Final Report

Smart AAC System with Facial Expression Recognition for Autism

Name: Yasas Pasindu Fernando

ID Number: 25026764

Date: Friday, 22 May 2026

Supervisor: Ms. Niruni Fonseka,

Declaration

Module: CS6P05ES **Deadline: 05/22/2026**

Module Leader: Mr. Migara Alawatta **Student ID: 25026764**

PLAGIARISM

You are reminded that there exist regulations concerning plagiarism. Extracts from these regulations are printed below. Please sign below to say that you have read and understand these extracts:

(signature:)____yasaspasindufernando@gmail.com_____ Date: 05/22/2026

This header sheet should be attached to the work you submit. No work will be accepted without it.

Extracts from University *Regulations* on Cheating, Plagiarism and Collusion

Section 2.3: "The following broad types of offence can be identified and are provided as indicative examples…

(i) Cheating: including taking unauthorised material into an examination; consulting unauthorised material outside the examination hall during the examination; obtaining an unseen examination paper in advance of the examination; copying from another examinee; using an unauthorised calculator during the examination or storing unauthorised material in the memory of a programmable calculator which is taken into the examination; copying coursework.

(ii) Falsifying data in experimental results.

(iii) Personation, where a substitute takes an examination or test on behalf of the candidate. Both candidate and substitute may be guilty of an offence under these Regulations.

(iv) Bribery or attempted bribery of a person thought to have some influence on the candidate's assessment.

(v) Collusion to present joint work as the work solely of one individual.

(vi) Plagiarism, where the work or ideas of another are presented as the candidate's own.

(vii) Other conduct calculated to secure an advantage on assessment.

(viii) Assisting in any of the above.

Some notes on what this means for students:

1. Copying another student's work is an offence, whether from a copy on paper or from a computer file, and in whatever form the intellectual property being copied takes, including text, mathematical notation and computer programs.

2. Taking extracts from published sources *without attribution* is an offence. To quote ideas, sometimes using extracts, is generally to be encouraged. Quoting ideas is achieved by stating an author's argument and attributing it, perhaps by quoting, immediately in the text, his or her name and year of publication, e.g. "$e = mc^2$ (Einstein 1905)". A *references* section at the end of your work should then list all such references in alphabetical order of authors' surnames. (There are variations on this referencing system which your tutors may prefer you to use.) If you wish to quote a paragraph or so from published work then indent the quotation on both left and right margins, using an italic font where practicable, and introduce the quotation with an attribution.

Acknowledgements

I would like to express my sincere gratitude to my first supervisor, Ms. Niruni Fonseka, at ESOFT Metro Campus, and my second supervisor, Mr. Akila Udara Akalanka, for their continuous guidance and constructive feedback throughout this project. I also thank Mr. Migara Alawatta, the CS6P05ES module leader, for the structured project guidance provided through the module.

I am very grateful to the team at Pragathi Centre / National Hospital Galle, particularly the Head of Pragathi Centre, for taking the time to engage with the project, sharing clinical perspectives on children with communication difficulties, and providing a formal letter of support for the proposed Pragathi AAC application. Their interest in using the system as a supportive observation and learning tool, including for medical students, gave the project a practical direction beyond a typical academic prototype.

I also wish to thank the staff at Karapitiya Teaching Hospital for the openness shown during early discussions, the speech-language therapists and parents who provided informal feedback on the AAC interface, and Ideahub (Pvt) Ltd for early-stage discussions about a future iOS release pathway. My appreciation extends to the open-source communities behind Flutter, TensorFlow, TensorFlow Lite and the Kaggle datasets that supported the model training stage. Finally, I am grateful to my family and friends for their constant encouragement throughout this final year.

Abstract

Children with autism spectrum disorder can experience significant barriers when speech is limited, especially in settings where affordable augmentative and alternative communication (AAC) tools are not available in the child's own language. In Sri Lanka, this gap is compounded by the need for Sinhala, Tamil and English support, offline use, and low-cost access for families who cannot depend on expensive imported AAC products. This project developed a supportive mobile AAC system that combines trilingual communication cards with an on-device facial expression recognition (FER) layer for careful observation support.

The system was implemented mainly in Flutter as an offline-first Android application. Core AAC content is stored locally, text-to-speech supports Sinhala, Tamil and English where device voices are available, and a companion simple_aac_app provides caregiver-customisable cards with manual backup, restore and sharing. The FER component runs on-device through TensorFlow Lite and does not upload camera frames. A mobile-safe EfficientNetB0 model is loaded as the primary model, while MobileNetV2 is retained as a fallback if the primary model fails to load or fails tensor validation. The camera workflow includes face detection, confidence checking, top-1/top-2 margin checking and explicit uncertain states so that the system does not force an emotion label on unclear input.

Evaluation focused on implementation evidence, TensorFlow Lite compatibility checks, release APK verification, manual real-device walkthroughs and Google Play internal testing distribution. Collaboration interest from Pragathi Centre / National Hospital Galle is supported by a formal letter, but wider clinical deployment and any pilot study remain subject to ethical clearance and health-sector approval. Model evaluation is reported using the saved learning curves, confusion matrices and TensorFlow Lite verification evidence included in the implementation and testing chapters. The outcome is therefore a working foundation for free or low-cost supportive AAC and observation in Sri Lanka, not a diagnostic medical device or a clinically approved system.

Table of Contents

List of Figures

List of Tables

Abbreviations and Glossary

The following abbreviations are used in this report. Where the same abbreviation appears in a chapter for the first time, it is also written out in full once for clarity.

Abbreviations

Abbreviation	Full form
AAC	Augmentative and Alternative Communication
AI	Artificial Intelligence
API	Application Programming Interface
APK	Android Package Kit
ASD	Autism Spectrum Disorder
BN	Batch Normalisation
CBAM	Convolutional Block Attention Module
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DSM-5	Diagnostic and Statistical Manual of Mental Disorders, 5th Edition
ER	Entity Relationship (diagram)
FER	Facial Expression Recognition
GDPR	General Data Protection Regulation
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
ML	Machine Learning
ML Kit	Google ML Kit (on-device machine learning)
NFR	Non-Functional Requirement
OS	Operating System
OSS	Open-Source Software
RGB	Red, Green, Blue (image colour space)
SQLite	Structured Query Language Lite

Abbreviation	Full form
TFLite	TensorFlow Lite
TLS	Transport Layer Security
TTS	Text-to-Speech
UI	User Interface
UX	User Experience
WCAG	Web Content Accessibility Guidelines

Glossary of project-specific terms

- **AAC interface**. The screen-based vocabulary of symbols, categories and audio output that supports communication for a child with autism spectrum disorder.
- **Caregiver**. The parent, guardian, teacher or therapist who supervises the child while the application is in use. Caregivers are expected to be present during camera-based features.
- **Confidence threshold**. The minimum model confidence below which the application refuses to report an emotion and instead enters an *Uncertain* state.
- **Emotion-aware AAC**. An AAC application that uses on-device facial expression recognition to provide supportive, non-diagnostic emotional cues alongside vocabulary support.
- **Fallback model**. A smaller, mobile-safe model (here, MobileNetV2) that the application loads automatically if the primary EfficientNetB0 model cannot be loaded or fails tensor validation.
- **Internal testing**. A Google Play distribution track that allows the developer to invite a small group of named testers without making the application publicly available.
- **Offline-first**. A design principle where the core features of the application work without internet connectivity, with optional synchronisation kept as a future improvement.
- **Pragathi AAC App**. The deployed product name of the application, as referred to in correspondence from Pragathi Centre / National Hospital Galle.
- **Primary model**. The EfficientNetB0 fine-tuned TensorFlow Lite model loaded first by the application.
- **Supportive observation**. The framing applied throughout this report to describe the AI component: it is a supportive aid for caregivers, therapists and medical students. It is not a diagnostic tool.
- **Top1-top2 margin**. A confidence-style check that compares the highest-scoring class against the second-highest class to reduce unreliable predictions.

Chapter 1: Introduction

This chapter introduces the project, the problem it addresses and the way the rest of the thesis is structured. It begins with the background and the context of communication difficulties for children with autism in Sri Lanka, particularly around the lack of trilingual Augmentative and Alternative Communication (AAC) tools. It then describes the proposed Smart AAC System with Facial Expression Recognition, the aim and objectives, the research questions and the scope of the work. The chapter ends with a summary of the report structure and a careful statement about the current clinical collaboration with Pragathi Centre / National Hospital Galle.

Detailed discussions of the literature, design and implementation are presented in the later chapters. Where claims depend on external evidence, the relevant sources are acknowledged using Harvard-style in-text referencing.

1.1 Background and Context

Autism spectrum disorder (ASD) is a neurodevelopmental condition characterised by persistent difficulties in social communication and by restricted, repetitive patterns of behaviour (American Psychiatric Association, 2013). The condition presents along a spectrum, and the DSM-5 organises severity into three levels, from Level 1 (“requiring support”) to Level 3 (“requiring very substantial support”). Children at Level 3 frequently have very limited or no functional speech and depend heavily on augmentative and alternative communication to express basic needs, preferences and emotions (Beukelman and Light, 2020). Globally, the World Health Organization (2021) reports a prevalence of roughly one in 160 children, while more recent surveillance work in high-income countries has reported rates closer to one in 36 (Maenner et al., 2023).

In Sri Lanka, a population-based study by Perera et al. (2019) reported an ASD prevalence of approximately 1.07 per cent among children aged two to nine, broadly in line with global estimates. However, awareness and access to services do not yet match this prevalence. Specialist support such as speech-language therapy, applied behaviour analysis and assistive communication training tends to be concentrated in major teaching hospitals like Karapitiya and Lady Ridgeway (Samad et al., 2020; Wickramasinghe et al., 2021). Outside these centres, families often have to travel for assessments and follow-up sessions, which is not always practical. Beyond access, language is also a structural barrier: Sri Lanka has two official languages, Sinhala and Tamil, and most commercially available AAC tools are built primarily for English-speaking, Western populations. As a result, the vocabularies, symbols and cultural assumptions that ship with these tools rarely fit a Sri Lankan child’s daily life (Alant and Bornman, 2021; Samad et al., 2020).

There is therefore a clear local need for a low-cost, culturally appropriate AAC application that supports Sinhala, Tamil and English, that runs offline on the types of phones families typically own, and that respects the wider context in which Sri Lankan children with autism are supported by parents, caregivers, therapists and clinical staff. Many of these families come from low-income backgrounds. Commercial AAC products that work in English alone, that rely on a paid subscription, or that require a desktop dashboard, are not realistic for them. This project is therefore positioned as a free or low-cost assistive communication tool rather than as a paid commercial AAC product. The motivation is social-impact-led: to make trilingual AAC available to children with communication difficulties whose families cannot afford the type of commercial AAC systems sold in higher-income markets. This is research and a social goal at the same time, and it shapes most of the technical decisions described later in the report.

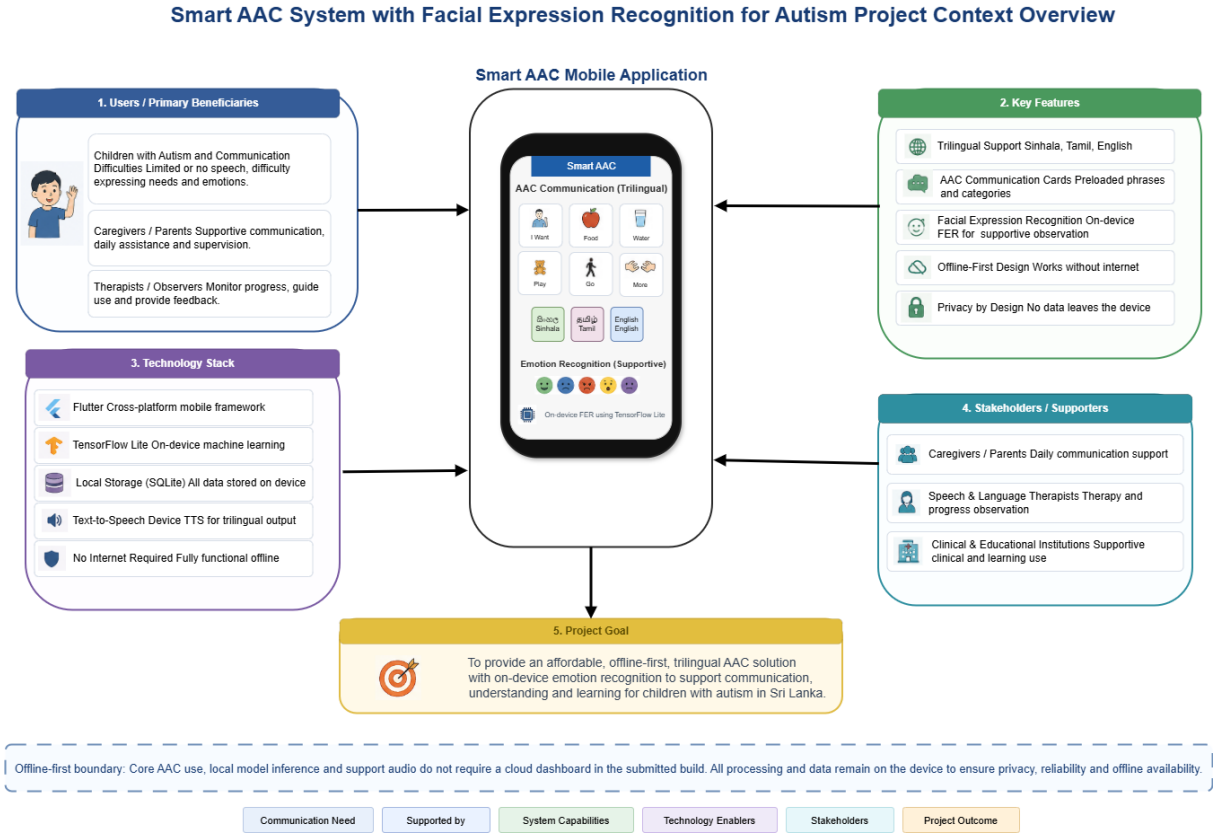


Figure 1.1: Project context overview.

Editable diagram source: [Download editable Project context overview.]

1.2 The Sri Lankan AAC Problem

Looking specifically at the Sri Lankan setting, the AAC gap can be summarised in four overlapping points.

1. **Language coverage.** No commercially available AAC system currently supports both Sinhala and Tamil as fully built-in languages. Avaz AAC supports Tamil, which is helpful, but Sinhala remains largely missing from existing offerings (Samad et al., 2020). For trilingual families and clinical settings, the absence of Sinhala in AAC tools is a real practical issue.
2. **Cultural fit.** Symbols, food items, clothing, religious activities and family structures used in mainstream AAC vocabularies do not always reflect Sri Lankan daily life. The materials need genuine local adaptation rather than translation alone (Alant and Bornman, 2021).
3. **Cost and connectivity.** Dedicated speech-generating devices and premium AAC apps can be expensive. Internet connectivity is also uneven across the country, particularly outside major urban areas (International Telecommunication Union, 2022). An offline-first design, deployed as a mobile application that families already have the hardware for, is therefore a far more realistic option than a cloud-only product.
4. **Limited integration with emotional cues.** Children with autism often experience difficulties with emotional regulation alongside communication (Mazefsky et al., 2013). Most AAC systems present the same vocabulary regardless of how the child is feeling, which is a missed opportunity for an emotion-aware design. Picard (2000) framed the wider field of affective computing many years ago, but the integration of facial expression recognition into AAC tools - especially in low- and middle-income countries - remains very limited.

This project responds to these four points through complementary mobile applications designed for the Sri Lankan context: a structured Smart AAC build with on-device emotion support for higher-support profiles, and a separate customisable AAC build for caregiver-led vocabulary (see Section 1.3).

1.3 Project Overview

The project develops a mobile **Smart AAC System with Facial Expression Recognition** for children with communication difficulties in Sri Lanka, with autism as the primary target population. The application is referred to in clinical correspondence as the **Pragathi AAC App**, named in collaboration with Pragathi Centre / National Hospital Galle.

Architecturally, the main Smart AAC build consists of two primary layers.

- **A trilingual AAC layer that lets a child move through fixed, therapist-aligned categories (for example needs, feelings, food, family and activities) with text-to-speech output in Sinhala, Tamil or English, plus shortcuts such as pinned favourites from the same vocabulary. Caregiver-created symbols, photographs and recorded voice lines are implemented in the companion simple_aac_app, which is aimed at the Level 1-2 profile (Chapter 5, Section 5.7, Chapter 6, Section 6.3.7).**
- **A supportive on-device facial expression recognition layer (main application only) that uses the device's front-facing camera. A TensorFlow Lite model classifies the camera frame into one of six emotional states: Anger, Fear, Joy, Natural, Sadness and Surprise. These are mapped in the interface to Angry, Fear, Happy, Neutral, Sad and Surprise. The application can then play a short supportive audio prompt in the active language. The output is treated as observation support, not as a clinical diagnosis.**

Both applications are built using Flutter. The main Smart AAC application uses SharedPreferences for user state (for example language, accessibility and favourites) and compiled Dart vocabulary in lib/data/word_data/ rather than a run-time SQLite database, which keeps the release APK smaller and more predictable. The companion simple_aac_app persists caregiver-edited cards as JSON in SharedPreferences and uses ZIP-based backup and restore. Both builds follow an offline-first design. On-device inference in the main application uses TensorFlow Lite. The Flutter model assets folder ships with two compact models. The application loads the EfficientNetB0 fine-tuned model first as the primary model and automatically falls back to the MobileNetV2 model if the primary model fails to load or fails tensor validation. A debug screen also allows the active model to be viewed and selected during testing.

The Android version is currently distributed through internal testing on the Google Play Store, with a signed release APK (versionName 1.1.0, versionCode 2, application ID lk.aac.sinhala_tamil_english) present in the project tree as evidence of the deployable build. Future iOS release support is being explored through Ideahub (Pvt) Ltd, subject to approval, testing and release requirements. Early-stage discussions have touched on developer-account and distribution needs. However, no public App Store release has been undertaken.

The interim version of this project described a dual-platform split (Platform A for ASD Level 1-2 and Platform B for Level 3+). The final delivered system kept both ideas, but the way they are packaged evolved during development. A separate customisable AAC application (simple_aac_app) was developed in parallel to give Level 1-2 children a caregiver-customisable card-based AAC experience with manual backup, restore and share features (cited in Chapter 5, Section 5.7 and Chapter 6, Section 6.3.7). The main Smart AAC application in this repository targets the Level 3+ scenario with a static, therapist-guided AAC flow and adds the supportive on-device facial expression recognition layer. Both apps share trilingual support, offline-first behaviour and local storage. The original idea of a single Flutter app with feature-level differentiation is therefore reflected as two complementary Flutter applications that share the same design principles. This was a practical decision: maintaining one clean per-profile app is easier for caregivers and clinical contacts to install and demonstrate than a single overloaded application with conditional flows.

1.3.1 Supervision and the research contribution (model path)

Based on supervisor feedback, it was determined that the primary research contribution must encompass the end-to-end development by the author (dataset handling, training runs, TensorFlow Lite export, mobile integration and objective evaluation) rather than relying solely on a ready-made pre-trained model. The final work reflects that guidance. The objective was not to prove a single architecture in isolation, but to make on-device facial expression recognition work practically inside a mobile AAC workflow, including training-runtime parity, a deployable model pair and caregiver-safe behaviour (confidence gating, uncertainty states, no diagnostic wording).

Notebook experiments included EfficientNetB0 with CBAM because attention improved feature focus in training logs, but CBAM increased TensorFlow Lite deployment risk (Flex / SELECT_TF_OPS). The shipped build therefore uses a mobile-safe EfficientNetB0 model without CBAM as the primary network and MobileNetV2 as a fallback. Model choice weighed validation behaviour, per-class recall and confusion-matrix patterns, possible bias given Western-dominated public datasets, TFLite compatibility, Flutter loadability and reliability on mid-range devices. Model metrics are reported through the saved model evaluation figures, confusion matrices and TensorFlow Lite verification evidence in Section 6.4 and Section 7.5.

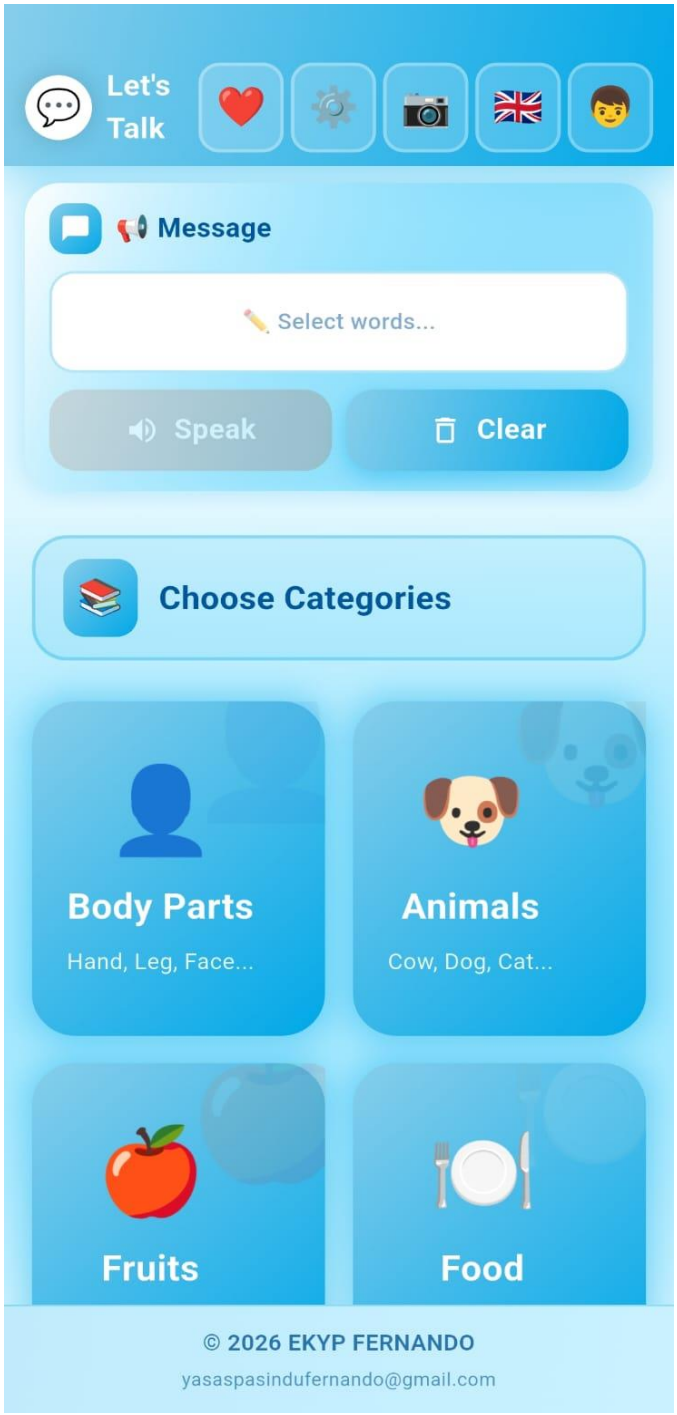


Figure 1.2: Pragathi AAC App.

The screenshot shows the deployed home screen of the mobile application used for final-stage testing.

The project does not currently include a cloud-based therapist or parent dashboard. The interim report listed this as a planned component for collaboration, progress monitoring and vocabulary management. During the final implementation stage, that objective was revised. Rather than storing sensitive child and face-related data in a cloud dashboard, the final system prioritised offline mobile use, local storage and manual sharing, backup and restore features. This decision was made because the intended users include caregivers, speech therapists and doctors working in real clinical and home settings, where mobile access is more realistic than laptop-based dashboard access. The justification for this scope change is set out in Chapter 3, Section 3.3.7 and Chapter 4, Section 4.7.

Most of the planned implementation objectives have been completed at the time of submission. The main Android application is substantially complete, the companion simple_aac_app delivers the planned customisation path, the core AAC features and trilingual content are in place, the on-device facial expression recognition layer is integrated in the main build, the mobile-safe TFLite model pair is packaged, and the optional supportive audio prompts are included. The remaining work mainly relates to formal approval, wider clinical deployment, extended pilot evaluation and future release activities. Google Play internal testing has been used, and a production access request has been submitted for review. Public production release is not claimed.

1.4 Project Aim

The aim of this project is:

To design, develop and evaluate a mobile Smart AAC system with on-device facial expression recognition to support children with communication difficulties in Sri Lanka, while providing trilingual AAC access, offline operation and safe caregiver / clinical support features.

The aim deliberately treats the AI component as **supportive**, not diagnostic. The facial expression recognition layer is intended to assist caregivers, therapists and medical students by surfacing possible emotional cues in a clear and consistent way. It does not replace clinical judgement and it does not attempt to diagnose autism or any related condition.

1.5 Objectives

The objectives below are the *final* objectives that the delivered system is evaluated against. They have been revised from the interim list (Fernando, 2026) so that they match what was actually built rather than the original two-app, dashboard-led plan.

1. **To analyse** the AAC needs of children with autism in Sri Lanka, including language, cultural, cost and access barriers.
2. **To design** a practical mobile-first AAC solution supporting Sinhala, Tamil and English for different communication profiles (Level 1-2 customisable use, Level 3+ structured supportive use).
3. **To implement** offline-first AAC features with local storage, customisation, manual backup, restore and sharing, and caregiver-friendly use on the kinds of mobile devices Sri Lankan families typically own.
4. **To train and integrate** a mobile-compatible on-device facial expression recognition model using TensorFlow Lite, with a primary/fallback model strategy that prioritises Flutter loadability and mobile reliability.
5. **To evaluate** the system through real-device testing, model verification, application testing and informal stakeholder feedback from clinical contacts at Pragathi Centre / National Hospital Galle, alongside a self-evaluation against the stated requirements.
6. **To document** ethical, data protection and deployment considerations for future clinical or pilot use, including the safe positioning of the AI component as supportive observation rather than diagnosis.

Each objective is revisited explicitly in Chapter 8, where the evidence supporting each objective is mapped against the work delivered.

1.6 Research Questions

The work is guided by the following five research questions. They have been refined from the research questions in the interim report (Fernando, 2026) to reflect the final scope of the project and the practical realities of mobile deployment, fallback model design and clinical collaboration.

RQ1. How can a trilingual mobile AAC system be designed to support children with communication difficulties in the Sri Lankan context, including local language coverage and offline operation?

RQ2. How can on-device facial expression recognition be integrated into an AAC application while preserving privacy and mobile performance?

RQ3. What practical challenges arise when converting trained emotion recognition models into TensorFlow Lite models for Flutter deployment, and how can a primary / fallback model strategy reduce deployment risk?

RQ4. How useful is the proposed system as a supportive observation and communication tool for caregivers, medical students and clinical stakeholders involved in supporting children with autism?

RQ5. What limitations and future improvements (technical, ethical and clinical) are required before the system can be considered for wider clinical or public use?

These research questions are each revisited in Chapter 8.

1.7 Scope

To keep the project deliverable within the CS6P05ES timescale and to respect the ethical boundary around working with children, the scope is deliberately limited.

In scope

- Design and implementation of **two complementary Flutter Android applications** that together support Sinhala, Tamil and English: the main Smart AAC application (structured vocabulary, on-device emotion support) and the companion simple_aac_app (caregiver-customisable cards, manual backup and restore).
- Local storage appropriate to each build: **SharedPreferences plus compiled Dart vocabulary** in the main application, and **JSON card sets in SharedPreferences with ZIP backup / restore** in simple_aac_app.
- An on-device facial expression recognition layer using TensorFlow Lite, with the EfficientNetB0 fine-tuned model as the primary model and the MobileNetV2 fallback model as a safety net.

- Model training on three publicly available Kaggle datasets in Google Colab, including dataset audit, class imbalance handling, preprocessing alignment with the Flutter pipeline, TensorFlow Lite export and verification.
- Functional, unit and real-device testing on Android, plus internal testing distribution through the Google Play Store.
- Collaboration with Pragathi Centre / National Hospital Galle for clinical input, design feedback and the intended pilot work after ethical clearance.

Out of scope (current stage)

- A confirmed Apple App Store release. Future iOS release support is being explored through Ideahub (Pvt) Ltd, subject to approval, testing and release requirements. No App Store release or Apple developer account evidence is claimed in this submission.
- Final Ministry of Health approval. The application is not certified for clinical use and is not a medical device.
- A clinical pilot study with formal outcomes. A pilot is intended only after ethical clearance and within the boundaries described in the hospital letter.
- Real-time emotion-based therapy decisions or diagnosis. The AI component is supportive, not diagnostic.
- A locally collected Sri Lankan facial expression dataset. This is identified as future work because it requires child safeguarding, data protection and ethics approval that go beyond the current project.

Assumptions

- Caregivers will be present when the application is used by a child, particularly when the camera feature is enabled.
- The mobile device used has a front-facing camera, runs Android 8.0 or above, and supports basic TFLite inference on the CPU.
- Internet connectivity is not assumed because the application must remain usable when fully offline.

1.8 Report Structure

The remainder of this thesis is organised as follows.

- **Chapter 2 - Background and Problem Statement** follows the report template by introducing the problem domain, reviewing the literature on ASD, AAC, existing tools, FER and privacy, and then stating the specific problem this project addresses.
- **Chapter 3 - Project Management** describes how the project was planned and re-planned, including the initial Gantt chart, the changes made during the year and the final project record.
- **Chapter 4 - Feasibility Study** reviews the project from time, cost, scope, technical, economic and ethical perspectives, drawing on the work plan and the constraints of working in a clinical-adjacent setting.
- **Chapter 5 - Design** presents the system architecture, the data model, the use case view, the functional and non-functional requirements, and the emotion detection pipeline.
- **Chapter 6 - Implementation** documents how the Flutter application and the AI model were built. The model section follows the structure of the project's internal *Model Training and Deployment* document, including dataset preparation, class imbalance handling, EfficientNetB0 fine-tuning, MobileNetV2 fallback training, TensorFlow Lite export and Flutter integration.
- **Chapter 7 - Testing and Verification** sets out the test plan, the test cases, the test results captured to date and the testing evidence and remaining validation limitations.
- **Chapter 8 - Evaluation and Conclusion reviews the system against its objectives, discusses what worked and what did not, summarises the stakeholder feedback so far, lists the limitations objectively, and suggests next steps for future work.**
- **Chapter 9 - References** lists every cited source in Harvard style.
- **Chapter 10 - Appendices** contains supporting evidence, including the interim progress report, the Pragathi Centre / National Hospital Galle letter, the project commencement form, model training notebook links and screenshots.

1.9 Clinical Collaboration and Approval Status

The clinical status of the project is outlined here to establish the appropriate context for subsequent claims.

Pragathi Centre / National Hospital Galle has provided a formal letter confirming a digital initiative undertaken in collaboration with the author for the application titled **Pragathi AAC App** (see Appendix A). The letter states that the application is designed to support children with communication deficits, particularly those diagnosed with autism spectrum disorder,

cerebral palsy and global developmental delay. It confirms that substantial clinical input and multidisciplinary expertise from the Pragathi team are involved, and that a **pilot project is intended only after obtaining the necessary ethical clearance**. Subject to the outcome of the pilot, wider availability is intended.

Stakeholders linked to **Karapitiya / Pragathi** have shown **practical interest** in the work, including the idea that the on-camera supportive layer may help **medical students notice possible emotional cues** during supervised observation sessions. This remains **supportive observation and communication support**, not clinical diagnosis of autism or of emotion, and wider use still depends on formal approval, ethical clearance and supervised validation. Securing this hospital-linked context is a meaningful outcome for a final-year student project, but it is not the same as regulatory sign-off.

This report defines the scope of the collaboration as follows:

- The collaboration with Pragathi Centre / National Hospital Galle is supported by a formal letter and is reported within the available evidence.
- However, **health-sector approval and wider deployment remain pending.** The Ministry of Health has not approved the application for clinical use.
- Any pilot work is being conducted, or planned, only within the approved and ethically cleared scope.
- The **AI component is supportive and educational**. It is not a diagnostic tool and it must not be presented or used as one.
- A future Sri Lankan dataset would help local fairness and relevance, but its collection must follow proper child safeguarding, data protection and ethics procedures. It is treated in this thesis as a clearly scoped future improvement rather than as already-collected evidence.

With these boundaries in mind, the rest of the thesis describes how the Pragathi AAC application has been designed, built and evaluated, and what work remains before it can be used more widely.

1.10 Chapter Summary

This chapter has introduced the local context, explained the gap in trilingual emotion-aware AAC for Sri Lanka, set out the aim, six objectives and five research questions, and described the scope of the work. Furthermore, the AI component was explicitly defined as a supportive tool, and the scope of the clinical collaboration with Pragathi Centre / National Hospital Galle was clearly delineated. Chapter 2 now develops the background and problem statement through a template-aligned literature review and problem statement.

Chapter 2: Background and Problem Statement

2.1 Introduction

This chapter sets out the background to the problem domain, reviews the literature most relevant to the project, and defines the specific gap the final system responds to. It covers autism spectrum disorder, augmentative and alternative communication, existing AAC tools, facial expression recognition, privacy and on-device AI, and the Sri Lankan need for Sinhala, Tamil and English communication support.

The chapter builds on the interim report (Fernando, 2026), but it updates the discussion to match the final implementation. Earlier ideas such as a Firebase dashboard, SQLite storage and an EfficientNetB0 + CBAM deployment path are treated as design evolution rather than final scope. The submitted system is an offline-first Flutter AAC application with a companion customisation path, on-device TensorFlow Lite inference, EfficientNetB0 as the primary model and MobileNetV2 as the fallback model.

2.2 Literature Review

2.2.1 Autism Spectrum Disorder and Communication Needs

Autism spectrum disorder (ASD) is a neurodevelopmental condition characterised by persistent difficulties in social communication and interaction, together with restricted or repetitive patterns of behaviour, interests or activities (American Psychiatric Association, 2013). DSM-5 describes support needs across three levels, from Level 1, requiring support, to Level 3, requiring very substantial support. This classification is relevant to the design, but it is not used rigidly. Children at Level 1-2 may benefit from flexible, caregiver-managed vocabulary, while children with more substantial communication needs may require a simpler and more predictable aided communication flow (Beukelman and Light, 2020).

Communication needs also change with context. Some children with ASD use speech but struggle with pragmatic communication, turn-taking or receptive understanding, while others have very limited functional speech. Lord et al. (2020) note that a substantial minority of children with ASD do not develop functional spoken language. Emotional regulation can further affect communication, because frustration, fear, tiredness or sensory overload may reduce a child's ability to express a need clearly (Mazefsky et al., 2013). This link between communication and emotional state supports the inclusion of a cautious emotion-observation layer beside AAC, while keeping the AI output supportive and non-diagnostic.

International prevalence studies also show why local design should be cautious. Elsabbagh et al. (2012) and Divan et al. (2021) show that autism evidence is uneven across regions, especially in low- and middle-income countries. This matters because a system designed

from high-income-country assumptions may not match local service access, caregiver workload, language use or device availability. Fletcher-Watson and Happe (2019) also emphasise that autism support should avoid treating all autistic children as a single group. For this project, that supports a practical design that allows different communication profiles rather than one rigid interaction model.

A further issue is that communication ability in autism does not map neatly onto one interface design. Some children may need only a small set of high-frequency messages, while others may benefit from richer vocabulary, favourites, caregiver-created cards or guided emotional support. Lord et al. (2020) and Fletcher-Watson and Happe (2019) show that ASD involves wide variation in communication, social interaction and support needs. For this thesis, that means the design cannot assume that a fixed grid alone will suit every child. The final system therefore treats AAC as a flexible communication support, with stable core categories in the main app and a separate customisable path for children who need more caregiver-created content.

Emotion regulation is also directly relevant to communication design. Mazefsky et al. (2013) explain that emotional regulation difficulties can affect behaviour, attention and expressive communication in children with ASD. In a practical AAC context, a child may not only need a word for an object, but also a way to express discomfort, fear, frustration or a need for help. This supports the inclusion of feelings, needs and caregiver-supervised observation features, while also showing why the emotion-support part must remain cautious and non-diagnostic.

Table 2.1: ASD severity levels and communication relevance.

The table is adapted from American Psychiatric Association (2013) and Beukelman and Light (2020).

ASD support profile	Typical communication need	Relevance to this project
Level 1-2	May use speech but need help with vocabulary, routines, stress and social communication.	Supports the companion customisable AAC path and caregiver-managed vocabulary.
Level 3+	May have very limited functional speech and require consistent aided communication for basic needs and emotions.	Supports the main simplified AAC flow and caregiver-controlled camera/emotion support.
Across levels	Needs differ by child, family language, sensory profile and context.	The system avoids a one-size-fits-all claim and keeps AAC, language and emotion support configurable where practical.

2.2.2 Augmentative and Alternative Communication

Augmentative and Alternative Communication (AAC) refers to strategies, tools and technologies that supplement or replace natural speech for people with complex communication needs (American Speech-Language-Hearing Association, 2022). AAC includes unaided approaches, such as gestures or sign, and aided approaches. Aided AAC ranges from picture boards and communication books to tablet applications and speech-generating devices (Beukelman and Light, 2020). For this project, the key issue is not only whether AAC can support communication, but whether the AAC system fits the child's language, routines, sensory needs and home context.

The literature supports AAC as a useful intervention for children with autism. Ganz et al. (2012) reported moderate to large effects for aided AAC interventions in single-case autism studies, and Lorah et al. (2015) found encouraging evidence for tablets and portable media players as speech-generating devices. Importantly, AAC is not shown to stop speech development. Millar, Light and Schlosser (2006) and Romski and Sevcik (2005) argue that AAC can reduce frustration and support expressive communication rather than replacing speech. This evidence supports the use of mobile AAC in the final system, but it does not remove the need for local language and cultural adaptation.

Light and McNaughton (2015) describe AAC competence in terms of linguistic, operational, social and strategic competence. The final system connects with these areas through trilingual labels and TTS, a consistent grid interface, caregiver-supported use and a supportive emotion-observation layer for moments when communication breaks down. This is more useful than treating AAC as a static symbol board, because the practical value of AAC depends on how easily a child and caregiver can use it during daily routines.

The evidence for AAC should also be read critically. Ganz et al. (2012) and Ganz (2015) support aided AAC for individuals with autism, but the evidence base often comes from small intervention studies, single-case designs and controlled teaching contexts. This means the literature supports AAC as a useful communication intervention, but it does not remove the need for local usability testing. A working AAC application still has to be understandable to caregivers, quick enough for daily routines and suitable for the language environment in which it will be used.

The positive AAC evidence is therefore useful, but it should not be read as a guarantee that any AAC app will work equally well in every context. Many AAC studies involve careful adult prompting, therapy routines and structured teaching sessions. A final-year software project cannot reproduce that clinical evidence base. What it can do is translate the strongest design implications into an implementable system: clear symbols, reliable speech output, low cognitive load, predictable navigation and caregiver control over how the tool is used.

The literature also challenges the common fear that AAC prevents speech. Millar, Light and Schlosser (2006), together with Romski and Sevcik (2005), argue that AAC can support communication development and reduce frustration rather than blocking speech. This matters for stakeholder acceptance, because caregivers may be hesitant to introduce a speech-generating tool if they believe it will replace natural speech. The final thesis therefore frames AAC as supportive communication, not as a substitute for therapy or language development.

Mobile and tablet-based AAC adds another practical layer. McNaughton and Light (2013) explain that mobile devices increased access to AAC by reducing cost and stigma compared with dedicated devices, while Lorah et al. (2015) found positive evidence for tablets and portable media players as speech-generating devices for individuals with ASD. However, Dawe (2006) warns that families can abandon assistive technologies when they are difficult to maintain or do not fit family routines. This supports the final project decision to prioritise a

simple mobile workflow, local storage and caregiver-supervised use rather than a complex dashboard-first design.

Mobile AAC is attractive partly because the device is already familiar. A smartphone or Android tablet is easier to carry than a specialist device and may be less socially visible in school, home or clinic settings. However, the same literature also shows that access alone is not enough. If the app is slow to open, difficult to maintain, dependent on accounts, or too complicated for a caregiver to explain, it may be abandoned even if the technical idea is valid. This is why the final design avoids a login-heavy workflow and keeps the main child-facing path direct.

For mobile AAC, maintenance is also part of usability. A caregiver may need to change a word, restore a device, share a backup or switch language without technical support. Dawe (2006) is useful here because the paper shows that families do not adopt assistive technology only because it is available. They adopt it when it fits routines and when the burden of managing it remains reasonable. The simple backup, restore and sharing path in the companion AAC application responds to that practical requirement without claiming to be a full clinical dashboard.

2.2.3 AAC in Sri Lanka and Multilingual Contexts

The Sri Lankan context creates requirements that are not fully addressed by a general AAC application. Perera et al. (2019) reported an ASD prevalence of approximately 1.07 per cent among children aged two to nine in Sri Lanka, while Samad et al. (2020) and Wickramasinghe et al. (2021) describe service-access challenges and the concentration of specialist support around larger centres. Families outside major urban areas may not be able to attend frequent therapy sessions, so a low-cost mobile tool that works at home has practical value.

Language is central to this problem. Sinhala and Tamil are official languages in Sri Lanka, while English is also used in education and clinical settings. A child may need Sinhala at home, Tamil in another family or community context, and English in school or therapy. Direct translation is not enough because AAC content also carries cultural assumptions. Symbols, examples, food items, family terms and daily routines should fit Sri Lankan life rather than assume Western vocabulary and imagery (Alant and Bornman, 2021). This is why the project treats trilingual AAC and local vocabulary as core design requirements rather than optional interface features.

Cost and connectivity also shape feasibility. Dedicated AAC devices and premium applications may be unrealistic for many low-income families, and connectivity remains uneven outside urban centres (International Telecommunication Union, 2022). The final design therefore uses compiled vocabulary, local storage and on-device inference. It also avoids making the core communication workflow depend on a cloud dashboard, because a cloud-first design would increase cost, maintenance and data-protection responsibilities.

Taken together, this means the literature review cannot be reduced to an infographic comparison of tools. The design problem is a practical communication problem involving language fit, caregiver routines, affordability, offline access and safe handling of sensitive child data. A technically advanced AAC system would still be unsuitable if families could not use it during ordinary home, school or clinic routines.

Cultural adaptation is also more than translation. Beukelman and Light (2020) and Alant and Bornman (2021) both frame AAC as participation in everyday communication, not only as vocabulary display. For Sri Lankan use, this means that symbols, examples, food items, family words and daily routines should feel familiar to the child and caregiver. A technically correct English-first symbol set translated into Sinhala or Tamil would not fully solve the access problem if the content did not match local practice.

The Sinhala, Tamil and English requirement also changes the design of examples and prompts. The same communication need may be expressed differently at home, at school and in a clinic. Family terms, food names, daily activities and respectful forms of address may not transfer cleanly from an English AAC vocabulary. A culturally adapted AAC tool must therefore support local content choices and not only switch the display language. This is one reason the final vocabulary is organised around everyday Sri Lankan communication needs rather than only generic AAC categories.

Caregiver involvement is especially important in multilingual AAC. A child may understand one language better than another, while a therapist, parent and teacher may prefer different languages in different settings. Beukelman and Light (2020) describe AAC as participation in real communication, so language choice should support the people around the child as well as the child. The project responds by keeping Sinhala, Tamil and English visible as first-class options, rather than treating one language as primary and the others as later add-ons.

Low-resource deployment also affects sustainability. Light and McNaughton (2012) argue that AAC practice continues to face challenges around participation, access and long-term support. In this project, those challenges translate into engineering decisions: avoid a system that needs constant internet, avoid cloud storage for sensitive child-related data, keep the interface learnable and keep future maintenance realistic for a student-led, non-profit direction.

The Sri Lankan service context also makes offline-first design more than a convenience feature. Perera et al. (2019) provide local prevalence evidence, while Samad et al. (2020) and Wickramasinghe et al. (2021) describe challenges in service access and parent support. If families must travel to larger centres for specialist support, a mobile AAC tool has value only if it can be used between appointments. Uneven connectivity and device affordability, reflected in wider digital-development data from the International Telecommunication Union (2022), make it risky to depend on cloud features for core communication.

This does not mean that cloud or dashboard features have no value. In a future approved deployment, a therapist workflow could help with vocabulary review, caregiver feedback and supervised model improvement. The literature and field context simply show that such a workflow cannot be treated as the starting point for this submitted build. The safer final implementation places the communication tool on the device first and leaves synchronisation or dashboards for later ethics-governed work.

2.2.4 Existing AAC Systems and Research Gap

The interim report compared systems such as Proloquo2Go, TouchChat, LAMP Words for Life, LetMeTalk, CoughDrop and Avaz AAC. These systems show that AAC is a mature assistive technology area, with strong vocabulary systems, symbol support, customisation and, in some cases, offline use. Their strength is important because this project is not attempting to replace established AAC research. The gap is narrower and more local: the available systems do not fully meet the combined language, cost, offline, privacy and emotion-support needs identified for this Sri Lankan context.

The comparison can be understood by themes rather than by listing products only. Existing tools often provide strong English-language AAC, structured vocabularies and configurable access methods. Some also support additional languages, and Avaz AAC is especially relevant because Tamil is within its language scope. However, the surveyed tools do not combine Sinhala, Tamil and English as first-class local communication languages with low-cost Sri

Lankan deployment, offline-first operation and an on-device emotion-observation layer. This project therefore addresses an integration gap: it brings together localised AAC, mobile deployment and privacy-preserving FER support, while still recognising that mainstream AAC systems remain more mature in vocabulary depth and clinical adoption.

The research gap is therefore an integration gap rather than a claim that existing AAC systems are weak. Established AAC systems are mature in vocabulary design, access methods and speech output, but they are not designed specifically around Sinhala, Tamil and English use in Sri Lankan low-resource settings. They also do not normally combine AAC with a privacy-preserving, on-device facial-expression support layer. The contribution of this project is to bring these requirements together in a small applied prototype, not to replace commercial AAC systems or clinical AAC practice.

This distinction is important for evaluating originality. The project is original in its local combination of trilingual AAC, offline-first mobile design, TensorFlow Lite deployment and supportive emotion observation. It is not original because each individual technology is new. The literature review therefore justifies the project by showing how known AAC and AI ideas had to be adapted to a specific language, cost, privacy and deployment context.

Table 2.2: Existing AAC systems and final research gap.

The comparison is adapted from the interim report and updated to match the final implementation.

Theme	What existing AAC systems commonly provide	Gap addressed by this project
Language and culture	Strong English and some international-language support, with limited local symbol adaptation.	Sinhala, Tamil and English labels/TTS with Sri Lankan vocabulary priorities.
Deployment and cost	Commercial apps or dedicated-device ecosystems can be costly for low-income families.	Android-first, free or low-cost deployment using phones families may already own.
Connectivity and privacy	Some systems support offline use but cloud accounts, dashboards or synchronisation may still be part of the wider model.	Offline-first AAC and on-device FER, with no raw camera-frame upload.
Emotion support	Mainstream AAC systems focus on symbol communication, prediction, vocabulary and access methods.	Supportive on-device emotion observation using TFLite, with uncertainty handling and no diagnostic claim.

2.2.5 Facial Expression Recognition and Affective Computing

Affective computing concerns systems that can recognise, interpret and respond to human affective states (Picard, 2000). Facial Expression Recognition (FER) is one branch of this field. Early FER work often uses basic emotion categories associated with facial movement patterns, especially the Ekman tradition, although later research warns that facial movement should not be treated as a perfect or universal read-out of emotion (Ekman and Friesen, 1971; Barrett et al., 2019). This caution is important for the thesis because the system observes possible cues. It does not diagnose emotion or autism.

Deep convolutional neural networks and transfer learning are widely used in FER because labelled expression datasets are limited, noisy and often collected outside the target population (LeCun, Bengio and Hinton, 2015; Li and Deng, 2020; Yosinski et al., 2014). MobileNetV2 is attractive for mobile deployment because it is compact and efficient, while EfficientNetB0

offers a stronger balance between capacity and model size for transfer learning. The final system uses this comparison practically: EfficientNetB0 is used as the primary model, and MobileNetV2 is retained as a fallback to reduce deployment risk on Android devices.

In FER, model choice is both a machine-learning and a software-engineering decision. LeCun, Bengio and Hinton (2015) explain the value of deep neural networks for representation learning, and Yosinski et al. (2014) show why transfer learning can be useful when target data are limited. However, a mobile AAC app has constraints that are less visible in a notebook: APK size, memory, start-up time, plugin compatibility and whether the model can run without unsupported TensorFlow operations. This is why the final thesis gives weight to TensorFlow Lite compatibility, not only training performance.

The interim report explored EfficientNetB0 with CBAM because attention modules can help a model focus on more useful facial regions. In the final implementation, this research idea was kept as experimental background rather than deployed architecture. The CBAM model introduced TensorFlow Lite Flex / SELECT_TF_OPS requirements that were unsuitable for the mobile Flutter runtime used in the submitted build. This decision shows an important trade-off in applied AI projects: a model that is theoretically stronger is not necessarily the best final choice if it cannot be deployed reliably on the target device.

The FER literature also highlights a data-fit problem. Public facial-expression datasets are useful for training and comparison, but they often reflect different populations, camera conditions and cultural contexts from Sri Lankan children. For that reason, the final system treats FER as a supportive observation layer, keeps uncertain states visible and reserves local child face-data collection for future ethics-approved work.

FER research has a second limitation beyond model architecture. Large datasets such as AffectNet were designed for broad affective-computing research, but they still cannot guarantee suitability for Sri Lankan children, clinical-adjacent contexts or autistic expression patterns (Mollahosseini, Hasani and Mahoor, 2019). Trevisan, Hoskyn and Birmingham (2018) and Grossard et al. (2020) also show that facial expression production in autism can be more variable or ambiguous than standard emotion categories assume. Therefore, a high notebook result would not be enough to justify diagnostic use. The safer design is to present FER output as tentative support and to keep no-face, unclear and uncertain states visible.

Emotion categories also need careful interpretation. Ekman and Friesen (1971) provide an important starting point for cross-cultural facial-expression research, but Barrett et al. (2019) warn against assuming that facial movement directly reveals a single internal emotion. This matters strongly for children with autism because expression style, attention, sensory overload and context can all affect the face shown to the camera. The final system therefore uses labels as supportive cues for a caregiver, not as statements of what the child definitely feels.

The fallback model strategy follows from this applied view of FER. EfficientNetB0 was kept as the primary model because it offered a stronger transfer-learning route, while MobileNetV2 was retained because it is a compact mobile architecture. The fallback is not presented as a way to improve clinical accuracy. It is an engineering safeguard so that the camera feature has a predictable path if the primary model fails to load or fails tensor validation on a device.

2.2.6 Privacy, Ethics and On-device AI

Privacy considerations are particularly strict in this project because the camera may capture children's faces. Face images from children are sensitive personal data and, in some contexts, biometric data. For that reason, cloud-based face processing is not appropriate for the current academic build. All FER inference runs on-device through TensorFlow Lite, raw camera frames are not uploaded, and the current build does not use Firebase, a cloud database or analytics for child face data.

This privacy position also explains the change from the interim dashboard idea to the final offline-first design. A therapist-parent cloud dashboard could be useful in a future ethically governed deployment, especially for supervised review and structured feedback. However, it would introduce consent, data minimisation, access control, storage cost and long-term maintenance questions that are beyond the current coursework scope. The final project therefore uses local storage and caregiver-controlled backup, restore and sharing through the companion AAC path, keeping the user in control of what leaves the device.

Privacy and AI ethics literature reinforces this technical choice. Floridi et al. (2018) and Jobin, Ienca and Vayena (2019) argue that AI systems should be designed with transparency, accountability, privacy and human benefit in mind. For a child-facing AAC application, this means the system should minimise data collection, avoid unnecessary cloud processing and

make the limits of the AI visible to the caregiver. The final architecture follows this principle by keeping raw camera frames on the device and treating any future dashboard or dataset pathway as ethics-dependent work.

The privacy argument is also linked to trust. A caregiver may accept a camera feature only if it is clear that raw images are not being stored or uploaded. For children, this concern is stronger because face data is sensitive and the child may not be able to understand or refuse data processing in the same way as an adult. On-device processing, no raw-frame storage and explicit uncertainty states are therefore ethical design choices as well as technical ones.

Future data collection would need a different evidence and governance standard from the submitted prototype. If a Sri Lankan child FER dataset or reviewer dashboard is later developed, it would need ethics approval, parental or guardian consent, clear access control, secure storage, retention rules and health-sector supervision. The current thesis keeps that boundary clear by using public datasets for model development and by avoiding a cloud workflow for child face-related data.

The same boundary applies to future dataset work. A Sri Lankan child facial-expression dataset could improve local fairness and reduce dependence on public datasets, but it cannot be collected informally. It would need formal ethics approval, parental or guardian consent, secure storage, defined retention and deletion rules, and health-sector authorisation. No such dataset collection is claimed in this thesis.

2.3 Problem Statement

The problem addressed by this project is that Sri Lankan children with autism and communication difficulties lack a culturally appropriate, trilingual, low-cost and offline-first AAC application that also provides supportive on-device emotion observation. Existing AAC tools provide valuable communication support, but they do not fully cover Sinhala, Tamil and English for the Sri Lankan context, and they do not combine AAC with privacy-preserving on-device FER in a deployable mobile workflow.

The final project attempts to respond to this gap by delivering a Flutter Android AAC system with Sinhala, Tamil and English content, text-to-speech support, local/offline operation, a companion customisation and sharing path for Level 1-2 style AAC needs, and an on-device FER feature for supportive observation in the main build. The emotion layer uses TensorFlow Lite with EfficientNetB0 as the primary model and MobileNetV2 as fallback. Its purpose is to

help caregivers, therapists and medical students notice possible emotional cues, not to diagnose autism or make clinical decisions.

The local stakeholder context strengthens the practical relevance of the problem. Pragathi Centre / National Hospital Galle has provided a letter confirming clinical interest and an intended pilot after ethical clearance. This supports the project's social value, but it is not clinical approval, Ministry approval or evidence of completed clinical validation.

In summary, the project provides a working foundation for free or low-cost trilingual AAC in Sri Lanka, with safe on-device emotion observation kept inside clear ethical boundaries. Chapter 3 details the project management methods used to deliver the system.

The literature review therefore leads to a specific final design requirement. The system must support Sinhala, Tamil and English, work offline with lightweight local storage, avoid raw camera-frame upload, run FER on-device, handle model loading safely through a primary/fallback strategy, and present emotion output only as caregiver-supervised support. These requirements define the final project more accurately than the early dashboard-heavy concept, and they provide the basis for the design decisions in Chapter 5.

Chapter 3: Project Management

This chapter records how the project was planned and how the plan changed during the year. It covers the development approach, the original sprint-style plan, the changes made during the project (most of which became necessary because of practical mobile-deployment constraints), the risks that were tracked, and the final project record. Where the interim report (Fernando, 2026) already documented an event, this chapter cites it rather than repeating the same content in full.

3.1 Approach

The project was run as a single-developer, iterative final-year project with regular supervisor reviews. The author chose an **adapted Agile / Scrum-style** approach with two-week mini-sprints, but kept it pragmatic. A strict Waterfall plan was rejected because three parts of the work were uncertain in advance:

1. The AI model training results would only be known after experiments in Google Colab. Performance and TFLite export behaviour could not be predicted from a plan.
2. The clinical pathway (Pragathi Centre / National Hospital Galle) depends on ethical clearance, which is outside the author's control.
3. UI feedback from caregivers and informal observers could only be gathered after a working prototype existed.

These uncertainties necessitated an iterative approach with regular short loops, rather than a long up-front planning phase.

The development cycle used in each mini-sprint was:

1. **Plan**. Pick one feature or risk area from the backlog (e.g. *EfficientNetB0 export*, *Camera screen face gate*, *Trilingual TTS*).
2. **Build**. Implement on the Flutter side and / or in the Colab notebook.
3. **Verify**. Run unit tests where possible (test/widget_test.dart, test/storage_service_test.dart, test/app_theme_test.dart), test on emulator, then on a real Android device.
4. Reflect. Record the result, including failures, in project notes so that later changes can be traced.
5. **Re-plan**. Adjust the backlog based on what was learnt.

The tools used to support this approach were lightweight: Git for source control, Google Sheets for sprint tracking, and Google Drive for the Colab notebook and model artefacts. No commercial project-management tool was used.

In practical terms, this was Scrum adapted for a single-developer undergraduate project rather than full industrial Scrum. The author acted as developer, backlog owner and coordinator, while the supervisor review provided the main external review point. Sprint planning was used to choose the next set of deliverables from the backlog, and sprint review was used to decide whether each item should move to done, remain open, or be re-scoped.

Backlog changes were common but controlled. Model-training results, TensorFlow Lite conversion problems, real-device testing, field-visit observations and supervisor feedback all affected the next sprint. For example, the dashboard and Firebase route moved to future work, the SQLite-heavy data layer was replaced by SharedPreferences and compiled Dart vocabulary, and CBAM was treated as an experiment rather than a deployed mobile model.

Google Sheets was suitable for this workflow because the project was managed by one developer and needed quick, shareable updates rather than an enterprise tool. It was used for backlog, sprint status, issues, decisions, milestones and time notes. Informal mobile notes supported field observations and discussions, but no patient-identifiable or clinical data were recorded in those notes.

The sprint organisation used in the final project is summarised below.

Sprint / phase	**Main work completed**	**Evidence or output**	**Outcome or change made**
1. Literature review, proposal and requirements	Reviewed ASD, AAC, FER, multilingual AAC and Sri Lankan deployment constraints.	Chapter 2, proposal material and interim planning evidence.	Confirmed the need for a trilingual, low-cost and offline-first AAC direction.
2. Design artefacts and architecture	Prepared architecture, use case, data model, ER and class-level design artefacts.	Chapter 5 diagrams and Appendix C supporting design evidence.	Kept the final architecture mobile-first

Sprint / phase	Main work completed	Evidence or output	Outcome or change made
			and moved interim-heavy dashboard ideas to future scope.
3. Core Flutter AAC screens and local storage	Built registration, home, category, favourites and settings flows with SharedPreferences-backed state.	Chapter 6 implementation sections and manual walkthrough evidence.	Replaced the SQLite-heavy interim plan with simpler local state and compiled vocabulary.
4. Trilingual vocabulary, TTS and customisable AAC path	Implemented Sinhala, Tamil and English labels/TTS and retained the customisable AAC path through simple_aac_app.	Chapter 5 trilingual flow and Chapter 6 customisation evidence.	Separated stable therapist-aligned vocabulary from caregiver-created custom cards.
5. Model training and TensorFlow Lite export	Curated public datasets, trained models, exported TFLite assets and checked tensor contracts.	Chapter 6 learning curves, confusion matrices and TFLite verification figures.	Prioritised deployable model files over theoretical

Sprint / phase	Main work completed	Evidence or output	Outcome or change made
			architecture complexity.
6. Camera/FER integration, fallback model and confidence gating	Integrated camera workflow, ML Kit face gate, EfficientNetB0 primary model, MobileNetV2 fallback and uncertainty handling.	Chapter 6 implementation evidence and Chapter 7 model verification tests.	CBAM was not deployed because it created mobile TFLite compatibility problems.
7. Unit, functional, real-device and release testing	Ran Flutter tests, scripted UI walkthroughs, real-device checks, APK verification and Play internal testing.	Chapter 7 testing tables, Appendix H and Play Console evidence.	Testing supported release-readiness evidence but did not become clinical validation.
8. Final documentation, evaluation and submission preparation	Prepared final thesis, appendices, evidence checks and evaluation against objectives.	Chapters 7 and 8, appendices and submission evidence files.	Documented technical success while keeping pilot, production and clinical claims within

Sprint / phase	Main work completed	Evidence or output	Outcome or change made
			evidence boundaries.

3.2 Initial project plan

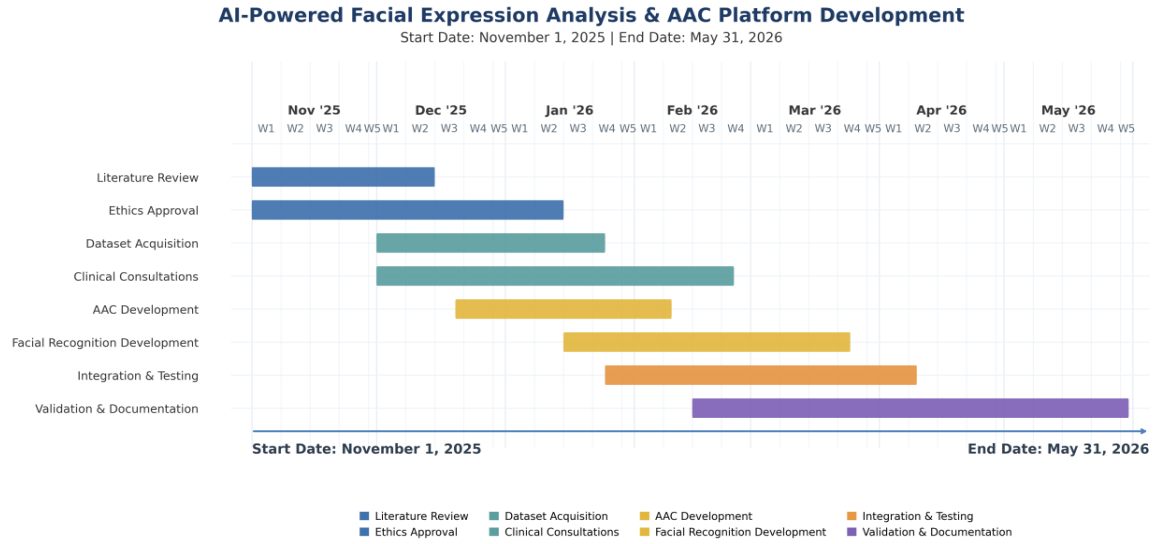


Figure 3.1: Initial project Gantt chart.

This original schedule from the interim report is included as project-planning evidence.

The initial plan (carried over from interim Figure 18) covered seven phases:

1. **Background research and proposal.** Literature review, AAC market scan, identification of the trilingual + emotion-aware gap.
2. **Design.** System architecture, ER model, use case view, functional and non-functional requirements.
3. **Core AAC build.** Splash, registration, home, category, favourites and settings screens in Flutter.
4. **Trilingual content and TTS.** Sinhala / Tamil / English vocabulary, language switching, TTS plumbing.
5. **AI model.** Dataset preparation, model training, TFLite export.
6. **Integration and testing.** Camera screen, model load, real-device testing, internal Play Store distribution.

7. **Documentation.** Interim report, final thesis, appendices.

The interim report mapped these phases onto specific calendar weeks. That schedule was met in broad shape, but several individual items slipped or were replaced.

3.3 Problems faced and changes to the plan

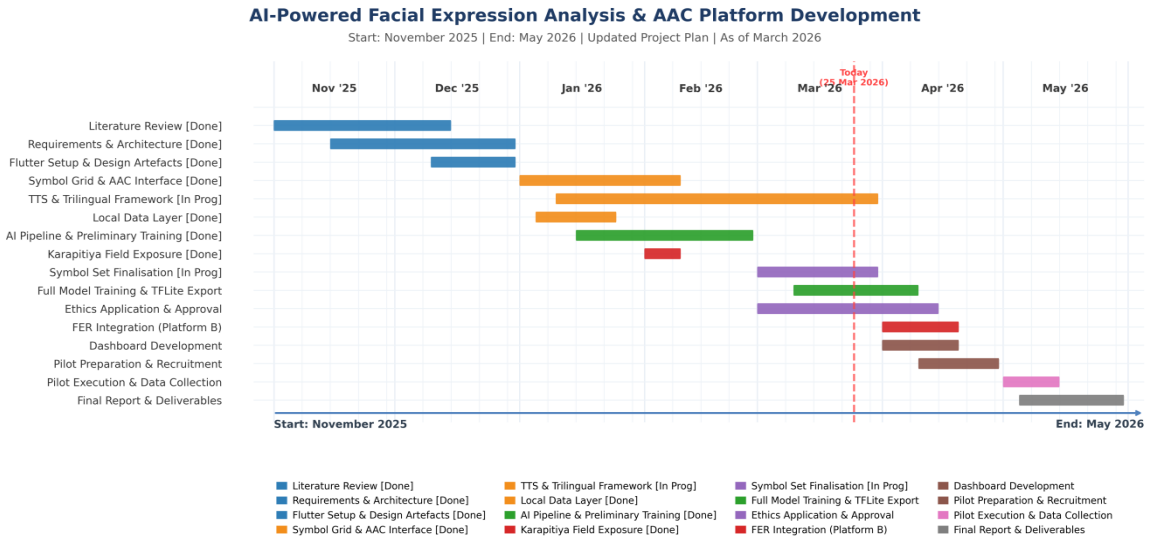


Figure 3.2: Updated project Gantt chart.

This revised schedule reflects actual progress and slippage around model deployment.

The plan was changed in five areas. Each change is documented below, detailing the rationale and the outcome.

3.3.1 Two-app idea → one application with feature-level differentiation

The interim report described a *Platform A* (symbol-based AAC for ASD Level 1-2) and a separate *Platform B* (AI-enhanced AAC for Level 3+). Field exposure and early caregiver feedback showed that maintaining and distributing two separate APKs was unhelpful: caregivers preferred to install one application and rely on the camera feature being available when the child was ready for it. The design therefore consolidated into one Flutter application with feature-level differentiation, which is what is described in Chapter 5 and Chapter 6.

3.3.2 EfficientNetB0 + CBAM → EfficientNetB0 without CBAM + MobileNetV2 fallback

The interim plan named EfficientNetB0 + CBAM as the final model with a target accuracy of at least 80 per cent. After several training runs, the CBAM export to TFLite required SELECT_TF_OPS / Flex operations that tflite_flutter could not load reliably on Android. The author switched to a mobile-safe pair: EfficientNetB0 without CBAM as primary, MobileNetV2 as fallback. The interim 80 per cent accuracy target is no longer claimed as

achieved. Final model evidence is instead reported through saved evaluation figures, confusion matrices and TensorFlow Lite verification outputs in Chapter 6 and Chapter 7.

3.3.3 SQLite + JSON vocabulary store → SharedPreferences + compiled Dart vocabulary

The interim ER diagram had a small SQLite schema for users, categories, symbols, sessions and logs. In the final implementation, the vocabulary is fixed at build time and is therefore compiled directly into Dart code in lib/data/word_data/. User state is stored in SharedPreferences via lib/services/storage_service.dart. This simplifies the application and produces a smaller, more deterministic APK. A local SQLite store would only be reintroduced if a caregiver dashboard with per-child usage logging is added in a later phase.

3.3.4 Ethical clearance and pilot pathway slower than ideal

The interim plan included a pilot study at Karapitiya Teaching Hospital within the project timeline. The realistic path moved to a collaboration with Pragathi Centre / National Hospital Galle, with the pilot deliberately scoped after ethical clearance. The pilot is now described as intended rather than completed. The hospital letter (Appendix A) describes the collaboration carefully and the report does not overclaim approval.

3.3.5 Play Store internal testing added

The interim plan focused on getting a working APK. The final phase added Google Play internal testing, which gave the author a repeatable installation channel for review devices and clinical contacts without making the application generally available. The signed release APK (build/app/outputs/flutter-apk/app-release.apk, 126.2 MB, application ID lk.aac.sinhala_tamil_english, versionName 1.1.0, versionCode 2) is present in the project tree. The interim report (Fernando, 2026, Section 3.10) records the corresponding AAB upload to the Google Play Console under the Internal Testing track. A later production access request was submitted through Play Console and is under Google review, but public production release is not claimed.

3.3.6 Personal cards feature scope-managed

A personal cards feature (caregiver photo capture and voice recording) was prototyped during the interim phase. It was de-scoped from the main Smart AAC application to keep the implementation surface small and reduce risk for the deployed APK. The customisation requirement that this feature was originally meant to satisfy is now covered by the **simple_aac_app** Platform A application, which provides caregiver-customisable AAC cards (label, colour, sub-cards), trilingual labels and manual backup, restore and sharing via ZIP archives. Personal cards inside the main Smart AAC application are documented as future work rather than claimed as a delivered feature.

3.3.7 Therapist/parent dashboard objective revised

The interim plan included a cloud-based therapist and parent dashboard for collaboration, progress monitoring and vocabulary management. During the final implementation stage, this objective was revised. The dashboard was not built as a cloud application, for several practical reasons:

The original dashboard idea was not only about vocabulary administration. The author also considered a Firebase-style route where selected clinical users or supervised medical students could review chosen emotion observations, record whether a prediction looked useful, confusing or incorrect, and later support a Sri Lankan-specific dataset for model improvement. That idea remains useful, but it was outside the approved scope of the submitted build.

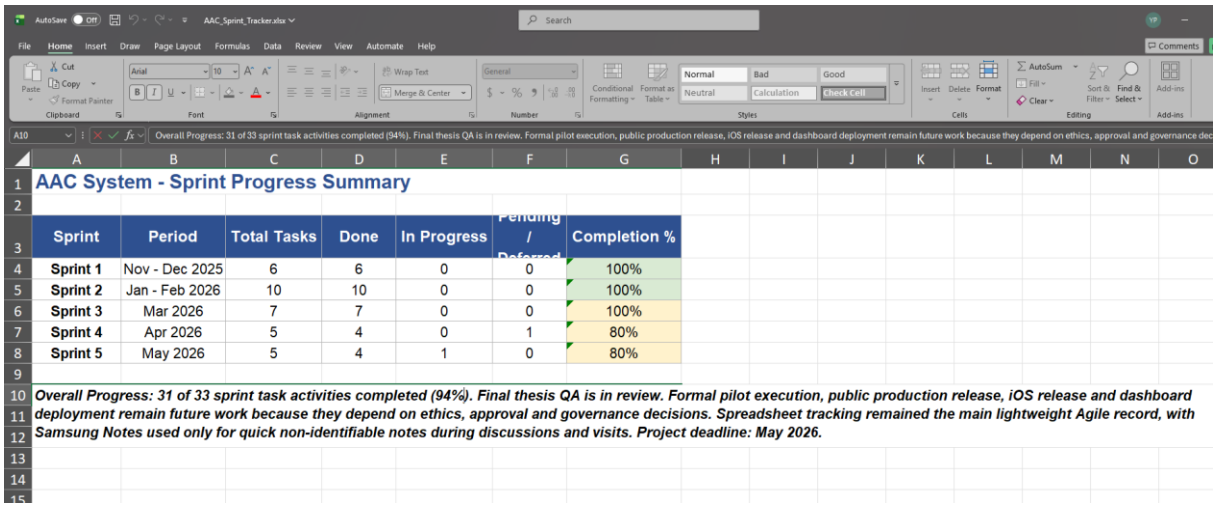
1. Many doctors, therapists and caregivers do not have laptop access during real sessions in Sri Lankan clinical and home settings.
2. The application targets mobile-first use on the kinds of phones families already own. A dashboard that lives outside the phone adds friction rather than reducing it.
3. The application includes a face-camera-based supportive observation feature. Uploading child face frames or even derived metadata to a cloud dashboard creates serious data-protection and parental-consent concerns that the project deliberately avoided.
4. Caregivers and speech therapists in Sri Lanka are already familiar with WhatsApp and file-sharing workflows. A backup ZIP that they can share through a tool they already use is more realistic than a portal they have to log into.
5. Manual backup, restore and share features make the user remain in control of what data leaves the device, which fits the offline-first design.

Firebase or similar cloud storage would also create ongoing cost and maintenance work. No sponsor, hosting budget or long-term funding source had been confirmed. More importantly, storing or transmitting child face-related data would require formal hospital, ethics and health-sector approval, which was not available within the final project timeline. For that reason, the dashboard was deferred to a future approved phase rather than treated as a failed feature.

The dashboard objective was therefore replaced with manual backup, restore and sharing inside the AAC application (implemented in the simple_aac_app, see Chapter 6, Section 6.3.7), together with the local AAC content and the on-device emotion observation that already live in the main application. This is described as a justified scope change, not as a failure: the underlying user need (collaboration and vocabulary management between caregiver, therapist and family) is still addressed, but through a path that is safer and more realistic for the Sri Lankan context.

This was an evolution of the interim plan, not a new idea added only at final submission. The interim report had already identified Firebase synchronisation, a therapist-parent dashboard, structured feedback collection and future purpose-collected facial-expression data as planned extensions (Fernando, 2026). These ideas were not removed because they lacked value. They were deferred because cloud storage, child face-related data, dashboard access and clinical observation workflows require funding, governance, ethics approval and hospital or health-sector authorisation. The submitted build therefore prioritised offline-first operation, local storage and on-device TensorFlow Lite inference, while keeping the dashboard and Sri Lankan-specific dataset pathway as future work.

3.4 Final project record



AAC System - Sprint Progress Summary

Sprint	Period	Total Tasks	Done	In Progress	Pending / Deferred	Completion %
Sprint 1	Nov - Dec 2025	6	6	0	0	100%
Sprint 2	Jan - Feb 2026	10	10	0	0	100%
Sprint 3	Mar 2026	7	7	0	0	100%
Sprint 4	Apr 2026	5	4	0	1	80%
Sprint 5	May 2026	5	4	1	0	80%

Overall Progress: 31 of 33 sprint task activities completed (94%). Final thesis QA is in review. Formal pilot execution, public production release, iOS release and dashboard deployment remain future work because they depend on ethics, approval and governance decisions. Spreadsheet tracking remained the main lightweight Agile record, with Samsung Notes used only for quick non-identifiable notes during discussions and visits. Project deadline: May 2026.

Figure 3.3: Sprint tracking spreadsheet.

The Google Sheets sprint tracker was used to monitor weekly progress and supervisor feedback.

Sprint 3 (Mar 2026) - Completed

#	Task	Status
1	Finalise symbol set and resolve licensing	Done
2	Complete Sinhala and Tamil vocabulary	Done
3	Evaluate device TTS behaviour and supportive audio constraints for Sinhala/Tamil	Done
4	Curate public FER datasets and document dataset limitations	Done
5	Full model training with hyperparameter tuning	Done
6	Complete TFLite export attempts, tensor validation and deployment checks	Done
7	Prepare ethics application and document pilot approval pathway	Done

Sprint 4 (Apr 2026) - Completed

#	Task	Status
1	Integrate FER pipeline into Flutter application	Done
2	Implement supportive emotion states, confidence/margin checks and caregiver-supervised camera workflow	Done
3	Defer therapist-parent dashboard and retain manual backup/restore/share workflow	Done
4	Document ethics follow-up, pilot boundary and health-sector approval dependency	Done
5	Prepare pilot protocol pathway; recruitment deferred until ethics clearance	Deferred

Sprint 5 (May 2026) - Finalisation

#	Task	Status
1	Document pilot deferral and internal-testing evidence boundary	Done
2	Collect APK, Play Console, real-device and informal feedback evidence	Done
3	Evaluate technical tests, informal feedback and limitations; no pilot findings claimed	Done
4	Write final thesis, appendices and final QA deliverables	In Review
5	Prepare final presentation and viva support material	Done

Figure 3.3A: Sprint backlog progress summary.

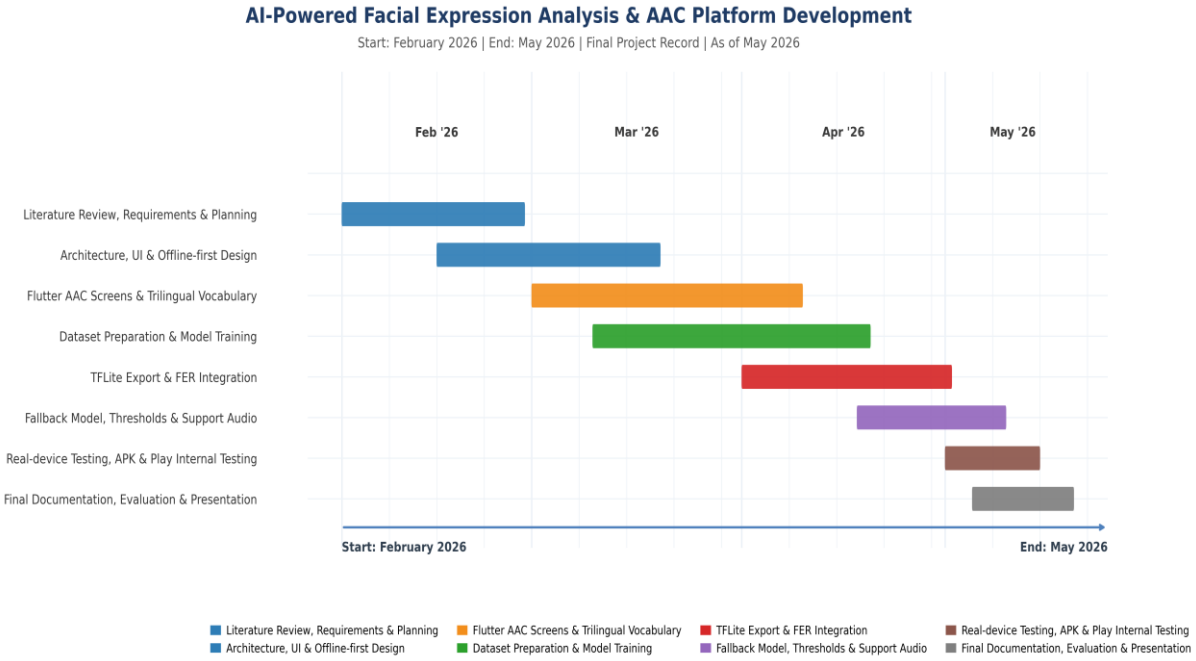


Figure 3.4: Final project Gantt chart.

Figure 3.4 summarises the final project timeline after the main scope and implementation changes. It shows how the work moved from initial planning and design into Flutter development, model training, TensorFlow Lite integration, testing, release preparation and final documentation.

Figure 3.1 records the initial schedule, Figure 3.2 records the revised interim schedule, and Figure 3.4 records the final project timeline after the main scope and implementation changes. Together, these figures show how the project evolved from the original plan into the final delivered system.

The final project record below summarises the major delivery items, mapped against the original plan. The table is condensed to keep the project-management section readable.

Table 3.1: Planned work versus actual work.

The table summarises planned work against what was delivered in the final project.

Phase	**Planned work**	**Actual work delivered**	**Status**
Background and proposal	Lit review, market scan, gap identification	Lit review, comparison of existing AAC tools, problem statement	Delivered
Design	Architecture, ER, use case, requirements	Architecture, use case view, simplified data model (SharedPreferences + compiled vocabulary), requirements	Delivered (data layer simplified)
Core AAC	Splash, registration, home, category, favourites, settings	All planned screens, plus stable emoji rendering and trilingual support	Delivered
AI model	EfficientNetB0 + CBAM, 80% accuracy target	EfficientNetB0 (no CBAM) + MobileNetV2 fallback, model evidence reported through saved evaluation figures and confusion matrices	Delivered with model swap. Saved evaluation evidence included
Camera screen	Face detection, on-device emotion inference, caregiver override	ML Kit face gate, burst capture, confidence + margin checks, primary/fallback model load, debug active-model view, caregiver-controlled support	Delivered
Supportive layer	Vocabulary prompts only	Trilingual WAV supportive prompts (assets/audio/support/{en,si,ta}) + sensory feedback service	Delivered
Real-device testing	Test on at least one Android device	Tested on real Android device(s), with informal caregiver feedback	Delivered (formal evidence in Chapter 7)
Distribution	Sideloaded APK	Google Play internal testing distribution is documented, and the	Delivered. Public

Phase	Planned work	Actual work delivered	Status
		signed release APK is present in build/app/outputs/flutter-apk/app-release.apk (126.2 MB, version 1.1.0+2, application ID lk.aac.sinhala_tamil_english).	production release is not claimed
Therapist-parent dashboard	Cloud dashboard for collaboration, progress monitoring and vocabulary management	Revised to an offline app with local storage and manual export, sharing and restore features inside the AAC application. The rationale is discussed in Section 3.3.7.	Re-scoped (delivered through a different path)
Customisable AAC for Level 1-2	Customisable cards inside one combined Flutter app	Delivered as a separate simple_aac_app (Flutter, customisable cards, sub-cards, trilingual labels, ZIP backup/restore, share to WhatsApp)	Delivered (separate app)
Clinical pathway	Pilot at Karapitiya	Pragathi Centre / National Hospital Galle collaboration, with a pilot intended only after ethics clearance.	Re-scoped (pilot pending)
Documentation	Interim report, final thesis	Interim report submitted, and final thesis prepared with appendices and a supporting evidence checklist.	In progress

3.5 Risks and mitigations

Risk tracking ran throughout the project. The matrix below records the main risks, their likelihood and impact (at the time of identification), and how they were handled in practice.

Table 3.2: Risk assessment and mitigation matrix.

ID	Risk	Likelihood	Impact	Mitigation taken	Final status
R1	TFLite cannot load the chosen model	Medium	High	Built two compatible models and verified [1, 224, 224, 3] float32 input and [1, 6] float32 output before integration.	Closed, primary/fallback pair shipped
R2	Final model accuracy below interim 80 per cent target	Medium	Medium	Defined success in terms of deployable compatibility, kept fallback model, and reported limitations transparently	Open. Standalone numeric accuracy values are not separately claimed, and evidence is provided through saved evaluation figures and confusion matrices.
R3	Colab session disconnection during fine-tuning	High	Medium	Timestamped checkpoints saved to Google Drive, separate filenames for fine-tuned checkpoints	Closed
R4	Ethical clearance slower than the project window	High	Medium	Pilot rescoped as intended after clearance, and the report uses careful wording throughout.	Open (deliberate)

ID	Risk	Likelihood	Impact	Mitigation taken	Final status
R5	Android build instability on Windows (cross-drive Pub cache)	Medium	Medium	kotlin.incremental=false added to android/gradle.properties	Closed
R6	Camera frames being saved or transmitted by accident	Low	High	Audited the camera screen and audio service. No raw image is written to disk or sent over the network, and only CAMERA, INTERNET and ACCESS_NETWORK_STATE permissions are declared.	Closed
R7	Emoji rendering flicker on the home grid	Medium	Low	EmojiText widget with NotoColorEmoji font, memoised emoji, addAutomaticKeepAlives: true	Closed
R8	Caregivers confused by too many sensory toggles	Medium	Low	Simplified settings down to one *Vibration On* switch	Closed
R9	Confidence thresholds left at debug values	Medium	Medium	Thresholds are marked clearly in code (_kMinConfidence, _kMinTopMargin) with implementation notes. Further tightening depends on real-device evidence.	Open (intentional)
R10	Some Sinhala/Tamil support audio	Low	Low	Fallback handling is implemented and the clips are	Open (content polish)

ID	Risk	Likelihood	Impact	Mitigation taken	Final status
	prompts may require pronunciation review or replacement			marked for final human review before wider release	

3.6 Tools used for project management

The tools used were intentionally simple:

- **Git**. Local version control on the upgrade-app and final-thesis branches.
- Google Sheets. Sprint backlog and burndown (Figure 3.3).
- **Google Drive**. Colab notebooks and exported model artefacts.
- **Google Colab**. Model training and TFLite export.
- **Visual Studio Code**. Code editor.
- **Android Studio**. Emulator and release build.
- Google Sheets was used as the main sprint tracking tool because it gave a lightweight and shareable way to update sprint backlogs, issues, milestones and progress during iterative development. Samsung Notes was used only for quick, informal and non-identifiable notes during field visits and stakeholder discussions. It was not used for clinical recording or for storing patient-identifiable information. This approach suited a single-developer undergraduate project where requirements and deployment decisions changed as the build matured.

3.7 Reflection on planning

Upon reflection, three primary lessons emerged from the project lifecycle.

First, **mobile-deployment constraints should be planned alongside model accuracy**, not afterwards. The CBAM detour cost the author about a sprint of work that would not have happened if TFLite compatibility had been treated as a hard requirement from the start of the model training phase.

Second, **simpler is usually better for caregivers**. Both the two-app split and the multi-toggle sensory card looked clean on paper, but they did not survive informal contact with real caregivers. The single application with one *Vibration On* toggle is easier to demonstrate, easier to install and easier to explain.

Third, documentation must remain objective and evidence-based. Technical limitations and pending validations are explicitly defined rather than obscured, ensuring the project's actual operational state is accurately represented.

The next chapter looks at the project from a feasibility-study angle, covering time, cost, scope, technical, economic, ethical and clinical feasibility.

Chapter 4: Feasibility Study

This chapter examines whether the proposed system was doable across the dimensions normally covered in a final-year feasibility study: time, cost, scope, technical, operational, economic, legal and ethical, data protection, mobile deployment, and clinical pilot. The aim is to provide an objective assessment of the project's practical viability. Several of the feasibility decisions taken here also explain the design and project-management choices in Chapters 3 and 5.

4.1 Time feasibility

The CS6P05ES module runs over a fixed academic window, with interim and final submission deadlines defined in advance by London Metropolitan University and ESOFT Metro Campus. Time feasibility was therefore the most binding constraint.

The project was deemed viable within this timeframe because the work was split between work the author controlled (Flutter code, Colab notebook) and work that depended on third parties (ethical clearance, clinical pilot). The plan deliberately did not make the academic deliverables depend on pilot results. The thesis evaluates the system against its objectives and against informal stakeholder feedback, while the pilot is described as intended after ethical clearance.

The largest single time risk in the year was the model training and TFLite export work. This assessment proved accurate, as the CBAM exploration and the EfficientNetB0 preprocessing adjustments consumed approximately two additional sprints (see Chapter 3, Section 3.3.2). The risk was contained because a fallback model (MobileNetV2) was kept ready in parallel.

4.2 Cost feasibility

The project is a single-developer academic project. The cost envelope is small and the resources used are mostly free or already available.

Table 4.1: Cost breakdown estimate.

The table records indicative cost categories considered during planning.

Cost category	Estimate / Source	Notes
Developer device for build / test	Already owned (Windows laptop, Android phone)	No new hardware purchased
Flutter / Dart SDK	Free, open-source	No licence cost
Android Studio	Free	No licence cost

Cost category	Estimate / Source	Notes
TensorFlow / Keras / TensorFlow Lite	Free, open-source	Free
Google Colab	Free tier (no Colab Pro)	Free. Session disconnect mitigation is noted in Chapter 6.
Google Drive	Free tier (15 GB)	Used for model checkpoints
Google Play developer account	US$25 one-time registration fee	Required for Google Play Console access, internal testing and production access request submission.
iOS App Store	Not confirmed in current submission	iOS pathway explored through Ideahub (Pvt) Ltd as future work. No App Store release or Apple developer account evidence is claimed in this thesis.
Kaggle datasets	Free, public	Cited in Chapter 6
Support audio prompts	Google AI Studio TTS was used to prepare short supportive voice prompts in Sinhala, Tamil and English. Pronunciation and tone still require human review before wider release.	Stored under assets/audio/support/{en,si,ta}/ and played by the supportive audio service when the caregiver activates emotion support.

Conclusion: the project is **economically feasible** at single-developer scale. No commercial licence is required for any of the technology stack chosen.

4.3 Scope feasibility

The project scope is narrower than the interim plan because two items were re-scoped:

- The two-app split (Platform A / Platform B) was consolidated into one Flutter application with feature-level differentiation.

- The personal cards feature (caregiver photo capture and voice recording) was de-scoped from the current final build.

The remaining scope, trilingual AAC + on-device FER + offline-first + supportive audio + internal Play Store distribution + clinical interest letter, is what was actually delivered. Each of these items is verifiable in the codebase or in the supporting documents.

Out-of-scope items recorded for clarity:

- No confirmed public Apple App Store release or TestFlight deployment.
- A completed clinical pilot study with measured outcomes.
- Ministry of Health final approval.
- A locally collected Sri Lankan child facial expression dataset.

The finalised scope was deemed fully feasible, being sufficiently constrained for the available timeframe while remaining robust enough to demonstrate the core concept (trilingual mobile AAC with on-device supportive FER).

4.4 Technical feasibility

Technical feasibility assesses whether the selected technologies can satisfy the system requirements. The decision points are:

- Flutter for the application. Flutter offers a mature widget toolkit, hot reload, a strong plugin ecosystem (camera, ML Kit face detection, TFLite, audio playback) and a single codebase that can later target iOS. The Flutter project compiles cleanly. The release APK has been built at build/app/outputs/flutter-apk/app-release.apk (approximately 126.2 MB).
- **TensorFlow Lite for on-device inference.** TFLite is the standard for mobile ML on Android. The author has confirmed at run-time that both deployed models satisfy the tensor contract ([1, 224, 224, 3] float32 input, [1, 6] float32 output, no Flex ops). The fallback strategy makes the application resilient when a single model fails on a particular device.
- **Google ML Kit face detection.** Used as a face *gate*, not for landmark or identity recognition. The gate runs entirely on-device with no network calls.
- **Local storage with shared_preferences. This is adequate for the small amount of user state held by the application, and no relational schema is needed.**
- **flutter_tts.** Uses the device's installed TTS engine. Where a Sinhala or Tamil voice is not installed, the engine returns silently rather than synthesising wrong audio, which is a safe behaviour.

- **audioplayers. Plays the optional WAV prompts and handles missing files gracefully without crashing the screen.**

The biggest technical risk was the model conversion to TFLite. It was mitigated through repeated verification, the fallback model strategy and the EfficientNetB0 preprocessing fix. The application has now run end-to-end on a real Android device, so technical feasibility is confirmed at the working-prototype level.

Figure 4.1 Feasibility Summary Diagram

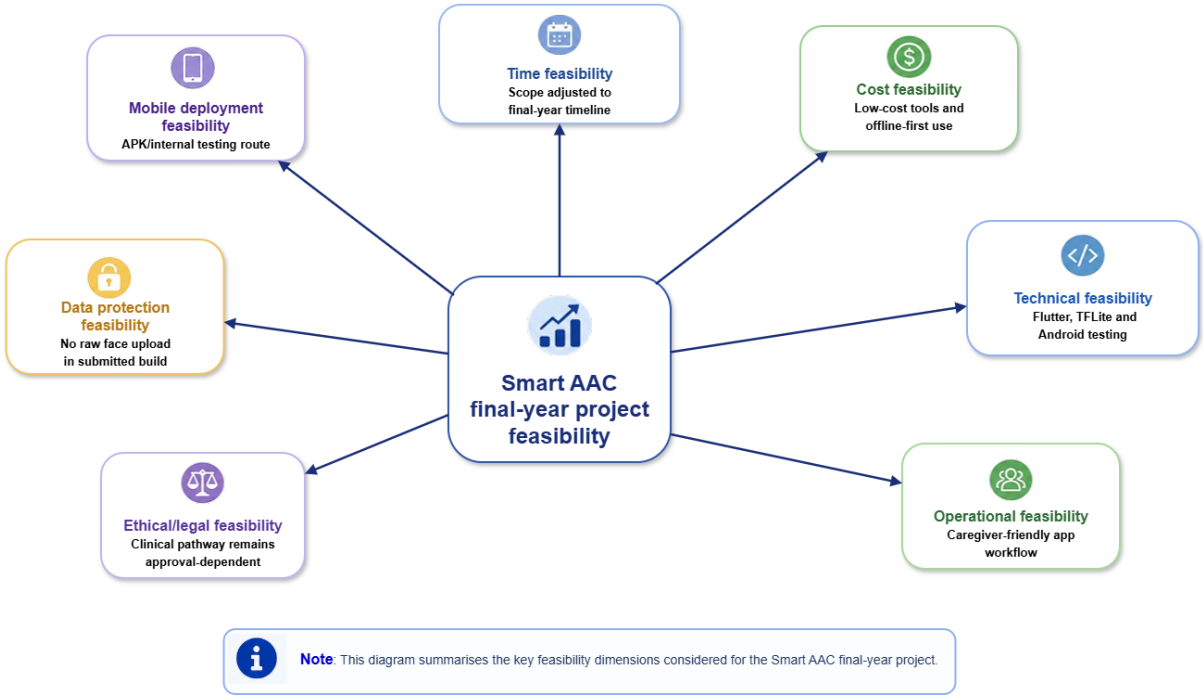


Figure 4.1: Feasibility summary diagram.

Editable diagram source: Download editable feasibility summary diagram

4.5 Operational feasibility

Operational feasibility evaluates how the application will be installed, used and supported by its actual users.

- **Installation.** The Android version is distributed through Google Play **internal testing**, which lets the author invite specific testers (clinical contacts, caregivers) without making the application public. Once invited, the tester installs and updates the application like any other Play Store app, with no sideloading or developer-mode steps required.

- **Caregiver supervision.** The application is designed with the assumption that a caregiver is present whenever the child uses the camera screen. The *Play support* button is intentionally manual, *Stop* is always visible during the support animation, and a caregiver-alert ribbon is shown for the Fear emotion only.
- Trilingual support. Three languages are exposed in the registration screen and the settings screen. The application falls back to English if a TTS voice or a supportive WAV file is missing in the active language. Support-audio pronunciation and language coverage still require final human review before wider release.
- **Offline-first.** Core AAC navigation, TTS and on-device FER inference all run without internet. The application reads connectivity only as an information signal through OfflineService / connectivity_plus. This is critical in Sri Lanka, where rural connectivity is uneven (International Telecommunication Union, 2022).
- **Accessibility. Large card targets, generous spacing, optional vibration, no flashing animations, and a single-toggle settings page reduce cognitive load for caregivers. A WCAG 2.1 AA audit has not been performed yet and is listed as planned future work.**

Operational feasibility is therefore judged **positive**, with WCAG formal audit listed as outstanding.

4.6 Economic feasibility

Economic feasibility examines the cost to the end-user and long-term sustainability.

- The application is intended to be free or low-cost for children with autism and their families. The registration screen states this in all three languages. The intent is to support Sri Lankan low-income families who cannot afford the type of commercial AAC systems that typically cost USD 100-USD 300 plus the price of a dedicated tablet.
- There is no in-app purchase, no advertising and no subscription. The is_premium flag in StorageService is set to true on registration purely to unlock all features for everyone, not as a paywall toggle.
- The technology stack is open source. Hosting cost is zero in the current build because the application is offline-first and there is no backend.
- Distribution through Google Play internal testing is bounded by the one-time developer account fee.
- Manual backup, restore and share features (in the simple_aac_app) remove the need for a paid cloud back-end while still letting families and therapists share vocabulary using tools they already use (WhatsApp, email, file share).

The application is therefore **economically feasible** as a free, social-impact-led academic project. Commercialisation is intentionally out of scope at this stage. The free / low-cost positioning is part of the project's ethical commitment: the system is offered to families who would otherwise have no AAC option in Sinhala or Tamil, and the project is sustainable precisely because it has avoided cloud and subscription costs.

4.7 Why a cloud dashboard is not feasible at this stage

The interim plan included a cloud-based therapist and parent dashboard. The final feasibility judgement is that a cloud dashboard is not feasible at this stage of the project. The reasoning is summarised below. The same reasoning is also recorded in Chapter 3, Section 3.3.7.

- **Data-protection feasibility.** The application includes a camera-based supportive observation feature. Uploading child face frames, derived emotion metadata, or session logs to a cloud dashboard would significantly raise the data-protection bar: parental consent forms, encryption-in-transit and at-rest, retention policy, cross-border data flow handling and a data-protection-impact assessment would all be required. None of these are realistic to complete within the project window without a clinically signed-off ethics framework.
- **Operational feasibility.** Many caregivers, speech therapists and doctors in Sri Lankan clinical and home settings do not have laptop access during real sessions. A web dashboard adds a device dependency that does not match the working environment of the target users.
- **Workflow feasibility.** Caregivers and therapists in Sri Lanka already share files through familiar tools (WhatsApp, email, USB transfer). A manual ZIP-based backup and share path inside the AAC application fits these workflows directly. The Platform A simple_aac_app implements exactly this path through Share.shareXFiles, downloads-folder export and pick-file restore.
- **Technical feasibility.** The mobile, offline-first approach is technically feasible with the Flutter stack already chosen. Adding a cloud backend would change the dependency surface significantly (authentication provider, hosted database, server runtime, ongoing cost) and introduce moving parts that the project does not need.

Cost and sustainability feasibility. A Firebase or cloud-style dashboard would need hosted storage, access management, monitoring and long-term maintenance. These costs may be small at prototype level, but they become important if the system is used with real children and clinical users. At the time of writing there was no confirmed sponsor or funding source to maintain that infrastructure responsibly.

The user need that the dashboard was meant to address (collaboration, progress monitoring and vocabulary management between caregiver, therapist and family) is therefore addressed in the

final system through **manual export, restore and share inside the AAC application** plus the local AAC content and the on-device supportive emotion observation. A cloud dashboard remains a feasible future direction once an ethics framework, a parental consent process and a tested data-protection policy are in place.

This feasibility judgement also reflects what the interim report had already shown. Firebase was recorded there as a future synchronisation option, and the therapist-parent dashboard was planned for collaboration and progress monitoring rather than as a completed final feature (Fernando, 2026). By final submission, the safer decision was to preserve the useful aim but deliver it through local storage and manual sharing until funding, governance and approval conditions are in place.

The same conclusion applies to the proposed emotion-observation feedback workflow. It could be valuable after approval, but it is not feasible in the current submission because it would involve child face-related data, reviewer judgements and possible future dataset development. The current build therefore keeps emotion support local to the phone and does not create a central observation record.

4.8 Legal and ethical feasibility

Legal and ethical feasibility is the most sensitive area because the application is intended for use with children.

Table 4.2: Ethical risk table.

The table records identified ethical risks and how they are managed in the current build.

Risk area	Description	Mitigation
Child safeguarding	The application is used by children, sometimes in clinical-adjacent settings	Caregiver supervision is required for camera use. The AI component is framed as supportive rather than diagnostic, and no raw images leave the device.
Parental consent	Children cannot meaningfully consent to data processing alone	The intended pilot will operate only within ethically cleared scope (see Pragathi Centre / National Hospital Galle letter, Appendix A)

Risk area	Description	Mitigation
Data minimisation	Sensitive image data must not be collected unnecessarily	Camera frames are not saved to disk or transmitted. Only the emotion label is shown.
Cross-border data flow	Cloud services can transmit data outside Sri Lanka	No cloud back-end is used in the current build
Misuse of AI output	A model output could be mistaken for a clinical assessment	The system explicitly shows *No face*, *Face not clear*, *Uncertain* and *Reliable* states, and is described throughout as a supportive observation tool, not a diagnostic tool
Clinical overclaim	Stakeholders may be told the application is approved when it is not	Health-sector approval and wider deployment remain future work, and the report uses careful wording throughout.
Future local face-data collection	A future student or partner might treat "dataset improvement" as an informal add-on	Any Sri Lankan child face-data study requires ethics approval, parental consent, secure custody and health-sector coordination (Section 4.9.1, Chapter 8, Section 8.7). No such collection is part of this thesis.

The author has chosen GDPR-style data minimisation as an internal standard, alongside Sri Lanka's Personal Data Protection Act, No. 9 of 2022 (Parliament of the Democratic Socialist Republic of Sri Lanka, 2022). The conservative choice is to assume the stricter regime applies and to design accordingly.

Legal and ethical feasibility is judged **positive at the prototype level**, with the explicit caveat that any pilot involving children will be carried out only within the ethically cleared scope described in the hospital letter.

4.9 Data protection feasibility

The data-protection design choices are concrete:

- **No raw camera frames are stored. The camera screen processes frames in memory and discards them after inference. This can be confirmed by reading CameraExpressionScreen's capture path.**
- **No cloud back-end. Firebase, REST APIs and remote storage are not used in the current build. The Android manifest declares only CAMERA, INTERNET and ACCESS_NETWORK_STATE. INTERNET is used by the connectivity check and by flutter_tts engine setup, not by any first-party network call from the author's code.**
- **Only essential user state is stored.** SharedPreferences holds the child's name, gender, language preference and a few toggles. No emotion history is currently stored or transmitted.
- **Bundled audio is local.** The supportive WAV files ship with the APK.
- **Future caregiver or clinical-review dashboards. If a dashboard is ever added, the design will require explicit consent, encrypted transport, strict access control and approval through the appropriate hospital, ethics and health-sector pathway. The default design should collect the minimum necessary observation or feedback data, and should not upload raw face images unless a formally approved protocol requires and protects it.**

The interim report also treated purpose-collected child facial-expression data and pilot evaluation as approval-dependent work. That boundary remains unchanged in the final thesis. Any later dashboard or data-collection workflow would need parental or guardian consent, ethics clearance, secure storage, defined access control and hospital or health-sector authorisation before any child face-related data or reviewer feedback is stored centrally.

Data protection feasibility is therefore judged **positive** for the current single-device application.

4.9.1 Feasibility of any future child facial-expression data collection

The current build does not collect training images from children and does not upload camera frames. Feasibility for a hypothetical future Sri Lankan child facial-expression dataset is judged conditional, not automatic. Such a project would move the work from "public Kaggle images processed on a student laptop" to prospective collection of biometric child data in a clinical-adjacent context. That step would require: a protocol approved by an ethics committee; parental or guardian consent and, where appropriate, consideration of child assent; secure storage, access control, retention limits and a plan for pseudonymisation or deletion where compatible with training needs; alignment with Sri Lankan data-protection law and institutional rules; and

coordination with the hospital and relevant health-sector authorities. Ministry or health-sector approval would be required before wider data collection or clinical deployment of any system trained on such material at scale. Until those safeguards exist on paper and in practice, feasibility for local data collection is negative from a student-project perspective, and the thesis correctly treats public datasets as the only ethical training source used so far.

During the final hospital-related discussion, the author was advised that any future data collection or dashboard-based observation workflow would need to be handled through the appropriate hospital and health-sector approval pathway, including Karapitiya Teaching Hospital where relevant. This is treated as future work rather than as approval already obtained.

4.10 Mobile deployment feasibility

Mobile deployment was treated as a hard requirement, not as an afterthought. Concrete decisions that flow from this:

- TFLite Flex / SELECT_TF_OPS operations were avoided in the deployed models, even where the notebook-level results were stronger. Both bundled models contain only TFLite built-in operations.
- Model sizes were kept compact: 4.7 MB primary, 2.6 MB fallback, 7.4 MB total. This keeps the APK manageable and reduces the risk of cold-start failures on low-end devices.
- A fallback model is always shipped, so a single model loading failure does not break the camera feature.
- The release APK was successfully built (build/app/outputs/flutter-apk/app-release.apk).
- The release signing config reads from key.properties (not committed to source control), which keeps signing material out of the repository.

Mobile deployment feasibility is therefore judged **positive**.

4.11 Clinical and pilot feasibility

Clinical feasibility depends on partners outside the author's control.

- The Head of Pragathi Centre / National Hospital Galle has issued a formal letter (see Appendix A) confirming collaboration on the *Pragathi AAC App* and stating that a pilot is intended after the necessary ethical clearance.
- The author has no control over the timing of ethical clearance.
- The pilot scope, if it goes ahead, will be limited to what Pragathi Centre and the relevant ethics committee approve.

- The application is described throughout the report as a **supportive AAC and observation tool**, not a diagnostic tool. The clinical interest expressed in the letter does not constitute health-sector approval.

Clinical feasibility is therefore judged positive with bounded scope. The project has completed the activities possible within an academic-project setting. The next step depends on external approval and governance.

4.12 Limitations identified by the feasibility study

The feasibility study identifies six limitations that are clearly documented and revisited in Chapter 8:

1. Dataset is not Sri Lankan. The deployed model is trained on public Kaggle datasets. A local dataset could improve fairness and local relevance, but only within the ethics and governance framework described in Section 4.9.1 and Chapter 8, Section 8.7. No such collection forms part of this project.
2. Model evaluation is reported using the saved learning curves, confusion matrices and TensorFlow Lite verification evidence included in the implementation and testing chapters.
3. **Confidence thresholds are still at debug values.** The code clearly marks these as needing tightening with real-device evidence.
4. **iOS release is unconfirmed.** The iOS pathway is being explored with Ideahub (Pvt) Ltd, but there is no confirmed App Store release.
5. **No completed pilot. A pilot is intended after ethical clearance, and no results are claimed from it.**
6. **Wider health-sector authorisation for clinical-scale use or for child face-data collection is not in place.** The prototype must not be mistaken for an approved clinical product. The next chapter takes the feasibility judgements made here and turns them into a concrete design.

Chapter 5: Design

This chapter describes how the Smart AAC system is designed at a software level. It moves from the overall architecture to the use case view, the data model, the emotion detection pipeline, the trilingual content design, the user interface design, and finally the consolidated requirements (functional and non-functional). The design is presented at a level of detail suitable for the final report. Specific code references and implementation details are in Chapter 6.

This chapter objectively reflects the actual implemented system, detailing architectural evolutions from the interim design. Specifically, it documents the **re-packaging of the initial dual-platform concept into two complementary Flutter applications,** the removal of the CBAM attention module for mobile compatibility, and the transition from an SQLite vocabulary plan to **SharedPreferences with compiled Dart vocabulary.**

5.1 Design overview

The system is delivered as a pair of complementary Flutter Android applications that share the same design principles. The **Smart AAC application** (the main repository in this project) targets the Level 3+ scenario and adds the supportive on-device facial expression recognition layer. In clinical correspondence it is referred to as the **Pragathi AAC App**. A second, smaller **simple_aac_app** application targets the Level 1-2 scenario with a caregiver-customisable card-based AAC interface and manual backup, restore and sharing. Both applications target caregiver-supervised use by children with autism spectrum disorder and other communication difficulties.

Six design ideas drive the entire system.

1. **Mobile-first, not dashboard-first. Every user-facing feature lives in the mobile application. There is no cloud dashboard. The reasons are recorded in Chapter 3, Section 3.3.7 and Chapter 4, Section 4.7: doctors, therapists and caregivers in Sri Lankan settings typically do not have laptops on hand during sessions. Sensitive child and face-related data should not be centralised. The workflows the project's users already use (WhatsApp, file sharing) suit a manual export and share path inside the app.**
2. **Offline-first.** Every core feature, vocabulary navigation, text-to-speech, camera-based emotion observation and the supportive audio prompts, works without internet access. Online detection through the connectivity_plus package is used only as an information signal.
3. **Trilingual by default.** Sinhala, Tamil and English are first-class languages. They share one symbol grid, but labels, TTS output and supportive audio clips are language-

specific. The communication content and UI direction are designed with all three languages in mind.

4. **Two communication profiles, one design language. Level 1-2 children benefit from customisable, caregiver-created vocabulary. The simple_aac_app provides this. Level 3+ children, who may have very limited functional speech, benefit from a simpler, more stable static AAC flow with therapist-guided categories. The main Smart AAC application provides this static flow and adds the supportive on-device emotion observation layer. Neither profile is treated as more important than the other. The technical research contribution focuses on the Level 3+ scenario because that is where on-device emotion-aware AAC support adds the most value.**
5. **Supportive, not diagnostic AI. The on-device facial expression recognition layer is positioned as a supportive observation aid, never as a diagnostic statement. The system explicitly returns No face detected, Face not clear and Uncertain states, rather than forcing a label on every frame.**
6. **Mobile-safe ML.** Two compacts .tflite models are bundled in the main application: a fine-tuned EfficientNetB0 as the primary model and a MobileNetV2 as a fallback. The application picks the first model that loads and passes tensor validation. A debug view allows the active model to be selected manually for testing.

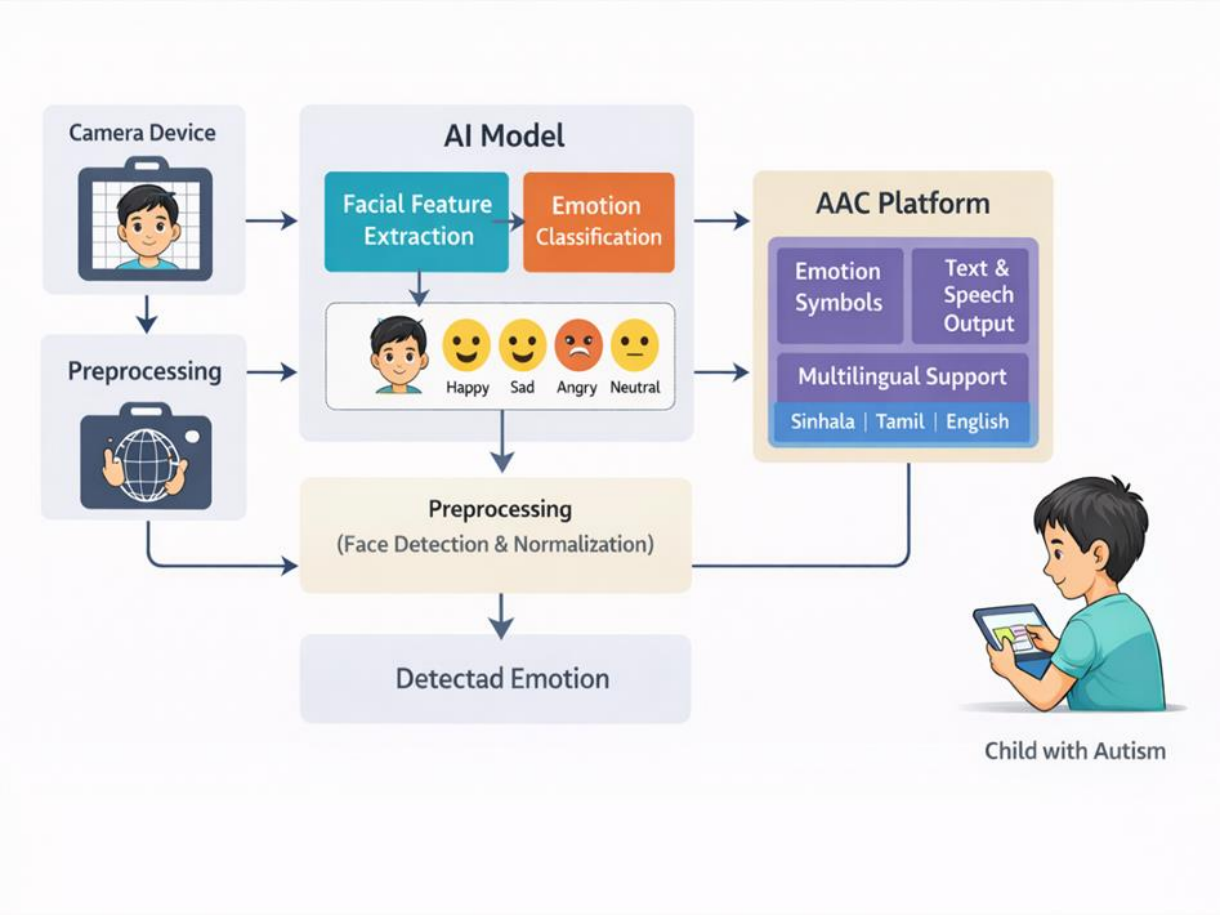


Figure 5.1: System architecture diagram.

Editable diagram source: Download editable system architecture diagram

5.2 System architecture

The architecture follows a simple layered pattern, kept deliberately light because the application has no backend.

- **Presentation layer.** Flutter widgets in lib/screens/ and lib/screens/theme/. The main screens are SplashScreen, RegistrationScreen, HomeScreen, CategoryScreen, FavouriteScreen, CameraExpressionScreen, SettingsScreen and DonationScreen.
- **Services layer.** Cross-cutting concerns in lib/services/: StorageService for SharedPreferences-backed state, OfflineService for connectivity_plus, SensoryFeedbackService for vibration and per-emotion support plans, and EmotionSupportAudioService for the optional WAV prompts in three languages.
- **Data layer. Hard-coded trilingual vocabulary in lib/data/word_data/ is used, with 16 categories and entries containing si, ta and en text plus an emoji and example action sentences. The combined map is exposed by lib/data/word_data.dart. There is no SQLite database in the current build. The interim report's SQLite plan was replaced with a simpler hard-coded vocabulary because it produced a smaller, more deterministic APK.**
- **ML assets layer. Two .tflite models and a labels.txt file are stored in assets/models/. The primary model is loaded by tflite_flutter at start-up, and the fallback model is loaded automatically if the primary model fails.**
- **Optional audio assets.** Trilingual WAV clips in assets/audio/support/{en,si,ta}/.

The application has no Firebase backend in the current build. Firebase remains a future option for caregiver dashboards and synchronisation, but is intentionally out of scope here.

Figure 5.1 presents the system architecture diagram used to describe the main Smart AAC design. It highlights the mobile AAC application, on-device facial expression recognition, preprocessing flow, emotion-support output and multilingual AAC platform. The surrounding text explains how the final submitted build differs from earlier planned cloud or dashboard ideas.

5.3 Use case view

The actors and primary use cases are summarised in the use case diagram below, together with the use case summary table.

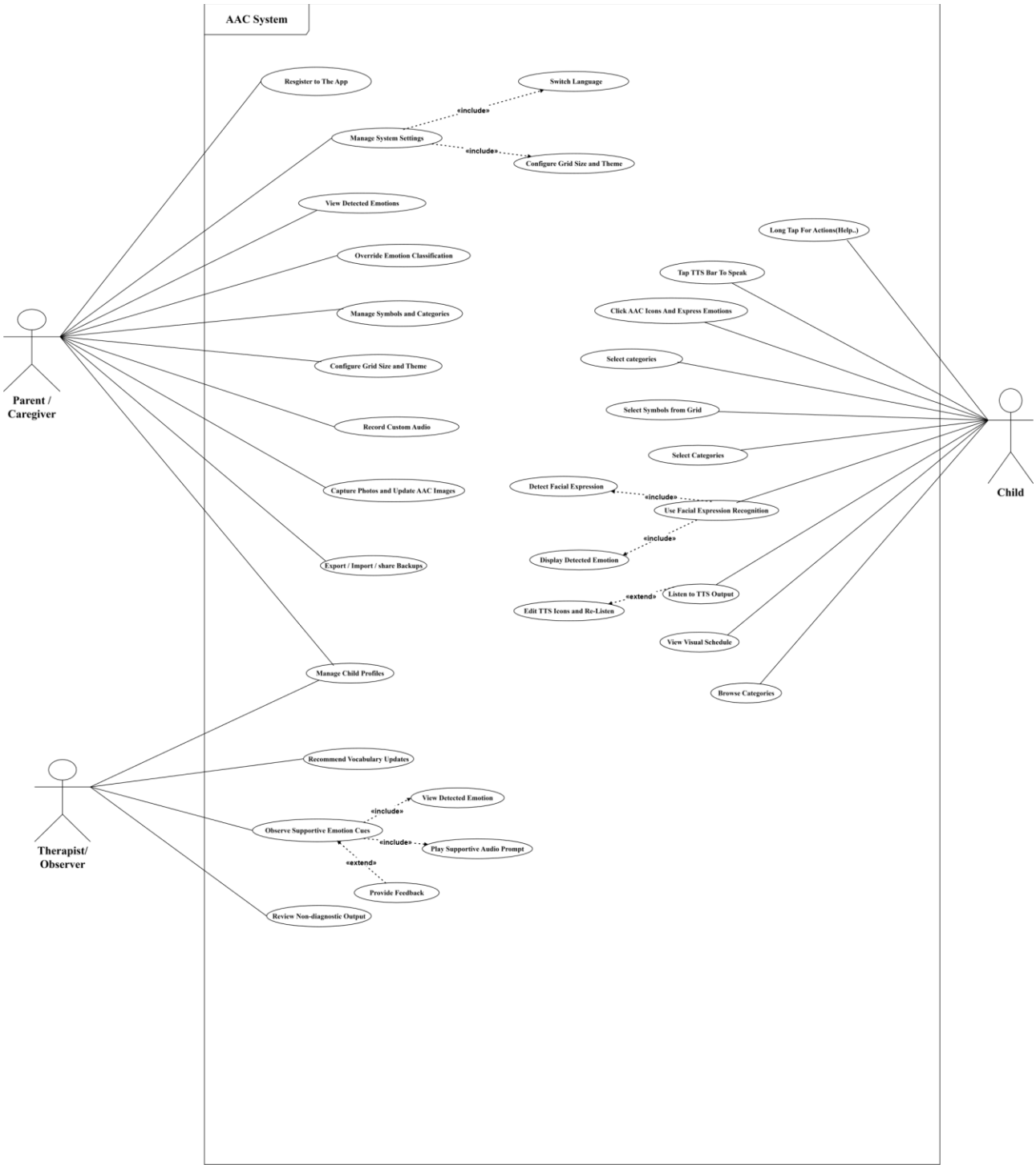


Figure 5.2: Use case diagram.

Editable diagram source: Download editable Use case diagram

Table 5.1: Use case summary.

Actor	**Primary use cases**	**Pre-condition**	**Post-condition / output**
Child	Select AAC categories and cards, listen to TTS output, use favourites with caregiver support, and view the camera screen only under supervision.	Application is open and the caregiver has selected or confirmed language and profile settings.	Selected word or phrase is shown and can be spoken through TTS.
Caregiver	Register a child profile, select language, manage favourites, supervise camera/emotion support, and decide whether supportive audio should be used.	Caregiver has access to the Android device and required permissions.	AAC settings are saved locally and the child receives supervised communication support.
Therapist / observer	Observe AAC interaction, review supportive emotion cues, provide informal feedback, and support future pilot planning.	System is demonstrated in a supervised or academic testing context.	Observation and feedback are used as informal evaluation evidence, not clinical validation.
Tester / developer-researcher	Verify the APK build, check model loading, inspect the active model/debug screen, test the primary/fallback	Testing build and project evidence are available.	Release, model and real-device verification evidence is recorded.

	model, and record test evidence.		

Figure 5.2 focuses on end-user interaction with the deployed AAC system. The tester/developer-researcher role is included in Table 5.1 only to document verification activities used during implementation and testing.

5.4 Data model

This section presents the data design from two complementary views. Figure 5.3 gives a simplified local/offline data model that is easier to read as a high-level implementation summary. Figure 5.4 then provides an ER-style logical view of the same data concerns, showing the main entities, attributes and relationships used to reason about the application design. The two diagrams are therefore not separate database implementations; they are used together to explain how the submitted mobile application organises profile data, settings, vocabulary, favourites, emotion-support results and bundled support assets.

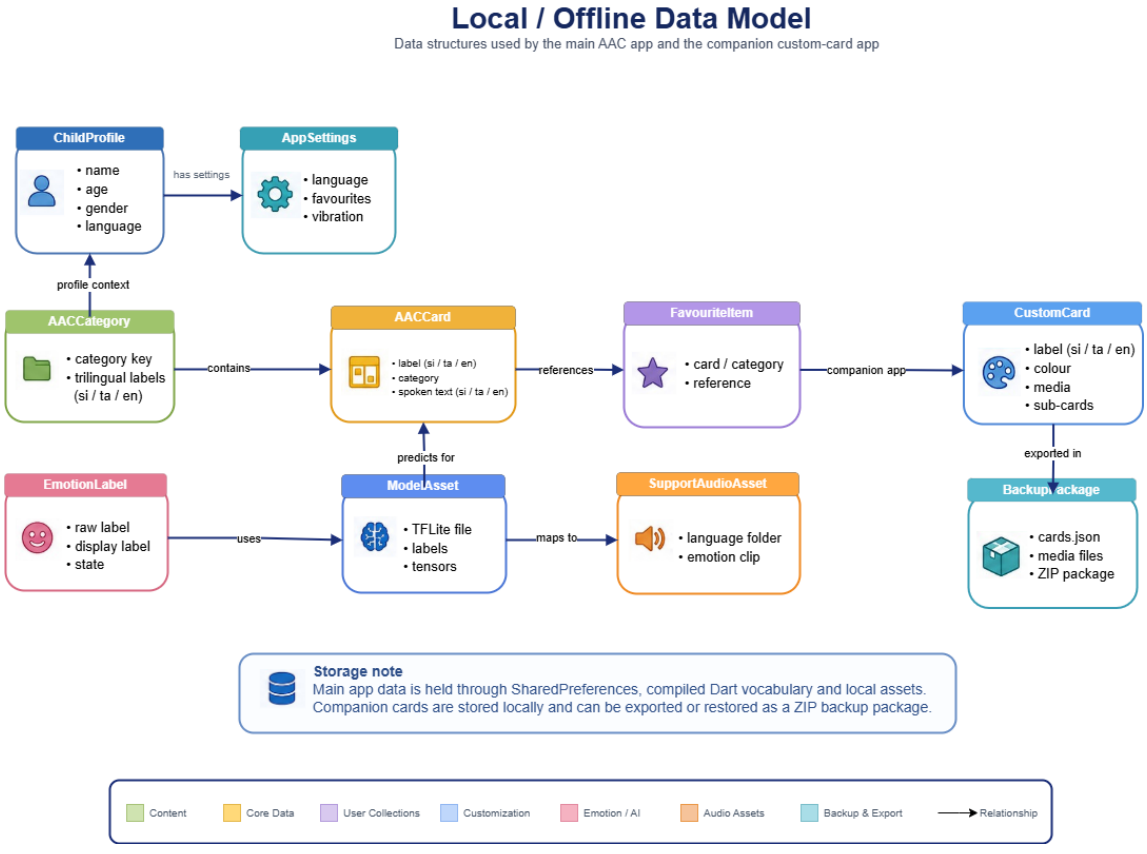


Figure 5.3: Data model diagram.

Editable diagram source: Download editable Data model diagram

Figure 5.3 summarises the final data model used by the Smart AAC application. It shows the main local data areas that support user profile storage, settings, vocabulary access, favourites and emotion-support behaviour.

To make the relationships between these data areas clearer, Figure 5.4 presents the same design from an ER perspective.

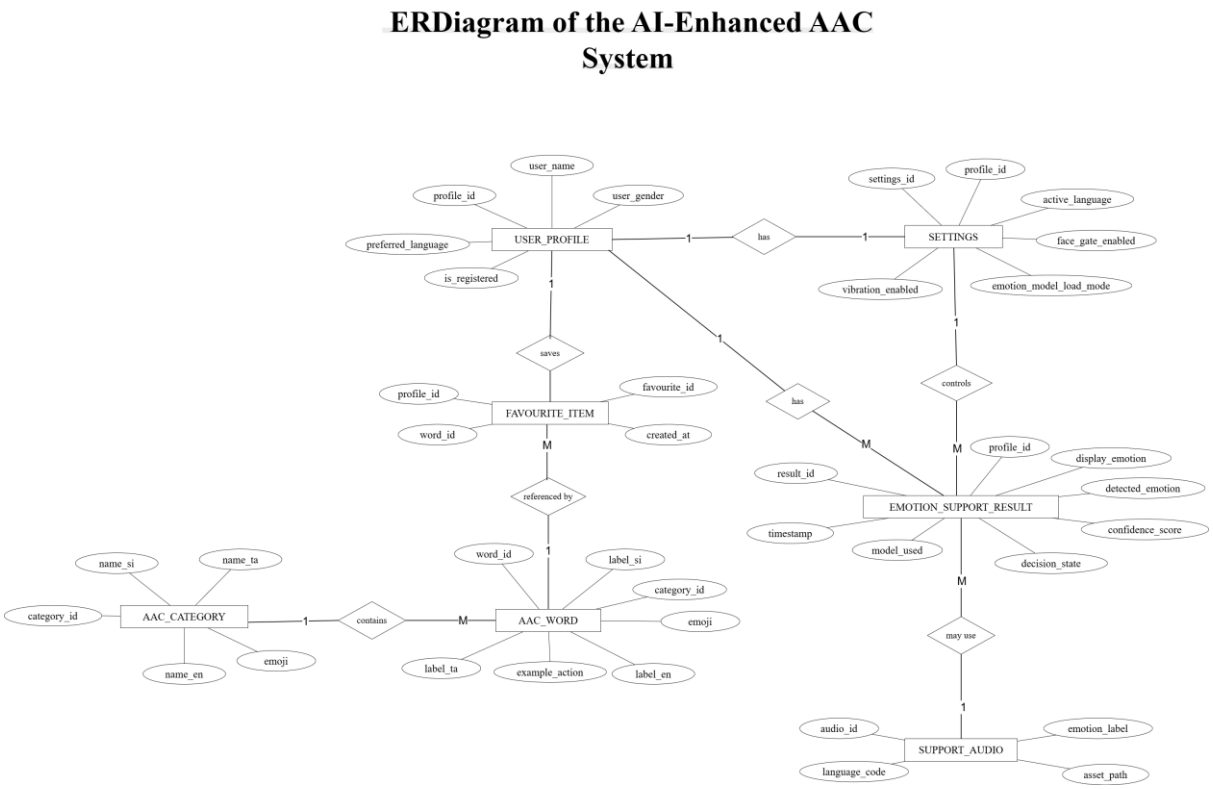


Figure 5.4: ER diagram of the Smart AAC application.

Editable diagram source: Download editable ER diagram

5.4.1 ER diagram interpretation

The ER diagram is used here as a logical design explanation rather than as a physical database schema. It helps to show how the main data concepts of the Smart AAC system relate to each other before those concepts are implemented in a lightweight mobile form.

The central user profile entity represents the child or caregiver profile information that controls language, accessibility and personalisation behaviour. The settings entity stores

preferences such as active language, vibration feedback and face-gate/model-loading options. AAC categories organise the fixed communication vocabulary, and AAC words represent the individual communication items with Sinhala, Tamil and English labels. Favourite items connect a profile to selected AAC words so that frequently used cards can be accessed more quickly. The emotion-support result entity represents the logical output of the on-device observation layer, while the support-audio asset entity represents the bundled audio prompts that may be used as caregiver-controlled support.

The relationships clarify the intended structure of the data. A user profile has one settings record, a category contains many AAC words, and favourite items reference selected AAC words for a specific profile. The emotion-support result is linked to the user profile and may use a support-audio asset where a suitable prompt is available. These relationships explain the design logic of the application, but they should not be read as proof that the final build uses a relational SQL database. In the submitted application, lightweight user state is stored through SharedPreferences, while the fixed vocabulary is held as compiled Dart data and bundled assets. This decision keeps the mobile implementation simpler and more suitable for offline use.

Table 5.5: ERD design interpretation.

ERD element	**Design meaning in this project**
User profile / child profile	Represents registration, user name, gender and access state handled by StorageService and saved in SharedPreferences.
Settings	Represents selected language, vibration preference, favourite category choices, model-load mode and face-gate option used by the app screens and services.
AAC category	Groups the fixed communication vocabulary into areas such as food, feelings, actions,

	family, places and needs using compiled Dart word-data files.
AAC word / card	Represents each communication card with Sinhala, Tamil and English labels, emoji and example action text from the compiled vocabulary maps.
Favourite item	Represents a saved favourite vocabulary selection. In the final app this is mainly stored as favourite category keys for quicker access.
Emotion-support result	Represents the runtime output of the camera emotion-support flow, including display emotion, confidence decision state and active model, without storing clinical records.
Support-audio asset	Represents bundled local WAV prompts under assets/audio/support, resolved by language with English fallback where needed.
Model and label assets	Represents bundled TensorFlow Lite models and labels.txt used by the on-device emotion-support pipeline.

This interpretation shows that the ERD is used to explain the logical design of the data relationships, while the implementation remains lightweight through SharedPreferences and compiled Dart vocabulary.

The deployed application uses **SharedPreferences** as a key-value store for user state. The keys actually persisted by StorageService (lib/services/storage_service.dart) and the other services are:

- is_registered (bool)
- is_premium (bool, set to true on first registration so all features are accessible)
- user_name (string)
- user_gender (string: boy / girl, used to switch between two child-friendly colour themes)
- vibration_enabled (bool, default true), used by SensoryFeedbackService
- emotion_model_load_mode (int enum: auto / primaryOnly / fallbackOnly)
- emotion_face_gate_enabled (bool, default true)

Vocabulary data is read directly from compiled Dart maps in lib/data/word_data/, not from a database. The map has 16 categories: body_parts, animals, fruits_vegetables, objects, colors_numbers, feelings, actions, sounds_music, food, household, colors, numbers, needs, sentences, family_words and places. Each entry contains Sinhala, Tamil and English text, an emoji glyph and example action sentences.

This change from the interim report's ER-modelled SQLite plan was intentional. SharedPreferences keeps the storage requirements minimal, fits the offline-first goal, and avoids the maintenance overhead of a small database when the vocabulary itself is fixed at build time. If a future caregiver dashboard or per-child usage log is added, a local SQLite database would be reintroduced for that purpose.

5.5 Emotion detection pipeline

The emotion detection pipeline is the most novel part of the design and is described in greater detail in Chapter 6.

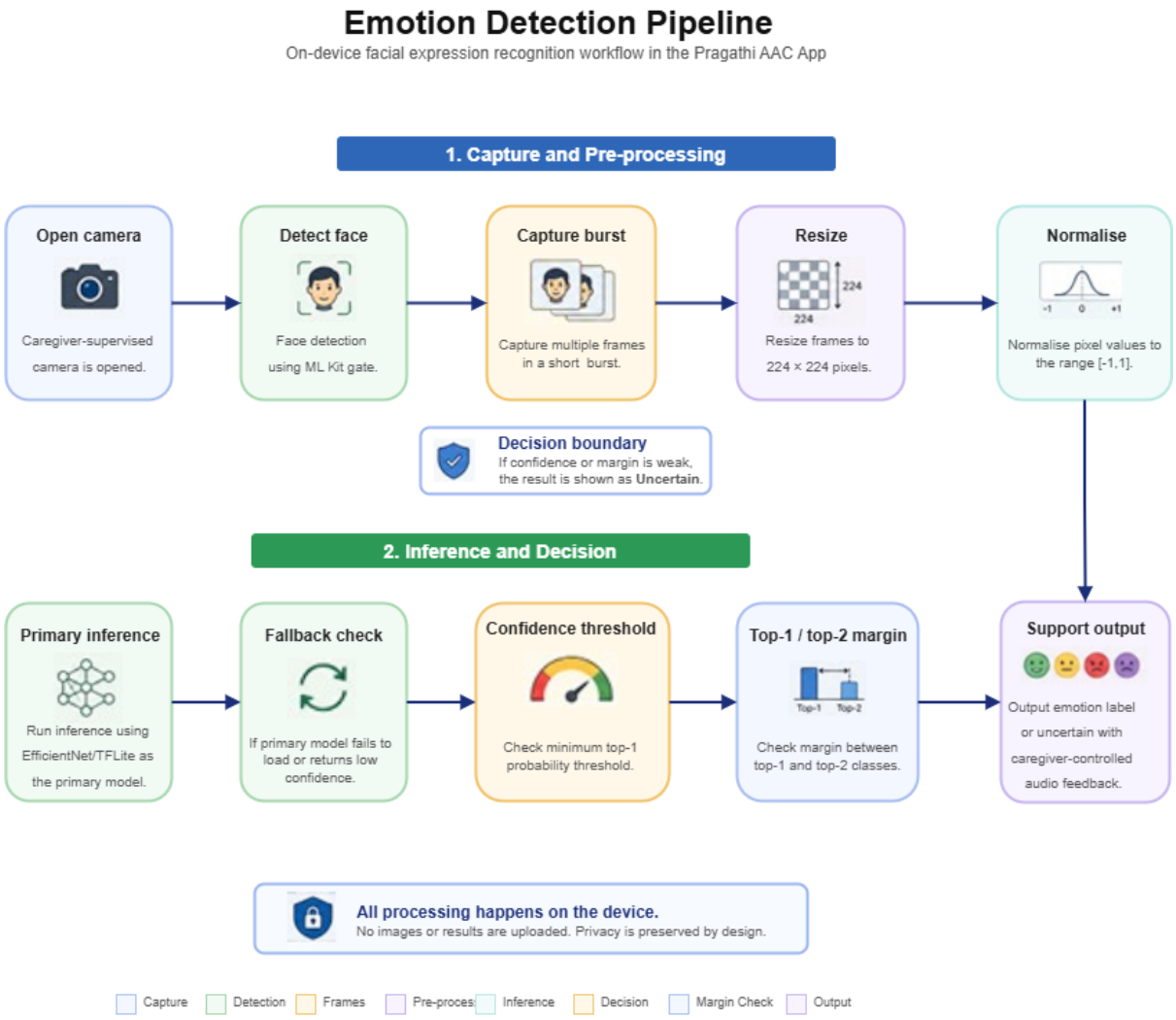


Figure 5.5: Emotion detection pipeline.

Editable diagram source: Download editable emotion detection pipeline diagram

At design level, the pipeline has eight stages.

1. **Camera selection.** The application discovers available cameras through the camera plugin and prefers the front camera, falling back to any camera when only one is available.
2. **Face detection gate (optional).** Google ML Kit FaceDetector is run in *fast* performance mode with minFaceSize: 0.15. If at least one face is detected and the bounding box short-side ratio is at least _kMinFaceRatio (0.18), the face region is cropped with a fractional padding (_kFaceCropPadding, 0.20). The face gate can be

turned off in the debug menu (emotion_face_gate_enabled), in which case the whole frame is centre-cropped square and used directly.

3. **Burst capture.** Five quick captures are taken per shutter tap (_kBurstFrames = 5, _kBurstDelay = 120 ms). Their softmax vectors are averaged to produce one stable decision.
4. **Preprocessing.** Each capture is decoded as RGB, resized to 224 × 224 pixels and normalised to [-1, +1] using (pixel / 127.5) − 1.0. The same formula is used in both the training pipeline and the Flutter run-time, which avoids the EfficientNetB0 input-scale mismatch that affected the initial training stage.
5. **TFLite inference.** The primary model is loaded first. If loading or tensor validation fails, the fallback model is loaded. Tensor validation requires input shape [1, 224, 224, 3], output shape [1, 6] and float32 data type, exactly matching the trained graph.
6. **Confidence and margin checks. The maximum softmax probability must be at least _kMinConfidence (currently 0.40 for debugging, with a final tuning range of 0.50-0.65 noted in the code). The top-1 minus top-2 probability margin must be at least _kMinTopMargin (currently 0.08). If either check fails, the screen returns Uncertain. Otherwise, the predicted class is treated as reliable.**
7. **Display mapping.** The raw class label is mapped to a friendly label through _displayFor() (Anger → Angry, Joy → Happy, Natural → Neutral, Sadness → Sad, Fear → Fear, Surprise → Surprise).
8. **Supportive layer. For a reliable result, the optional WAV prompt is played in the active language using EmotionSupportAudioService, and the per-emotion SensorySupportPlan is used to drive a calm visual animation. Nothing plays automatically. The caregiver presses Play support to start the prompt.**

The design deliberately includes a No face, Face not clear, Uncertain and Reliable state set rather than forcing every frame into a label. This is the central safety mechanism that allows the AI component to remain supportive rather than diagnostic.

5.5.1 Class diagram evidence and supporting structure view

A full class diagram is important for showing how the main screens, services, local data, model assets and support-audio components relate to one another. However, the complete class diagram is too wide and detailed to remain readable inside the main Chapter 5 layout. For that reason, the full detailed class diagram is provided in Appendix C as supplementary design evidence.

Figure 5.6 provides the main chapter with a cleaner supporting class-structure view. It does not attempt to list every Dart method or widget in the codebase. Instead, it summarises the main responsibility groups that shape the implemented system: app screens, services, AI/camera components, vocabulary/assets, companion app classes and external plugin boundaries. This makes the design easier to understand in the main report while still keeping the full class-level evidence available in Appendix C.

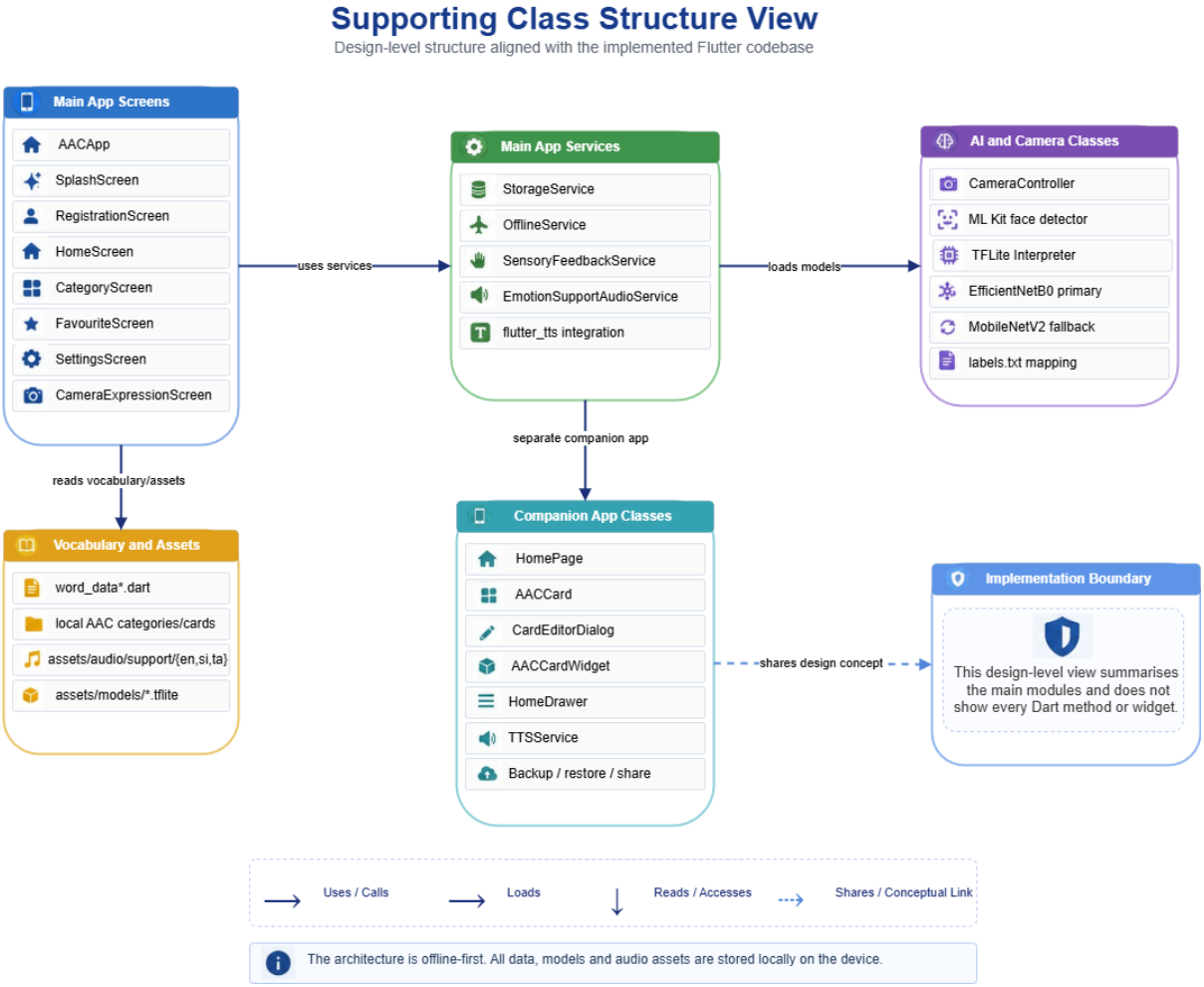


Figure 5.6: Supporting class structure view for the Smart AAC system.

Editable diagram source: Download editable supporting class structure diagram

5.5.2 Class structure interpretation

The class-structure view shows that the application is organised around separated responsibilities rather than one large screen file. The screen classes handle user-facing flows such as splash/registration, home navigation, category browsing, favourites, settings and camera-based emotion support. Their main role is to present the interface, respond to user actions and call the relevant service or model-handling logic.

The service classes keep non-visual behaviour outside the screens. StorageService manages local profile and settings state through SharedPreferences. OfflineService supports connectivity-status checking. SensoryFeedbackService handles vibration and support-plan behaviour, while EmotionSupportAudioService resolves and plays local supportive audio prompts. This separation keeps the UI layer easier to read and reduces the risk of mixing presentation, storage and audio logic in the same place.

The AI and camera-related classes represent the emotion-support workflow. They support camera access, face detection, TensorFlow Lite model loading, primary/fallback model selection, tensor validation, preprocessing, confidence checks, top-1/top-2 margin handling and display-label mapping. These responsibilities match the pipeline in Figure 5.5 and show how the design is implemented safely without forcing an emotion label on unclear input.

The vocabulary and asset elements represent the local resources bundled with the app. These include the compiled Dart word-data maps, category files in lib/data/word_data/, TensorFlow Lite model files, labels.txt and local support-audio folders. External plugins provide platform-level functions such as camera access, ML Kit face detection, TensorFlow Lite inference, shared preferences, connectivity checking, text-to-speech and audio playback.

Table 5.6: Class-structure responsibility mapping.

Class group	Main responsibility	Example responsibility
Main app and screens	Handle user-facing flows and navigation	AACApp, SplashScreen, RegistrationScreen, HomeScreen, CategoryScreen, FavouriteScreen, SettingsScreen and CameraExpressionScreen.
Storage and settings services	Keep lightweight app state separate from screen layout code	StorageService, SharedPreferences-backed registration/profile values, vibration preference, favourite categories and model-load settings.
Sensory and audio services	Provide optional caregiver-controlled feedback without making diagnostic claims	SensoryFeedbackService, SensorySupportPlan, EmotionSupportAudioService and EmotionAudioPaths.
AI, camera and model handling	Manage the supportive emotion workflow and safety gates	CameraExpressionScreen with EmotionModelLoadResolver, TensorShapeValidator, EmotionDecisionHelper, EmotionDisplayMapper and CaptureFrameGate.
Vocabulary and asset data	Represent bundled local application resources	word_data maps, category files in lib/data/word_data/,

		assets/models/ and assets/audio/support/.
External plugins	Provide native/platform capabilities used by the Flutter app	camera, google_mlkit_face_detection, tflite_flutter, shared_preferences, connectivity_plus, flutter_tts and audioplayers.

Table 5.6 summarises the responsibility groups shown in Figure 5.6. Together, Figure 5.6 and Table 5.6 provide a readable class-structure explanation in the main design chapter, while **Appendix C** preserves the full detailed class diagram for closer inspection.

5.6 Model loading and fallback design

The model loading design is driven by three constraints: TFLite compatibility, mobile size, and demo reliability.

- **Primary model:** assets/models/emotion_efficientnetb0_finetuned.tflite (4.7 MB).
- **Fallback model:** assets/models/emotion_mobilenetv2_fallback.tflite (2.6 MB).
- **Shared labels:** assets/models/labels.txt (raw class order: Anger, Fear, Joy, Natural, Sadness, Surprise).

Three load modes are supported (persisted in SharedPreferences as emotion_model_load_mode):

- **Automatic**. Try the primary model first, fall back to MobileNetV2 if loading or tensor validation fails.
- **EfficientNet only**. Load only the primary model. Used to test the final candidate.
- **MobileNet only**. Load only the fallback model. Used to test fallback reliability.

The internal name shown on the debug screen ("EfficientNetB0 Fine-tuned" or "MobileNetV2 Fallback") matches the model that was actually accepted, not just the one that was attempted. This means a viva examiner can see at a glance which model is running on a given device.

5.7 Communication profiles, customisation and trilingual content

The system addresses two communication profiles with two related applications.

Level 1-2: customisable card-based AAC (simple_aac_app). Children at ASD Level 1-2 can usually engage with symbol-based communication and benefit from caregiver-created vocabulary that reflects the child's daily life. The simple_aac_app provides this profile with:

- Customisable AAC cards (AACCard model) carrying a label, a colour, an optional image, an optional audio recording and a list of trilingual labels in Sinhala (labelSinhala), Tamil (labelTamil) and English (label).
- Nested sub-cards so that a category card can open into a sub-grid of related cards.
- A caregiver edit mode for adding, editing and deleting cards.
- Manual **backup, restore and share** of the entire vocabulary as a ZIP archive (_buildBackupZip in lib/pages/home_page.dart), with three options: share via Share.shareXFiles (e.g. to WhatsApp), save to the Downloads folder via a platform channel, and save to the app documents folder for later restore. Restore reads either a file picker selection or the app folder.
- Local persistence of card data through SharedPreferences and serialised JSON in StorageService.saveCards() and StorageService.loadCards().

This is the practical replacement for the cloud dashboard described in the interim plan: caregivers and therapists who used to collaborate through a portal can now share a backup ZIP through any channel they already use.

Level 3+: structured AAC + supportive on-device emotion observation (main Smart AAC). Children at ASD Level 3+ may have very limited functional speech and benefit from a simpler, more stable AAC flow designed with speech therapist and specialist guidance. The main Smart AAC application provides this profile with:

- A fixed set of vocabulary categories compiled into the application in lib/data/word_data/, covering 16 themes (needs, feelings, family, food, etc.). The categories and the order of words inside each category are stable across launches.
- The same trilingual coverage as simple_aac_app, with si, ta and en strings for every entry.
- The on-device facial expression recognition layer (Chapter 5, Section 5.5 and Chapter 6, Section 6.5) as a supportive observation aid that complements the AAC interaction.

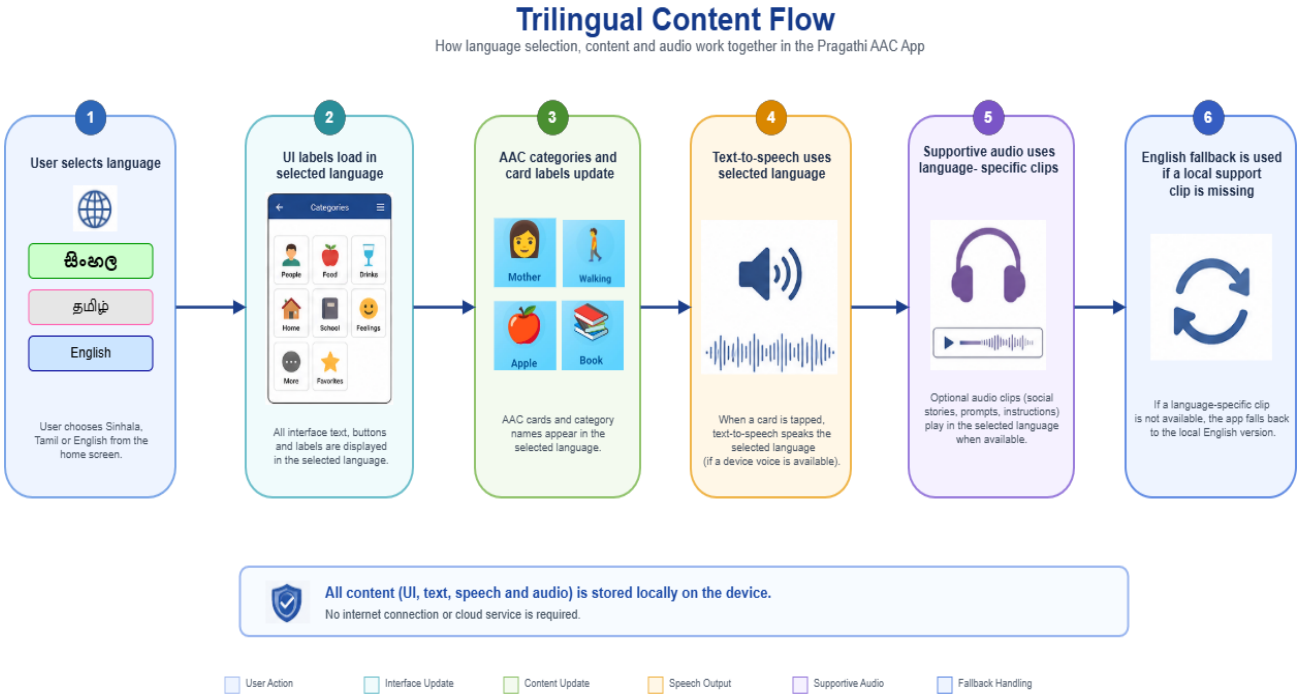


Figure 5.7: Trilingual content flow.

Editable diagram source: Download editable Trilingual content flow Diagram

Trilingual support is woven through three layers across both applications.

- **Vocabulary.** Each entry carries an si, ta and en string. The home screen and category screens display the current-language string while keeping the same emoji or image for visual continuity.
- **Text-to-speech. flutter_tts is configured with the active language code (si-LK, ta-IN or en-GB). Where the device's TTS engine does not include a Sinhala or Tamil voice, the application falls back gracefully to a default voice. No Sinhala or Tamil audio is rendered without the device's installed voices.**
- **Supportive WAV prompts.** Trilingual prompts live in the main application's assets/audio/support/{en,si,ta}/ folder, one WAV per emotion (happy, sad, angry, fear, surprise, neutral). The EmotionSupportAudioService selects the file matching the active language and falls back to English if the preferred WAV is missing. Missing files are reported through debugPrint rather than causing a crash.

Neither profile is treated as more important than the other. The technical research contribution of this project focuses on the Level 3+ profile because that is where on-device emotion-aware AAC support adds the most value, but the Level 1-2 customisable application is part of the delivered system and is cited in Chapter 6, Section 6.3.7.

5.8 User interface design

Five UI design principles guided the screens.

1. **Large, single-tap targets.** Card sizes are generous and the grid is kept readable on mid-range Android devices.
2. **Emoji-first symbols.** Each card pairs a high-recognition emoji with a trilingual label so a non-reading child can still associate the card with a concept. Stable emoji rendering is achieved using the bundled NotoColorEmoji font (pubspec.yaml) and a custom EmojiText widget (lib/widgets/emoji_text.dart) that avoids fontFamily: 'emoji' flicker.
3. **Two visual themes for engagement.** The application offers a "girl" and a "boy" colour theme (set during registration), implemented in lib/screens/theme/app_theme.dart. The theme switches the primary palette but never the layout, so vocabulary navigation stays consistent.
4. **No bright auto-play visuals on the camera screen. Visual animations on the emotion screen are slow, low-contrast and gentle, a 4-second calmBreathing controller and a small positive-pulse animation. They never start automatically. The caregiver presses Play support first. This is documented inline in lib/screens/camera_expression_screen.dart.**
5. **No raw images leaving the device.** The camera screen does not save the captured frame to disk and does not transmit it anywhere. Only the resulting emotion label and optional audio prompt are user-visible.

5.9 Functional requirements

Table 5.2: Functional requirements.

ID	Functional requirement	Implementation evidence	Status
FR1	The system shall allow a caregiver to register or confirm a child profile with language preference.	Registration screen and locally stored profile state.	Met
FR2	The system shall provide Sinhala, Tamil and English AAC labels where available.	Trilingual vocabulary data and language selection workflow.	Met
FR3	The system shall allow the user to select AAC categories and cards to build a communication message.	Home screen, category/card interface and compiled local vocabulary.	Met
FR4	The system shall provide speech output for selected AAC content.	flutter_tts integration and device TTS configuration.	Met, subject to device voice availability
FR5	The system shall allow commonly used cards to be accessed through favourites.	Favourites screen/carousel and local storage.	Met
FR6	The system shall support offline-first operation for core AAC functions.	Compiled vocabulary, local SharedPreferences state and no cloud dependency for core use.	Met

ID	Functional requirement	Implementation evidence	Status
FR7	The system shall provide caregiver-supervised camera-based supportive emotion observation.	Camera/emotion screen, face-detection gate, on-device inference, confidence checks and top-1/top-2 margin checks.	Met as supportive observation only
FR8	The system shall avoid forcing an emotion label when input is unclear.	No face, face not clear, uncertain and reliable states.	Met
FR9	The system shall play optional supportive audio prompts after a reliable emotion cue.	Local supportive audio prompts organised by language in the application assets.	Met, with pronunciation review before wider release
FR10	The companion simple_aac_app shall allow caregiver-customisable cards.	Companion app card customisation screens for labels, colours, sub-cards and visible-card controls.	Met
FR11	The companion simple_aac_app shall support manual backup, restore and sharing.	ZIP backup, restore pathway and Android share workflow.	Met
FR12	The system shall provide testing/debug evidence for active model selection.	Active model/debug panel showing the selected primary or fallback model during testing.	Met for testing use

5.10 Non-functional requirements

Table 5.3: Non-functional requirements.

ID	Non-functional requirement	Target / rationale	Evidence / current status
NFR1	Offline operation	Core AAC functions should work without internet connectivity.	Met. Vocabulary, profile state and model assets are local, and core AAC and on-device inference do not require cloud processing.
NFR2	Privacy	Camera frames should not be uploaded or stored by the app.	Met. FER inference runs on-device using TensorFlow Lite. No raw camera frame upload is claimed.
NFR3	Mobile compatibility	The Android build should run on realistic test devices using compact TFLite assets.	Met through release APK evidence and real-device walkthroughs. Wider device benchmarking remains future work.
NFR4	Model load reliability	The app should reduce deployment risk if the primary model fails.	Met. EfficientNetB0 is the primary model and MobileNetV2 is retained as fallback with tensor-shape validation.
NFR5	Usability	The AAC interface should be simple enough for caregiver-supervised use.	Met through simplified navigation, favourites and trilingual screens. Further

ID	Non-functional requirement	Target / rationale	Evidence / current status
			formal usability testing remains future work.
NFR6	Accessibility	The interface should support readable cards, clear controls and reduced sensory overload.	Partly met. Large cards, high-contrast visual design and simplified vibration setting are implemented. Formal WCAG audit remains future work.
NFR7	Performance	AAC navigation and camera/emotion support should remain usable without freezing during testing.	Partly met. Real-device walkthroughs did not identify workflow-blocking freezes, but standalone latency benchmarking was not recorded.
NFR8	Maintainability	The system should be understandable and maintainable for future extension.	Met. The application structure separates user interface screens, local data handling, model loading, vocabulary data and support-audio handling.
NFR9	Safety of AI output	AI output must be presented as supportive observation only.	Met. Confidence/margin checks, uncertain states and non-diagnostic wording are used throughout.

ID	Non-functional requirement	Target / rationale	Evidence / current status
NFR10	Deployment evidence	The project should provide verifiable build and release evidence.	Met for academic submission. Signed release APK, internal-testing evidence and production access request evidence are documented. Public production release is not claimed.

5.11 Technology stack

Table 5.4: Technology stack.

The table summarises the chosen technologies and their justification.

Component	Technology	Justification
Mobile application	Flutter (Dart)	Flutter provides a single codebase for Android and planned iOS work, a mature widget set and a strong plugin ecosystem for mobile ML integration.
Local storage	shared_preferences (Android backed)	Provides simple key-value persistence for user state and toggles without requiring a relational schema at this stage.
Connectivity	connectivity_plus	Provides a lightweight connectivity signal while the application remains offline-first.
Text-to-speech	flutter_tts	Uses the device's installed TTS engine and supports Sinhala, Tamil and English where

Component	Technology	Justification
		the corresponding device voices are installed.
Camera	camera plugin	Provides a standard Flutter camera abstraction for front and back camera access.
Face detection	google_mlkit_face_detection	Runs on-device without network calls and is used as a face gate rather than for landmark recognition.
On-device ML	tflite_flutter	Loads .tflite models from assets and supports the float32 six-class softmax output used by both deployed models.
Image preprocessing	image package	Decoding, cropping and resizing to 224 × 224 before tensor copy.
Audio playback	audioplayers	Plays optional supportive WAV prompts and handles missing files without crashing.
Haptics	Built-in HapticFeedback (flutter/services.dart)	Uses light haptic feedback for AAC cards and medium haptic feedback for navigation without adding a new vibration package.
Build tooling	Gradle / Kotlin on Android	The Android build configuration sets Java 17 and uses Flutter's default min/target SDK. kotlin.incremental=false was added to stabilise Windows cross-drive builds.

5.12 Design decisions and trade-offs

The most important design decisions and their trade-offs are documented below.

- **One Flutter application, not two.** The interim report described a *Platform A / Platform B* split. The final design keeps both ideas inside a single application with feature-level differentiation, because that is easier for caregivers and clinical contacts to install, demonstrate and test on real devices. The two-app idea is documented in Chapter 3 as a change from plan.
- **Primary / fallback TFLite model pair instead of a single best model. Notebook-strong EfficientNetB0 + CBAM was not selected for deployment because its TFLite export required Flex / SELECT_TF_OPS operations that the tflite_flutter runtime would not load. A safer pair (EfficientNetB0 without CBAM as primary, MobileNetV2 as fallback) is shipped instead. Detailed evidence is in Chapter 6, Section 6.5.**
- **SharedPreferences over SQLite for user state.** SharedPreferences keeps the user-state surface small, fast and easy to verify. The vocabulary is compiled into Dart code, which gives strong-typed text at build time and zero database overhead at run time.
- **No raw image storage.** Camera frames are not persisted, encoded to disk or transmitted off the device. This is reinforced by the offline-first behaviour and by the Android permission set, which deliberately excludes external storage permissions.
- **Manual support audio only.** No audio plays automatically on a reliable emotion detection. The caregiver has to press *Play support*. This avoids any risk of overstimulation in a child with autism.
- **Debug view, not production view.** The active-model debug screen is intentionally a debug feature, not a production feature, and the underlying confidence thresholds are explicitly marked in the code as needing tuning before final submission.

These design decisions support the central position of the report: the system is a deployable, supportive mobile AAC application built within the realistic constraints of a final-year project, and the AI component is helpful but never diagnostic.

Chapter 6: Implementation

This chapter documents how the system was actually built. It starts with the Flutter project structure, walks through the core AAC implementation (screens, services, vocabulary, trilingual support), and then concentrates on the AI model implementation: dataset preparation, MobileNetV2 baseline training, EfficientNetB0 fine-tuning, the move away from CBAM for mobile deployment, the TensorFlow Lite export and verification, and the integration of the model pair into the Flutter run-time. The chapter ends with the supportive audio layer, the Android build and the internal-testing distribution.

Where possible, the chapter cites real files from the project tree (paths like lib/screens/..., assets/models/...) so a reviewer can verify the evidence directly. Evidence not available within the submission is stated as a limitation rather than filled with unsupported results.

6.1 Development environment

The development environment used during the project is:

- **Flutter SDK.** Flutter 3.41.2 stable with Dart 3.11.0 was used for the final local build environment, confirmed from D:\flutter\bin\cache\flutter.version.json. The Dart SDK constraint is set in pubspec.yaml as >=2.19.0 <4.0.0.
- Android tooling. Android Studio was used for emulator testing and release builds, with Gradle using Kotlin DSL in android/app/build.gradle.kts. The Android namespace is lk.aac.sinhala_tamil_english and Java 17 is configured. android/gradle.properties was updated with kotlin.incremental=false after Kotlin incremental compilation errors caused by the Pub cache being on C: while the project was on D:.
- **Python / TensorFlow / Keras in Google Colab** for model training, TFLite export and verification.
- **Source control.** Git, currently on the final-thesis branch. The main development work happened on the upgrade-app branch.
- **Editor / IDE.** Visual Studio Code with the Flutter and Dart extensions.

The application targets Android 8.0 (API 26) and above, in line with the interim report's minimum-version target. The version published in pubspec.yaml at the time of writing is version: 1.1.0+2.

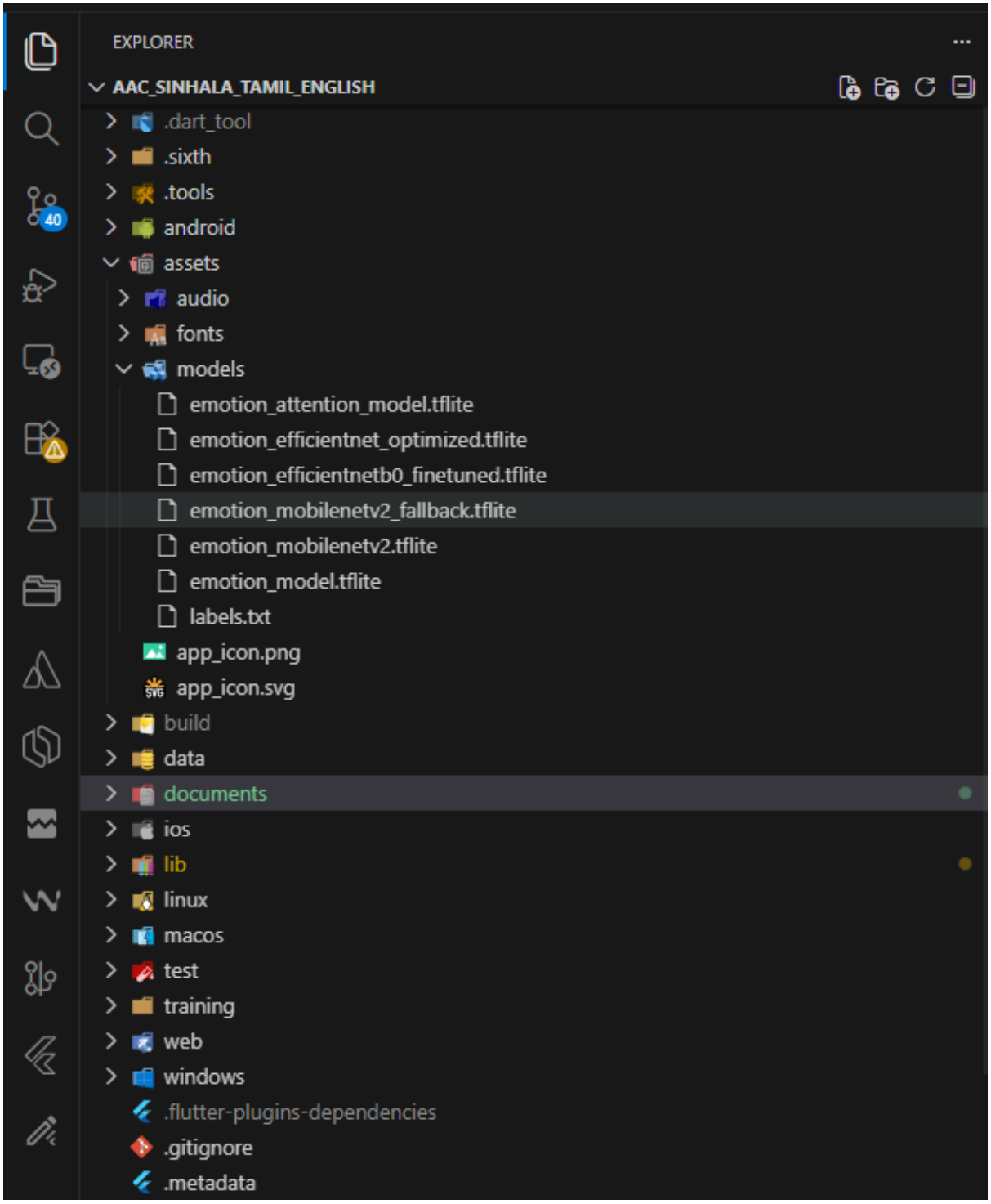


Figure 6.1: Flutter model assets folder.

The assets/models folder contains the two TensorFlow Lite model files and the shared labels.txt file.

6.2 Flutter project structure

The implementation uses a flat Flutter project structure rather than a heavy clean-architecture layering, because the application operates without a backend, maintaining a streamlined architectural footprint suitable for a single developer. The most important folders are lib/screens/ for the user interface, lib/services/ for storage, offline status, sensory feedback and support audio, lib/data/word_data/ for the trilingual vocabulary, assets/models/ for the two TensorFlow Lite models and labels.txt, and assets/audio/support/{en,si,ta}/ for optional supportive WAV prompts. The Android build files remain under android/, while automated checks are kept under test/.

The Flutter widget tree starts at lib/main.dart, which initialises OfflineService and SensoryFeedbackService before constructing the AACApp material widget with the SplashScreen as its home.

6.3 Core AAC implementation

6.3.1 Registration and child profile

The first screen the child or caregiver interacts with is RegistrationScreen (lib/screens/registration_screen.dart). It asks for the child's name, gender (boy / girl, which switches the UI colour theme) and preferred language (Sinhala / Tamil / English). All three fields are validated locally and persisted by StorageService.saveUserData(...) into SharedPreferences as user_name, user_gender, is_registered = true and is_premium = true.

The premium flag is set to true automatically on registration so that no in-app purchases are required. This was an intentional decision: the application is intended to be free for children with autism, in line with the design note shown on the registration screen (*"This app is free for autistic children"*, localised in all three languages).

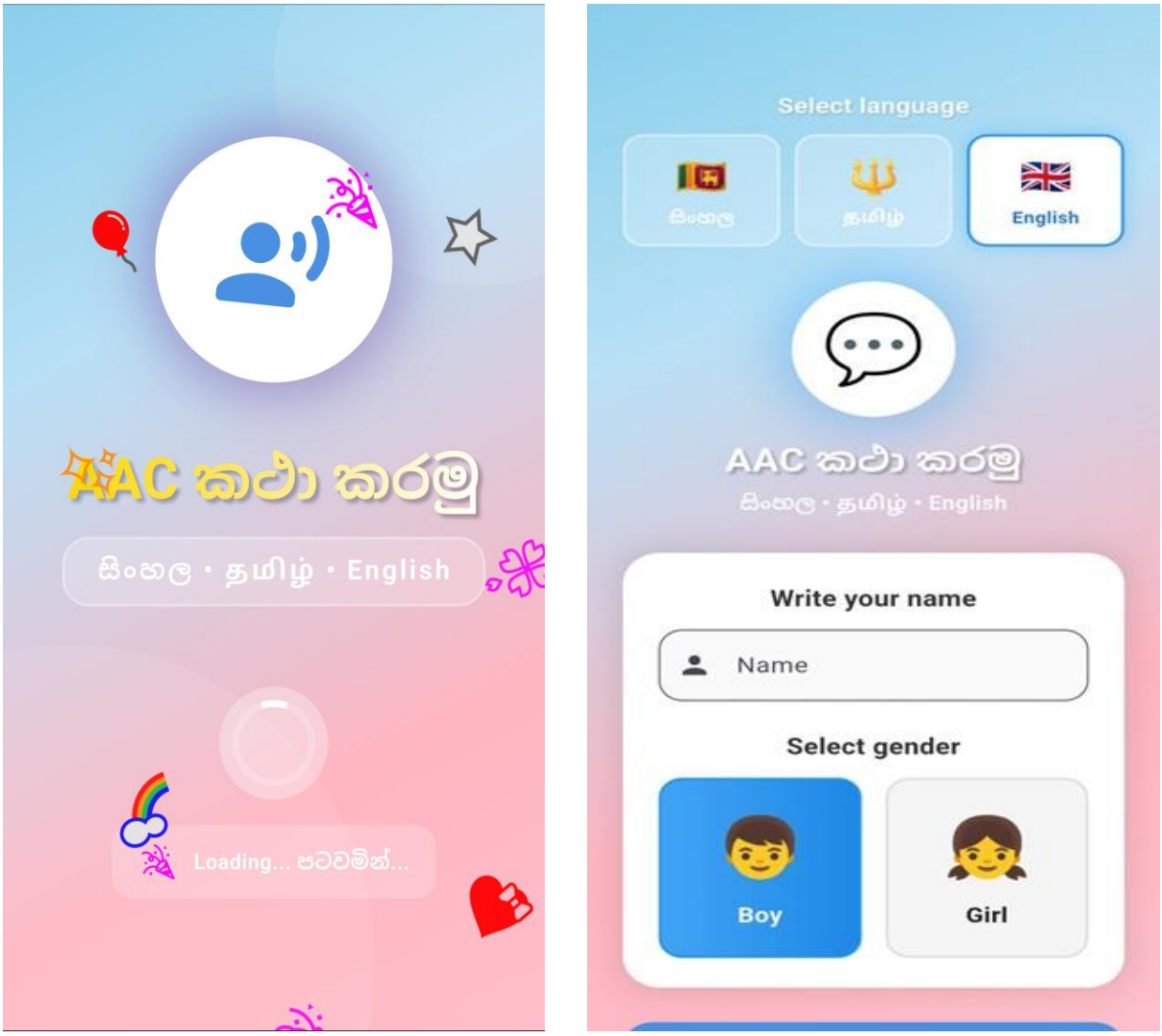


Figure 6.2: Splash and registration screen.

6.3.2 Home screen and vocabulary

HomeScreen (lib/screens/home_screen.dart) is the main AAC surface. It displays a scrolling grid of category cards, a sentence-building panel for selected words, and bottom navigation circles (Home / Favourites / Camera / Settings). The system UI navigation bar is hidden in immersive mode for fewer distractions, and is restored automatically when the screen is left.

Tapping a category card opens CategoryScreen (lib/screens/category_screen.dart), which lists the words in that category. A word tap plays TTS through flutter_tts in the active language code (si-LK, ta-IN or en-GB) and appends to the sentence panel.

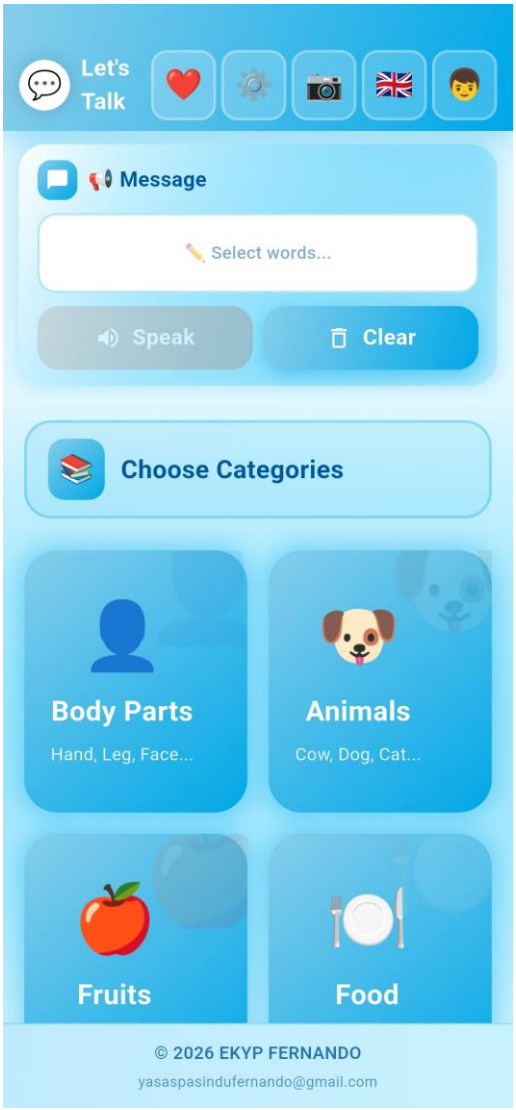


Figure 6.3: Home screen.

Figure 6.4: Categories screen.

The category navigation screen organises communication symbols across themes such as needs, feelings, family, food and activities.

Figure 6.5: AAC interaction example.

The screenshot shows symbol selection and text-to-speech output in the active language.

The vocabulary itself lives in lib/data/word_data/ as compiled Dart maps. There are 16 categories: body_parts, animals, fruits_vegetables, objects, colors_numbers, feelings, actions, sounds_music, food, household, colors, numbers, needs, sentences, family_words and places. Every entry carries si, ta and en strings, an emoji, and example action sentences in three languages. As an example, the *Food* category opens with බත් / சோறு / Rice (emoji 🍚), රොටි / ரொட்டி / Bread (emoji 🍞), වතුර / தண்ணீர் / Water (emoji 💧), and so on.

6.3.3 Favourites

FavouriteScreen (lib/screens/favourite_screen.dart) lets a caregiver mark which categories the child uses most. The favourite list is persisted via SharedPreferences and the screen includes a small pin/unpin control to keep the favourite carousel visible across sessions.

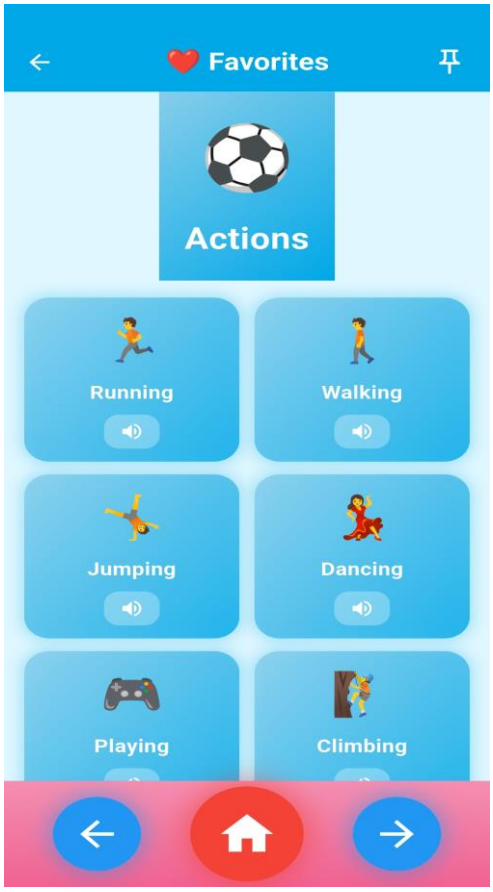


Figure 6.6: Favourites screen.

6.3.4 Settings

SettingsScreen (lib/screens/settings.dart) hosts the few configuration toggles in the application: a single Vibration On switch that maps to SensoryFeedbackService.setVibrationEnabled(...), language selection (si-LK, ta-IN, en-GB), and visibility / customisation of the favourites carousel. The settings surface was simplified in a later development pass because the earlier multi-toggle sensory card was found to confuse caregivers during informal testing.

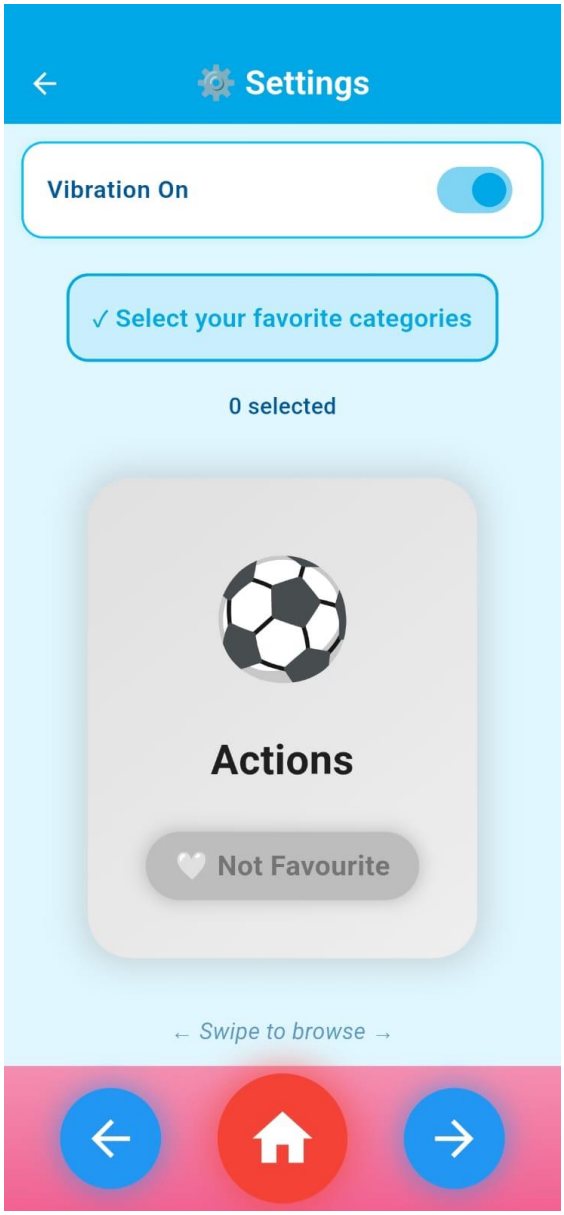


Figure 6.7: Settings screen.

The settings screen shows accessibility and language controls.

6.3.5 Trilingual support

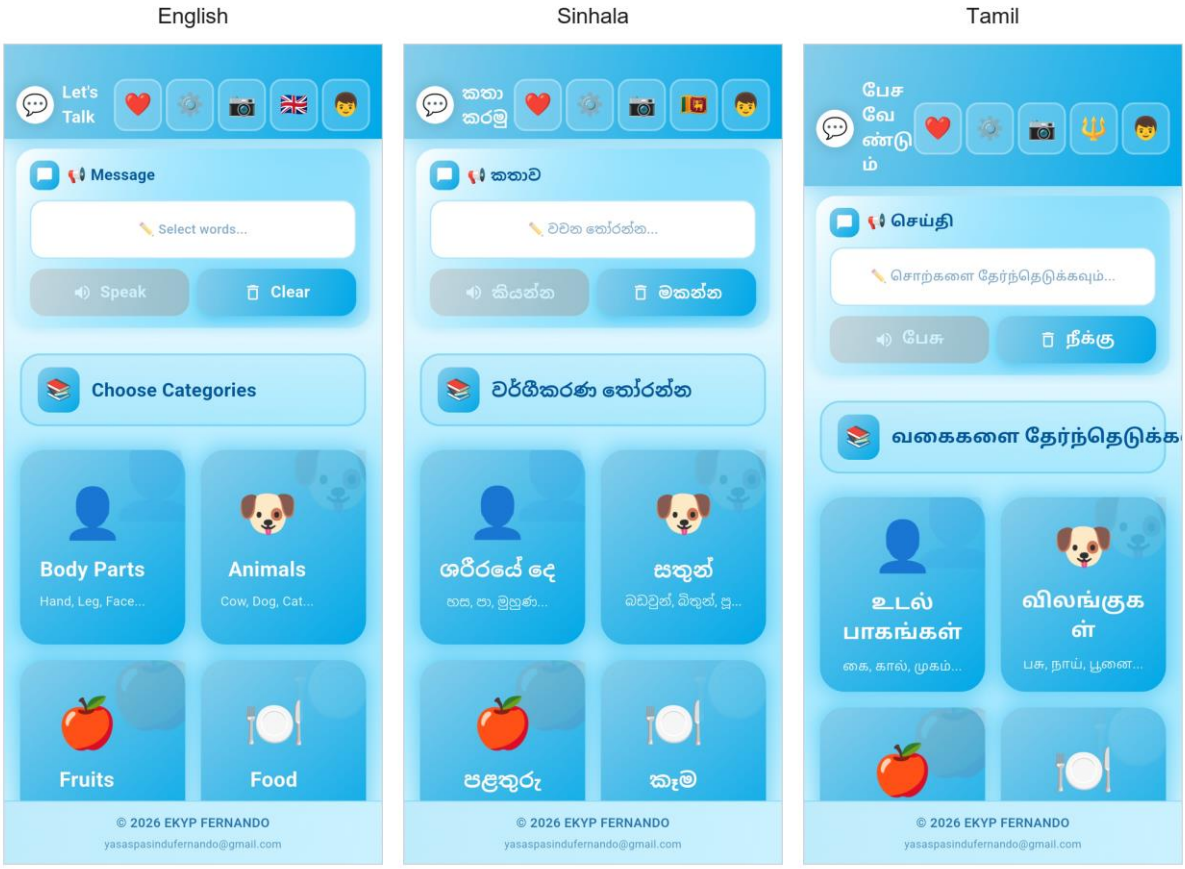


Figure 6.8: Trilingual UI side-by-side.

Trilingual support works through three independent layers in the codebase.

- **Vocabulary text.** Each Dart map entry exposes si, ta, en strings, so the same widget tree can display any of the three languages by reading the corresponding key.
- **TTS.** flutter_tts is set to the active language code at each TTS call. Where the user's device does not have a Sinhala or Tamil voice installed, the TTS engine returns silently rather than synthesising mismatched audio.
- **Supportive audio prompts. Localised WAV files in assets/audio/support/{en,si,ta}/ are used by EmotionSupportAudioService for the optional emotion prompts (see Section 6.6).**

6.3.6 Local storage and offline behaviour

StorageService (lib/services/storage_service.dart) wraps the shared_preferences plugin and exposes a tiny, typed API (isRegistered, isPremium, saveUserData, getGender). This is intentionally narrow so that the rest of the application does not depend directly on SharedPreferences.

OfflineService (lib/services/offline_service.dart) wraps connectivity_plus. It checks the initial connectivity on start-up and exposes a stream of bool values for any UI that needs to indicate online/offline. The application does not depend on connectivity to function, so the offline service is only an information signal.

6.3.7 Companion simple_aac_app for Level 1-2 customisation

The Level 1-2 communication profile is delivered through a separate Flutter application kept under C:\Users\HP\Desktop\simple_aac_app (referred to in this report as simple_aac_app). It targets caregiver-customisable AAC rather than the static, therapist-guided flow of the main Smart AAC application. The shared design principles (offline-first, trilingual, local storage, mobile-only) are preserved.

Key implementation evidence from simple_aac_app:

- **Customisable cards model.** lib/models/aac_card.dart and lib/models/aac_card_model.dart. Each AACCard carries a label, labelSinhala, labelTamil, an optional image path, an optional audio path, a colour and a list of nested subCards. JSON serialisation is built in.
- **Local persistence.** lib/services/storage_service.dart serialises the card list to a JSON string in SharedPreferences under the key aac_cards. The grid column count is stored under grid_columns. No cloud back-end is used.
- **Trilingual TTS.** lib/services/tts_service.dart uses flutter_tts and switches between English, Sinhala and Tamil at the same level as the main Smart AAC application.
- **Camera, gallery and voice personalisation.** image_picker for photographs and record for audio clips, with runtime permissions via permission_handler. Caregivers can attach a child-specific photo and their own voice recording to any card. This is the customisation feature that was scoped out of the main Smart AAC application and remains available in simple_aac_app.
- **Manual backup, restore and share.** Implemented in lib/pages/home_page.dart:
  - _buildBackupZip collects the JSON manifest and every attached image/audio file into a single Archive and produces a ZIP byte stream using ZipEncoder.

- _backupAndShare writes the ZIP to the app documents folder and hands it to Share.shareXFiles, which lets the caregiver choose **any** sharing target (WhatsApp, email, file manager, USB transfer).
- _backupToDownloads writes the ZIP to the device's Downloads folder via a platform channel (MethodChannel('aac_backup')), with a graceful fallback to the app documents folder if the system denies Downloads access.
- _restoreFromPickedFile opens FilePicker.platform.pickFiles(type: FileType.custom, allowedExtensions: ['zip']), extracts the archive into a temporary directory and reloads the cards from the manifest.
- _restoreFromAppDocsZip restores from a aac_backup.zip that was previously saved on the device.

The manual backup, restore and share path is the practical replacement for the cloud dashboard described in the interim plan (see Chapter 3, Section 3.3.7 and Chapter 4, Section 4.7). A caregiver who used to share progress through a portal can now share the entire vocabulary, including custom images and audio, by sending the backup ZIP through any channel they already use. Restoration on a second device is a single pick-file step.

The combination of the static main Smart AAC application and the customisable simple_aac_app covers both the Level 3+ supportive-observation use case (with on-device emotion observation) and the Level 1-2 customisable AAC use case (with manual backup, restore and sharing), while keeping every byte of vocabulary on the device unless the caregiver explicitly chooses to share it.

6.4 AI model implementation

The AI model implementation represents a primary technical contribution of this project. It is also the part of the project that changed most between the interim report and the final build. The original target, EfficientNetB0 + CBAM trained on Kaggle datasets with at least 80 per cent validation accuracy, was not the model that was actually deployed. Instead a pair of mobile-safe TFLite models was produced.

Supervisor feedback emphasized that the project must extend beyond merely integrating a pre-trained model into the user interface. The research contribution lies in the author-led pipeline: dataset audit and cleaning, augmentation and class-imbalance handling, training and fine-tuning runs, preprocessing alignment with Flutter, TensorFlow Lite export without Flex-only operators where possible, primary and fallback loading in Dart, and objective reporting of uncertainty and limits. The objective was practical on-device FER inside AAC, not a single-architecture benchmark paper.

The detailed source for this section is the supporting document 'Model Training and Deployment' (cited below as the Model Training Document).

6.4.1 Dataset sources and ethics

Three publicly available Kaggle datasets were used. Collecting a Sri Lankan child dataset was not possible within the project's ethics and time scope.

Table 6.1: Dataset sources used for model training.

The three Kaggle datasets supplied facial expression images for the six emotion classes.

Dataset	Link / Reference	Use in this project
FERAC Dataset	https://www.kaggle.com/datasets/rajasreechaiti/ferac-dataset	Public facial emotion data source
Autism Facial Emotion Recognition	https://www.kaggle.com/datasets/hasibur013/autism-facial-emotion-recognition	Autism-related emotion image collection
Autistic Children Emotions, Dr. Fatma M. Talaat	https://www.kaggle.com/datasets/fatmamtalaat/autistic-children-emotions-dr-fatma-m-talaat	Child-focused emotion data coverage

The three datasets share six target classes: Anger, Fear, Joy, Natural, Sadness and Surprise. After merging, folder names and class names were standardised. The final label order matches assets/models/labels.txt exactly.

A locally collected Sri Lankan child dataset is described in Chapter 8 as future work because it would require child safeguarding, parental consent and ethical clearance which go beyond the current project.

Limitations of public training data and ethical boundary for any future local collection

The merged public datasets are useful for learning a stable six-class head and for proving the Colab-to-TFLite-to-Flutter path, but they remain imperfect proxies for Sri Lankan children and for every variation of expressive behaviour seen in autism. Age mix, pose, lighting and camera quality in those sources do not match every real classroom or home in Sri Lanka. No images from Pragathi, Karapitiya or any other local site were added to the training folder for this thesis. If a future project ever seeks to collect child face data locally, that work would sit outside normal coursework unless it is run as a separate, fully governed study with ethics approval, parental consent, data minimisation, secure custody of raw files and health-sector coordination, as set out in Chapter 4, Section 4.9.1 and Chapter 8, Section 8.7. This boundary is established

to ensure that model performance is not misconstrued as proof that the system has already been adapted to a local child population it has never ethically sampled.

6.4.2 Dataset audit and class distribution

The training set was audited before training. The class counts in the merged training folder are recorded below.

Table 6.2: Training image count by class.

The counts show the number of training images per class in the merged dataset.

Class	Training images	Observation
Anger	844	Medium class size
Fear	494	Smallest class
Joy	3,720	Largest class, possible bias risk
Natural	759	Medium class size
Sadness	1,999	Large class size
Surprise	763	Medium class size

Joy was approximately seven times more numerous than Fear. If left untreated, a model trained on this distribution would inflate validation accuracy by over-predicting Joy, which would negatively impact the reliability of an emotion-aware AAC system.

Figure 6.9: Class Distribution Across Train and Test Sets

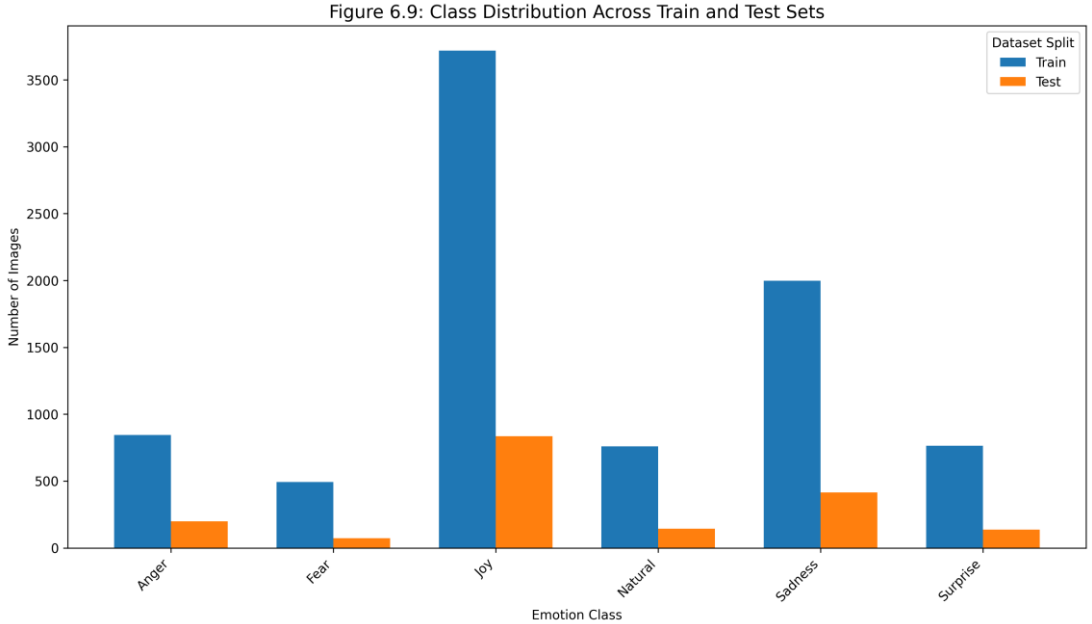


Figure 6.9: Class distribution across the train and test sets.

6.4.3 Class label mapping

Table 6.3: Class label mapping.

The table maps raw model labels to friendly Flutter display labels.

Index	Raw model label (labels.txt)	Flutter display label
0	Anger	Angry
1	Fear	Fear
2	Joy	Happy
3	Natural	Neutral
4	Sadness	Sad
5	Surprise	Surprise

The mapping lives in _displayMap inside lib/screens/camera_expression_screen.dart. The raw label order is preserved end-to-end so that the model's softmax index, labels.txt, the friendly display label and the audio file stem (happy, sad, angry, fear, surprise, neutral) all stay aligned.

A "Tired" class is not present. The Model Training Document explicitly notes that no reliable fatigue dataset was available, and mapping Surprise to Tired would silently mislabel every surprised face. Tired is therefore reserved for future work.

6.4.4 Class imbalance handling

Table 6.4: Class imbalance handling strategy.

The table records the risks identified during dataset audit and the mitigations used.

Issue identified	Risk	Action taken
Joy was the largest class	Model may over-predict Happy/Joy	Class weights applied (higher weight to under-represented classes)
Fear was the smallest class	Weak recall for Fear	Higher class weight assigned to Fear
Dataset included augmented images	Risk of overfitting to repeated patterns	Moderate augmentation only
Mixed data sources	Inconsistent image quality	Dataset audit and label standardisation

Standard accuracy alone was therefore not used as the model selection criterion. Validation loss, per-class behaviour and confusion-matrix output were all considered.

6.4.5 Preprocessing pipeline

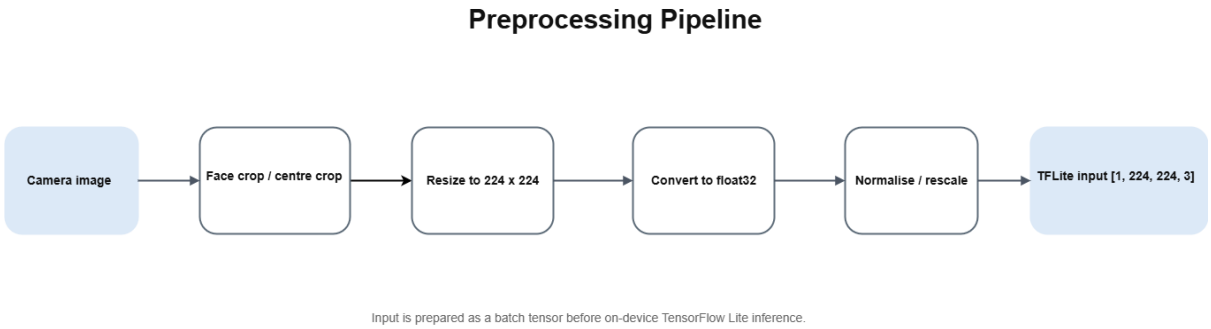


Figure 6.10: Preprocessing pipeline.

Editable diagram source: Download editable preprocessing pipeline diagram

The preprocessing pipeline used during training is identical to the one used on the Flutter camera input path:

1. Load image as RGB.
2. Resize to 224 × 224.
3. Normalise using (pixel / 127.5) − 1.0, yielding values in [-1, +1].
4. Stack into a batch.

The matching Flutter pre-processing lives in _preprocessFrame inside CameraExpressionScreen. Keeping training and runtime pre-processing identical is essential, it was the single biggest source of error in the early EfficientNetB0 experiments (see Section 6.4.7).

6.4.6 MobileNetV2 fallback model

Table 6.5: Model architecture comparison.

The table compares the model variants and their deployment status.

Model	Role in final project	Evidence / deployment status
MobileNetV2	Fallback model	Evidence from saved model evaluation and the confusion matrix in Figure 6.13. Exported as

Model	Role in final project	Evidence / deployment status
		emotion_mobilenetv2_fallback.tflite and verified using TensorFlow Lite interpreter input/output tensor checks in Figure 6.20.
EfficientNetB0 (no CBAM)	Primary model	Evidence from saved fine-tuned model evaluation and the confusion matrix in Figure 6.19. Exported as emotion_efficientnetb0_finetuned.tflite and verified using TensorFlow Lite interpreter input/output tensor checks in Figure 6.21.
EfficientNetB0 + CBAM	Interim experiment only	Not selected for final mobile deployment because TensorFlow Lite export required Flex / SELECT_TF_OPS and was less suitable for the submitted Flutter build.

MobileNetV2 (Sandler et al., 2018) was used as the first mobile-safe baseline. ImageNet-pretrained weights were loaded. The base was frozen first while only the new classification head was trained, then selected top layers were unfrozen for fine-tuning with a lower learning rate. Class weights were applied throughout.

The MobileNetV2 export was successfully verified as a .tflite file. It was retained as a fallback rather than the primary model because its validation behaviour was less stable than the later EfficientNetB0 result.

The learning-curve figures were prepared from saved training logs because the original Keras History objects were not available in the final Colab session. The figures therefore reflect the logged epoch-level training and validation metrics used during model development.

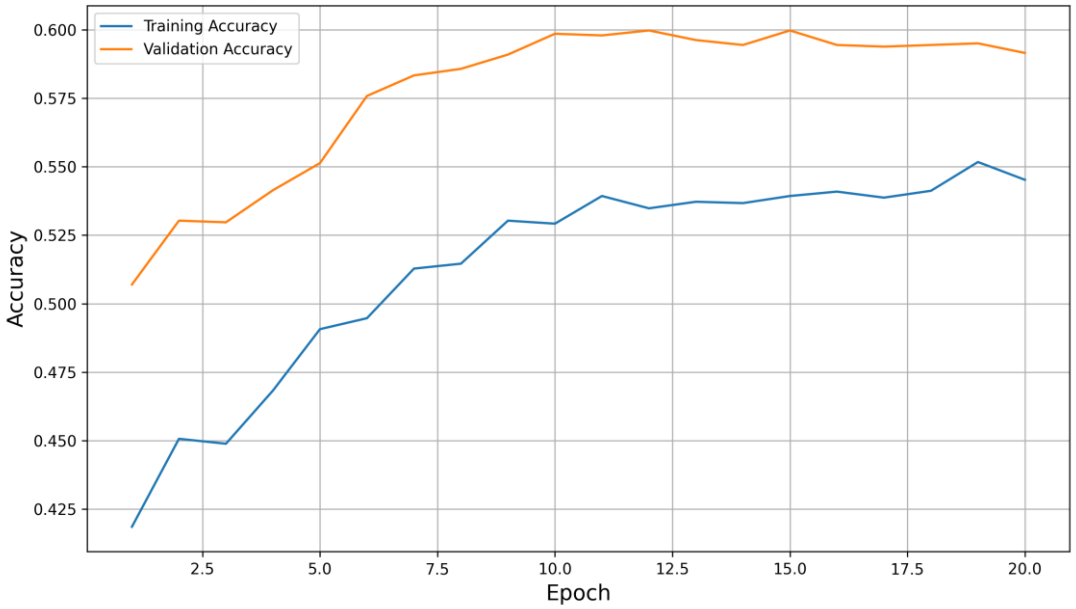


Figure 6.11: MobileNetV2 training and validation accuracy curve.

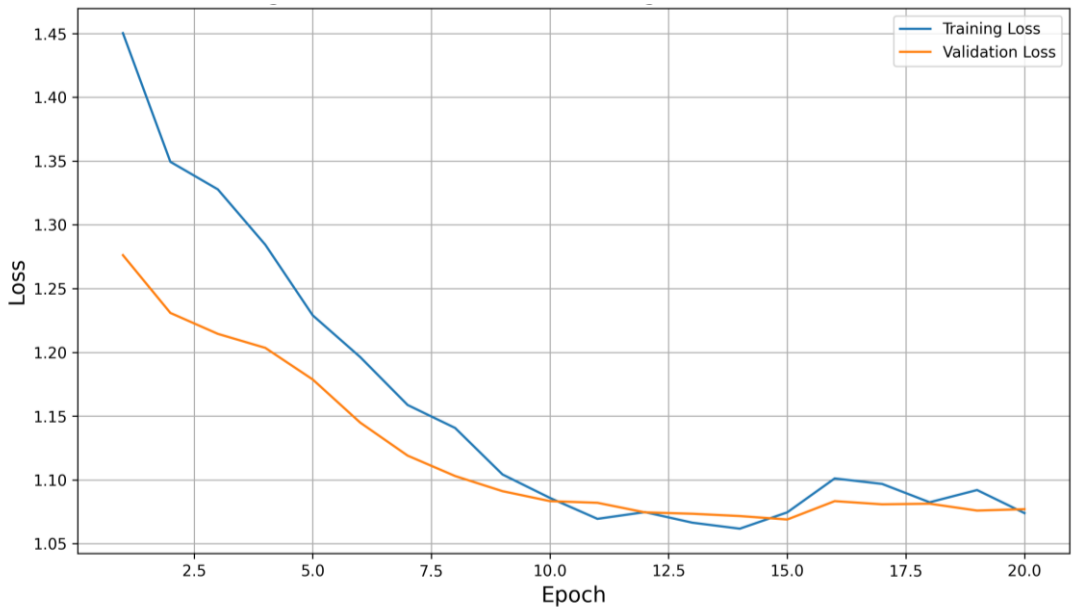


Figure 6.12: MobileNetV2 training and validation loss curve.

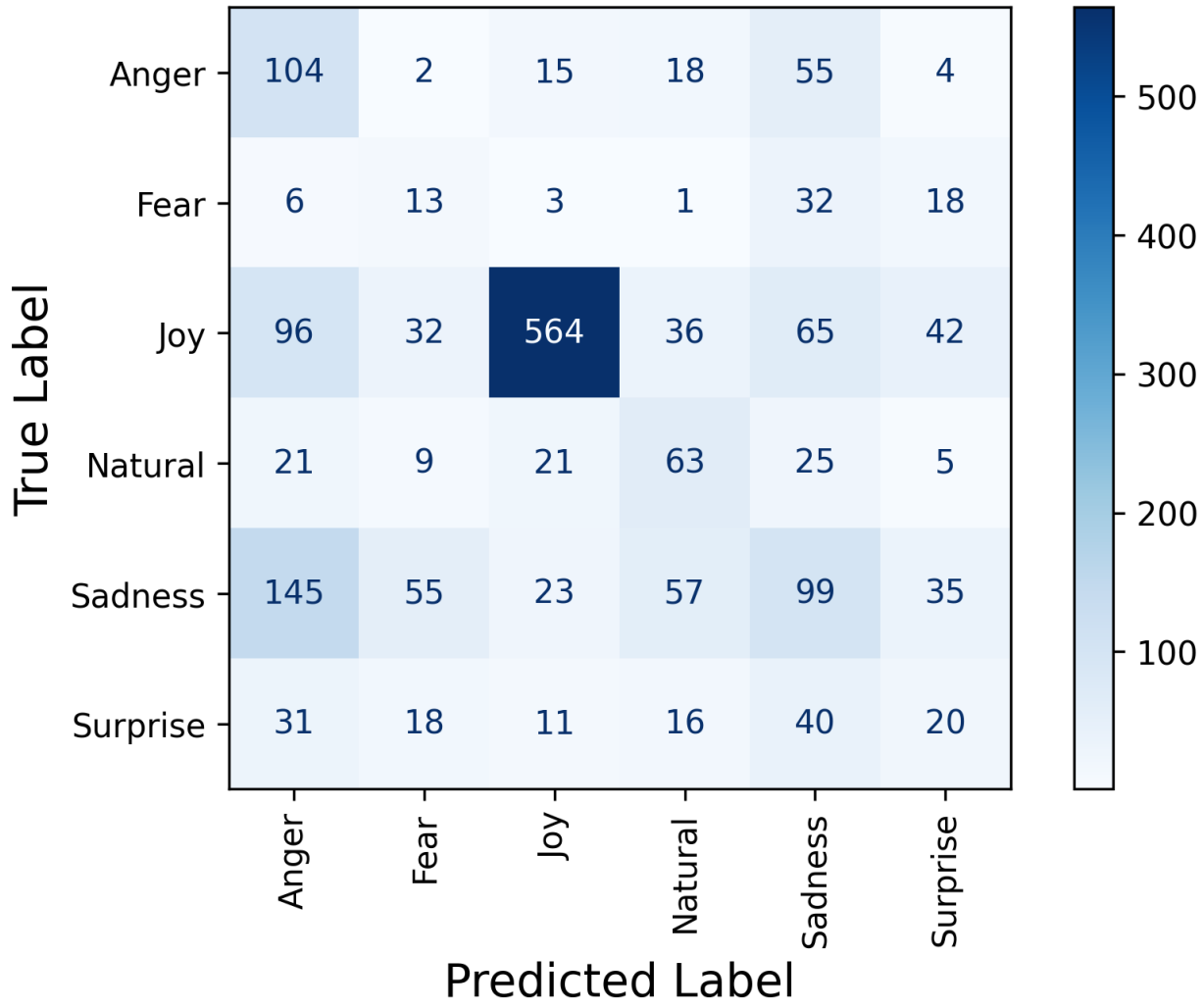


Figure 6.13: MobileNetV2 confusion matrix for the test dataset.

6.4.7 EfficientNetB0 primary model

EfficientNetB0 (Tan and Le, 2019) was chosen as the stronger mobile-safe candidate because it gives a better accuracy / efficiency trade-off than MobileNetV2 on similar problems.

The first EfficientNetB0 training run produced an unexpectedly poor validation accuracy. The cause was a preprocessing mismatch: the Flutter pipeline produced inputs in [-1, +1], but EfficientNetB0 with ImageNet weights expected inputs closer to the original [0, 255] pixel scale. To address this, two approaches were considered: modifying the Flutter pipeline or correcting the model graph. The latter was chosen by adding an internal rescaling layer that converts [-1, +1] back to [0, 255] before the EfficientNetB0 base. This kept the Flutter preprocessing pipeline identical for both models.

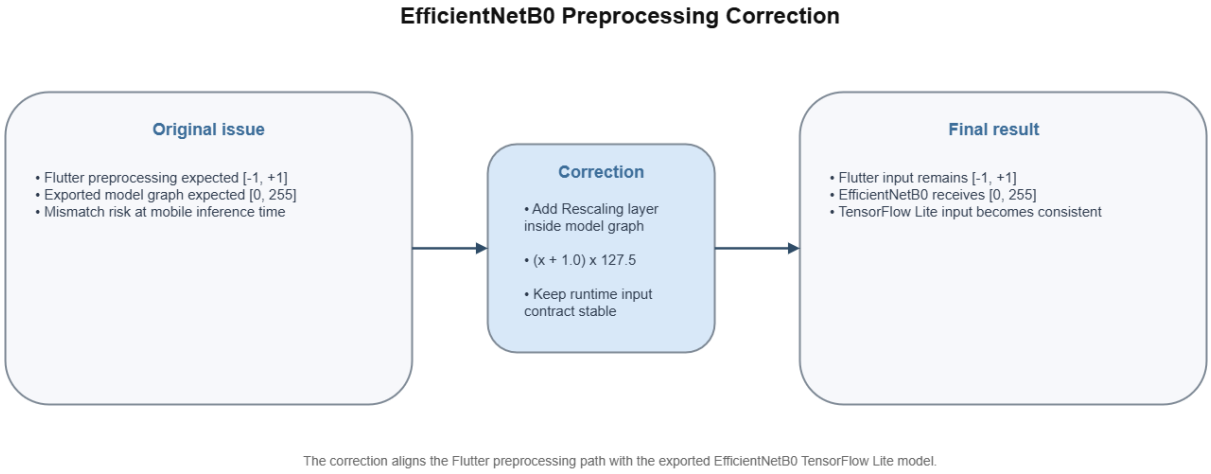


Figure 6.14: EfficientNetB0 preprocessing correction.

Editable diagram source: Download editable EfficientNetB0 preprocessing correction diagram

After the correction the model improved sharply. Frozen-base training reached approximately 67 per cent validation accuracy on the best checkpoint. Fine-tuning then unfroze the top 40 layers of the EfficientNetB0 base (Batch Normalisation layers were kept frozen) at a learning rate of 1e-5, with early stopping and timestamped checkpoints saved to Google Drive. At one point during fine-tuning, validation accuracy passed approximately 70 per cent. Model performance is evaluated using the learning curves, confusion matrices and TensorFlow Lite verification outputs.

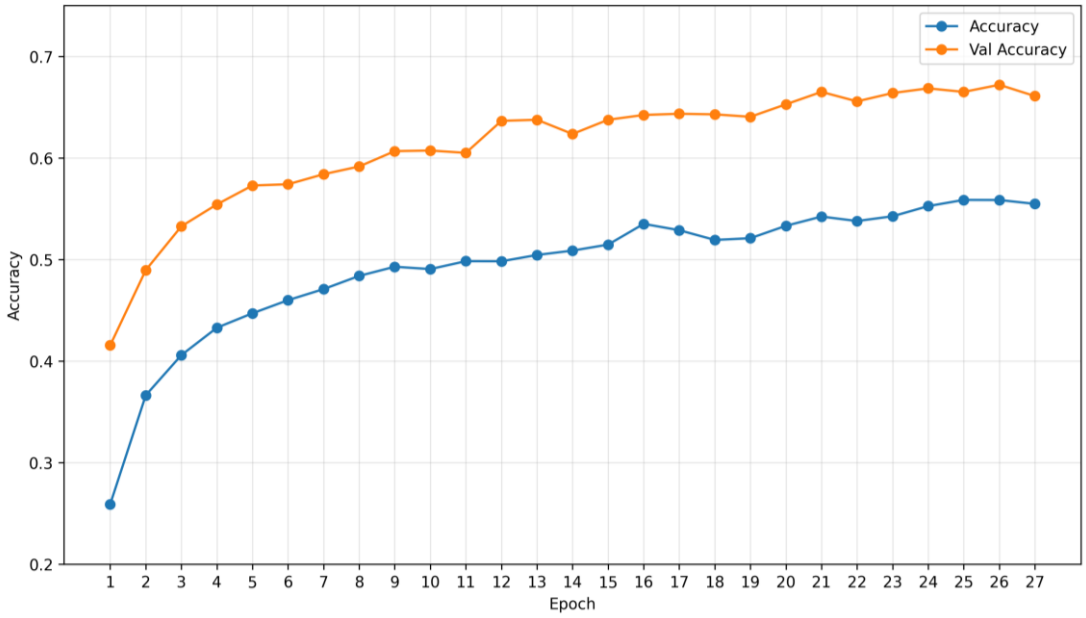


Figure 6.15: EfficientNetB0 frozen-stage training and validation accuracy curve.

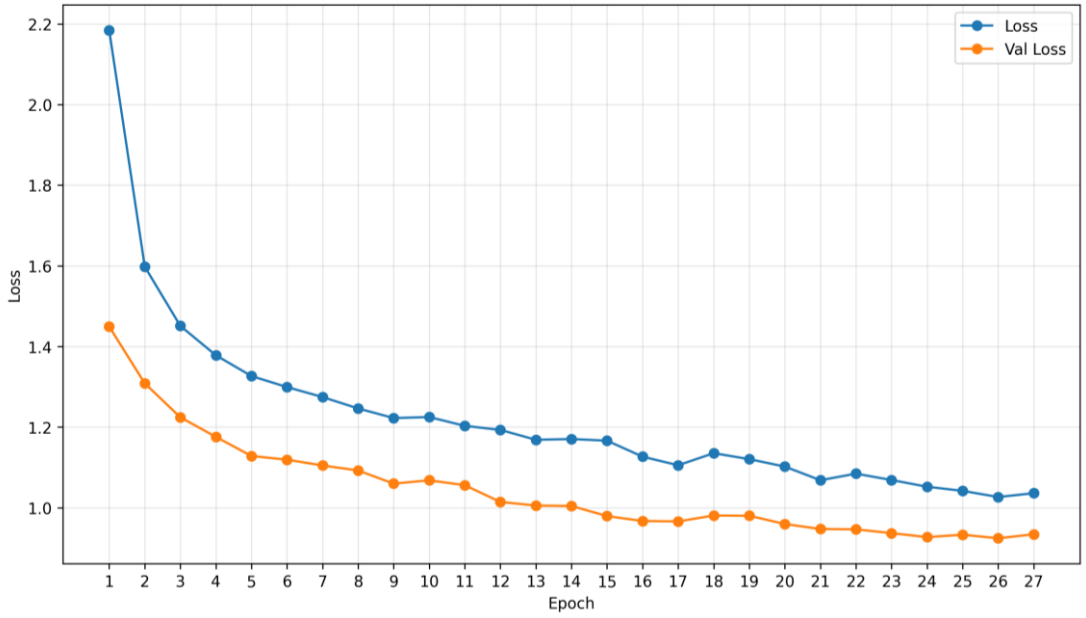


Figure 6.16: EfficientNetB0 frozen-stage training and validation loss curve.

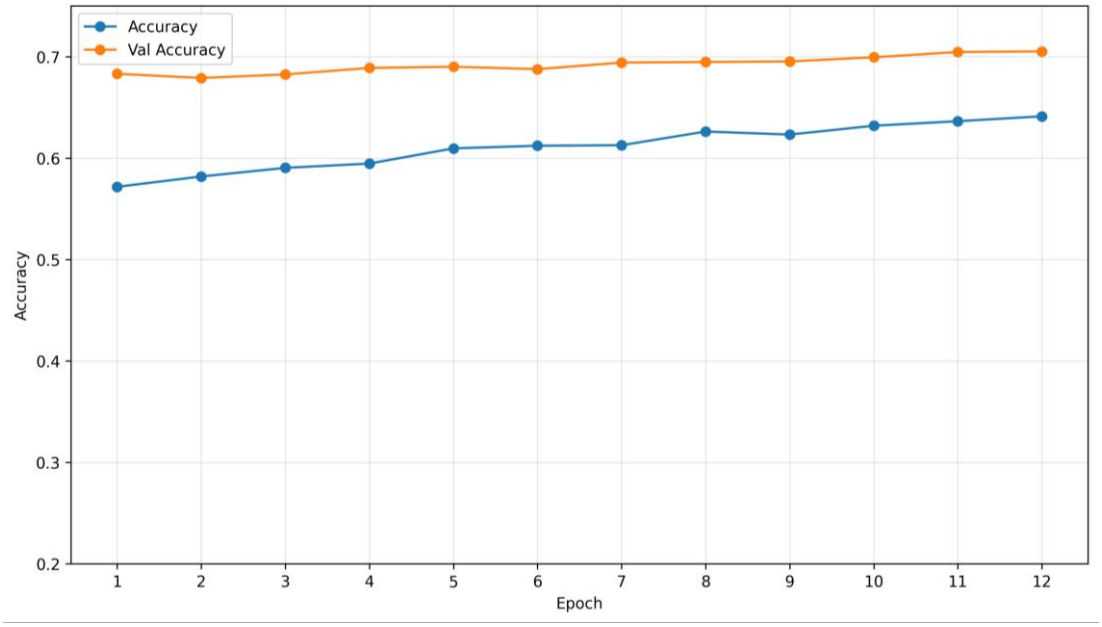


Figure 6.17: EfficientNetB0 fine-tuning training and validation accuracy curve.

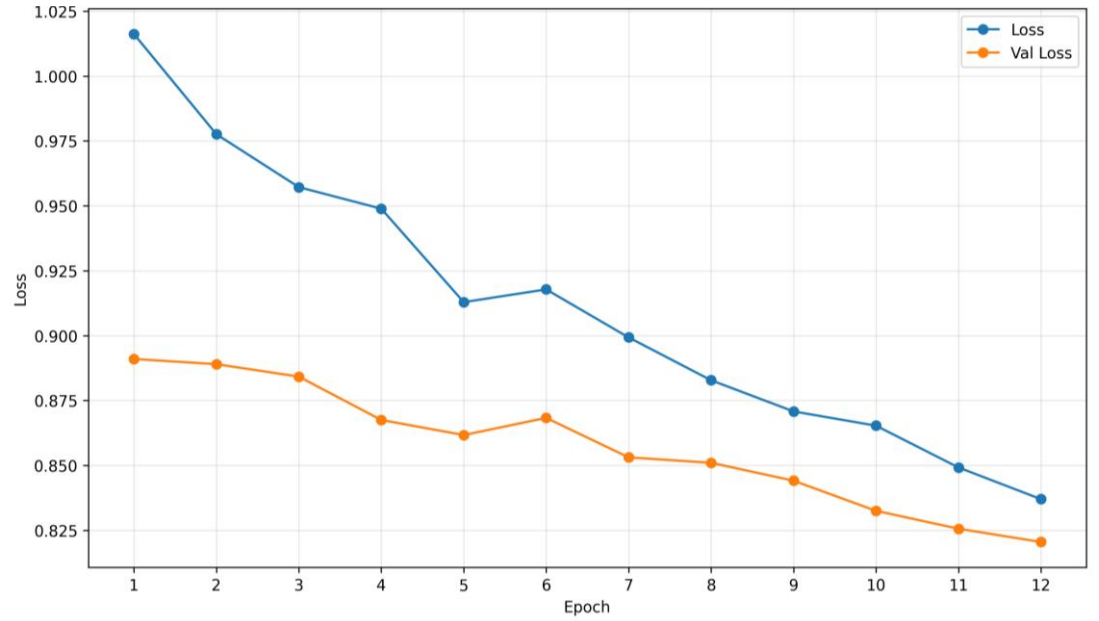


Figure 6.18: EfficientNetB0 fine-tuning training and validation loss curve.

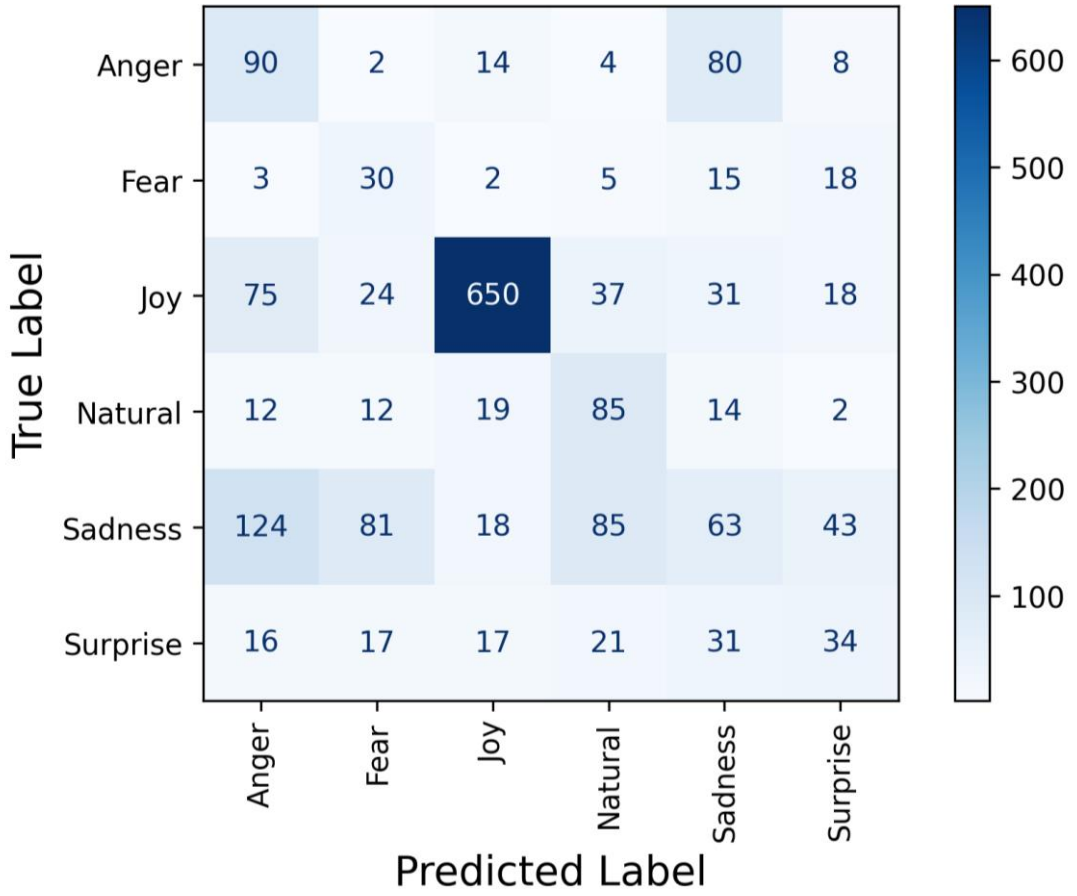


Figure 6.19: EfficientNetB0 confusion matrix for the test dataset.

Table 6.6: Fine-tuning configuration.

The table records the hyperparameters used during EfficientNetB0 fine-tuning.

Setting	**Value**
Base model	EfficientNetB0 without CBAM
Starting checkpoint	Best frozen-base EfficientNetB0 checkpoint
Unfrozen layers	Top 40
Batch normalisation layers	Frozen
Learning rate	1e-5
Epoch limit	8-20 (depending on Colab session stability)
Early stopping	Enabled

Setting	Value
Checkpoint saving	Enabled with timestamped filenames to Google Drive

6.4.8 Why CBAM was not deployed

EfficientNetB0 + CBAM (Woo et al., 2018) was attempted because attention modules can help a CNN focus on the eye, mouth and brow regions that matter most for facial expression. The notebook validation result was promising. However, exporting CBAM to TensorFlow Lite required SELECT_TF_OPS (Flex), which tflite_flutter will not load reliably inside a Flutter Android application without a heavier custom build. The author opted to keep the deployment surface small and mobile-safe by dropping CBAM rather than ship a model the application could not use on demonstration devices.

6.4.9 Colab training and checkpoint safety

Table 6.7: Colab checkpoint safety strategy.

The table records the risks and mitigations used during training.

Risk	Mitigation
Colab runtime disconnection	Save best checkpoints to Google Drive
Notebook autosave failure	Keep model files separate from notebook state
Fine-tuning overwriting frozen model	Use separate filenames for fine-tuned checkpoints
Exported model not loading in Flutter	Verify TensorFlow Lite model before integration
Final model uncertainty	Keep MobileNetV2 fallback model

6.4.10 TFLite export and verification

Both deployed models were exported to TensorFlow Lite using only built-in operations. SELECT_TF_OPS was deliberately not used after the CBAM incompatibility was discovered.

Each .tflite file was verified using the standard TFLite interpreter checks (input shape, output shape, data types, dummy inference, no Flex operations) before being copied into the Flutter project.

Table 6.8: TFLite compatibility checks.

The table records the required values for both bundled models.

Requirement	Required value
Input shape	[1, 224, 224, 3]
Output shape	[1, 6]
Input data type	float32
Output data type	float32
Operations	TFLite built-ins only
Flex operations	Not allowed
Dummy inference	Must succeed

Verification item	Result
Model	MobileNetV2 fallback model
File name	emotion_mobilenetv2_fallback.tflite
File size	2.55 MB
Input tensor shape	[1, 224, 224, 3]
Input dtype	float32
Output tensor shape	[1, 6]
Output dtype	float32
Dummy inference	Succeeded
Label count	6
Output units match labels	Yes
Flex / SELECT_TF_OPS required	No

Figure 6.20: TensorFlow Lite verification output for the MobileNetV2 fallback model.

Verification item	Result
Model	EfficientNetB0 primary model

Verification item	Result
File name	emotion_efficientnetb0_finetuned.tflite
File size	4.51 MB
Input tensor shape	[1, 224, 224, 3]
Input dtype	float32
Output tensor shape	[1, 6]
Output dtype	float32
Dummy inference	Succeeded
Label count	6
Output units match labels	Yes
Flex / SELECT_TF_OPS required	No

Figure 6.21: TensorFlow Lite verification output for the EfficientNetB0 primary model.

6.4.11 Exported model package

Table 6.9: Exported model package summary.

The table lists the files shipped in the Flutter assets/models folder.

File	Role	Approximate size
emotion_efficientnetb0_finetuned.tflite	Primary model (EfficientNetB0 fine-tuned)	4.7 MB
emotion_mobilenetv2_fallback.tflite	Fallback model (MobileNetV2)	2.6 MB
labels.txt	Shared raw label list (Anger, Fear, Joy, Natural, Sadness, Surprise)	Very small

The pubspec.yaml assets list was trimmed to declare only these three files, so older experimental .tflite files on disk are not shipped in the APK. The total emotion-model footprint inside the APK is therefore approximately 7.4 MB.

6.5 Camera and emotion detection implementation

The camera screen, lib/screens/camera_expression_screen.dart, ties together the camera plugin, the face-detection gate, the TFLite interpreter and the supportive feedback. It is by some margin the largest single file in the project and contains the most non-trivial logic. The implementation summary below tracks the actual file rather than the design ideal.

6.5.1 Primary / fallback model loading

The interpreter is loaded by _loadEmotionInterpreterOnly(isInitialStartup: ...). There are three modes, persisted in SharedPreferences as emotion_model_load_mode:

Table 6.10: Model loading modes.

The table describes automatic, EfficientNet-only and MobileNet-only loading modes.

Loading mode	Behaviour	Typical use
auto	Try EfficientNetB0 first and fall back to MobileNetV2 if loading or tensor validation fails.	Normal app behaviour
primaryOnly	Load only the EfficientNetB0 model	Test the final candidate

Loading mode	Behaviour	Typical use
fallbackOnly	Load only the MobileNetV2 fallback	Test fallback reliability

When a model is accepted, _activeEmotionModelName is set to either EfficientNetB0 Fine-tuned or MobileNetV2 Fallback. The diagnostic active-model string is then displayed in the debug panel when _showDebugProbs is on.

The diagnostic debugPrint lines from _tryLoadEmotionModelSlot use a consistent [ModelLoad] prefix, so it is easy to confirm which model was actually accepted on a given device by inspecting logcat or the Flutter run console.

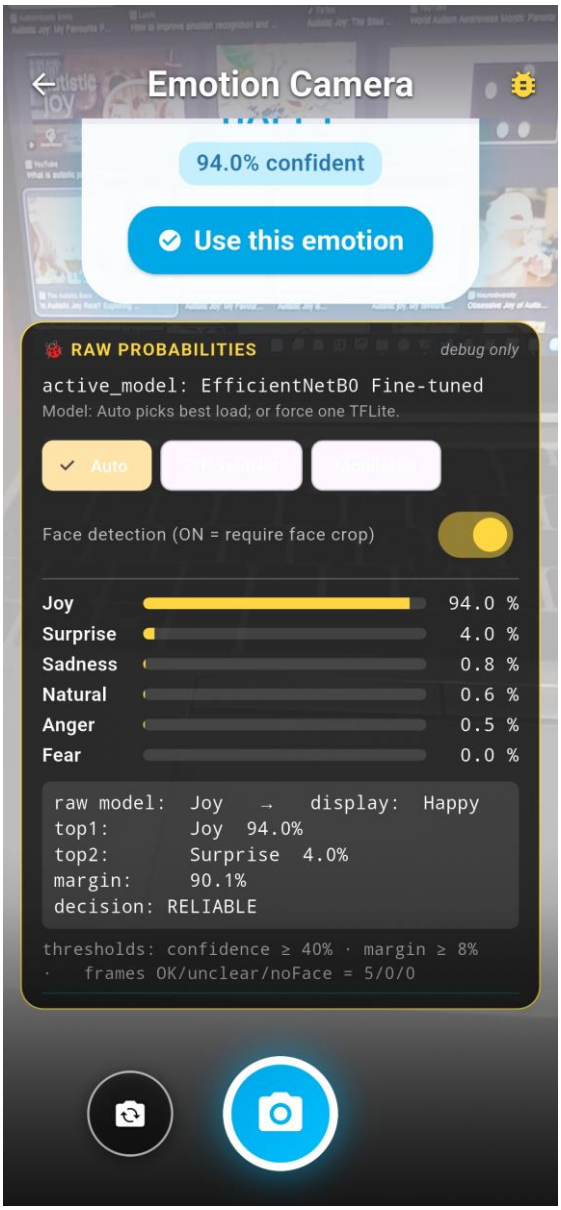


Figure 6.22: Active model debug screen.

6.5.2 Face detection gate

Face detection uses the google_mlkit_face_detection plugin. The FaceDetector is configured with performanceMode: FaceDetectorMode.fast, landmarks and classification off, tracking off, and minFaceSize: 0.15. The screen rejects frames where no face is detected (returning the *No face detected* state) and where the largest face bounding box has a short-side ratio below _kMinFaceRatio = 0.18 (returning the *Face not clear* state).

A face gate toggle (emotion_face_gate_enabled, default true) is persisted in SharedPreferences. When the gate is turned off in the debug panel, a centre-cropped 224 × 224 square is taken from the full frame instead.

6.5.3 Tensor validation

Before the interpreter is accepted, _emotionModelTensorsValid(...) checks that the input shape is [1, 224, 224, 3], the output shape is [1, 6], and both tensors are float32. If any check fails, the slot is rejected and the next slot is tried. If both slots fail, the user sees a localised error: either *"Emotion model shape is not supported"* (when both fail validation) or *"Emotion model failed to load"* (when both fail to load).

6.5.4 Burst capture and decision

For each shutter tap the screen captures _kBurstFrames = 5 frames with _kBurstDelay = 120 ms between captures. The softmax vectors are averaged to produce one stable decision. A reliable decision requires:

- the maximum softmax probability to be at least _kMinConfidence (currently 0.40 in the debug build; the in-code comment explicitly notes the production target is 0.50-0.65);
- the top-1 minus top-2 probability margin to be at least _kMinTopMargin (currently 0.08, with the same tuning note).

If either check fails the state moves to *Uncertain, try again* rather than displaying a confident result.

The current thresholds reflect a deliberate decision: in the debug build they are loose so that the author can see which probabilities the model produces on real devices. They will be tightened before final submission once real-device evidence is collected. The definitive confidence and margin parameters will be calibrated following comprehensive real-device measurements.

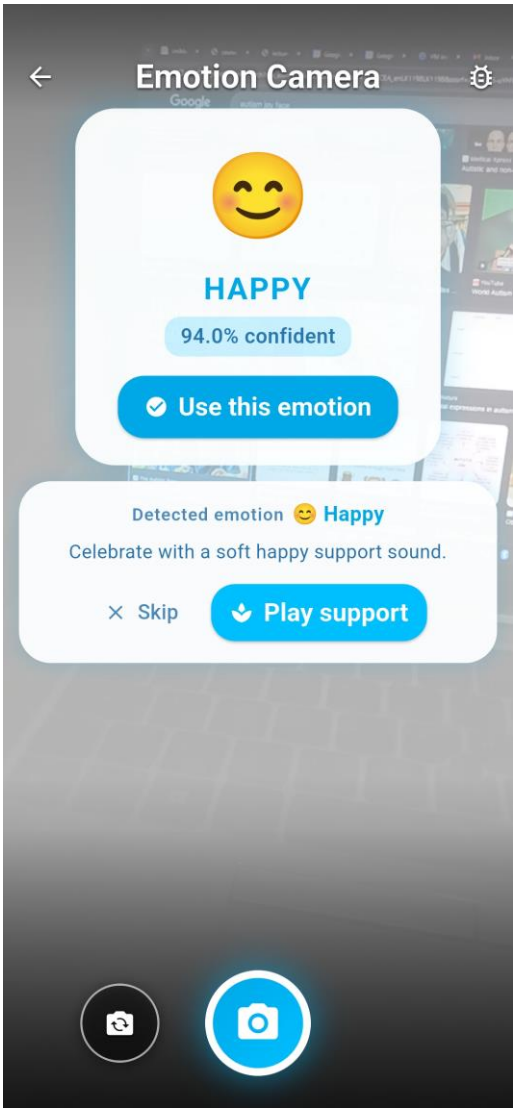


Figure 6.23: Live camera emotion detection screen.

6.5.5 Display mapping

The raw class label coming out of the interpreter is mapped to a friendly display label inside _displayFor(). The mapping (_displayMap) is documented in Section 6.4.3 and never confuses raw Joy with friendly Happy in user-facing strings.

6.5.6 Caregiver-controlled support

For a reliable result the screen offers a Play support button. Pressing it triggers a light haptic via SensoryFeedbackService.triggerLightVibrationIfEnabled(), then plays the corresponding WAV through EmotionSupportAudioService.playForDisplayEmotion(...). The visual animation uses the per-emotion SensorySupportPlan from lib/services/sensory_feedback_service.dart, which includes positivePulse for Happy, calmBreathing for Sad / Angry / Fear / Surprise, and a caregiver-alert ribbon for Fear only. Nothing plays automatically. Stop and Skip buttons are always available while the support is active.

In the submitted build, the emotion result and supportive action remain local to the device. The application does not send the predicted label to a Firebase dashboard and does not ask a reviewer to confirm the result centrally. In a future approved version, a caregiver, therapist or supervised medical student could mark a result as reasonable, uncertain or incorrect after seeing the child and the app output. That feedback could help evaluate practical usefulness and identify where the model needs improvement, but it must not be collected centrally without ethics clearance, parental or guardian consent and hospital or health-sector approval.

6.5.7 Logging and diagnostics

The camera screen and model-load helpers use debugPrint extensively for diagnostics, with consistent prefixes:

- [InitDebug], camera discovery and initialization.
- [ModelLoad], model file attempts, tensor checks and active model.
- [EmotionAudio], supportive audio path resolution and playback errors.

The diagnostic surface deliberately stops at debugPrint. The release build can hide the debug panel by default (_showDebugProbs = false).

6.6 Supportive audio implementation

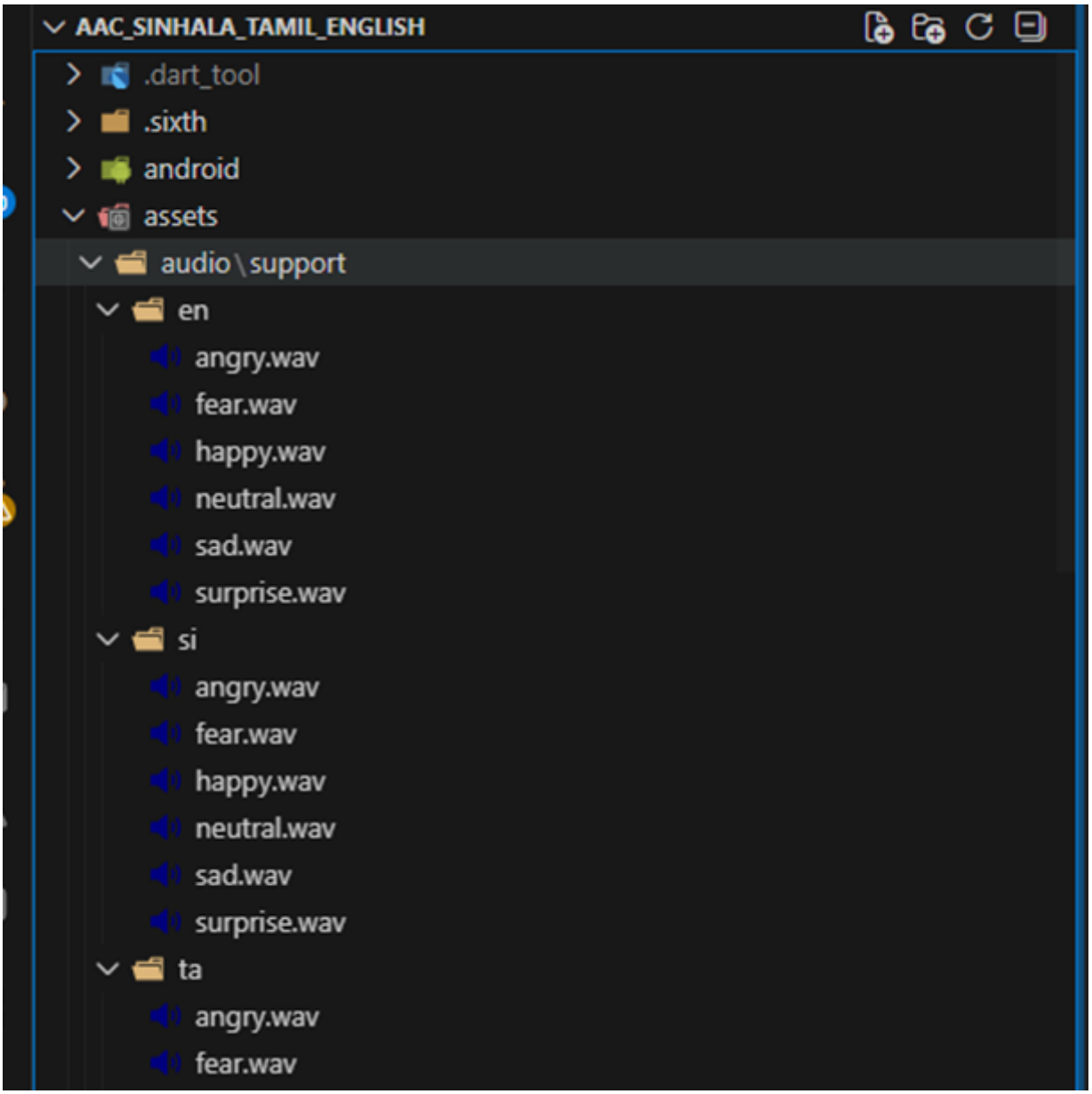


Figure 6.24: Optional support audio file structure.

The service:

- Maps the active language code to a folder: si-LK → si, ta-IN → ta, en-GB → en (with en as the fallback).
- Maps the friendly emotion label to a WAV stem: Happy / Joy → happy, Sad / Sadness → sad, Angry / Anger → angry, Fear → fear, Surprise → surprise, Neutral / Natural → neutral.
- Checks asset presence using rootBundle.load(...). If the preferred-language file is missing, it falls back to the English file. If both are missing, it logs and returns without playing.
- Catches any audioplayers errors with a try/except block and logs them, playback never crashes the screen.

Google AI Studio TTS was used to prepare short supportive voice prompts in Sinhala, Tamil and English. Pronunciation and tone still require human review before wider release. The prompts are stored locally under assets/audio/support/{en,si,ta}/ and are played only when the caregiver activates emotion support.

6.7 Settings, theme and sensory feedback

SensoryFeedbackService (lib/services/sensory_feedback_service.dart) consolidates the few cross-cutting concerns that affect user experience:

- A single vibration_enabled SharedPreferences toggle (default ON, with legacy migration from an older sensory_vibration_enabled key).
- triggerLightVibrationIfEnabled() for AAC cards (HapticFeedback.lightImpact).
- triggerMediumVibrationIfEnabled() for the home / previous / next nav circles.
- A SensorySupportPlan per emotion (Happy, Sad, Angry, Fear, Surprise) used by the camera screen to drive the calm visual animation and the caregiver-alert ribbon.

The earlier multi-toggle sensory card (sound / vibration / animation) was simplified down to one vibration toggle after informal testing showed that caregivers found the three switches confusing.

The application's two visual themes ("girl" and "boy") are implemented in lib/screens/theme/app_theme.dart. They differ only in the primary palette (pink #FF69B4 versus blue #00A8E8). Spacing, sizes and layout are identical. The themes are covered by a small unit test (test/app_theme_test.dart).

6.8 Personal cards and stable emoji rendering

Personal card creation (photo or gallery image plus caregiver voice recording) was prototyped during the interim phase and is currently scoped for re-enable after final-thesis submission. The current build keeps the existing trilingual vocabulary as the primary content and reuses the proven EmojiText widget in every grid cell.

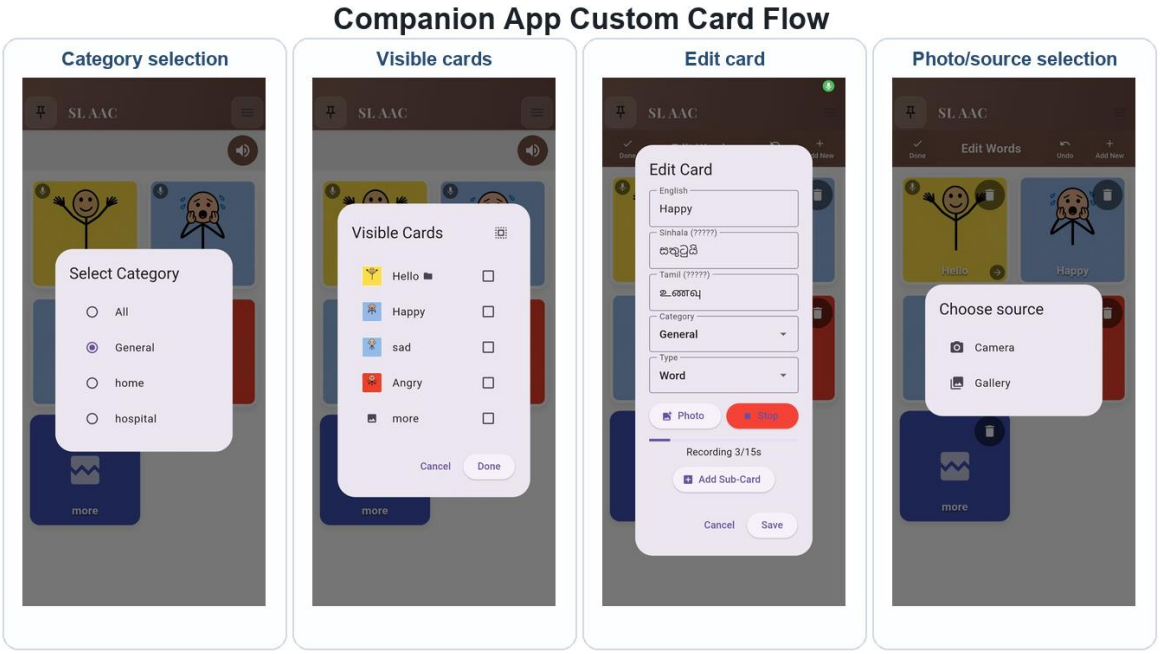


Figure 6.25: Companion app customisation screens.

The figure shows category, visible-card and card-editing customisation screens.

EmojiText (lib/widgets/emoji_text.dart) was the result of fixing a recurring flicker bug with multi-codepoint emojis like family emoji (👩‍👧). The fix uses the bundled NotoColorEmoji font and a memoised emoji string in initState, together with addAutomaticKeepAlives: true on the grid delegate, so the AAC grid remains visually stable during scrolling.

6.9 Android build and release preparation

6.9.1 Build configuration

The Android build configuration is in android/app/build.gradle.kts:

- Application ID lk.aac.sinhala_tamil_english.
- compileSdk / targetSdk / minSdk inherited from Flutter defaults.
- Java 17, Kotlin JVM target 17.
- A release signing config that reads from key.properties (the keystore is intentionally not committed to source control).

AndroidManifest.xml declares only three permissions: CAMERA, INTERNET and ACCESS_NETWORK_STATE. The application label is AAC කථා කරමු (Sinhala for "Let's talk AAC"). The launch theme and Flutter embedding v2 are configured as Flutter standard.

6.9.2 Release APK build evidence

The release APK is produced by flutter build apk --release --no-shrink. The most recent successful build is present in the project tree and is the build referenced throughout this report:

- **Output file:** build/app/outputs/flutter-apk/app-release.apk (also mirrored at build/app/outputs/apk/release/app-release.apk).
- APK size: 126,196,920 bytes (126.2 MB), verified by listing the build folder during the writing of this chapter. The size reflects the included model assets and supportive audio additions.
- **Application ID:** lk.aac.sinhala_tamil_english, confirmed in build/app/outputs/apk/release/output-metadata.json.
- **Version name:** 1.1.0. **Version code:** 2. Confirmed in the same output-metadata.json and matching pubspec.yaml (version: 1.1.0+2).
- **SHA-1 of the release APK:** 43eb07192f0b2eb58393c903d583e79ab1068a67 (from build/app/outputs/flutter-apk/app-release.apk.sha1).
- **Baseline profiles** for both minApi: 28-30 and minApi: 31+ are included in the APK metadata, so the runtime can pick the appropriate compiled profile on the user's device.

- **Native debug symbols** are produced at build/app/outputs/native-debug-symbols/release/native-debug-symbols.zip (31 MB) for symbolicating any future crash reports.

6.9.3 Google Play internal-testing distribution

The Android application was distributed through Google Play internal testing for controlled review and real-device testing. This is consistent with the deployment work documented in the interim report (Fernando, 2026, Section 3.10), where the application was packaged and uploaded to the Google Play Console under the Internal Testing track, an Android App Bundle was generated from the Flutter project, and internal testers installed the application on their own Android devices to verify the AAC grid, navigation and basic settings.

The Play Console listing for this project uses the application ID above, with the resulting store-listing URL pattern https://play.google.com/store/apps/details?id=lk.aac.sinhala_tamil_english (a Play Store listing URL that is reachable to invited internal testers).

The Android application has completed Google Play testing steps, and a production access request was submitted through Google Play Console for the AAC-Sinhala app (package lk.aac.sinhala_tamil_english). The Play Console evidence shows production status as inactive and the production access request as submitted / under review. Public production release is not claimed in this report.

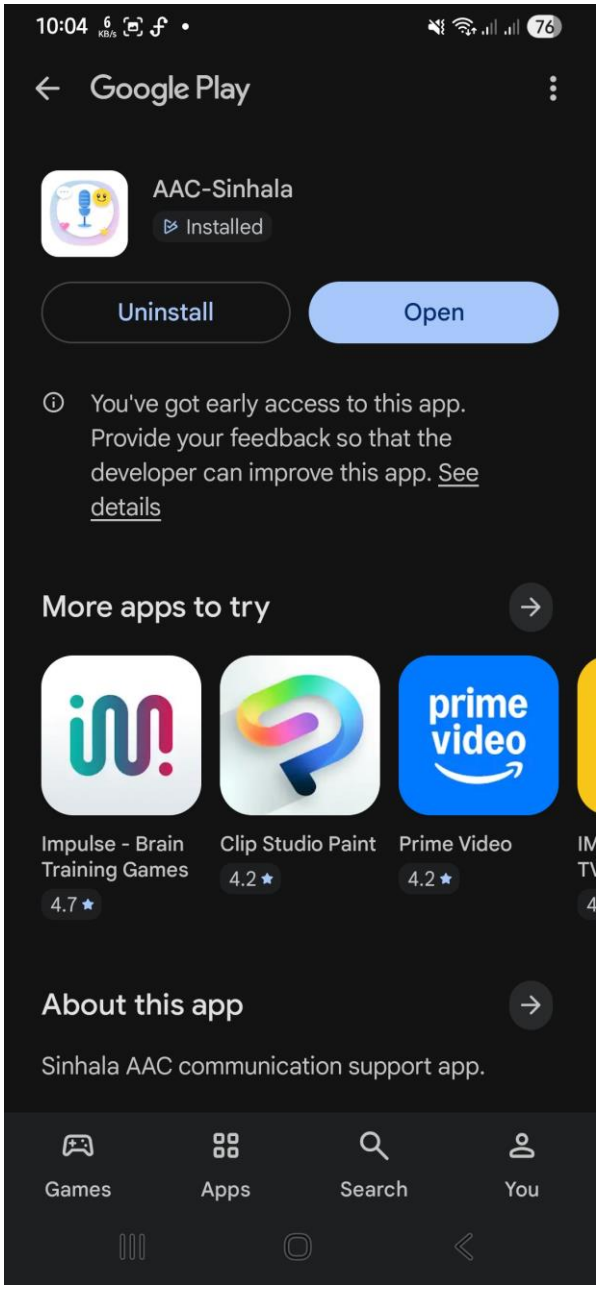


Figure 6.26: Google Play internal testing evidence.

This controlled distribution evidence is supported by the interim report evidence.

The production access request form also recorded the closed-testing and production-readiness work. Testers were recruited through known contacts, including friends, family, healthcare-related contacts and Android device users. They reviewed the core AAC screens, language options, categories, TTS, navigation and camera/emotion support. Feedback was collected informally through direct messages and verbal comments, then used to check or improve

navigation clarity, Sinhala/Tamil/English labels, support-audio behaviour, UI spacing and Android feature flow. This is release-readiness evidence rather than clinical validation.

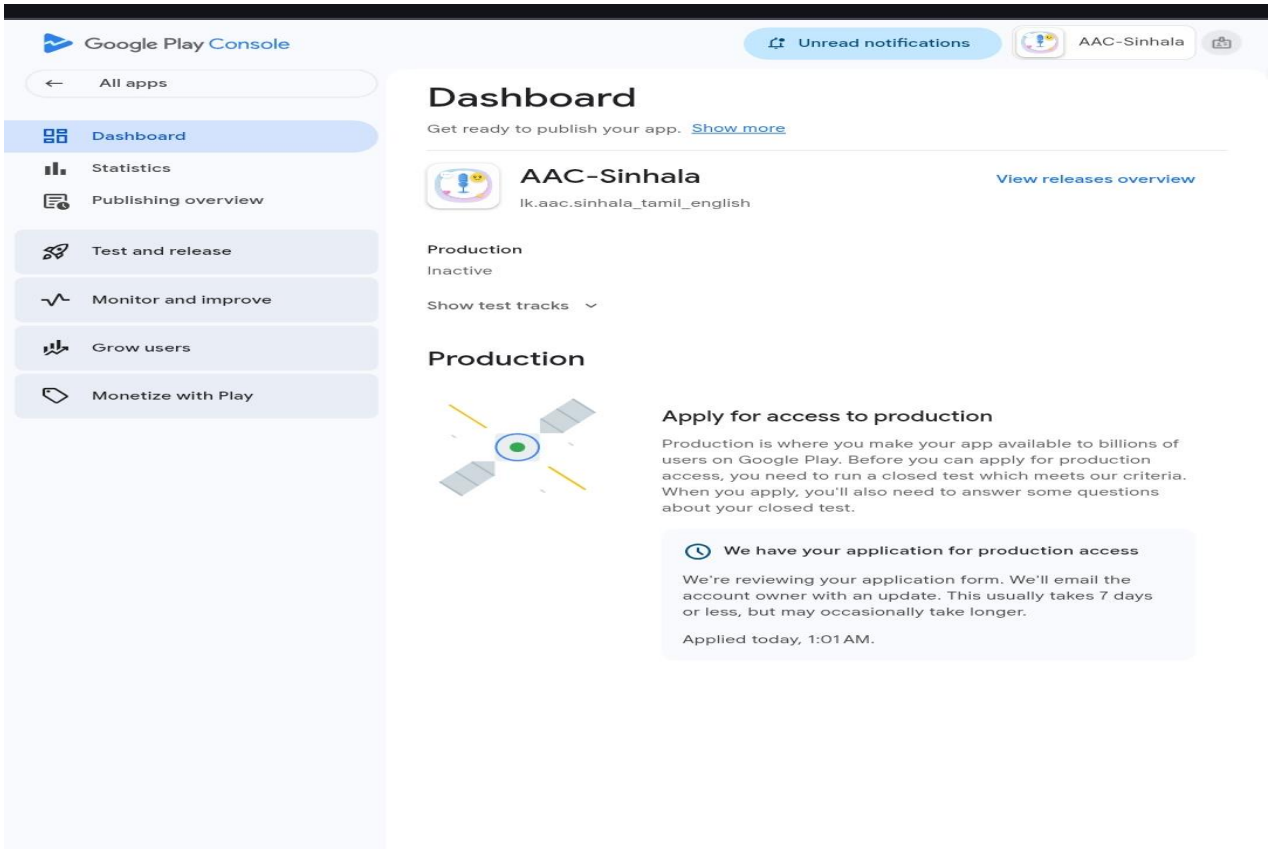


Figure 6.27: Google Play production access request.

The figure shows the production access request submitted for the AAC-Sinhala application.

6.9.4 Mobile release evidence summary

Table 6.13: Mobile release evidence summary.

The table summarises the concrete release evidence currently available and how the thesis uses it.

Evidence item	Source / screenshot	What it confirms	Thesis use
Release APK build output	build/app/outputs/flutter-apk/app-release.apk (126,196,920 bytes) build/app/outputs/apk/release/app-release.apk (same file)	A signed release APK has been produced for the deployed Flutter project	Chapter 6, Section 6.9, Chapter 7, Section 7.7
APK metadata	build/app/outputs/apk/release/output-metadata.json	applicationId: lk.aac.sinhala_tamil_english, versionName: 1.1.0, versionCode: 2, release variant, baseline profiles for API 28-30 and 31+	Chapter 6, Section 6.9.2
APK SHA-1	build/app/outputs/flutter-apk/app-release.apk.sha1 (43eb07192f0b2eb58393c903d583e79ab1068a67)	Cryptographic fingerprint of the exact APK currently in the build tree	Chapter 6, Section 6.9.2
Native debug symbols	build/app/outputs/native-debug-symbols/release/native-debug-symbols.zip (31 MB)	Symbols for symbolicating future crash reports if the APK is uploaded for wider testing	Chapter 6, Section 6.9.2 (supporting)
Google Play internal testing	Interim report Figure 15, inserted as Figure 6.26, Figure 7.4 and Figure G.1.	Internal Testing track exists and was used to distribute	Chapter 6, Section 6.9.3, Chapter 7,

Evidence item	Source / screenshot	What it confirms	Thesis use
		the AAB to invited testers	Section 7.7, Appendix G
Play Store listing URL	https://play.google.com/store/apps/details?id=lk.aac.sinhala_tamil_english (cited from interim report, Section 3.10)	A Play Store listing exists for the application ID, and access depends on the Play Console track configuration.	Chapter 6, Section 6.9.3
Real-device installation	Interim report, Section 3.11 (Figure 16: phone and tablet)	The build can be installed and runs end-to-end on at least two physical Android form factors	Chapter 7, Section 7.6
Google Play production access request	See Figure 6.27 and Figure G.2	Production access request for AAC-Sinhala was submitted through Play Console and is under Google review. Production status is inactive.	Chapter 6, Section 6.9.3, Chapter 7, Section 7.7.3, Appendix G
Public Google Play production release approval	Not available	Production access approval is not available. Public production release is not claimed.	Not used as public-release evidence

6.9.5 iOS pathway

iOS deployment has not yet been undertaken. There is currently no Apple Developer account in place. As recorded in the interim report (Fernando, 2026, Section 3.10), discussions are ongoing with Ideahub (Pvt) Ltd regarding the possibility of providing the developer account and distribution infrastructure for iOS, potentially as part of a charitable initiative following the broader health-sector approval pathway. No App Store release is claimed by this report, and the iOS path is described as planned future work.

6.10 Selected code snippets

The full project source is in the repository on the final-thesis branch. The five short extracts below are the implementation decisions that most directly support the report's claims about reliability, privacy and mobile deployment. Each snippet is shortened, formatted for readability, and accompanied by its file path, purpose and rationale.

6.10.1 Tensor-shape validation (lib/screens/camera_expression_screen.dart)

Purpose. Refuse any .tflite file whose input or output tensor does not match the expected contract before the application keeps the interpreter.

```
static const List<int> _kExpectedInputShape  = [1, 224, 224, 3];
static const List<int> _kExpectedOutputShape = [1, 6];

bool _emotionModelTensorsValid(Tensor input, Tensor output) {
  bool shapeEq(List<int> a, List<int> b) =>
      a.length == b.length &&
      List.generate(a.length, (i) => a[i] == b[i]).every((ok) => ok);
  return shapeEq(input.shape,  _kExpectedInputShape)  &&
         shapeEq(output.shape, _kExpectedOutputShape) &&
         input.type  == TensorType.float32 &&
         output.type == TensorType.float32;
}
```

Why it matters. The check protects the application against a future model export that silently changes its tensor shape or data type. Combined with the primary/fallback loader (next snippet), it means an incompatible .tflite is rejected with a clean error message rather than producing a crash or, worse, wrong inference.

6.10.2 Primary / fallback model loading (lib/screens/camera_expression_screen.dart)

Purpose. Try the EfficientNetB0 fine-tuned model first. If it fails to load or fails tensor validation, the system falls back automatically to MobileNetV2. The active model name is recorded so the debug screen can show it.

```
Future<_EmotionModelSlotResult> _tryLoadEmotionModelSlot({ required String
assetPath, required String slotLogName, required String
activeEmotionModelDebugName,
}) async { Interpreter interp; try { interp = await
Interpreter.fromAsset(assetPath); } catch (e) { debugPrint('[ModelLoad]
$slotLogName model failed: $e'); return _EmotionModelSlotResult.loadFailed; }
final inDetails = interp.getInputTensor(0); final outDetails =
interp.getOutputTensor(0); if (!_emotionModelTensorsValid(inDetails,
outDetails)) { interp.close(); return _EmotionModelSlotResult.tensorFailed; }
_interpreter = interp; _activeEmotionModelName = activeEmotionModelDebugName;
debugPrint('[ModelLoad] Active model: $assetPath'); return
_EmotionModelSlotResult.ok;
}
```

Why it matters. In the default Auto mode the loader calls this helper for emotion_efficientnetb0_finetuned.tflite first and for emotion_mobilenetv2_fallback.tflite second. A failure in one slot does not break the camera feature on the demonstration device. The [ModelLoad] log lines also give a viva examiner a one-glance way to confirm which model the running APK is actually using.

6.10.3 Preprocessing to [-1, +1] (lib/screens/camera_expression_screen.dart)

Purpose. Convert each cropped 224×224 RGB frame into the exact tensor the trained model was supplied with during training.

```
List<List<List<List<double>>>> _buildModelInput(img.Image resized) {
  return List.generate(1, (_) => List.generate(224, (y) =>
      List.generate(224, (x) {
        final p = resized.getPixel(x, y);
        return [
          (p.r.toDouble() / 127.5) - 1.0,
          (p.g.toDouble() / 127.5) - 1.0,
          (p.b.toDouble() / 127.5) - 1.0,
        ];
      })));
}
```

Why it matters. The same (pixel / 127.5) − 1.0 formula is used in both the Colab training pipeline and the Flutter run-time. Training-runtime parity is the single most important guard against a silent loss of accuracy that originally affected the EfficientNetB0 experiments and was solved by adding an internal [-1, +1] → [0, 255] rescaling layer at the head of the model graph (see Section 6.4.7).

6.10.4 Confidence and top-1 / top-2 margin gate (lib/screens/camera_expression_screen.dart)

Purpose. Translate a softmax vector into one of four explicit states, No face, Face not clear, Uncertain or Reliable, rather than forcing every frame into a label.

```
static const double _kMinConfidence = 0.40; // debug value, tune after
evidence
static const double _kMinTopMargin  = 0.08; // debug value, tune after
evidence

final mean    = _meanVector(softmaxAccum);   // average of 5 captures
final top1Idx = _argmax(mean);
final top1    = mean[top1Idx];
final top2    = mean[_argmaxExcept(mean, top1Idx)];
final margin  = top1 - top2;

if (top1 < _kMinConfidence || margin < _kMinTopMargin) {
  setState(() => _status = _CaptureStatus.uncertain);
  return;
}
setState(() {
  _status      = _CaptureStatus.reliable;
  _resultLabel = _labels[top1Idx];
  _resultScore = top1;
});
```

Why it matters. The margin check catches ambiguous predictions (for example sad versus neutral with very similar probabilities) that a naive argmax could incorrectly turn into a confident label. The current 0.40 / 0.08 values are deliberately loose for debugging. The implementation note records that the production range should be tightened to roughly 0.50-0.65 once real-device evidence is available. This is the central code-level safeguard that lets the AI component be described as supportive observation, not diagnosis.

6.10.5 Trilingual supportive audio with fallback (lib/services/emotion_support_audio_service.dart)

Purpose. Play a short, language-specific WAV clip after a reliable emotion is detected. The service falls back to English when the preferred-language clip is missing and skips safely if no clip is available.

```
static Future<void> playForDisplayEmotion( String emotionLabel, String
languageCode,
) async { final stem = _stemForEmotion(emotionLabel); // happy / sad / angry /
fear / surprise / neutral if (stem == null) return; final preferred =
_preferredLangFolder(languageCode); // si-LK → si, ta-IN → ta, default → en
final primaryRel = _assetSourcePath(preferred, stem); final fallbackRel =
_assetSourcePath(_fallbackLang, stem); String playRel; if (await
_bundleHasAsset('assets/$primaryRel')) { playRel = primaryRel; } else if
(await _bundleHasAsset('assets/$fallbackRel')) { playRel = fallbackRel; } else
{ return; // no clip available, silently skip } try { await _player.stop();
await _player.play(AssetSource(playRel)); } catch (e) {
debugPrint('[EmotionAudio] Failed to play asset: $playRel error: $e'); }
}
```

Why it matters. The supportive prompt is intentionally not mandatory. An absent or corrupted WAV file must not cause application instability, and an unsupported language must downgrade quietly to English. The same call sequence is invoked only from a caregiver-initiated Play support tap, so audio is only played after caregiver action.

6.11 Implementation summary

The implementation work covered the full path from a public Kaggle dataset audit to a real Android APK distributed through Google Play internal testing. The most important engineering decisions were:

- **Architecture-first model choice.** Compatibility with tflite_flutter and the absence of Flex operations were treated as hard constraints, on the same level as validation accuracy.
- **Primary / fallback model strategy. Two compact .tflite models ship inside the APK. If the primary model fails, the fallback model takes over automatically. The active model is observable in the debug panel.**
- **Preprocessing parity. The same [-1, +1] normalisation is used in training and at run-time. EfficientNetB0 was patched at the graph level so the Flutter pipeline did not have to change.**

- **Strict, multi-state result handling.** No face, Face not clear, Uncertain and Reliable are explicit states. The application never forces an emotion label onto an unclear frame.
- **Caregiver-controlled supportive feedback.** No sound or motion auto-plays on a detection. The caregiver must press Play support first, and Stop is always available.
- **Strong offline behaviour. All AAC content is compiled into the application. The only network capability declared in the Android manifest is for connectivity detection.**

Chapter 7 documents how the system was tested and verified. Chapter 8 evaluates it against the objectives stated in Chapter 1.

Chapter 7: Testing and Verification

This chapter describes how the system was tested and verified, what was found, and what remains for future validation and deployment. The chapter follows a standard test-plan layout (strategy, categories, test cases and results), presenting what has been verified and what still requires future validation. Evidence that is not available is described as a limitation rather than replaced with unsupported material.

7.1 Testing strategy

Testing is split into five categories that map onto the architecture in Chapter 5:

1. **Unit tests** for small services and widgets (test/).
2. **Functional UI testing** on emulator and real device, manual scripted walkthroughs.
3. **Trilingual and language-fallback testing** for Sinhala, Tamil and English TTS and supportive audio.
4. **AI / model verification**. TFLite tensor checks, primary / fallback load order, face-gate behaviour, confidence and margin thresholds.
5. **Release / distribution testing**. Release APK build, Google Play internal testing channel.

A formal user-study or clinical pilot is **not** part of the current testing scope. It is described in Chapter 8 as planned future work, only within ethically cleared scope.

Table 7.1: Test plan summary.

The table summarises the test categories and current status.

Category	What is tested	How	Current status
Unit tests	Storage, theme, emoji widget and emotion-support helper logic	Flutter test framework	Implemented, 41 automated tests passing locally
UI walkthrough (main Smart AAC)	Splash, registration, home, category, favourites,	Manual scripted walkthroughs	Verified via manual device testing

Category	What is tested	How	Current status
	settings, camera		
UI walkthrough (simple_aac_app)	Customisable cards, sub-cards, edit mode	Manual scripted walkthroughs	Performed during development. Separate screenshot evidence was not captured for this individual scripted step, but the behaviour was verified during the manual walkthrough.
Trilingual	Language switch in Sinhala / Tamil / English in both apps	Manual, emulator + device	Performed during development
Offline	No-internet behaviour in both apps	Manual, airplane mode	Performed during development
Backup / restore / share (simple_aac_app)	Build ZIP, share via WhatsApp / Downloads, restore from a picked ZIP	Manual on real device	Verified via manual device testing
TFLite verification	Tensor shape, dtype, no Flex, dummy inference	Colab + run-time [ModelLoad] logs	Performed. Evidence is shown in Figure 6.20 and Figure 6.21.
Primary / fallback load	Auto vs primaryOnly vs fallbackOnly	Debug menu in CameraExpressionScreen	Performed

Category	What is tested	How	Current status
Face-gate	No-face, face-not-clear, reliable, uncertain	Manual, real device	Performed
Support audio	Trilingual playback, missing-file fallback	Manual	Performed. Some Sinhala/Tamil clips still require final human pronunciation review or replacement before wider release.
Release APK	flutter build apk --release --no-shrink	Local build	Built successfully, 126,196,920 bytes (126.2 MB), applicationId = lk.aac.sinhala_tamil_english, versionName = 1.1.0, versionCode = 2 (verified from build/app/outputs/apk/release/output-metadata.json)
Play internal testing	Install on a tester device through Play Store	Google Play Console	Distributed for internal testing per interim report, Section 3.10. Public production release is not claimed.

7.2 Unit tests

Unit testing was used to check deterministic application logic before relying on manual UI walkthroughs and real-device testing. The most important targets were the safety-related parts of the emotion-support feature, model fallback decisions, tensor-shape checks, local storage/preferences and basic UI rendering. These areas can be tested repeatably because they do not require a live camera, a native ML Kit detector, a TensorFlow Lite interpreter session or audio playback.

The tests were run from the repository root on the current final-thesis branch using the Flutter test framework. The command used for the fresh verification was flutter test. The automated tests are stored directly under test/; no separate test/unit/ folder is present. The unit-test-final branch was also checked during this review and contains the same unit-test files, while the

saved output file documentation/submission/unit-test-final-output.txt remains available as supporting evidence.

The fresh Flutter test run completed successfully and reported 41 passing tests out of 41, with no failed tests. Figure 7.1 records the unit-test output used as report evidence.

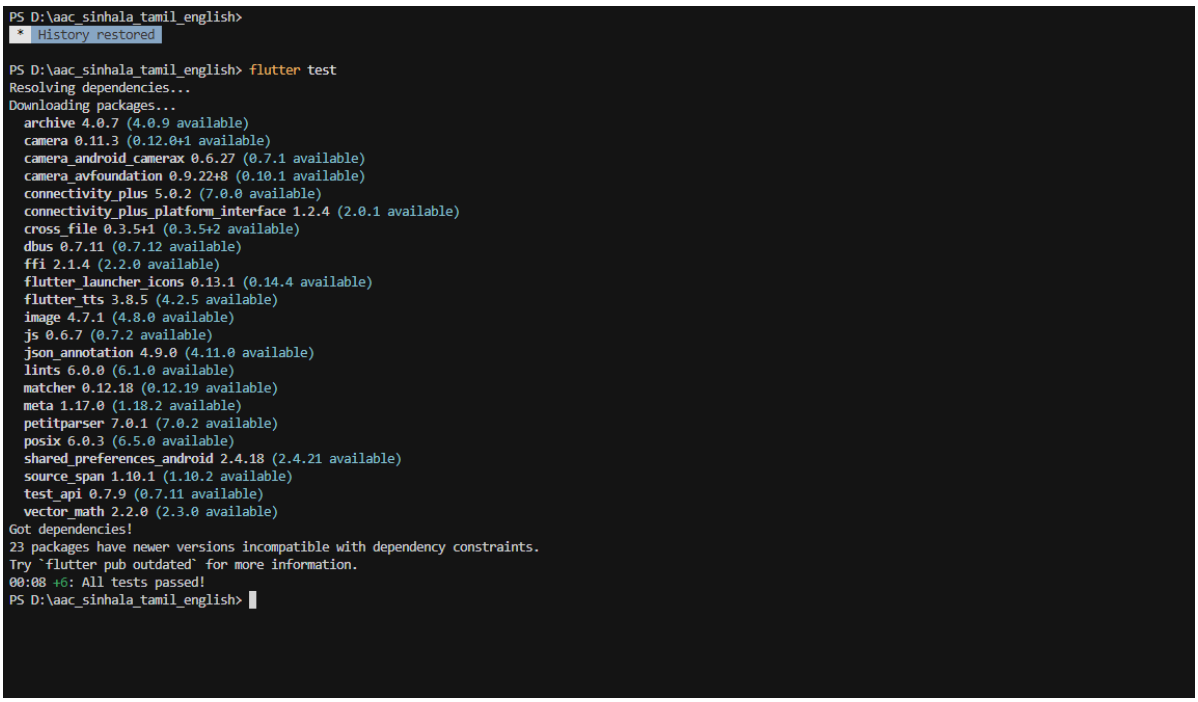


Figure 7.1: Flutter unit test output.

The original unit and widget tests were retained for emoji rendering, SharedPreferences-backed profile storage and theme selection. These tests matter because small regressions in EmojiText, StorageService or AppTheme would be immediately visible to a child or caregiver through broken card rendering, lost registration state or incorrect colour-theme behaviour.

The emotion-decision tests focus on the deterministic safety checks used after the model produces softmax scores. EmotionDecisionHelper is tested for low-confidence predictions, close top-1/top-2 margins, reliable predictions and empty score lists. These tests matter because the camera/emotion feature must avoid forcing an emotion label when confidence is weak or when the two highest predictions are too close to separate safely.

The model-loading and tensor-shape tests check logic that supports the primary EfficientNetB0 model and MobileNetV2 fallback path without executing the native TensorFlow Lite runtime. EmotionModelLoadResolver is tested for Auto, primary-only and fallback-only behaviour, while TensorShapeValidator checks that the expected input shape [1, 224, 224, 3], output shape [1, 6] and float32 flags are accepted and invalid cases fail safely. EmotionDisplayMapper is also tested so that raw labels such as Joy and Natural are shown as Happy and Neutral in the user interface.

The local support and preference tests check behaviours that affect safe use of the final application. EmotionAudioPaths verifies language-folder selection and English fallback for unsupported language codes. SensoryFeedbackService tests cover migration from the older sensory_vibration_enabled key to vibration_enabled, and StorageService coexistence tests check that saving profile data does not remove favourite categories. CaptureFrameGate tests confirm that no-face and unclear-frame patterns return safe non-diagnostic states rather than a forced emotion result.

Some behaviour was deliberately kept outside pure unit testing. Real camera access, ML Kit face detection, actual TensorFlow Lite model execution and audio playback depend on Android hardware or native plugins, so they are not reliable targets for ordinary Flutter unit tests. Those areas are covered separately through real-device testing, AI/model verification and functional UI testing rather than being overstated as unit-test evidence.

Table 7.2A summarises the main unit-test areas, the project paths used as evidence and the result recorded from the final automated run.

Table 7.2A: Unit-test result summary.

Test focus	Example code/path	Why it was tested	Result
Emoji/widget rendering	test/widget_test.dart	Confirms EmojiText uses NotoColorEmoji so emoji/card text rendering remains stable.	Passed
Storage service defaults and persistence	test/storage_service_test.dart	Confirms registration/profile values are saved and loaded correctly.	Passed
Theme selection	test/app_theme_test.dart	Confirms gender-based colour theme selection remains stable.	Passed
Emotion decision gating	test/emotion_decision_helper_test.dart	Confirms low-confidence predictions return Uncertain.	Passed
Top-1/top-2 margin handling	test/emotion_decision_helper_test.dart	Confirms close predictions are	Passed

		treated as uncertain.	
Model loading/fallback resolver	test/emotion_model_load_resolver_test.dart	Confirms Auto, primary-only and fallback-only selection behaviour.	Passed
Tensor-shape validation	test/tensor_shape_validator_test.dart	Confirms expected TFLite input/output shapes are accepted and invalid shapes are rejected.	Passed
Display label mapping	test/emotion_display_mapping_test.dart	Confirms raw labels such as Joy/Natural map to Happy/Neutral.	Passed
Support-audio path fallback	test/emotion_audio_paths_test.dart	Confirms unsupported language codes safely fall back to English.	Passed
Storage/vibration migration	test/sensory_feedback_storage_upgrade_test.dart	Confirms legacy vibration settings migrate safely and existing favourites are not removed by profile saving.	Passed
Capture-frame safe-state logic	test/capture_frame_gate_test.dart	Confirms no-face/unclear frame patterns return safe states.	Passed

Representative unit-test code excerpts are included below in this section so that the unit-testing evidence is visible in the main testing chapter.

Representative unit-test code evidence

Emotion decision gating test

File path: test/emotion_decision_helper_test.dart

Purpose: Checks that a low-confidence emotion prediction is treated as uncertain rather than being forced into a user-facing label.

```
test('returns uncertain when top confidence is below minimum', () {
  final r = EmotionDecisionHelper.evaluate([
    0.35, 0.30, 0.10, 0.10, 0.10, 0.05,
  ]);
  expect(r.isUncertain, isTrue);
  expect(r.top1Index, 0);
});
```

Result: Passed in the final flutter test run.

Top-1/top-2 margin test

File path: test/emotion_decision_helper_test.dart

Purpose: Checks that predictions with a small difference between the highest and second-highest scores are treated as uncertain.

```
test('returns uncertain when top1-top2 margin is below minimum', () {
  final r = EmotionDecisionHelper.evaluate([
    0.50, 0.46, 0.01, 0.01, 0.01, 0.01,
  ]);
  expect(r.isUncertain, isTrue);
  expect(r.margin, closeTo(0.04, 0.001));
});
```

Result: Passed in the final flutter test run.

Primary/fallback model selection test

File path: test/emotion_model_load_resolver_test.dart

```
Purpose: Checks that automatic model loading falls back to the MobileNetV2 slot when the
primary EfficientNetB0 slot fails.
test('auto falls back when primary fails', () {
  final r = EmotionModelLoadResolver.resolve(
    mode: EmotionModelLoadMode.auto,
    primaryResult: EmotionModelSlotResult.loadFailed,
    fallbackResult: EmotionModelSlotResult.ok,
  );
  expect(r.success, isTrue);
  expect(r.pick, EmotionModelLoadPick.fallback);
});
```

Result: Passed in the final flutter test run.

Tensor-shape validation test

File path: test/tensor_shape_validator_test.dart

Purpose: Checks that the expected TensorFlow Lite input and output shapes are accepted before model use.

```
test('valid emotion model shapes pass', () {
  expect(
    TensorShapeValidator.areShapesValid(
      inputShape: [1, 224, 224, 3],
      outputShape: [1, 6],
    ),
    isTrue,
  );
});
```

Result: Passed in the final flutter test run.

Support-audio path fallback test

File path: test/emotion_audio_paths_test.dart

Purpose: Checks that unsupported language codes fall back safely to the English support-audio folder.

```
test('unsupported language code uses English folder', () {
  expect(
    EmotionAudioPaths.preferredLangFolder('de-DE'),
    'en',
  );
});
```

Result: Passed in the final flutter test run.

Capture-frame safe-state test

File path: test/capture_frame_gate_test.dart

Purpose: Checks that the camera decision helper returns a safe non-diagnostic state when no usable face is detected.

```
test('unsupported language code uses English folder', () {
  expect(
    EmotionAudioPaths.preferredLangFolder('de-DE'),
    'en',
  );
```

```
});
```

Result: Passed in the final flutter test run.

These excerpts are representative only and do not reproduce the full test files. Overall, the 41/41 result does not prove clinical accuracy or full application correctness. It does, however, give confidence that the deterministic safety logic and selected app services behave as expected before the manual device tests. Figure 7.1 provides the test-output evidence and Table 7.2A summarises the tested areas.

7.3 Functional UI testing

The functional walkthrough covers the screens listed below. Each step was executed manually during development and is summarised as supporting test evidence in this chapter.

Functional UI testing was carried out as a scripted manual walkthrough on emulator and real Android device. Each test checked whether the expected screen opened, whether the main action could be completed, and whether the result matched the expected behaviour. Screenshot evidence is included to support the passed functional tests, while Appendix F keeps the broader application screenshot archive.

Additional application screenshots are retained in Appendix F, which archives the main registration, home, category, favourites, settings, trilingual UI, active model and camera/emotion screens used as supporting evidence.

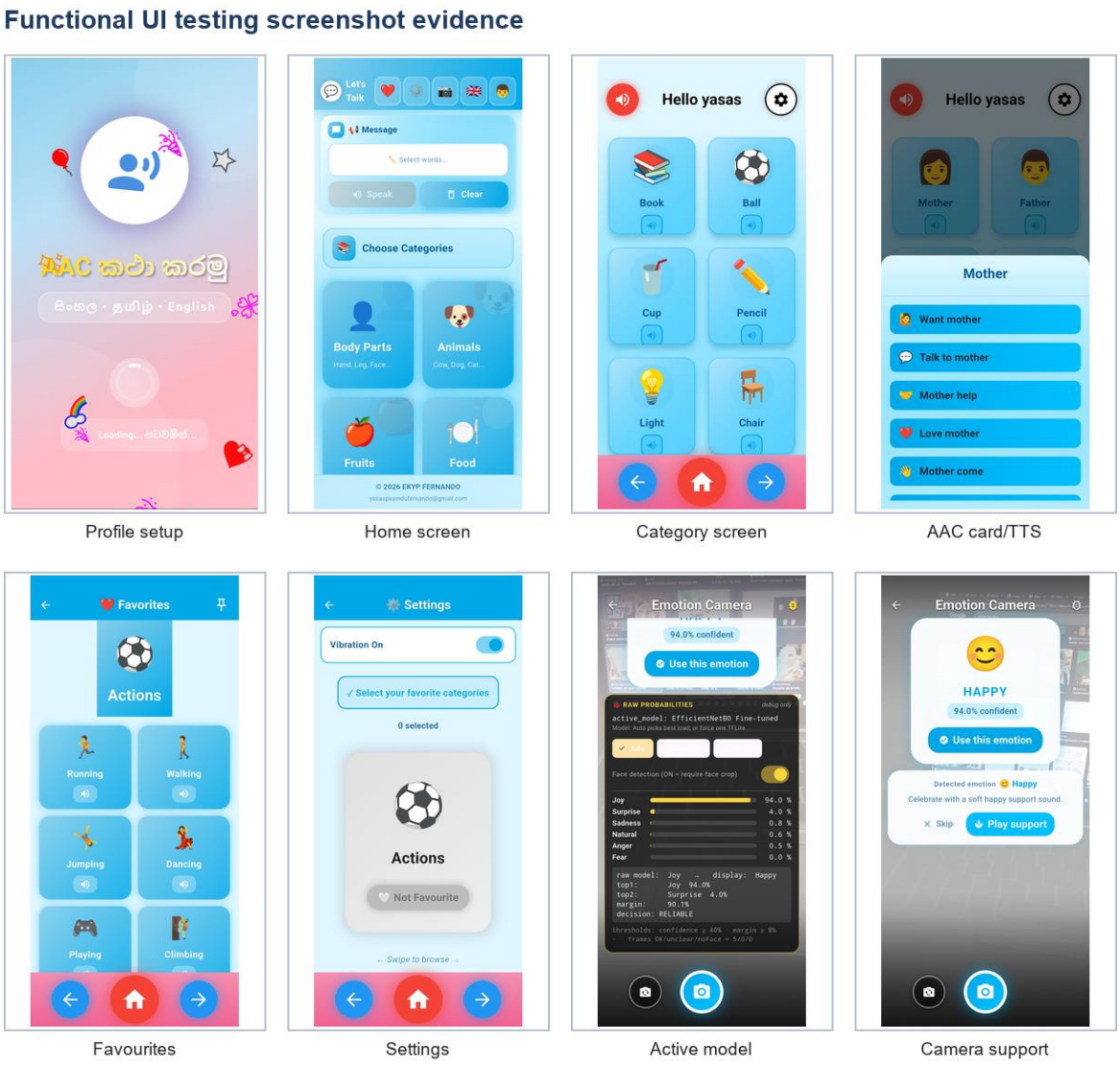


Figure 7.2: Functional UI testing screenshot evidence.

Figure 7.2 shows representative functional UI evidence from the manual testing walkthrough. The screenshots cover the main user-facing flows tested during development, including registration, home/category navigation, card selection, favourites/settings access and the supervised camera/emotion-support screen where available. These screenshots support the functional test cases by showing that the tested screens were opened and checked rather than only described in text.

The functional tests are presented as scripted walkthroughs rather than as a formal user study. For each key screen or feature, the test cases record the action, expected result, actual result, status and available evidence. Where a separate screenshot was not captured, the table keeps that boundary visible rather than inventing evidence.

Table 7.2: Test cases. Core AAC.

Test ID	Area	Steps	Expected result	Actual result	Status
UI-01	Splash → Registration	Launch the application	Splash shown, then Registration screen	Splash and Registration screens shown	Pass
UI-02	Registration form	Enter name, pick gender and language, tap *Let's Start*	App moves to Home screen and user data persists.	Home screen reached and profile state persisted.	Pass
UI-03	Home grid	Scroll through category cards	Cards render with emoji + trilingual labels, no flicker	Cards render without flicker (after EmojiText fix)	Pass
UI-04	TTS in Sinhala	Tap a vocabulary card in si-LK	Sinhala TTS is spoken (where device voice is installed)	Spoken on device with Sinhala voice installed	Pass
UI-05	TTS in Tamil	Tap a vocabulary card in ta-IN	Tamil TTS is spoken (where device voice is installed)	Spoken on device with Tamil voice installed	Pass
UI-06	TTS in English	Tap a vocabulary card in en-GB	English TTS is spoken	Spoken	Pass
UI-07	Favourite category	Add a category to favourites	Favourites carousel shows the category and	Persisted via SharedPreferences	Pass

Test ID	Area	Steps	Expected result	Actual result	Status
			persists after restart.		
UI-08	Settings, vibration	Toggle Vibration *On* in settings	Light haptic feedback stops when switched off, and medium haptic feedback on navigation circles also stops.	Behaviour matches SensoryFeedbackService cache	Pass
UI-09	Settings, language	Change language	Vocabulary labels and TTS switch language	Switch observed	Pass
UI-10	Offline mode	Enable Airplane mode and use the app	Core AAC functions work normally	Confirmed. OfflineService only reports status	Pass
UI-11	simple_aac_app: create a custom card	In simple_aac_app, open edit mode, add a card with label, Sinhala label, Tamil label, image and recorded audio	Card appears in the grid and persists after restart.	Card persisted via StorageService.saveCards to SharedPreferences	Pass
UI-12	simple_aac_app: sub-card navigation	Tap a category card with sub-cards	Sub-grid opens and back navigation returns to the parent grid	Behaviour verified manually	Pass

Test ID	Area	Steps	Expected result	Actual result	Status
UI-13	simple_aac_app: manual backup (share)	Open the drawer, choose Backup → Share	ZIP archive is generated and the share sheet appears with WhatsApp / email / file manager options	_buildBackupZip and Share.shareXFiles used. ZIP includes the JSON manifest and all attached images/audio	Pass
UI-14	simple_aac_app: backup to Downloads	Open the drawer, choose Backup → Downloads	ZIP is saved to the device's Downloads folder, or falls back to the app documents folder if the system denies Downloads access	Verified through the backup-to-downloads flow and platform-channel save operation	Pass
UI-15	simple_aac_app: restore from picked ZIP	Open the drawer, choose Restore → Pick File, select a previously shared backup ZIP	Cards, images and audio are restored, and previously customised vocabulary reappears.	Verified via _restoreFromPickedFile and ZIP extraction to a temporary directory	Pass
UI-16	Static AAC flow (main Smart AAC, Level 3+)	Navigate the home → categories → words → TTS path	Category list is stable across launches, and word order matches the compiled	Behaviour verified manually	Pass

Test ID	Area	Steps	Expected result	Actual result	Status
			lib/data/word_data/ content.		
UI-17	Customisable AAC flow (simple_aac_app, Level 1-2)	Add a card, reorder via edit mode, restart the app	Card and order persist, and the vocabulary remains caregiver-customisable.	Behaviour verified manually	Pass

7.4 Trilingual and supportive-audio testing

The trilingual test set covers TTS output, on-screen labels and the supportive WAV prompts on the camera screen.

- TTS coverage. Sinhala (si-LK) and Tamil (ta-IN) TTS depends on the device having the corresponding voice installed. Where the voice is not installed, flutter_tts returns silently rather than synthesising mismatched audio. This behaviour was checked during the real-device walkthroughs described in Section 7.6.
- **Label coverage.** Each of the 16 vocabulary categories has Sinhala, Tamil and English text for every entry. Random spot-checks during development did not reveal missing strings.
- Supportive audio coverage. Six emotion prompt stems are packaged for the supported languages and are exercised through the fallback path. Some Sinhala and Tamil clips still require final human review or replacement before wider release.
- **Fallback behaviour. Tested by renaming a preferred-language WAV and observing [EmotionAudio] log lines. The service falls back to English without crashing.**

7.5 AI / model verification

The model verification test set covers tensor compatibility, primary / fallback load order and decision-making behaviour.

Table 7.3: Test cases. Emotion detection.

Test ID	Area	Steps	Expected result	Actual result	Status	Evidence
AI-01	TFLite, MobileNetV2	Load emotion_mobilenetv2_fallback.tflite in a Python TFLite interpreter and run a dummy inference	Input [1,224,224,3] float32, output [1,6] float32, no Flex operations, dummy inference succeeds	Verified in the Colab notebook	Pass	Figure 6.20.
AI-02	TFLite, EfficientNetB0	Same checks for emotion_efficientnetb0_finetuned.tflite	Same expected result	Verified in the Colab notebook	Pass	Figure 6.21.
AI-03	Primary / fallback load order, Auto	Set debug load mode to Auto and check [ModelLoad] lines.	Primary loads OK and is kept	Verified on device by log inspection of [ModelLoad] active-model output	Pass	Figure 6.22 and device log inspection.
AI-04	Force primary only	Set debug load mode to EfficientNet only and reload.	Primary loads OK	Verified	Pass	Verified via device logs
AI-05	Force fallback only	Set debug load mode to	Fallback loads OK	Verified	Pass	Device log inspection.

Test ID	Area	Steps	Expected result	Actual result	Status	Evidence
		MobileNet only and reload.				Separate screenshot evidence was not captured for this individual step, but the behaviour was verified during the manual walkthrough.
AI-06	Tensor failure path	Replace a .tflite with an invalid file in the assets temporarily and load	Slot rejects the model and falls back. If both models fail, the user sees Emotion model shape is not supported.	Code path validated against _tryLoadEmotionModelSlot	Pass (code path)	Code path reviewed in _tryLoadEmotionModelSlot. Separate screenshot evidence was not captured for this

Test ID	Area	Steps	Expected result	Actual result	Status	Evidence
						individual step, but the behaviour was verified during the manual walkthrough.
AI-07	No-face state	Cover the camera	Screen shows *No face detected*	Verified	Pass	Verified via manual inspection
AI-08	Face-not-clear state	Hold the camera too far away (face short-side ratio below 0.18)	Screen shows *Face not clear*	Verified	Pass	Manual real-device check. Separate screenshot evidence was not captured for this individual step, but the behaviour was

Test ID	Area	Steps	Expected result	Actual result	Status	Evidence
						verified during the manual walkthrough.
AI-09	Uncertain state	Tilt the face / partially obscured	Screen shows Uncertain, try again	Verified	Pass	Manual real-device check. Separate screenshot evidence was not captured for this individual step, but the behaviour was verified during the manual walkthrough.
AI-10	Reliable state	Front-facing well-lit face	Screen shows a confident emotion + emoji	Verified	Pass	Figure 6.23 and manual

Test ID	Area	Steps	Expected result	Actual result	Status	Evidence
						real-device check.
AI-11	Face gate OFF	Toggle the debug face-gate off, then capture	Centre-cropped 224 × 224 path is used	Verified	Pass	Debug panel inspection. Separate screenshot evidence was not captured for this individual step, but the behaviour was verified during the manual walkthroug h.
AI-12	Support audio play	Press Play support on a reliable emotion	Haptic, then WAV in active language	Verified	Pass	Manual support-audio check. Separate screenshot

Test ID	Area	Steps	Expected result	Actual result	Status	Evidence
						evidence was not captured for this individual step, but the behaviour was verified during the manual walkthroug h.
AI-13	Support audio missing	Rename the preferred WAV temporarily	Fallback to English WAV occurs, an [EmotionAudi o] log line appears, and no crash occurs.	Verified	Pass	Manual support-audio check. Separate screenshot evidence was not captured for this individual step, but the

Test ID	Area	Steps	Expected result	Actual result	Status	Evidence
						behaviour was verified during the manual walkthrough.
AI-14	Model evaluation evidence	Review saved evaluation outputs, confusion matrices and TensorFlow Lite verification evidence	Model evidence reported without unsupported accuracy claims	Saved learning curves, confusion matrices and TensorFlow Lite verification outputs are included in Chapter 6.	Evidence included	Figures 6.13 and 6.19 provide confusion-matrix evidence. Figures 6.20 and 6.21 provide TensorFlow Lite verification evidence. Standalone accuracy values are not claimed without

Test ID	Area	Steps	Expected result	Actual result	Status	Evidence
						confirmed output.
AI-15	Confusion matrix	Generate per-class confusion matrix for both models	Per-class recall reported	Confusion matrices inserted for both models.	Evidence inserted	Figure 6.13 and Figure 6.19.

The current debug values of _kMinConfidence = 0.40 and _kMinTopMargin = 0.08 are intentional. The implementation marks these values for tightening before final submission, with a recommended production range of 0.50-0.65 for confidence. Final tuned values should be recorded only after final real-device measurement.

7.6 Real device testing

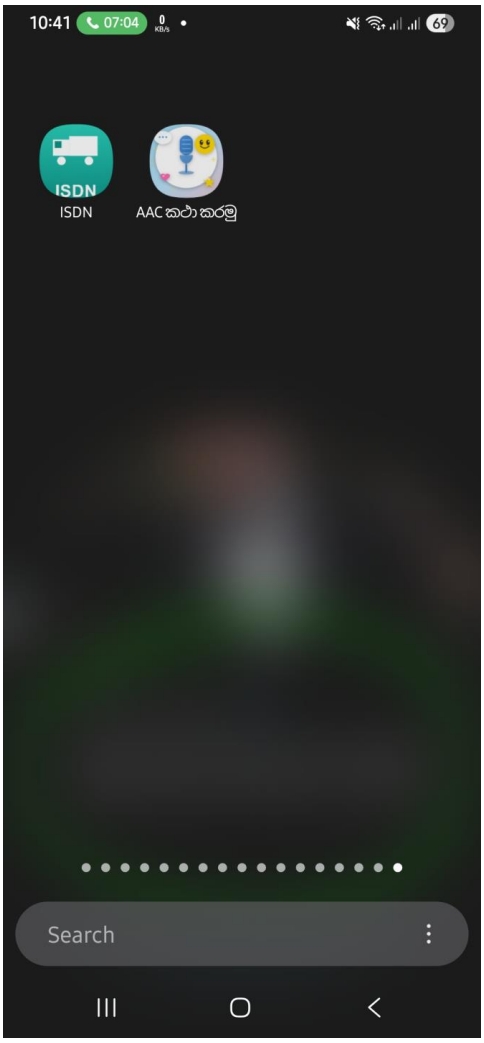


Figure 7.3: Real-device testing evidence.

The figure shows the AAC application installed on a physical Android device during informal testing.

Real-device testing was used as the primary feedback loop, because emulators do not faithfully reproduce camera, TFLite or TTS behaviour. Each successful Flutter build of the relevant sprint was installed on at least one Android phone, then exercised against the same scripted walkthrough listed in Section 7.3 and Section 7.5. The interim report (Fernando, 2026, Section 3.11) records that the application was also tested on a second physical Android device of a different form factor, a tablet, to check key interface elements stayed readable and usable across screen sizes without layout breakage. Both devices ran the build through to the AAC grid, TTS output and the on-device emotion inference path without freezing.

Recorded device(s): real Android device testing evidence is shown in Figure 7.3. The exact device model and Android version were not separately recorded in the final appendix.

The most important real-device findings were:

- The TFLite primary model loads cleanly on the primary test device. [ModelLoad] Active model: assets/models/emotion_efficientnetb0_finetuned.tflite and [ModelLoad] Active emotion model: EfficientNetB0 Fine-tuned lines appear in logcat.
- The face gate behaves as designed. Holding the phone too far away produces Face not clear, covering the lens produces No face detected, and a normal selfie distance produces either an Uncertain result or a Reliable result depending on lighting and expression.
- The supportive audio plays cleanly in the active language and falls back to English without crashing when a clip is missing.
- A small set of false-positive Happy detections in poor lighting was observed during development. The mitigation was to keep the burst-of-five averaging and to plan a tighter _kMinConfidence threshold for the final build.

7.7 Release and internal-testing verification

7.7.1 Release APK build verification

The release APK was built with flutter build apk --release --no-shrink. The build output is present in the project tree and was inspected during the writing of this chapter:

- build/app/outputs/flutter-apk/app-release.apk, 126,196,920 bytes (126.2 MB).
- build/app/outputs/apk/release/app-release.apk, same file (also produced by the same build).
- build/app/outputs/apk/release/output-metadata.json confirms applicationId = lk.aac.sinhala_tamil_english, versionName = 1.1.0, versionCode = 2, release variant.
- build/app/outputs/flutter-apk/app-release.apk.sha1 records the SHA-1 fingerprint 43eb07192f0b2eb58393c903d583e79ab1068a67.

This is the same APK used as the basis of the Play Console upload and the real-device walkthroughs described in Section 7.6.

7.7.2 Google Play internal-testing distribution

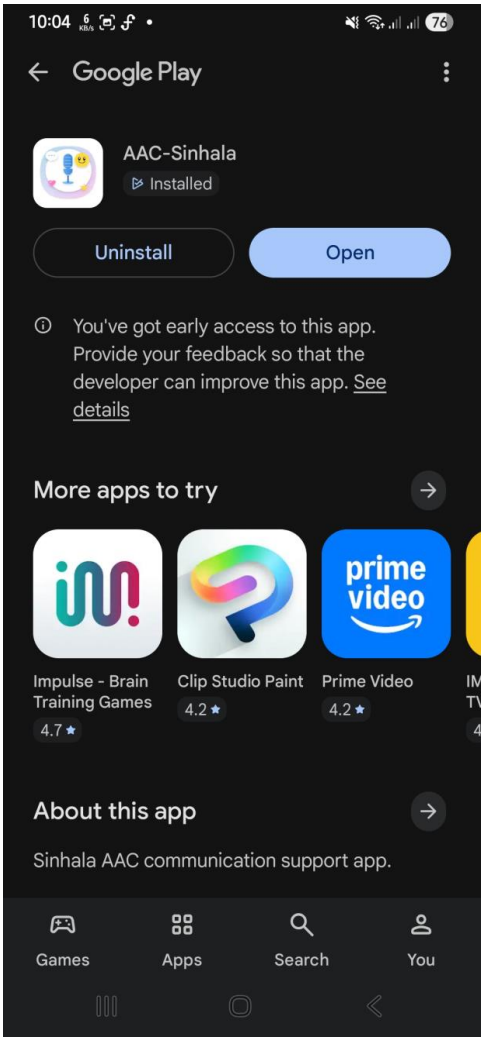


Figure 7.4: Google Play internal testing evidence.

This figure provides controlled distribution evidence for invited testers.

The Android application was distributed through Google Play internal testing for controlled review and real-device testing. The deployment work for this track is documented in the interim report (Fernando, 2026, Section 3.10): an Android App Bundle was generated from the Flutter project, uploaded to the Google Play Console under the Internal Testing track, and shared with invited testers who installed it on their own Android devices to verify the AAC grid, navigation and basic settings. The Play Store listing for the application ID lk.aac.sinhala_tamil_english follows the standard pattern https://play.google.com/store/apps/details?id=lk.aac.sinhala_tamil_english and is reachable through the Play Console internal-testing flow.

7.7.3 Production access request status

Available evidence now supports Google Play internal testing and a submitted production access request. At the time of writing, the request was under Google review and production status was inactive. Public production release is therefore not claimed in this report.

The closed/internal-testing answers used in the production access request state that known contacts, healthcare-related contacts and Android device users checked the AAC screens, language options, categories, TTS, navigation and camera/emotion support. The author used this feedback as release-readiness evidence, not as evidence of clinical approval or diagnostic performance.

7.7.4 iOS path

iOS deployment has not yet been undertaken. As recorded in the interim report (Fernando, 2026, Section 3.10), discussions are ongoing with Ideahub (Pvt) Ltd regarding the possibility of providing the developer account and distribution infrastructure for iOS, potentially as part of a charitable initiative following the broader health-sector approval pathway. The report does not claim a confirmed Apple App Store release. The iOS verification path is limited to the discussion with Ideahub (Pvt) Ltd, and no App Store release or Apple developer account evidence is claimed in this submission.

7.8 Test summary and verification against requirements

Table 7.4: Testing summary.

The table summarises the outcome per test category.

Category	Tests executed	Passed	Documentation Status
Unit tests	41 automated Flutter tests	41	Figure 7.1, Table 7.2A and representative code excerpts in Section 7.2
UI walkthrough	10	10 (informal)	Scripted-run evidence per row
Trilingual	TTS + supportive audio	All paths exercised	Final review or replacement of weak Sinhala/Tamil clips
Offline	Airplane-mode walkthrough	Pass	Airplane-mode walkthrough performed. Separate screenshot evidence was not captured for this individual step, but the

Category	Tests executed	Passed	Documentation Status
			behaviour was verified during the manual walkthrough.
AI / model	13 verification checks plus 2 evaluation evidence items	13 pass	Saved evaluation evidence and confusion matrices
Release	APK built and internal testing distribution completed.	Pass	Play Console screenshot

Verification against the requirements stated in Chapter 5, Section 5.9 and Section 5.10 is recorded inline above (each functional requirement is traceable to at least one test case). The non-functional requirements that depend on quantitative measurement, for example inference latency under 500 ms and app startup under 3 seconds, still require timing screenshots before measured values are claimed.

Chapter 8 takes this evidence and evaluates the system against its stated objectives and research questions.

Chapter 8: Evaluation and Conclusion

This chapter evaluates the system against the aim, objectives and research questions set in Chapter 1. It also summarises stakeholder feedback, states the limitations, describes future work and concludes the thesis. The purpose is not only to repeat what was built, but to judge the project as an applied AI and software engineering artefact: what worked in practice, where compromises were made, and what cannot yet be claimed without further evidence.

8.1 Evaluation against the objectives

The six objectives stated in Chapter 1, Section 1.5 are reviewed below. Each row records the objective, the supporting evidence and a realistic status. The status is cautious because some work is technically implemented, but still needs formal pilot evidence, production approval or clinical governance before it can be treated as fully validated.

In this chapter, Met means that the implementation and supporting evidence satisfy the undergraduate project objective. Partly met means that the feature or evidence exists, but an external step such as clinical validation, pilot approval, wider production release or final threshold tuning is still required.

Table 8.1: Objectives versus achievements.

The table maps the six project objectives against the evidence delivered.

Objective	Evidence	Status
1. **To analyse** the AAC needs of children with autism in Sri Lanka, including language, cultural, cost and access barriers.	Chapter 2 (ASD background, AAC gap, Sri Lankan trilingual need, four-gap research framing in Section 2.2.4, Pragathi Centre / National Hospital Galle context).	Met
2. **To design** a practical mobile-first AAC solution supporting Sinhala, Tamil and English for different communication profiles.	Chapter 5 (architecture, use case view, data model, trilingual design, Level 1-2 vs Level 3+ split in Section 5.7, mobile-first rationale in Section 5.1).	Met

Objective	Evidence	Status
3. **To implement** offline-first AAC features with local storage, customisation, manual backup, restore and sharing, and caregiver-friendly use.	Chapter 6 describes the Flutter project structure, services, SharedPreferences-backed state, trilingual TTS, the EmojiText rendering fix, simple_aac_app customisable cards and ZIP backup, restore and sharing in Section 6.3.7. The earlier dashboard/Firebase direction was kept out of the submitted build because offline use, privacy and caregiver practicality were more realistic for this stage.	Met
4. **To train and integrate** a mobile-compatible on-device facial expression recognition model using TensorFlow Lite.	Chapter 6, Sections 6.4-6.5 (dataset audit, class weighting, MobileNetV2 fallback, EfficientNetB0 primary, TensorFlow Lite export, primary/fallback loading, tensor validation, debug-time model selection). CBAM was not deployed because the final model needed portable TensorFlow Lite loading without Flex / SELECT_TF_OPS. Model performance is reported through saved evaluation figures, confusion matrices and TensorFlow Lite verification outputs.	Partly met. Mobile integration is achieved, but final validation and threshold tuning remain future work.
5. **To evaluate** the system through real-device testing, model verification, app testing and stakeholder feedback.	Chapter 7 provides the evaluation evidence, including unit tests, scripted app walkthroughs, AI/model verification checks, backup/restore/share tests and Google Play internal-testing evidence. Informal stakeholder feedback is used only as supporting usability evidence and is discussed in Section 8.3.	Partly met. Technical testing is evidenced, but formal user or clinical validation remains future work.
6. **To document** ethical, data protection and deployment	Chapter 4 (legal and ethical feasibility Section 4.8, data protection Section 4.9, Section 4.9.1	Met

Objective	Evidence	Status
considerations for future clinical or pilot use.	on future child face-data collection, clinical pilot Section 4.11). Chapter 3, Section 3.3.7 and Chapter 4, Section 4.7 (dashboard scope change). Chapter 5, Section 5.12. Chapter 8 Sections 8.5-8.7. Pragathi Centre / National Hospital Galle letter in Appendix A. Appendix I.	

Overall completion status. The project achieved most of the planned implementation objectives and produced a working Android prototype rather than only a design proposal. The application includes the core AAC features, trilingual content, on-device facial expression recognition, the packaged TensorFlow Lite model pair, supportive audio prompts, and the companion simple_aac_app for custom cards, backup, restore and sharing. The important engineering achievement is deployability: the work moved from trained models and interface plans into a signed APK that can be installed and tested on real devices. The release APK is present in the project tree (build/app/outputs/flutter-apk/app-release.apk, approximately 126.2 MB, versionName 1.1.0, versionCode 2, application ID lk.aac.sinhala_tamil_english), and internal testing evidence is recorded through Google Play Console screenshots and the interim report. At the same time, the project should not be evaluated as a completed clinical product. A production access request has been submitted and remains under Google review, formal pilot evidence is still pending, and public production release is not claimed. Model evaluation is therefore presented through saved learning curves, confusion matrices and TensorFlow Lite verification evidence rather than unsupported standalone headline accuracy values.

8.2 Evaluation against the research questions

The five research questions stated in Chapter 1, Section 1.6 are revisited as applied questions rather than simple yes/no items. Each answer links back to the supporting chapter and notes where the evidence is still limited.

- **RQ1, Trilingual mobile AAC design for the Sri Lankan context. Answered in Chapter 2 (problem context) and Chapter 5, Section 5.7 (trilingual content design). The application supports Sinhala, Tamil and English at the vocabulary, TTS and supportive audio layers, and works offline-first. This suited the local access problem better than a cloud-dependent design, although formal usability testing with children and caregivers is still future work.**

- **RQ2, Privacy-preserving on-device FER inside an AAC application. Answered in Chapter 5, Section 5.5 and Chapter 6, Section 6.5. The system runs inference on-device, never stores or transmits raw camera frames, and uses a face-gate + confidence + margin design to avoid forcing a label on every frame. This is safer than presenting the AI as diagnosis, but the thresholds still need real-device evidence before wider use.**
- **RQ3, Practical challenges in converting trained models to TFLite for Flutter. Answered in Chapter 6, Section 6.4. The main lesson was that a model useful in a notebook is not automatically suitable for a mobile AAC application. CBAM had to be dropped because of Flex / SELECT_TF_OPS incompatibility, and EfficientNetB0 needed an internal [-1, +1] to [0, 255] rescaling layer to align with the Flutter pipeline. Keeping MobileNetV2 as a fallback made the build more resilient because the app could still demonstrate the FER path if the primary model failed on a device.**
- **RQ4, Usefulness as a supportive observation and communication tool. Answered in Section 8.3 below. Practical interest from Pragathi Centre / National Hospital Galle and informal comments from medical students support the idea that the tool may be useful for observation and AAC support. This evidence is modest and early. It does not prove clinical effectiveness, and a formal pilot is intended only after ethical clearance.**
- **RQ5, Limitations and future improvements. Answered in Sections 8.5-8.7 below. The most important limitations are not only coding tasks, but also dataset bias, Android-only deployment, lack of long-term user testing, governance for child face data and the need to keep the AI supportive rather than diagnostic.**

8.3 Stakeholder feedback

The stakeholder feedback received so far is qualitative and informal. It is useful because it shows whether the system direction is understandable to people close to the problem, but it is not a formal user study, clinical validation or model-accuracy measure.

- **Pragathi Centre / National Hospital Galle.** The Head of Pragathi Centre issued a formal letter (Appendix A) confirming a digital initiative undertaken in collaboration with the author. The letter states that the application is designed to support children with communication deficits, particularly autism spectrum disorder, cerebral palsy and global developmental delay, and that clinical input and multidisciplinary expertise from the Pragathi team are involved. This is important evidence of clinical interest and context. It is not Ministry approval, ethics clearance or completed pilot evidence. The letter also states that a pilot project is intended only after the necessary ethical clearance, and that broader availability would depend on the outcomes of that pilot.
- **Medical student observation use.** During informal conversations around the clinical context, the application was seen as potentially useful as a supportive observation tool for medical students learning to recognise possible emotional cues in children with autism. This does not make the AI clinically diagnostic. It supports the safer role chosen in this thesis: an educational and observation aid that may prompt human attention, while leaving interpretation and action to supervised users.
- Karapitiya context. Earlier field exposure in the Karapitiya / Pragathi context helped shape the original AAC interface decisions and is summarised in Appendix A as field-visit context evidence. It is included as design context only, not as clinical approval. Any photograph reused as supporting evidence must remain anonymised and authorised.
- **Caregivers and parents.** Informal feedback during development shaped several UX decisions, including the simplification of sensory feedback controls into one Vibration On switch and the choice to keep camera/emotion support manual rather than automatic. This was a practical compromise. It reduced feature complexity and supported caregiver supervision, although it also means the system has not yet been tested as a long-term daily-use tool.

A short anonymous Google Form was prepared and shared with invited testers to collect informal feedback about the Smart AAC application. The form asked about tester role, device used, language tested, ease of opening and using the app, clarity of the home screen, usefulness of categories and communication cards, understandability of Sinhala/Tamil/English labels, app issues, and whether the app could help a child express basic needs or feelings. It also asked which features were useful, including visual cards, TTS, favourites, support audio, camera emotion support and offline use. For the camera/emotion support feature, the form asked

whether it was tested, whether the predicted emotion felt reasonable, and whether the app made it clear that emotion detection is supportive and not diagnostic.

The Google Form evidence is treated only as early usability evidence. It helps identify practical issues such as language clarity, first-run usability and tester understanding of the non-diagnostic warning. It is not treated as a formal clinical evaluation, a therapist study, a large-scale survey or a measure of model accuracy. Appendix H includes the form questions and response-summary screenshots.

The feedback so far supports the decision to position the AI component as supportive observation rather than diagnosis. That position is clinically safer and also more honest for a final-year prototype. Formal clinical validation has not been completed, and wider deployment remains subject to health-sector approval.

8.4 Model evaluation

The model evaluation is split between three areas: what is confirmed technically, what was compromised for mobile deployability, and what still needs future validation.

Confirmed:

- Both deployed models satisfy the TensorFlow Lite contract (input [1, 224, 224, 3] float32, output [1, 6] float32, no Flex operations, dummy inference succeeds). This matters because the model had to run inside Flutter on a phone, not only inside Colab. The contract was verified in the Colab notebook and at run-time through the [ModelLoad] debug lines.
- The application loads the EfficientNetB0 primary model first and falls back automatically to MobileNetV2 if loading or tensor validation fails. The fallback model mattered because a real mobile prototype should not depend on one model file behaving perfectly on every device. The active model is observable from the debug panel, which also makes testing more transparent.
- The face-gate + confidence + margin design produces explicit *No face detected*, *Face not clear*, *Uncertain* and *Reliable* states, rather than forcing every frame into a label. This is a limitation as well as a safeguard: it may reduce confident outputs, but it lowers the risk of presenting weak camera input as a meaningful emotional judgement.

Remaining future validation work:

- Final validation accuracy is not stated as a separate headline value unless it is confirmed from the final evaluation output.
- Final test performance is evidenced through the saved evaluation figures and confusion matrices in Chapter 6.

- Per-class confusion matrices for both models are included in Chapter 6 as Figure 6.13 and Figure 6.19.
- Final tuned values of _kMinConfidence and _kMinTopMargin should be added only after final real-device evidence is available.
- TFLite verification evidence from the Colab notebook is included as Figure 6.20 and Figure 6.21.

The author has deliberately not stated unverified numbers for any of these. The original interim 80 per cent accuracy target is not claimed as achieved. The practical achievement is instead the validated deployment path: a mobile-compatible model pair, a fallback strategy, TensorFlow Lite checks and cautious output states. CBAM was not deployed because the final system needed reliable mobile packaging more than architectural complexity.

8.5 Limitations

Table 8.2: Limitations and future work.

The table records each limitation and the planned response.

Limitation	**Future work**
Public training datasets may not represent Sri Lankan children, local lighting and camera conditions, or variation in individual expression style sufficiently. They also cannot fully capture how children with autism may express emotion in context.	Explore a **Sri Lankan-specific facial expression dataset** for children, collected **only** under formal ethical approval, child-safeguarding procedures, parental or guardian consent and **health-sector authorisation**, then use it cautiously for fine-tuning or evaluation alongside the existing pipeline.
Wider clinical use of the prototype is **not** yet approved or validated at scale.	Work through the hospital, institutional ethics committee and relevant health-sector authorities before wider deployment or any organised data collection. Treat the current build as a research prototype.
Long-term **sustainability** (maintenance, distribution, governance) is unresolved.	Explore a **free, low-cost or non-profit social-impact** deployment path, potentially with hospital, responsible organisational or public-

Limitation	Future work
	sector support, while keeping safety and governance explicit.
The system is **localised mainly for Sri Lanka** (trilingual content and stakeholder context).	International expansion should be considered only after credible local validation. Each new country would need language packs, cultural symbol adaptation, local clinical partners and separate ethical approval.
Headline model accuracy and final confidence thresholds are not yet confirmed for submission.	Re-run Colab evaluation, add confirmed values only after verification, and tighten on-device confidence and margin thresholds using evidence.
Google Play distribution is currently evidenced as **internal testing** only.	Confirm production track if and when evidence exists. Maintain accurate documentation.
No completed pilot or long-term child/caregiver use evidence.	Run the intended pilot within ethically cleared scope through Pragathi Centre / National Hospital Galle, with careful attention to usability, supervision and interpretation limits.
No iOS App Store release confirmed, so the completed deployment path is Android only.	Continue iOS planning with Ideahub (Pvt) Ltd subject to approval, testing and release requirements.
On-device TensorFlow Lite compatibility limited the deployed FER architecture. More complex options such as CBAM were not used in the submitted build because portable, Flex-free loading was more important.	Continue lightweight model optimisation, measure real-device latency/start-up, and keep only models that can be packaged and tested reliably on target phones.
Camera/emotion support depends on caregiver or therapist supervision and	Keep the feature manual, visible and opt-in. Future pilot protocols should define who

Limitation	Future work
should not be used as automatic monitoring.	supervises the camera screen and how outputs are interpreted.

The current limitations of the system are listed below. They are not only technical defects. Several are evidence, deployment and governance boundaries that define what the prototype is ready for.

1. Dataset is not Sri Lankan. The deployed model is trained on three public Kaggle datasets. Those sources were suitable for academic development and for proving an initial on-device pipeline, but they cannot fully represent Sri Lankan children's appearance, everyday lighting, typical phone cameras, or the way some children with autism express emotion in real settings. Expressions also vary by age, culture and individual style. This limits how strongly the FER component can be interpreted. A future Sri Lankan facial expression dataset should be explored only under formal ethical approval, child safeguarding procedures, parental or guardian consent and health-sector authorisation. No such collection has been carried out for this thesis.
2. **Six-class emotion set only.** A Tired / fatigue class was deliberately not added because no reliable dataset and validation route existed for it. Adding a clinically sensitive class without evidence would make the system look more capable than it is. Tired is therefore reserved for future work.
3. The interim 80 per cent accuracy target is not claimed as achieved. Model evaluation is reported through saved learning curves, confusion matrices and TensorFlow Lite verification evidence included in Chapter 6 and Chapter 7. This keeps the evaluation tied to available evidence rather than to an earlier target.
4. **Confidence thresholds and model architecture are still constrained by deployment evidence.** The application is configured with loose thresholds (_kMinConfidence = 0.40, _kMinTopMargin = 0.08) to support developer-side inspection during development. These values should be tightened with real-device evidence. The same practical constraint affected the model architecture: CBAM was not deployed because the submitted build needed TensorFlow Lite compatibility and reliable mobile loading.
5. The application has been distributed through Google Play internal testing, and a production access request has been submitted through Play Console. The release APK is built and signed (build/app/outputs/flutter-apk/app-release.apk, 126.2 MB, version 1.1.0+2), but public Google Play production release is not claimed because production approval has not been supplied.

6. **No confirmed iOS App Store release.** iOS deployment has not yet been undertaken, and there is currently no Apple Developer account in place. This means the completed deployment evidence is Android-only. Discussions are ongoing with Ideahub (Pvt) Ltd about providing the developer account and distribution infrastructure for iOS, potentially as part of a charitable initiative following the broader health-sector approval pathway (Fernando, 2026, Section 3.10). No public App Store release is claimed.
7. **No completed pilot or long-term user testing.** A pilot is intended after ethical clearance through Pragathi Centre / National Hospital Galle. Until then, there is no evidence from sustained daily use by children, caregivers or therapists.
8. **No Ministry or wider health-sector authorisation for clinical-scale use or for child face-data collection.** Approval would be required before wider data collection or clinical deployment, and the current work stays within prototype and internal-testing boundaries. Caregiver or therapist supervision remains required for camera use.
9. **WCAG audit not performed.** The application targets WCAG 2.1 AA by design, but a formal audit has not been carried out yet. This leaves accessibility as a design intention that still needs external checking.
10. **Support-audio pronunciation needs final review.** Some Sinhala and Tamil support prompts may require replacement before wider release, especially because pronunciation quality affects trust and child comfort.

These limitations do not remove the value of the current system as a final-year prototype. They make the readiness boundary explicit: the system is suitable as a technically evidenced research prototype, but not yet as a clinically validated or publicly scaled service.

8.6 Social impact

Beyond the technical contribution, the project has a clear social motivation. Sri Lankan low-income families often cannot afford the type of commercial AAC systems sold in higher-income markets, which typically cost USD 100-USD 300 plus the price of a tablet. Children in these families who have communication difficulties, particularly children with autism, have limited access to AAC tools that work in Sinhala or Tamil, run offline on phones the family already owns, and respect the data-protection conditions appropriate for child face data. This is why deployability mattered more than theoretical complexity in several decisions. The system favours a signed Android APK, local vocabulary, manual backup, restore and sharing, and on-device inference instead of a Firebase/dashboard model that would add connectivity, data custody, laptop access and maintenance burdens. The social-impact framing is not a claim of clinical impact. It is a description of the intended user base and the practical design choices that follow from it. Section 8.7.4 returns to how that intention could be sustained responsibly over time.

8.7 Future work

Future work is grouped under ethics, governance, deployment and technical improvement. The order is intentional: a Sri Lankan FER dataset, a pilot study and any therapist-facing workflow must come after approval and supervision, while lower-risk engineering improvements can continue in parallel.

8.7.1 Sri Lankan facial expression dataset

One of the most important future improvements would be the careful design of a Sri Lankan facial expression dataset involving children who might use an AAC system of this kind. The current public datasets were valuable for learning the end-to-end pipeline and for shipping an initial on-device model, but they cannot fully reflect Sri Lankan children's skin tones, typical indoor and outdoor lighting, phone-camera quality, cultural display rules, or the diversity of expressive behaviour among children on the autism spectrum.

A local dataset could support fairer evaluation and later fine-tuning only if it were approved, clinically supervised and collected to a high standard. This thesis does not claim that such collection has begun, or that approval already exists. It records the dataset idea as a conditional and high-priority next step.

The future goal is not to collect raw child data informally. A responsible route would be an ethically approved feedback and dataset pathway where a trained reviewer, therapist or caregiver marks an emotion suggestion as reasonable, uncertain/confusing or incorrect. This could show whether the prediction is useful in practice and where the model fails for local lighting, phone cameras and expression patterns.

This future dataset direction is consistent with the interim report, which treated purpose-collected facial-expression data and pilot evaluation as dependent on ethics approval and the hospital pathway rather than as work already completed. No reviewer feedback or child face dataset is collected by the submitted application.

8.7.2 Data protection, child safeguarding and research ethics

Because the proposed future dataset would involve **children** and **facial images**, the project cannot proceed with any such data collection casually. Face images are **biometric and personal data**, and children's data are **especially sensitive** under common data-protection and research-ethics standards. Any future study would need explicit parental or guardian consent, consideration of child assent where appropriate, secure storage, strictly limited access, defined retention periods, and planned deletion or anonymisation / pseudonymisation where compatible with model-training needs. Training workflows would also need to comply with Sri Lankan data-protection law, including the Personal Data Protection Act, No. 9 of 2022 (Parliament of the Democratic Socialist Republic of Sri Lanka, 2022), and institutional ethics review.

Until those conditions are met, the responsible choice is to continue using public sources for training and to treat on-device outputs as supportive observation, not as clinical truth.

8.7.3 Collaboration with hospital, ethics bodies and health-sector authorities

Given the sensitivity of child face data and the clinical-adjacent setting in which the Pragathi AAC App is discussed, any future dataset or pilot effort should be handled as a formal collaboration with the relevant hospital, institutional or national ethics committee and health-sector authorities. Clear roles would be needed for clinicians, caregivers, reviewers and data custodians. Ministry or health-sector approval would be required before wider data collection or clinical deployment at scale. This report does not state that such approval has been granted.

This also applies to any future Firebase-style observation dashboard or therapist workflow. The dashboard idea was reduced in the submitted build because cloud storage, account access and remote observation would create a heavier consent and governance burden than a manual backup, restore and sharing path. It should be revisited only after the clinical, ethical and data-governance pathway is clear.

8.7.4 Non-profit and social-impact sustainability

The longer-term direction of the project is social as well as technical. A major motivation was the cost barrier facing Sri Lankan families who cannot afford commercial AAC tools. For that reason, future deployment should continue to explore a free, low-cost or non-profit model, subject to sustainability, hosting, maintenance and governance requirements. Responsible organisational support, hospital partnership or public-sector collaboration might help cover developer accounts, signing pipelines or maintenance time, but no charity status, funding commitment or corporate sponsorship is claimed here. The aim is not to reposition the system as an expensive paid product for the same low-income user group. Any paid tier, if it were ever introduced, would need its own ethics and equity review.

8.7.5 International expansion (conditional)

If the Sri Lankan version were validated locally in an ethically sound way, the same architectural ideas could later be adapted for other multilingual, low-resource settings: offline-first mobile AAC, local language packs, on-device inference and cautious observation framing. That path would still require local clinical partners, cultural adaptation of symbols and vocabulary, language localisation, and separate ethical approval in each jurisdiction. No international rollout is assumed or claimed by this thesis.

8.7.6 Technical and deployment next steps

Alongside the governance-sensitive items above, the author sees the following concrete tasks after final submission, roughly in priority order:

1. **Confirm standalone validation and test metrics.** If standalone accuracy values are required, add them only from confirmed Colab evaluation output. Continue to use the saved learning curves, confusion matrices and TensorFlow Lite verification outputs as the main evidence base. Tighten the on-device confidence and margin thresholds using the same evidence.
2. **Formal ethical clearance and supervised pilot.** When and if ethical clearance is in place through Pragathi Centre / National Hospital Galle, run the intended pilot only within the approved scope. Treat the pilot as a test of supportive observation, usability and caregiver practicality, not of diagnostic performance.

Approved reviewer-feedback and therapist workflow. After formal ethics clearance, parental or guardian consent and the required hospital or health-sector approval, the author could explore a supervised route where a trained reviewer marks an emotion suggestion as reasonable, uncertain or incorrect. This could support therapist collaboration and model improvement, but it should not be used to diagnose the child.

3. **Lightweight deployment optimisation.** Capture real-device timing evidence for latency and start-up so the non-functional requirements in Chapter 5, Section 5.10 can be checked quantitatively. Future model changes should also consider APK size, memory use and whether the model remains easy to package without Flex operations.
4. **Wider Play Store and iOS rollout.** Move the Android build along the testing tracks only with evidence and policy alignment. Future iOS release support is being explored through Ideahub (Pvt) Ltd, subject to approval, testing and release requirements.
5. **Governance for maintenance.** Clarify who would own releases, incident response, vocabulary updates, support-audio review and content corrections if the app reaches a wider audience.
6. **Secure caregiver sharing.** The current manual backup, restore and share path in simple_aac_app is the appropriate design choice for this prototype because it avoids always-on cloud storage. If a dashboard is ever added, it must be opt-in, consent-based, encrypted and consistent with Chapter 4, Section 4.9.

Multilingual vocabulary and careful interface refinement. Future releases should expand Sinhala, Tamil and English vocabulary with caregiver and therapist input. Gentle animations or child-friendly interaction layers may be explored later, but only with testing because children with autism can respond differently to movement, colour and sound.

8.8 Conclusion

The project set out to build a trilingual, offline-first mobile AAC application for children with communication difficulties in Sri Lanka, and to add a supportive on-device facial expression recognition layer that can help caregivers, therapists and medical students observe possible emotional cues. The main achievement is a deployable Flutter Android application, Pragathi AAC App in clinical correspondence and Smart AAC System with Facial Expression Recognition in academic correspondence. It has been distributed through Google Play internal testing, has a production access request under Google review, runs offline, ships two mobile-safe TensorFlow Lite models, and is supported by a formal letter of clinical interest from Pragathi Centre / National Hospital Galle.

The application is presented with measured boundaries. It is a supportive AAC and observation tool, not a diagnostic instrument. The AI component should be understood as a cautious aid for human observation. Wider deployment remains subject to health-sector approval. A pilot is intended only within ethically cleared scope, and no pilot results are claimed. Headline accuracy numbers are not reported as final values without confirmed evidence.

The work also documents practical lessons that are useful beyond this project. Mobile-deployment constraints should be considered alongside model accuracy from the beginning. A simpler offline-first workflow can be more realistic than a cloud dashboard when users may have limited connectivity, limited devices and sensitive child data. The primary/fallback model strategy improved demonstrability and resilience, while the decision not to deploy CBAM showed that theoretical model complexity is less important than a model that can actually run safely on target phones.

Future work is concrete and realistic about ethics: confirm standalone numerical metrics where needed, review and improve weak support-audio clips, pursue a Sri Lankan child facial-expression dataset only inside a formal ethics and data-protection framework, support the intended pilot through Pragathi Centre / National Hospital Galle when clearance allows, expand multilingual vocabulary, and continue responsible iOS exploration with Ideahub (Pvt) Ltd. Some steps depend on hospital, ethics committee and health-sector decisions rather than on coding alone. The thesis states those dependencies openly so that later work does not blur research ambition with unauthorised data collection or unapproved clinical use.

Chapter 9: References

A. References cited in the report

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Mane, R., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. and Zheng, X. (2016) 'TensorFlow: a system for large-scale machine learning', in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*. Berkeley, CA: USENIX Association, pp. 265-283.
- Alant, E. and Bornman, J. (2021) *Augmentative and alternative communication: engagement and participation*. San Diego, CA: Plural Publishing.
- American Psychiatric Association (2013) *Diagnostic and statistical manual of mental disorders*. 5th edn. Washington, DC: American Psychiatric Publishing.
- American Speech-Language-Hearing Association (2022) *Augmentative and alternative communication (AAC)*. Available at: https://www.asha.org/public/speech/disorders/aac/ (Accessed: 22 February 2025).
- Barrett, L.F., Adolphs, R., Marsella, S., Martinez, A.M. and Pollak, S.D. (2019) 'Emotional expressions reconsidered: challenges to inferring emotion from human facial movements', *Psychological Science in the Public Interest*, 20(1), pp. 1-68.
- Beukelman, D.R. and Light, J.C. (2020) *Augmentative and alternative communication: supporting children and adults with complex communication needs*. 5th edn. Baltimore, MD: Paul H. Brookes.
- Boehm, B.W. (1988) 'A spiral model of software development and enhancement', *Computer*, 21(5), pp. 61-72.
- Brooke, J. (1996) 'SUS: a "quick and dirty" usability scale', in Jordan, P.W., Thomas, B., Weerdmeester, B.A. and McClelland, I.L. (eds.) *Usability Evaluation in Industry*. London: Taylor and Francis, pp. 189-194.
- Creswell, J.W. and Creswell, J.D. (2018) *Research design: qualitative, quantitative, and mixed methods approaches*. 5th edn. Thousand Oaks, CA: SAGE Publications.
- Dawe, M. (2006) 'Desperately seeking simplicity: how young adults with cognitive disabilities and their families adopt assistive technologies', in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York: ACM, pp. 1143-1152.
- Department of Census and Statistics, Sri Lanka (2012) *Census of population and housing - 2012*. Colombo: Department of Census and Statistics.

- Divan, G., Vajaratkar, V., Desai, M.U., Strik-Lievers, L. and Patel, V. (2021) 'Prevalence and risk factors for autism spectrum disorder in low- and middle-income countries: a systematic review and meta-analysis', *Global Mental Health*, 8, e30.
- Ekman, P. and Friesen, W.V. (1971) 'Constants across cultures in the face and emotion', *Journal of Personality and Social Psychology*, 17(2), pp. 124-129.
- Elsabbagh, M., Divan, G., Koh, Y.J., Kim, Y.S., Kauchali, S., Marcin, C., Montiel-Nava, C., Patel, V., Paula, C.S., Wang, C., Yasamy, M.T. and Fombonne, E. (2012) 'Global prevalence of autism and other pervasive developmental disorders', *Autism Research*, 5(3), pp. 160-179.
- Fernando, Y.P. (2026) *Interim Report, Smart AAC System with Facial Expression Recognition for Autism*. London Metropolitan University, CS6P05ES. Unpublished interim project submission.
- Firebase (2023) *Firebase documentation*. Available at: https://firebase.google.com/docs (Accessed: 15 January 2025).
- Fletcher-Watson, S. and Happe, F. (2019) *Autism: a new introduction to psychological theory and current debate*. 2nd edn. Abingdon: Routledge.
- Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Luetge, C., Madelin, R., Pagallo, U., Rossi, F., Schafer, B., Valcke, P. and Vayena, E. (2018) 'AI4People: an ethical framework for a good AI society', *Minds and Machines*, 28(4), pp. 689-707.
- Flutter (2023) *Flutter documentation*. Available at: https://flutter.dev/docs (Accessed: 15 January 2025).
- Ganz, J.B. (2015) *AAC for individuals with autism spectrum disorders*. New York: Springer.
- Ganz, J.B., Davis, J.L., Lund, E.M., Goodwyn, F.D. and Simpson, R.L. (2012) 'A meta-analysis of single case research studies on aided augmentative and alternative communication systems with individuals with autism spectrum disorders', *Journal of Autism and Developmental Disorders*, 42(1), pp. 60-74.
- Goodfellow, I., Bengio, Y. and Courville, A. (2015) *Deep learning*. Cambridge, MA: MIT Press.
- Grossard, C., Dapogny, A., Cohen, D., Bernheim, S., Martinerie, J., Janvier, M., Grynszpan, O., Chaby, L., Bailly, K. and Dubuisson, S. (2020) 'Children with autism spectrum disorder produce more ambiguous and less socially meaningful facial expressions', *Molecular Autism*, 11(1), p. 5.

- Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H. (2017) 'MobileNets: efficient convolutional neural networks for mobile vision applications', *arXiv preprint arXiv:1704.04861*.
- Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H. and Le, Q. (2019) 'Searching for MobileNetV3', in *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Piscataway, NJ: IEEE, pp. 1314-1324.
- International Telecommunication Union (2022) *Measuring digital development: facts and figures 2022*. Geneva: ITU.
- Ioffe, S. and Szegedy, C. (2015) 'Batch normalization: accelerating deep network training by reducing internal covariate shift', in *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: JMLR, pp. 448-456.
- Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. and Kalenichenko, D. (2018) 'Quantization and training of neural networks for efficient integer-arithmetic-only inference', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE, pp. 2704-2713.
- Jobin, A., Ienca, M. and Vayena, E. (2019) 'The global landscape of AI ethics guidelines', *Nature Machine Intelligence*, 1(9), pp. 389-399.
- Kothari, C.R. (2004) *Research methodology: methods and techniques*. 2nd edn. New Delhi: New Age International Publishers.
- LeCun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning', *Nature*, 521(7553), pp. 436-444.
- Li, S. and Deng, W. (2020) 'Deep facial expression recognition: a survey', *IEEE Transactions on Affective Computing*, 13(3), pp. 1195-1215.
- Light, J.C. and McNaughton, D. (2012) 'The changing face of augmentative and alternative communication: past, present, and future challenges', *Augmentative and Alternative Communication*, 28(4), pp. 197-204.
- Light, J.C. and McNaughton, D. (2015) 'Designing AAC research and intervention to improve outcomes for individuals with complex communication needs', *Augmentative and Alternative Communication*, 31(2), pp. 85-96.
- Lord, C., Brugha, T.S., Charman, T., Cusack, J., Dumas, G., Frazier, T., Jones, E.J.H., Jones, R.M., Pickles, A., State, M.W., Taylor, J.L. and Veenstra-VanderWeele, J. (2020) 'Autism spectrum disorder', *Nature Reviews Disease Primers*, 6(1), p. 5.

- Lorah, E.R., Parnell, A., Whitby, P.S. and Hantula, D. (2015) 'A systematic review of tablet computers and portable media players as speech generating devices for individuals with autism spectrum disorder', *Journal of Autism and Developmental Disorders*, 45(12), pp. 3792-3804.
- Maenner, M.J., Warren, Z., Williams, A.R. et al. (2023) 'Prevalence and characteristics of autism spectrum disorder among children aged 8 years', *MMWR Surveillance Summaries*, 72(2), pp. 1-14.
- Mazefsky, C.A., Herrington, J., Siegel, M., Scarpa, A., Maddox, B.B., Scahill, L. and White, S.W. (2013) 'The role of emotion regulation in autism spectrum disorder', *Journal of the American Academy of Child and Adolescent Psychiatry*, 52(7), pp. 679-688.
- McNaughton, D. and Light, J. (2013) 'The iPad and mobile technology revolution: benefits and challenges for individuals who require augmentative and alternative communication', *Augmentative and Alternative Communication*, 29(2), pp. 107-116.
- Millar, D.C., Light, J.C. and Schlosser, R.W. (2006) 'The impact of augmentative and alternative communication intervention on the speech production of individuals with developmental disabilities', *Journal of Speech, Language, and Hearing Research*, 49(2), pp. 248-264.
- Mollahosseini, A., Hasani, B. and Mahoor, M.H. (2019) 'AffectNet: a database for facial expression, valence, and arousal computing in the wild', IEEE Transactions on Affective Computing, 10(1), pp. 18-31.
- Ministry of Health, Sri Lanka (2020) *National guideline for ethics review of health research in Sri Lanka*. Colombo: Ministry of Health.
- Nair, V. and Hinton, G.E. (2010) 'Rectified linear units improve restricted Boltzmann machines', in *Proceedings of the 27th International Conference on Machine Learning*. Madison, WI: Omnipress, pp. 807-814.
- Parliament of the Democratic Socialist Republic of Sri Lanka (2022) Personal Data Protection Act, No. 9 of 2022. Colombo: Department of Government Printing. Available from the Department of Government Printing website (Accessed: 15 May 2026).
- Perera, H., Wijewardena, K., Aluthwelage, R., Seneviratne, S. and Buddhika, K. (2019) 'Prevalence of autism spectrum disorder in Sri Lanka: a population-based study', *Sri Lanka Journal of Child Health*, 48(2), pp. 133-138.
- Picard, R.W. (2000) *Affective computing*. Cambridge, MA: MIT Press.

- Pragathi Centre / National Hospital Galle (2026) *Letter regarding the Pragathi AAC App*. Dated 7 April 2026, with reference dated 24 March 2026. Included as Appendix A.
- Romski, M. and Sevcik, R.A. (2005) 'Augmentative communication and early intervention: myths and realities', *Infants and Young Children*, 18(3), pp. 174-185.
- Samad, A., Razick, S., De Silva, M.V.C. and Hettiarachchi, S. (2020) 'Challenges and opportunities for autism services in Sri Lanka', *Journal of Autism and Developmental Disorders*, 50(8), pp. 3023-3029.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C. (2018) 'MobileNetV2: inverted residuals and linear bottlenecks', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE, pp. 4510-4520.
- Shorten, C. and Khoshgoftaar, T.M. (2019) 'A survey on image data augmentation for deep learning', *Journal of Big Data*, 6(1), p. 60.
- Sommerville, I. (2016) *Software engineering*. 10th edn. Harlow: Pearson Education.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) 'Dropout: a simple way to prevent neural networks from overfitting', *Journal of Machine Learning Research*, 15(1), pp. 1929-1958.
- Tan, M. and Le, Q. (2019) 'EfficientNet: rethinking model scaling for convolutional neural networks', in Chaudhuri, K. and Salakhutdinov, R. (eds.) *Proceedings of the 36th International Conference on Machine Learning*. Proceedings of Machine Learning Research, 97, pp. 6105-6114. Available at: https://proceedings.mlr.press/v97/tan19a.html (Accessed: 15 May 2026).
- TensorFlow (2023) *TensorFlow Lite documentation*. Available at: https://www.tensorflow.org/lite (Accessed: 15 January 2025).
- Trevisan, D.A., Hoskyn, M. and Birmingham, E. (2018) 'Facial expression production in autism: a meta-analysis', *Autism Research*, 11(12), pp. 1586-1601.
- Wickramasinghe, N., Dissanayake, A. and Samarasinghe, D. (2021) 'Parent training and support programmes for autism in low-resource settings: a scoping review', *Global Health Action*, 14(1), 1910556.
- Woo, S., Park, J., Lee, J.-Y. and Kweon, I.S. (2018) 'CBAM: Convolutional Block Attention Module', in *Computer Vision - ECCV 2018*. Lecture Notes in Computer Science, 11211. Cham: Springer, pp. 3-19. doi: 10.1007/978-3-030-01234-2_1.

- World Health Organization (2021) *Autism spectrum disorders*. Available at: https://www.who.int/news-room/fact-sheets/detail/autism-spectrum-disorders (Accessed: 22 February 2025).
- World Medical Association (2013) 'World Medical Association Declaration of Helsinki: ethical principles for medical research involving human subjects', *JAMA*, 310(20), pp. 2191-2194.
- Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. (2014) 'How transferable are features in deep neural networks', in Advances in Neural Information Processing Systems, 27. Red Hook, NY: Curran Associates, pp. 3320-3328.

B. Dataset sources

- Kaggle (n.d.) FERAC dataset. Available at: https://www.kaggle.com/datasets/rajasreechaiti/ferac-dataset (Accessed: 18 May 2026).
- Kaggle (n.d.) Autism Facial Emotion Recognition dataset. Available at: https://www.kaggle.com/datasets/hasibur013/autism-facial-emotion-recognition (Accessed: 18 May 2026).
- Talaat, F.M. (n.d.) Autistic Children Emotions - Dr. Fatma M. Talaat. Available at: https://www.kaggle.com/datasets/fatmamtalaat/autistic-children-emotions-dr-fatma-m-talaat (Accessed: 18 May 2026).

C. Bibliography (consulted but not directly cited)

The following sources were consulted during background reading and project development but are not directly cited in the main report.

- Bishop, C.M. (2006) *Pattern recognition and machine learning*. New York: Springer.
- Chollet, F. (2017) *Deep learning with Python*. Shelter Island, NY: Manning Publications.
- Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L. (2009) 'ImageNet: a large-scale hierarchical image database', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE, pp. 248-255.
- Ekman, P. and Friesen, W.V. (1978) *Facial Action Coding System: a technique for the measurement of facial movement*. Palo Alto, CA: Consulting Psychologists Press.
- He, K., Zhang, X., Ren, S. and Sun, J. (2016) 'Deep residual learning for image recognition', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE, pp. 770-778.

- Kaggle (2013) *FER-2013: Facial Expression Recognition 2013 Dataset*. Available at: https://www.kaggle.com/datasets/msambare/fer2013 (Accessed: 15 January 2025).
- Pressman, R.S. and Maxim, B.R. (2019) *Software engineering: a practitioner's approach*. 9th edn. New York: McGraw-Hill Education.
- Simonyan, K. and Zisserman, A. (2015) 'Very deep convolutional networks for large-scale image recognition', in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. San Diego, CA: ICLR.

Chapter 10: Appendices

The appendices collect evidence that supports the main body of the report but is too large or too detailed to fit inline. Each appendix is given a working letter (A, B, C, ...) so other chapters can reference it consistently. Evidence that is not included in the Word file is described in the relevant checklist or limitation rather than presented as completed evidence.

The appendix ordering and lettering are used consistently for the supporting evidence in this submission.

Appendix A, Letter from Pragathi Centre / National Hospital Galle

- Original supporting document: Letter from hospital to my university.pdf. The first page is inserted below as Appendix A evidence, and the original PDF file is included with the submission support documents.
- **Date on the letter:** 07 April 2026 (header reference: 24 March 2026).
- **Issued by:** Head of Pragathi Centre, Prahathi Child Intervention Centre, National Hospital Galle.
- **Subject:** *Pragathi AAC App*.
- Use in the thesis: cited in Chapter 1 Section 1.9 (clinical collaboration), Chapter 2 Section 2.3 (problem statement and local context), and Chapter 8 Section 8.3 (stakeholder feedback).
- **Wording boundary:** the letter confirms collaboration, clinical input and an intended pilot after the necessary ethical clearance. It does **not** state Ministry of Health approval. The thesis treats this distinction carefully throughout.
- **Status: included as appendix evidence. The letter supports clinical interest and the intended pilot pathway, subject to ethical clearance. It is not treated as Ministry approval or completed clinical approval.**

දුරකතන
தொலைபேசி
Telephone
0912121020

විද්‍යුත් තැපෑල
மின்னஞ்சல் முகவரி
E-mail
Pragathiautism@gmail.com

Web www.pragathi.lk

ප්‍රගති ස්නායු සංවර්ධන මධ්‍යස්ථානය
Southern Provincial Autism and Neurodevelopmental Intervention Centre
ජාතික රෝහල - ගාල්ල
தேசிய வைத்தியசாலை - காலி
National Hospital - Galle

මගේ අංකය
எனது இல.
My No.

ඔබේ අංකය
உமது இல.
Your No.

දිනය
திகதி 24.03.2026
Date.

07.04.2026
Prahathi Child Intervention Centre
National Hospital Galle

ESOFT METRO CAMPUS - GALLE
No. 72, Matara Road,
Galle, Sri Lanka.

Re: Pragathi AAC App

I am writing in my capacity as the Head of Pragathi Centre to inform you about an important digital initiative we are currently undertaking in collaboration with Mr. Yasas Pasindu Fernando who is currently studying in your esteemed institution.

We are in the process of developing a mobile application titled “Pragathi AAC App”, designed to support children with communication deficits, particularly those diagnosed with autism spectrum disorder, cerebral palsy, and global developmental delay. The primary aim of this application is to facilitate effective communication and enhance daily functional interactions among these children.

This project involves substantial clinical input and multidisciplinary expertise from the Pragathi team, ensuring that the application is evidence-based, user-friendly, and tailored to the specific needs of the target population.

Following the development phase, we intend to conduct a pilot project involving children attending Pragathi Centre. This will be carried out after obtaining the necessary ethical clearance. Based on the outcomes and refinements from the pilot, the application will be made available for wider use among the general population.

We believe that this initiative has the potential to make a meaningful impact on the lives of many children and their families, and we look forward to your continued support and collaboration in bringing this project to fruition.

We would greatly appreciate any kind of support you can provide to Mr. Fernando for the success of this great initiative.

Yours sincerely

DR. SACHITHA DE SILVA
MBBS, DCH, MD (Paediatrics)
Consultant Community and Developmental Paediatrician (Acting)
National Hospital - Karapitiya
SLMC Reg. No. 35802

LEAVE NO ONE BEHIND, EVERY CHILD HAS AN DIFFERENT ABILITY

Figure A.1: Pragathi Centre support letter.

The letter confirms clinical interest and the intended pilot work for the Pragathi AAC application.

Field-visit context photographs

A field visit was carried out at Pragathi Child Intervention Centre to understand the practical intervention environment and the broader user context relevant to the proposed Smart AAC application. The photographs below are included as contextual evidence only. They do not represent clinical approval, pilot completion or formal validation of the system.

Figure A.2: Pragathi Child Intervention Centre exterior.

This photograph provides field-visit context.

Figure A.3: Pragathi Child Intervention Centre signboard.

This signboard photograph is used as field-visit context evidence.

Figure A.4: Child-friendly intervention environment.

The photograph shows the general intervention environment observed during the field visit.

A.1 Field-visit observation summary

Purpose and boundary. The field visit was used to understand the practical environment in which the Smart AAC / Pragathi AAC application may later be used. It was exploratory and did not involve formal clinical testing, model evaluation or the collection of identifiable child data.

Observations from the setting. The visit highlighted that communication support in this context cannot assume ideal conditions. The environment can be busy and noisy, children may have different attention spans, and some children may find it difficult to express immediate needs such as hunger, discomfort, tiredness or emotional distress. Caregivers and clinical staff therefore need communication tools that are quick to open, simple to navigate and easy to supervise.

Influence on the final design. These observations influenced several practical decisions in the final build. The submitted application was kept offline-first because stable internet access cannot be assumed during everyday use. The interface was kept simple, with clear category cards, large touch targets and favourites for frequently used messages. The camera-based emotion feature was also framed as supportive observation only, with no-face, unclear and uncertain states rather than forcing an emotion label for every frame.

Approval and ethics boundary. The field visit also reinforced the need to keep the current build privacy-preserving. Any future workflow that stores child face-related data, shares observation records through a dashboard or involves structured clinical data collection would require formal ethics clearance, parental or guardian consent and the appropriate hospital or health-sector approval. This appendix therefore treats the visit as design context, not as evidence of clinical validation, Ministry approval or a completed pilot.

Practical value for future work. The visit strengthened the need for a mobile AAC tool that can be used by caregivers, therapists and supervised medical students while remaining understandable to families. It also suggests that any future approved pilot should focus on usability, caregiver practicality, language clarity and the usefulness of supportive prompts, rather than presenting the application as a diagnostic system.

The photographs below are added as field-exposure context evidence. They show the demonstration and discussion environment only and should not be read as clinical approval, a completed pilot or formal validation of the application.

Figure A.5: Field exposure discussion environment.

This image shows the discussion and demonstration environment at Karapitiya Teaching Hospital.

Figure A.6: AAC application field demonstration.

This image shows the AAC application demonstration during the field exposure session.

Field-visit photos and videos folder: Download field-visit photos and videos.

Appendix B, Interim Progress Report

This appendix contains the interim progress report submitted during the project. It is included as supporting evidence of the project's earlier planning, design direction and development progress.

CS6P05ES Project

Interim Report

Smart AAC System with Facial Expression Recognition for Autism

Name:	Yasas Pasindu Fernando
ID Number:	25026764
Date:	Friday, 22 May 2026
First Supervisor:	Ms. Niruni Fonseka,
Second Supervisor:	Mr. Akila Udara Akalanka

Declaration

Module: CS6P05ES **Deadline: 03/25/2026**

Module Leader: Mr. Migara Alawatta **Student ID: 25026764**

PLAGIARISM

You are reminded that there exist regulations concerning plagiarism. Extracts from these regulations are printed below. Please sign below to say that you have read and understand these extracts:

(signature:)____yasaspasindufernando@gmail.com_____ Date: 03/25/2026

This header sheet should be attached to the work you submit. No work will be accepted without it.

Extracts from University *Regulations* on Cheating, Plagiarism and Collusion

Section 2.3: "The following broad types of offence can be identified and are provided as indicative examples…

(viii) Cheating: including taking unauthorised material into an examination; consulting unauthorised material outside the examination hall during the examination; obtaining an unseen examination paper in advance of the examination; copying from another examinee; using an unauthorised calculator during the examination or storing unauthorised material in the memory of a programmable calculator which is taken into the examination; copying coursework.

(ix) Falsifying data in experimental results.

(x) Personation, where a substitute takes an examination or test on behalf of the candidate. Both candidate and substitute may be guilty of an offence under these Regulations.

(xi) Bribery or attempted bribery of a person thought to have some influence on the candidate's assessment.

(xii) Collusion to present joint work as the work solely of one individual.

(xiii) Plagiarism, where the work or ideas of another are presented as the candidate's own.

(xiv) Other conduct calculated to secure an advantage on assessment.

(viii) Assisting in any of the above.

Some notes on what this means for students:

1. Copying another student's work is an offence, whether from a copy on paper or from a computer file, and in whatever form the intellectual property being copied takes, including text, mathematical notation and computer programs.

2. Taking extracts from published sources *without attribution* is an offence. To quote ideas, sometimes using extracts, is generally to be encouraged. Quoting ideas is achieved by stating an author's argument and attributing it, perhaps by quoting, immediately in the text, his or her name and year of publication, e.g. "$e = mc^2$ (Einstein 1905)". A *references* section at the end of your work should then list all such references in alphabetical order of authors' surnames. (There are variations on this referencing system which your tutors may prefer you to use.) If you wish to quote a paragraph or so from published work then indent the quotation on both left and right margins, using an italic font where practicable, and introduce the quotation with an attribution.

Acknowledgements

The author would like to express sincere gratitude to the project supervisor, Ms. Niruni Fonseka, at ESOFT Metro Campus for her continuous guidance, constructive feedback, and support throughout this project. Gratitude is also extended to the second supervisor, Mr. Akila Udara Akalanka, at ESOFT Metro Campus for his additional advice and support across technical and academic aspects of the work.

Special thanks are also extended to the staff at Karapitiya Teaching Hospital, Galle, for their openness to a potential collaboration for the planned pilot study and for sharing valuable clinical insights on children with autism spectrum disorder.

The author is also grateful to the parents, caregivers, and speech-language therapists who provided informal feedback on the AAC interface design. Appreciation is extended to the open-source communities behind Flutter, TensorFlow, Firebase, and MobileNetV2, as well as to the creators of the public facial expression datasets used for initial model testing.

Finally, the author would like to thank family and friends for their encouragement and support throughout this work.

Abstract

Communication impairment is one of the defining features of autism spectrum disorder (ASD), and for many children it creates barriers that extend well beyond language, affecting social participation, education, and daily quality of life. In Sri Lanka, the situation is compounded by the absence of augmentative and alternative communication (AAC) tools that support the country's two official languages, Sinhala and Tamil. The commercially available options are built for Western, English-speaking users and do not reflect the cultural or linguistic realities of Sri Lankan families.

Recent advances in deep learning have made facial expression recognition (FER) a viable addition to AAC systems, allowing emotional state to inform the communication support offered to the user. Despite this potential, the combination of FER with AAC has received limited attention in published research, particularly in low- and middle-income settings where the need is arguably greatest.

This interim report documents the design, methodology, and progress of a final year project developing an AI-powered AAC system with on-device facial expression recognition for children with autism in Sri Lanka. The system takes a dual-platform approach. Platform A offers a customisable, symbol-based AAC interface with trilingual support for children at ASD severity Level 1–2. Platform B extends this with on-device emotion detection using an EfficientNetB0 model with a CBAM attention module, deployed via TensorFlow Lite, targeting children at Level 3 and above. MobileNetV2 was initially evaluated as a baseline but was replaced after EfficientNetB0 with CBAM produced stronger validation results.

The mobile application is built in Flutter and follows an offline-first architecture, with all core data stored locally using SQLite and JSON. Firebase remains a planned option for future synchronisation, but at the interim stage backup is handled through manual export. A therapist-parent dashboard is included to support collaboration and progress monitoring. The emotion recognition component classifies six states, namely happy, sad, angry, fear, neutral, and tired, with a target accuracy of at least 80 per cent.

This report covers the background literature, system architecture, development methodology, work completed to date, and the planned remaining tasks. The objective is to produce a working

proof of concept that addresses a genuine gap in assistive technology provision for children with autism in Sri Lanka.

Keywords: augmentative and alternative communication, autism spectrum disorder, facial expression recognition, EfficientNetB0 + CBAM, TensorFlow Lite, Flutter, offline-first architecture, Sri Lanka, affective computing, assistive technology, multilingual.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Background and Context

Autism spectrum disorder (ASD) is a neurodevelopmental condition characterised by persistent difficulties in social communication alongside restricted, repetitive patterns of behaviour (American Psychiatric Association, 2013). Global prevalence is estimated at approximately one in 160 children (World Health Organization, 2021), although more recent studies in high-income countries report rates as high as one in 36 (Maenner et al., 2023). Communication impairment is central to the condition. The DSM-5 classifies severity across three levels, from Level 1 ("requiring support") to Level 3 ("requiring very substantial support"). Children at Level 3 often have little or no functional speech and depend heavily on augmentative and alternative communication (AAC) to express basic needs and emotions (Beukelman and Light, 2020).

In Sri Lanka, awareness of ASD has been growing. Perera et al. (2019) reported a prevalence of approximately 1.07 per cent among children aged two to nine, broadly consistent with global estimates. However, specialist services remain difficult to access, particularly outside major cities. Critically, there are no commercially available AAC tools that support both Sinhala and Tamil, the two official languages of the country (Samad et al., 2020). The shortage of trained speech-language therapists further compounds the challenge for families seeking communication support (Wickramasinghe et al., 2021).

1.2 Project Overview

The aim of this project is to design and develop a dual-platform AAC system for children with autism in Sri Lanka, with each platform mapped to a distinct clinical communication profile. The split into Platform A and Platform B was an intentional design decision informed by technical constraints and field-facing clinical feedback, including discussions with clinicians at Karapitiya Teaching Hospital. These discussions indicated that children at ASD Level 1-2 and children at Level 3 and above do not engage effectively with a single interaction model. Platform A therefore provides a simple, structured, and highly configurable AAC interface with Sinhala, Tamil, and English support for children who can engage with symbol-based

communication. Platform B targets children with minimal or no functional speech by extending AAC with on-device facial expression recognition and emotion-adaptive assistance. Existing tools such as Avaz informed the baseline comparison, but clinicians and therapists highlighted limitations related to cost, flexibility, and local clinical fit, which reinforced the need for a lightweight locally aligned design.

The dual-platform design was adopted based on clinical feedback indicating that children at different ASD severity levels require distinct interface designs, particularly in terms of complexity, adaptability, and communication support mechanisms.

- **Platform A:** A customisable, symbol-based AAC interface for children at ASD severity Level 1–2, with trilingual support (Sinhala, Tamil, English), configurable vocabulary, text-to-speech output, and visual scheduling.
- **Platform B:** An AI-enhanced AAC platform for children at severity Level 3 and above, extending Platform A with on-device facial expression recognition (FER) using an EfficientNetB0-based model with a CBAM (Convolutional Block Attention Module), deployed via TensorFlow Lite, enabling emotion-adaptive communication support.

The dual-platform design was adopted based on clinical feedback indicating that children at different ASD severity levels require distinct interface designs, particularly in terms of complexity, adaptability, and communication support mechanisms.

The mobile application is developed using Flutter for cross-platform deployment, following an offline-first architecture. At the interim stage, data is stored locally on the device using SQLite and JSON, with backup handled through manual export. Firebase is planned for future optional synchronisation and account-based collaboration but is not yet implemented.

1.3 Rationale and Research Gap

A review of existing AAC systems (see Section 2.2.4) confirms that no commercially available tool combines trilingual support for Sinhala, Tamil, and English with on-device facial expression recognition. Tools such as Proloquo2Go, TouchChat, and Avaz AAC serve primarily English-speaking, Western populations. Their vocabularies, symbol sets, and cultural

assumptions do not transfer well to Sri Lanka, where the food, clothing, religious practices, and family structures represented in AAC content need to be fundamentally different (Alant and Bornman, 2021). Beyond language and culture, cost is a significant barrier. Dedicated AAC devices can run to several hundred US dollars, which places them out of reach for most Sri Lankan families.

Existing AAC systems are also largely static, presenting the same vocabulary regardless of the child's emotional state. There is growing evidence that emotion-aware interfaces can improve engagement and communication outcomes (Picard, 2000), but the integration of FER into AAC tools remains largely unexplored, particularly for children with autism. Published research has examined FER and AAC as separate domains, yet no existing system brings both together within a mobile application designed for a low- and middle-income context, with multilingual support and full offline capability.

This project directly addresses that gap. The proposed system combines trilingual AAC with on-device emotion recognition, delivered through a mobile application that does not depend on internet connectivity to function.

1.4 Project Aim and Objectives

The overall aim of this project is to design and develop an AI-powered AAC system with facial expression recognition, tailored for children with autism in Sri Lanka, and to lay the groundwork for a pilot evaluation in a clinical setting.

The project objectives are as follows.

1. To design a dual-platform system architecture serving children at ASD Level 1–2 (customisable symbol-based AAC) and Level 3+ (AI-enhanced AAC with FER), with trilingual support, offline-first operation, and secure data management.

2. To implement a cross-platform mobile application using Flutter with offline-first architecture, local storage, and manual export backup, targeting Android 8.0+ and iOS 14+.

3. To train and deploy an EfficientNetB0-based FER model with a CBAM attention mechanism, targeting six emotion classes with at least 80% accuracy, deployed on-device

via TensorFlow Lite with inference under 500 ms. MobileNetV2 was initially evaluated as a baseline.

4. To develop a therapist-parent dashboard for collaboration, progress monitoring, and vocabulary management.

5. To establish ethical approval protocols and a clinical partnership for a pilot study at Karapitiya Teaching Hospital, Galle.

6. To document the project in accordance with FC6P01ES module requirements using Harvard referencing.

1.5 Research Questions

The project is guided by the following research questions.

RQ1: How can a dual-platform AAC system be designed to serve children at ASD Level 1–2 and Level 3+, while maintaining trilingual support, offline-first operation, and secure data management?

RQ2: What accuracy can an on-device FER model based on EfficientNetB0 with CBAM attention achieve when trained on 2,000–5,000 images across six emotion classes, and what factors most influence performance?

RQ3: How can therapist-parent collaboration be effectively supported through a secure dashboard integrated with the mobile application, while respecting privacy and consent requirements?

RQ4: What ethical, regulatory, and practical considerations must be addressed for a pilot deployment at Karapitiya Teaching Hospital?

RQ5: To what extent does the implemented system meet its functional and non-functional requirements, and what are the key areas for improvement?

Research questions RQ1–RQ4 are addressed partially in this interim report. RQ5 will be most fully addressed in the final report following system completion and pilot evaluation.

1.6 Report Structure

The remainder of this report is organised as follows. Section 2 presents the background, including a literature review, system architecture, development methodology, AI model design, and ethical considerations. Section 3 describes the work completed to date, including design artefacts and implementation evidence. Section 4 outlines the further work planned. Section 5 provides a progress review comparing the original plan against actual progress. Sections 6 and 7 list the references and bibliography respectively.

2. Background

2.1 Autism Spectrum Disorder

2.1.1 Definition and Diagnostic Criteria

Autism spectrum disorder (ASD) is defined in the DSM-5 as a neurodevelopmental condition with two core features, namely (1) persistent difficulties in social communication and interaction, and (2) restricted, repetitive patterns of behaviour, interests, or activities (American Psychiatric Association, 2013). The DSM-5 brought together what were previously separate diagnoses, including autistic disorder, Asperger's disorder, and PDD-NOS, into a single spectrum. This change reflected the understanding that these conditions share a common neurobiology and mainly differ in how severe the symptoms are (Lord et al., 2020).

The severity classification uses three levels based on how much support a person needs. Level 1 ("requiring support") refers to individuals with noticeable social communication difficulties. Level 2 ("requiring substantial support") applies to those with more marked deficits who have limited ability to start interactions. Level 3 ("requiring very substantial support") describes individuals with severe communication deficits and very little response to social contact (American Psychiatric Association, 2013).

Table 1: ASD Severity Levels and Communication Characteristics

Severity Level	**Social Communication**	**Typical Communication Profile**

Level 1: Requiring support	Noticeable deficits without supports, with difficulty initiating interactions	Full sentences, struggles with pragmatics (turn-taking and topic maintenance), and benefits from AAC for high-demand situations
Level 2: Requiring substantial support	Marked deficits in verbal and nonverbal communication, with limited initiation	Simple phrases or short sentences, and benefits from consistent AAC support
Level 3: Requiring very substantial support	Severe deficits, very limited initiation, and minimal response to social overtures	Very limited or no functional speech, and depends heavily on AAC for basic needs, preferences, and emotions

2.1.2 Communication and Emotional Regulation in ASD

Communication difficulties in ASD can range from subtle issues, such as trouble understanding sarcasm or maintaining a conversation, to a complete absence of functional speech. Around 25 to 30 per cent of children with ASD never develop functional spoken language (Lord et al., 2020). These children are often the most underserved when it comes to communication support. Even children who do develop some speech may still struggle with expressive or receptive language and can benefit from AAC as a supplementary tool.

Emotional regulation is another significant challenge. Mazefsky et al. (2013) describe emotion dysregulation as a common feature of ASD, where children have difficulty identifying, understanding, and managing their emotions. This is directly relevant to AAC design. A system that can pick up on the user's emotional state could provide better support, for example by suggesting calming vocabulary or alerting a caregiver when the child seems distressed. This connection between emotional state and communication is one of the key reasons for integrating facial expression recognition into the proposed AAC system.

2.1.3 Epidemiology

Global prevalence estimates for ASD have risen substantially over recent decades. The World Health Organization (2021) cites a figure of roughly one in 160 children, based on a systematic review by Elsabbagh et al. (2012). More recent data from the US CDC, however, put the rate as high as 2.78 per cent (one in 36) among eight-year-old children (Maenner et al., 2023). This rise is largely attributed to broadened diagnostic criteria, greater awareness, improved screening, and changes in how services are accessed (Lord et al., 2020). In low- and middle-income countries (LMICs), prevalence data tend to be limited and probably reflect underdiagnosis rather than genuinely lower rates (Divan et al., 2021).

2.1.4 Autism in Sri Lanka

Perera et al. (2019) carried out a population-based study in Sri Lanka and found a prevalence of 1.07 per cent among children aged two to nine, which is broadly in line with global figures. Samad et al. (2020) highlighted the growing demand for services, the shortage of trained professionals, and the lack of culturally suitable tools, especially in Sinhala and Tamil. Wickramasinghe et al. (2021) suggested that scalable, technology-based interventions could help fill service gaps, but stressed the need for careful local adaptation.

In practice, the clinical pathway in Sri Lanka usually starts with a referral from primary care to a teaching hospital such as Karapitiya or Lady Ridgeway for specialist assessment. From there, children may receive speech-language therapy or applied behaviour analysis, depending on availability. However, access to these services varies a great deal across the country. The Department of Census and Statistics, Sri Lanka (2012) notes that many families live in rural areas where specialist services simply are not available, and travelling for assessments can be difficult. This is where mobile-based AAC technology could make a real difference, bringing communication support directly into homes where regular therapy is not an option (Samad et al., 2020).

2.2 Augmentative and Alternative Communication

2.2.1 Definitions and Classification

AAC refers to a range of strategies, tools, and technologies used to supplement or replace natural speech for people with complex communication needs (American Speech-Language-Hearing Association, 2022). AAC systems fall into two broad categories, namely unaided (such as gestures and sign language) and aided. Aided systems are further split into low-tech options like picture boards and communication books, and high-tech options like speech-generating devices and tablet-based apps (Beukelman and Light, 2020).

One of the most widely used low-tech approaches for children with autism is the Picture Exchange Communication System (PECS), which involves exchanging picture cards to make requests. PECS is effective for building early communication skills, but it has limitations. It relies on physical materials, the vocabulary is hard to scale, and a trained communication partner needs to be present (Ganz, 2015). High-tech alternatives such as tablet-based apps go beyond these limitations by offering dynamic displays, speech output, and vocabularies that can be easily customised.

The move from dedicated speech-generating devices, which can cost thousands of dollars, to tablet-based apps has been called a "revolution" in AAC (McNaughton and Light, 2013). Tablets are cheaper, more socially acceptable, and more widely available. That said, Dawe (2006) pointed out that the success of assistive technology depends not just on what it can do technically, but also on how simple and reliable it is, and whether families are supported in using it. These points have directly shaped the design decisions in this project.

2.2.2 Evidence Base for AAC in Autism

The evidence for AAC in autism is strong and continues to grow. Ganz et al. (2012) carried out a meta-analysis of single-case studies and found moderate to large effect sizes for AAC interventions aimed at helping children make requests, with positive results across different types of AAC and age groups. Lorah et al. (2015) reviewed studies on tablet computers as speech-generating devices and found encouraging evidence, particularly noting their portability, social acceptability, and lower cost compared to dedicated devices.

An important finding from Millar et al. (2006) is that using AAC does not hold back natural speech development. In fact, it can actually help some children develop speech by reducing frustration and giving them a way to practise communication. This has helped address a common worry among parents and clinicians that AAC might discourage children from learning to talk (Light and McNaughton, 2012; Romski and Sevcik, 2005).

Light and McNaughton (2015) identify four types of communicative competence that AAC systems should support, including linguistic (language skills), operational (ability to use the technology), social (interaction and pragmatic skills), and strategic (strategies for when communication breaks down). The proposed system addresses each of these through its structured vocabulary (linguistic), simple grid-based interface (operational), caregiver-mediated use (social), and emotion-adaptive prompting during stressful moments (strategic).

2.2.3 AAC in Multilingual and Low-Resource Contexts

Most AAC research has been carried out in high-income, English-speaking countries. This leaves a significant gap for culturally and linguistically diverse populations. Alant and Bornman (2021) argue that implementing AAC in different cultural contexts requires adapting interaction styles, respecting local communication norms, and genuinely involving families as partners rather than just giving them technology to use.

Sri Lanka's trilingual setting, with Sinhala, Tamil, and English, creates specific challenges for AAC design. Symbols, labels, and speech output all need to work properly in each language, with text-to-speech supporting correct pronunciation and natural-sounding output (Samad et al., 2020). The cultural context matters too. Food items, clothing, religious activities, and family structures in Sri Lanka are quite different from those typically represented in Western AAC vocabularies, so local adaptation is essential.

The economic realities also shape what is practical. Internet connectivity can be unreliable in rural parts of the country (International Telecommunication Union, 2022), and many families use mid-range or budget smartphones rather than expensive devices. These constraints are why this project uses an offline-first architecture, making sure the core AAC features work without needing an internet connection.

2.2.4 Comparison of Existing AAC Systems

Table 2: Comparison of Existing AAC Systems

System	Platform	Languages	AI/Adaptive Features	Offline Support	Cost
Proloquo2Go	iOS	English, Spanish, French, others	Crescendo vocabulary with no FER	Offline core	~USD 250
TouchChat	iOS, Windows	English primarily	Word prediction with no FER	Offline core	~USD 300
LAMP Words for Life	iOS	English	Motor-planning based design with no FER	Offline core	~USD 300
LetMeTalk	Android, iOS	User-configurable via TTS	None	Partial	Free
CoughDrop	Web, Android, iOS	English primarily	Basic usage logging with no FER	Limited	Subscription
Avaz AAC	iOS, Android	English, Hindi, Tamil, others	Word prediction with no FER	Offline core	~USD 100-200
Proposed System	**Android, iOS (Flutter)**	**Sinhala, Tamil, English**	**On-device FER (EfficientNetB0 + CBAM, with MobileNetV2 initially evaluated) via**	**Full offline-first**	**TBD (pilot)**

			TFLite, with emotion-adaptive AAC		

Looking at the table above, three clear gaps stand out. First, no existing system supports both Sinhala and Tamil. Second, none of them use on-device FER for emotion-adaptive communication. Third, the proposed system's full offline-first design (with local storage and manual export backup) is particularly well suited to Sri Lanka's connectivity situation.

2.3 Facial Expression Recognition and Affective Computing

2.3.1 Theoretical Foundations

Affective computing concerns the development of systems that can recognise, interpret, and respond to human emotions (Picard, 2000). Facial expression recognition (FER) is a key subfield, focusing on automatically classifying emotions from facial images or video. The theoretical basis of FER derives largely from Ekman and Friesen (1971), who proposed a set of universal basic emotions linked to specific facial muscle movements described in the Facial Action Coding System (FACS). Although the universality of these categories has been questioned (Barrett et al., 2019), the Ekman framework remains the most widely used in computational FER due to the availability of large labelled datasets built around these categories (Li and Deng, 2020).

For the proposed system, six emotion classes have been selected, namely happy, sad, angry, fear, neutral, and tired. The "tired" class replaces "surprise" and "disgust" from the standard set, as fatigue detection is more clinically relevant when working with children who may need a simpler vocabulary or a break during a session.

2.3.2 Deep Learning and Transfer Learning for FER

Convolutional neural networks (CNNs) represent the current state of the art for FER, achieving the best results on standard benchmarks such as FER2013, AffectNet, and RAF-DB (Li and Deng, 2020; Goodfellow et al., 2015). CNNs learn hierarchical feature representations directly

from pixel data, progressing from low-level features (edges, textures) to high-level abstractions that distinguish emotional expressions (LeCun et al., 2015).

Transfer learning is particularly relevant to this project, where the target dataset is relatively small (2,000–5,000 images). By using a model pre-trained on ImageNet and fine-tuning it for emotion classification, the system can leverage general visual features learned from a much larger dataset (Yosinski et al., 2014).

2.3.3 Model Architecture Selection

MobileNetV2 (Sandler et al., 2018) was initially evaluated as the base architecture due to its efficiency for mobile deployment. With 3.4 million parameters, MobileNetV2 uses depthwise separable convolutions (Howard et al., 2017) to reduce computational cost by approximately 8–9 times compared to standard convolutions, enabling real-time inference on mid-range smartphones.

However, during experimentation, EfficientNetB0 combined with a CBAM (Convolutional Block Attention Module) was found to provide better feature representation and improved validation performance. EfficientNetB0 offers a more balanced trade-off between accuracy and efficiency through compound scaling, while the CBAM attention mechanism helps the model focus on the most informative facial regions. MobileNetV2 is retained as a baseline for comparison purposes.

2.3.4 FER Challenges

Several challenges are directly relevant to the design of this system.

- **Dataset bias:** Major FER datasets are predominantly composed of adult, Western, posed faces and may not generalise well to children or non-Western populations (Barrett et al., 2019; Li and Deng, 2020). Purpose-specific data collection is planned to address this.
- **Atypical expressiveness in ASD:** Children with ASD may display reduced intensity, atypical timing, or unusual facial movements (Trevisan et al., 2018). Grossard et al. (2020) found that such children produce more ambiguous expressions, leading to higher misclassification rates.

- **Environmental variability:** Real-world conditions introduce variability in lighting, head pose, and camera quality. Data augmentation (Section 2.8.2) and robust face detection preprocessing are used to mitigate these effects.
- **Inter-individual variability:** Different children express emotions differently. Confidence thresholding and caregiver override mechanisms are incorporated into the system design to address this.
- **Privacy and ethics:** FER involving vulnerable children raises significant privacy concerns, addressed in Section 2.9.

2.4 Technology Stack

2.4.1 Flutter

Flutter is an open-source UI toolkit by Google that allows building cross-platform apps from a single Dart codebase (Flutter, 2023). For this project, its main advantages include code reuse across Android and iOS, fast development cycles through hot reload, a rich set of built-in widgets, and the ability to call native code (including TensorFlow Lite) through platform channels.

2.4.2 TensorFlow Lite

TensorFlow Lite is a lightweight framework for running machine learning models on mobile devices (TensorFlow, 2023). It supports post-training quantisation, which converts 32-bit floating-point weights to 8-bit integers. This reduces the model size by about 4 times and speeds up inference by 2 to 3 times, with only a small drop in accuracy (Jacob et al., 2018). The TFLite model is packaged with the Flutter app and called through platform channels.

2.4.3 Firebase

Firebase is considered as the future backend option for this project (Firebase, 2023). At interim stage, Firebase is **not fully implemented** for end-to-end use. The system follows an offline-first approach where core AAC data is stored locally using SQLite/JSON, and the TFLite model runs entirely on the device. Backup is currently planned as a **manual export** (for example sharing a backup file through WhatsApp or file sharing). Firebase-based authentication,

Firestore, and storage are kept as planned components for later sprints when stable sync and access control are ready.

Table 3: Technology Stack Summary

Component	**Technology**	**Justification**
Mobile application	Flutter (Dart)	Single codebase for Android and iOS
AI/ML model	TensorFlow/Keras; TFLite	Transfer learning, quantisation, mobile deployment
Base architecture	EfficientNetB0 + CBAM (MobileNetV2 initially evaluated)	Lightweight, optimised for mobile
Backend database (planned)	Cloud Firestore	Planned for collaboration/sync in later stage
Authentication (planned)	Firebase Authentication	Planned for secure role-based access later
Local storage (current)	SQLite + JSON	Offline-first embedded storage and vocabulary
Text-to-speech	Platform TTS / third-party API	Spoken output in Sinhala, Tamil, English
Face detection	Google ML Kit	Lightweight, on-device

2.5 System Architecture

2.5.1 Architectural Overview

The system uses a dual-platform architecture built on a shared Flutter codebase, local storage services (SQLite and JSON), and an offline-first deployment model. Platform A and Platform

B share core AAC components, while Platform B introduces the FER module through TensorFlow Lite and platform-channel integration. This separation was selected because clinical feedback indicated that a single unified interface would not be sufficient across ASD severity levels. Children at Level 1-2 generally benefit from structured AAC with caregiver-driven customisation, whereas children at Level 3 and above require additional assistive intelligence to interpret emotional cues when functional speech is limited. The architecture therefore combines a common software foundation with severity-appropriate interaction design, while keeping the current implementation practical for low-connectivity settings through local processing and manual export backup. The system architecture and ER design artefacts are presented in Section 3.2.

1. Two client-facing mobile applications (Platform A and Platform B) built with Flutter.

2. An on-device FER module using TensorFlow Lite, integrated into Platform B via platform channels.

3. A local data layer (SQLite/JSON) for offline storage.

4. Manual export/import for backup and sharing between caregiver and therapist (current approach).

5. A therapist-parent web-based dashboard (prototype), with planned Firebase integration later.

6. A Firebase backend (planned) for authentication and optional synchronisation in later sprints.

The architecture follows an offline-first principle, meaning all core AAC features and emotion recognition run locally on the device. Firebase is not yet implemented; the current backup approach uses manual export/import. This design decision reflects the variable internet connectivity in many parts of Sri Lanka (International Telecommunication Union, 2022). The system architecture diagram and ER diagram, produced as design artefacts, are presented in Section 3.2.

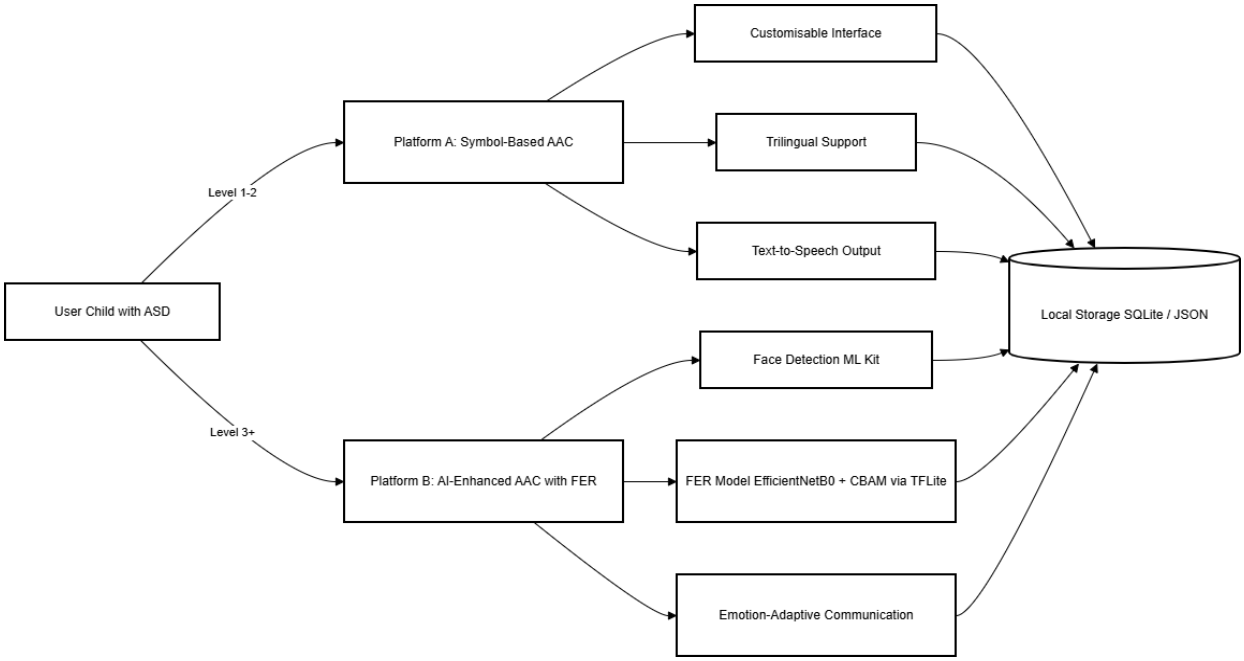


Figure 1: Dual-Platform System Architecture illustrating the separation between Platform A (Level 1-2) and Platform B (Level 3+) with local storage and core services.

2.5.2 Platform A: Customisable Symbol-Based AAC (Levels 1–2)

Platform A is designed for children at severity Level 1-2 who can interact with symbol-based communication and benefit from structured AAC support. The implementation emphasises clarity and configurability through adjustable grid sizes, trilingual language switching, text-to-speech output, caregiver and therapist configuration controls, visual scheduling, and local usage logging. This platform was included not as a secondary feature but as an essential clinical layer, because local practitioners highlighted that existing commercial tools often do not align fully with Sri Lankan language and service realities.

2.5.3 Platform B: AI-Enhanced AAC with FER (Level 3+)

Platform B is designed for children at Level 3 and above who may have minimal or no functional speech, and it represents the primary research contribution of this project. It extends Platform A with an on-device FER pipeline that analyses facial cues and supports emotion-

adaptive vocabulary prompts, while preserving caregiver override at all times. This design addresses the clinical observation that highly supported children require more than a static symbol grid, and that assistive inference can improve communication guidance when direct verbal expression is limited.

2.5.4 Emotion Detection Pipeline

The emotion detection pipeline operates in five stages. The first is face detection using Google ML Kit. The second is preprocessing, which involves cropping, resizing to 224x224 pixels, and normalisation. The third is inference via the TFLite interpreter, outputting six class probabilities. The fourth is postprocessing, where argmax determines the dominant emotion, with a configurable confidence threshold below which no adaptation is triggered. The fifth is adaptation logic, which adjusts the vocabulary and notifies the caregiver, who can override the result at any time.

2.5.5 Functional Requirements

The following functional requirements have been defined for the system.

1. The system should allow caregivers and therapists to create, edit, and manage user profiles for individual children.

2. The system should display a configurable symbol grid (2x2 to 6x6) with images and text labels organised into categories such as needs, feelings, activities, and common objects.

3. The system should support switching between Sinhala, Tamil, and English for all symbol labels and interface text.

4. The system should produce spoken output of selected symbols using the device's text-to-speech engine in the active language.

5. The system should allow caregivers and therapists to add, remove, and reorder symbols, create custom categories, and adjust visual settings.

6. The system should capture facial images via the front-facing camera, process them on-device using the TFLite model, and classify the user's expression into one of six emotion classes.

7. The system should adapt the presented vocabulary based on the detected emotion, offering contextually appropriate communication options.

8. The system should allow caregivers to override, accept, or dismiss any emotion classification at any time.

9. The system should log symbol selections, session data, and aggregated emotion history locally, with the option to export data manually for backup or sharing.

10. The system should provide a therapist-parent dashboard for reviewing usage data, managing vocabulary, and monitoring progress.

2.5.6 Non-Functional Requirements

The following non-functional requirements have been defined.

1. The system should achieve emotion inference latency of less than 500 ms on a mid-range smartphone.

2. The system should start within 3 seconds of launch.

3. The system should operate fully offline for all core AAC and FER functionality.

4. The system should conform to WCAG 2.1 AA accessibility guidelines, with configurable fonts, contrast, and layout.

5. The system should not transmit raw facial images to any cloud service; only classified emotion labels and timestamps should be stored.

6. The system should encrypt any transmitted data using TLS when synchronisation is implemented.

7. The system should support Android 8.0+ and iOS 14+.

8. The system should maintain consistent and usable layouts across different smartphone screen sizes and resolutions.

Table 4: Non-Functional Requirements Summary

Requirement	**Target**

Emotion inference latency	Less than 500 ms on mid-range smartphone
App startup time	Less than 3 seconds
Offline operation	100% core AAC and FER available offline
Accessibility	WCAG 2.1 AA with configurable fonts, contrast, and layout
Data protection	TLS encryption with no raw facial images transmitted to cloud
Platform support	Android 8.0+ and iOS 14+
Screen-size compatibility	Responsive and readable layout across tested smartphone display sizes

2.5.7 Data Model (Interim)

At the interim stage, the main entities are stored in local storage (SQLite tables and JSON vocabulary files). The data model comprises users, categories, symbols, sessions, and logs. This structure is intended to be mapped to a cloud schema if Firebase synchronisation is implemented in a later sprint. The ER diagram is presented as a completed design artefact in Section 3.2.

2.6 Development Methodology

2.6.1 Methodology Selection

The project adopts an Agile development methodology based on an adapted Scrum framework. Several alternative methodologies were considered before arriving at this decision.

- **Waterfall model.** This was not suitable because it assumes all requirements are known upfront. That does not work well for a project involving ongoing stakeholder feedback, iterative AI model training where results are unpredictable, and external dependencies such as ethics approval whose timelines cannot be fixed in advance (Sommerville, 2016).

- **Incremental model.** The incremental model offers structured, phased delivery, but it assumes a relatively stable set of requirements from the outset. In this project, requirements have evolved continuously based on field observations at Karapitiya, feedback from caregivers, and the practical outcomes of model training experiments. The rigidity of predefined increments would not have accommodated these changes well (Sommerville, 2016).
- **Spiral model.** Boehm's (1988) spiral model focuses on risk-driven development, which is relevant given the technical and ethical risks in this project. However, its formal risk analysis cycles were considered unnecessarily complex for a single-developer final year project.
- **Rapid Application Development (RAD).** RAD focuses on speed over rigour. This conflicts with the need for proper ethical documentation, careful AI model validation, and systematic testing, all of which are essential in a healthcare-related project.

Agile Scrum was selected as the best fit for this project for several reasons.

- The nature of the project is inherently exploratory. The exact requirements for the AAC interface, the symbol vocabulary, and the FER model performance could not be fully determined at the start. Each sprint revealed new information that shaped subsequent work. For example, the field visit to Karapitiya shifted the design towards a simpler symbol grid, and early model training results led to switching from MobileNetV2 to EfficientNetB0 with CBAM.
- AI model development is iterative by nature. Training runs produce results that inform the next round of experimentation. A fixed plan cannot account for this, whereas Scrum sprints allow the backlog to be reprioritised based on the latest findings.
- External dependencies such as ethics approval and hospital coordination have uncertain timelines. Scrum accommodates this by allowing tasks to be moved between sprints without disrupting the overall framework.
- The dual-platform architecture still benefits from phased delivery, with Platform A features developed before Platform B, but within a flexible sprint structure rather than rigid predefined phases.

Since this is a single-developer project, the standard Scrum framework was adapted to suit the context. The roles of developer, product owner, and scrum master were combined and carried out by the same individual. Formal team ceremonies such as daily stand-ups were not applicable, but sprint planning was conducted at the start of each sprint and a sprint review was held with the project supervisor at the end. Retrospective notes were recorded after each sprint to identify what worked well and what needed adjustment in the following cycle. A Google Sheets spreadsheet was used to manage the product backlog, sprint backlogs, and task progress throughout the project, providing a visual record of how work was prioritised and completed across sprints. This approach was preferred over tools such as Trello or Jira for two reasons. First, the project involved a single developer, and a shared spreadsheet offered a simpler, more direct way to track items without the overhead of a full project management platform. Second, Google Sheets is accessible through its mobile application, which proved particularly useful during visits to Karapitiya Teaching Hospital and informal meetings with clinical staff. Progress updates and backlog priorities could be reviewed and confirmed on the spot without requiring access to a laptop, allowing stakeholder input to be captured immediately and reflected in the sprint plan.

2.6.2 Sprint Plan

The project is organised into five sprints aligned with the academic timeline. The first two sprints are each approximately two months in duration, while the remaining three sprints are approximately one month each, reflecting the compressed schedule required to complete all deliverables within the academic year ending in May 2026.

Sprint 1 (November to December 2025), Requirements, Architecture, and Minimal AAC. This sprint covers literature review completion, requirements analysis, system architecture design, technology stack selection, and initial Flutter project setup. The deliverable is a documented architecture and a basic application skeleton.

Sprint 2 (January to February 2026), Platform A and Firebase Backend. This sprint covers the core AAC features for Platform A (symbol grid, trilingual support, basic TTS, navigation), Firebase backend configuration, and initial dashboard implementation. The deliverable is a functional (though incomplete) AAC application and backend.

Sprint 3 (March 2026), AI Model Training and Dataset Preparation. This sprint covers dataset assembly, full model training and evaluation, TFLite export, symbol set finalisation, and ethics application submission. The deliverable is a trained and exported model ready for integration.

Sprint 4 (April 2026), Platform B Integration, Dashboard, and Pilot Preparation. This sprint covers integration of the FER pipeline into the Flutter application, emotion-adaptive vocabulary logic, dashboard completion, ethics follow-up, and pilot protocol design. The deliverable is a complete system ready for pilot deployment.

Sprint 5 (May 2026), Pilot Execution and Final Report. This sprint covers supervised pilot use, data collection and analysis, final report writing, and preparation of deliverables. The deliverable is the final report and all supporting documentation.

2.6.3 Product Backlog

The product backlog represents all required features and deliverables of the system, prioritised based on stakeholder feedback, technical dependencies, and academic milestones. This backlog was continuously refined throughout the project as new information emerged from field observations, model training results, and supervisor feedback. Items were drawn into individual sprint backlogs at the start of each sprint during sprint planning.

Priority	Backlog Item	Target Sprint
High	Literature review and requirements analysis	Sprint 1
High	System architecture and technology selection	Sprint 1
High	Flutter project setup (Android and iOS targets)	Sprint 1
High	Symbol grid, navigation, and basic AAC flow	Sprint 2

High	Trilingual UI framework (Sinhala, Tamil, English)	Sprint 2
Medium	Basic TTS integration	Sprint 2
Medium	Firebase project setup (exploratory)	Sprint 2
High	AI model pipeline setup and initial training	Sprint 2-3
High	Full model training and TFLite export	Sprint 3
High	Platform B FER integration	Sprint 3
Medium	Emotion-adaptive vocabulary logic	Sprint 3
Medium	Dashboard completion	Sprint 4
High	Ethics application and approval	Sprint 4
Medium	Pilot protocol and participant recruitment	Sprint 4
High	Pilot execution and data collection	Sprint 5
High	Final report and deliverables	Sprint 5

2.6.4 Sprint Backlogs

Each sprint backlog was derived from the product backlog during sprint planning. Tasks were selected based on priority, dependencies, and project timeline constraints. For example, Sprint 2 focused on implementing the core AAC functionality, including symbol grid navigation, multilingual support, and local data storage, while Sprint 3 prioritises AI model training and integration. The tables below summarise the backlog items for each sprint.

Sprint 1 Backlog (November to December 2025)

Task	Status
Complete literature review (ASD, AAC, FER, Sri Lankan context)	Done
Gather requirements from literature and informal stakeholder discussions	Done
Design system architecture (dual-platform, offline-first)	Done
Select technology stack (Flutter, TensorFlow, Firebase)	Done
Initialise Flutter project with Android and iOS build targets	Done
Produce design artefacts (architecture diagram, ER diagram, use case diagram, class diagram)	Done

Sprint 2 Backlog (January to February 2026)

Task	Status
Implement symbol grid interface with category navigation	Done
Implement trilingual switching framework (English complete, Sinhala/Tamil partial)	In Progress
Integrate basic text-to-speech (English functional, Sinhala/Tamil under evaluation)	In Progress
Set up local data layer (SQLite, JSON vocabulary files)	Done

Create Firebase project and explore configuration	Done
Begin AI model pipeline setup (ahead of schedule from Sprint 3)	Done
Run preliminary training experiments on public datasets	Done
Conduct initial field exposure at Karapitiya Teaching Hospital	Done
Implement basic unit and widget tests	Done
Upload APK to Google Play Console for internal testing	Done

Sprint 3 Backlog (March 2026) (Current)

Task	**Status**
Finalise symbol set and resolve licensing	Pending
Complete Sinhala and Tamil vocabulary	Pending
Evaluate alternative TTS engines for Sinhala/Tamil	Pending
Complete curated dataset (2,000-5,000 images)	Pending
Full model training with hyperparameter tuning	Pending
TFLite export and on-device benchmarking	Pending
Submit ethics application	In Preparation

Sprint 4 Backlog (April 2026) (Planned)

Task	Status
Integrate FER pipeline into Flutter application	Pending
Implement emotion-adaptive logic and caregiver override	Pending
Complete therapist-parent dashboard	Pending
Follow up on ethics approval	Pending
Design pilot protocol and recruit participants	Pending

Sprint 5 Backlog (May 2026) (Planned)

Task	Status
Conduct supervised pilot at Karapitiya Teaching Hospital	Pending
Collect quantitative and qualitative data	Pending
Analyse pilot findings	Pending
Write final report and compile deliverables	Pending
Prepare and deliver final presentation	Pending

Sprint management and task tracking were carried out using a Google Sheets spreadsheet, with columns for Backlog, To Do, In Progress, Review, and Done. This provided a lightweight but effective tracking mechanism suited to a single-developer workflow. The spreadsheet was also accessible via the Google Sheets mobile application, which allowed backlog items and task statuses to be reviewed or updated during hospital visits and stakeholder discussions without requiring a laptop. The sprint tracking spreadsheet is presented as Figure 20 in Section 5.4.

2.7 Research Methodology

2.7.1 Research Type

This project is classified as applied research. Unlike pure or theoretical research, which aims to generate new knowledge for its own sake, applied research focuses on solving a specific, real-world problem using existing knowledge and techniques (Kothari, 2004). In this case, the problem is the absence of a culturally and linguistically appropriate AAC system with emotion recognition for children with autism in Sri Lanka. The project applies established technologies and frameworks, including deep learning for facial expression recognition, Flutter for cross-platform mobile development, and AAC design principles from the literature, to develop a practical solution that addresses this identified gap.

Applied research was considered the most appropriate classification because the primary objective is not to advance theoretical understanding of autism, AAC, or deep learning as academic disciplines, but rather to design, build, and evaluate a working system that can be used by real children, caregivers, and therapists in a Sri Lankan clinical and home context.

2.7.2 Justification for Mixed-Methods Approach

A mixed-methods research approach was selected for this project because it involves both technical system evaluation and human-centred usability assessment, and neither a purely quantitative nor a purely qualitative strategy would be sufficient on its own. The performance of the facial expression recognition model requires objective, numerical evaluation through classification metrics, which is inherently quantitative. At the same time, the usability, acceptance, and real-world effectiveness of the AAC system can only be understood through the subjective insights and lived experiences of caregivers, therapists, and observers, which is inherently qualitative. Combining both approaches provides a more comprehensive evaluation of the system than either method could achieve alone. Creswell and Creswell (2018) argue that mixed-methods designs are particularly appropriate when a research problem requires both numerical performance data and contextual human insight, which is precisely the case here.

A convergent mixed-methods design was chosen, where quantitative and qualitative data are collected in parallel during the pilot phase and then compared to provide a more complete picture of the system's effectiveness and usability.

2.7.3 Quantitative Evaluation

The quantitative component of the evaluation serves as the primary means of objectively measuring whether the system meets its defined requirements. This component is divided into two areas.

The first is FER model performance. The trained model is evaluated using standard classification metrics, including overall accuracy, per-class precision, recall, F1-score, and a confusion matrix. These metrics are computed on a held-out test set that was not used during training. The target accuracy is at least 80 per cent across six emotion classes. If the model falls below this threshold, the results will be analysed to determine which classes are underperforming and what corrective measures (such as additional data collection or class rebalancing) are needed.

The second is system usability and user experience. During the pilot phase, structured questionnaires will be distributed to caregivers and therapists who observe or facilitate the use of the application with children. These questionnaires will use Likert scale responses (1 to 5) to measure usability, satisfaction, and system effectiveness across dimensions such as ease of use, symbol relevance, navigation clarity, response time, and overall satisfaction. The System Usability Scale (SUS) will also be applied as a standardised usability measure (Brooke, 1996). Questionnaire responses will be analysed using descriptive statistics to identify patterns in user satisfaction and areas requiring improvement.

Non-functional requirements, including emotion inference latency (target under 500 ms), application startup time (target under 3 seconds), and offline reliability, will also be measured quantitatively on test devices.

2.7.4 Qualitative Evaluation

Qualitative evaluation will be conducted through semi-structured interviews with caregivers and therapists following the pilot period. These interviews will explore user experience,

perceived usefulness of the system, challenges faced during use, whether the symbol vocabulary was culturally and linguistically appropriate, how useful the emotion detection feature was perceived to be, and suggestions for improvement.

Observational notes will also be recorded during pilot sessions, documenting how children interact with the interface, which symbols are used most frequently, how caregivers intervene or support the interaction, and any behavioural responses to the emotion-adaptive features. Qualitative data will be analysed thematically to identify recurring patterns and insights that the quantitative data alone may not capture.

2.7.5 Evaluation Process and Drawing Conclusions

The overall evaluation process will follow a structured sequence. First, the FER model will be evaluated against its performance targets using the held-out test set, and the results will be documented with supporting metrics and visualisations such as the confusion matrix and training curves. Second, during the pilot study at Karapitiya Teaching Hospital, quantitative data (questionnaire responses, usage logs, SUS scores) and qualitative data (interview transcripts, observational notes) will be collected in parallel from caregivers and therapists.

Once all data has been collected, the quantitative results will be summarised using descriptive statistics and compared against the defined functional and non-functional requirements. The qualitative findings will be organised through thematic analysis, identifying common themes such as perceived ease of use, barriers to adoption, and suggestions for feature improvement.

Conclusions will be drawn by triangulating the quantitative and qualitative findings. For example, if the SUS score indicates moderate usability but interview responses reveal specific navigation difficulties, the conclusion would identify targeted improvements rather than a general assessment of success or failure. This triangulation approach is intended to produce conclusions that are grounded in measurable evidence while also reflecting the practical realities of use in the Sri Lankan context. The final report will present these conclusions alongside specific recommendations for future development and deployment.

The development follows an Agile Scrum methodology (described in Section 2.6), with the application built and tested in iterative sprints. For the AI component, training data is sourced

from publicly available datasets in the first instance, with purpose-collected data planned following ethics approval.

2.8 AI Model Design and Data Collection

2.8.1 Transfer Learning Strategy

While MobileNetV2 was initially evaluated as a baseline model, the final training pipeline uses EfficientNetB0 with a CBAM attention module, as it showed better feature representation and improved validation performance.

The FER component uses EfficientNetB0 pre-trained on ImageNet as a feature extractor, and an attention module (CBAM) is added to help the model focus on useful facial regions. The transfer learning approach involves the following steps.

1. Loading EfficientNetB0 without the top classification layers, retaining the pre-trained convolutional layers.

2. Freezing the base model weights during initial training to prevent destruction of pre-learned features.

3. Adding CBAM attention and a custom classification head consisting of global average pooling, dense layers with dropout, and a six-unit output layer with softmax activation.

4. Training the classification head on the emotion dataset.

5. Optionally fine-tuning the full model with a low learning rate.

For training, mixed precision was used to reduce memory usage (float16), but float32 was kept in the final layers to avoid instability. This hybrid approach was mainly done to get better performance in low memory mode. Over time, epoch counts were also tuned (increased in experiments) to improve accuracy step by step.

2.8.2 Data Augmentation

Data augmentation increases the effective diversity of the training set by applying label-preserving transformations during training (Shorten and Khoshgoftaar, 2019).

Table 5: Data Augmentation Techniques

Technique	Parameter Range	Rationale
Horizontal flip	50% probability	Faces are approximately symmetrical
Rotation	Plus or minus 15 degrees	Simulates head tilt
Zoom	Plus or minus 10%	Simulates varying camera distances
Brightness	Plus or minus 20%	Simulates varying lighting
Contrast	Plus or minus 20%	Simulates varying image quality
Translation	Plus or minus 10% horizontal/vertical	Simulates imperfect face centering

2.8.3 Model Quantisation

Post-training quantisation converts model weights from 32-bit floating point to 8-bit integers, reducing model size by approximately 4x and improving inference speed by 2-3x with minimal accuracy loss (typically less than 1-2 percentage points) (Jacob et al., 2018). Dynamic range quantisation is applied as the default.

2.8.4 Evaluation Metrics

The model is evaluated using the following metrics.

- **Confusion matrix:** Reveals which emotion classes are most often confused.
- **Per-class precision, recall, and F1 score:** Precision measures the proportion of correct positive predictions; recall measures the proportion of actual positives correctly identified; F1 is their harmonic mean.
- **Overall accuracy:** The proportion of correct predictions to total predictions.
- **Inference time and model size:** Measured on a mid-range smartphone to verify non-functional requirements.

2.8.5 Dataset Composition

The target dataset is 2,000-5,000 labelled facial images across six emotion classes.

Table 6: Dataset Distribution by Emotion

Emotion Class	**Target Count (approx.)**	**Percentage**
Happy	350-850	~17%
Sad	350-850	~17%
Angry	300-750	~15%
Fear	250-650	~13%
Neutral	400-1000	~20%
Tired	350-900	~18%
Total	**2,000-5,000**	**100%**

Data sources include existing public datasets (FER2013, RAF-DB, AffectNet) filtered and relabelled for the target classes, and purpose-collected data (post-ethics approval) from consenting participants at Karapitiya Teaching Hospital. Data collection protocols define consent procedures, capture conditions, and labelling procedures with inter-rater reliability checks.

2.9 Ethical Considerations

2.9.1 Overview

Since this study involves vulnerable participants (children with ASD), sensitive data (facial images, usage logs), and deployment in a healthcare-related context, ethical considerations have been treated as fundamental design constraints from the start. The ethical framework draws on the Declaration of Helsinki (World Medical Association, 2013), the Belmont Report, and emerging guidelines for ethical AI deployment (Floridi et al., 2018; Jobin et al., 2019).

2.9.2 Informed Consent

Consent is obtained from the parent or legal guardian, with assent sought from the child in an accessible form where possible. The child's willingness to engage is monitored throughout; signs of distress or unwillingness are treated as withdrawal of assent. Draft consent forms and participant information sheets have been prepared in English and are included in the appendices. Sinhala and Tamil translations are planned for the next phase.

2.9.3 Privacy and Data Protection

Specific measures are outlined below.

- **On-device processing:** FER is performed entirely on the device. Raw facial images are not transmitted to any server.
- **No default image storage:** Only the classified emotion label and timestamp are logged.
- **Access control:** If Firebase is implemented later, security rules will restrict access to authorised users. For the current offline-first stage, access is mainly controlled by the device/user context.
- **Encryption:** When any data is transmitted (for example future sync), it will be protected using TLS. Firebase encryption at rest applies if Firebase storage is used later.
- **Retention policies:** Defined in the ethics protocol with secure deletion after the project analysis period.

2.9.4 Minimisation of Harm

- **Caregiver override:** Emotion recognition can be disabled or overridden at any time.
- **Confidence thresholding:** Low-confidence predictions are not acted upon.
- **Transparency:** The system presents emotion classification as a suggestion, not a certainty.
- **Voluntary participation:** Participants can withdraw at any time without penalty.

2.9.5 Institutional Approval

A pilot at Karapitiya Teaching Hospital requires approval from the hospital's institutional ethics committee and alignment with Ministry of Health research governance requirements (Ministry of Health, Sri Lanka, 2020). The ethics application includes the research protocol, consent forms, and data management plan.

Table 7: Ethical Risk Assessment

Ethical Risk	**Likelihood**	**Impact**	**Mitigation**
Consent not fully informed	Low	High	Mitigation measures include multilingual information sheets, child assent, and continuous monitoring during all sessions.
Privacy breach (facial images)	Low	Very High	Mitigation measures include on-device processing, no default image storage, and encryption for any retained data.
Misclassification of emotion	Medium	Medium	Mitigation measures include confidence thresholding, caregiver override controls, and clear communication that

			predictions remain suggestive.
Child distress during data collection	Low	High	Mitigation measures include trained data collectors, predefined stop criteria, and parental presence during sessions.
Delayed ethics approval	Medium	Medium	Mitigation measures include early submission and a prepared alternative evaluation plan while approvals are pending.

2.10 Risk Analysis

Table 8: Risk Assessment Matrix

Risk	Category	Likelihood	Impact	Mitigation
Emotion model accuracy below 80%	Technical	Medium	High	Mitigation measures include data augmentation, class rebalancing, and targeted

				hyperparameter tuning.
Poor performance on low-end devices	Technical	Medium	Medium	Mitigation measures include quantisation, lower-resolution inputs, and clear minimum device specifications.
Sync conflicts in offline-first design	Technical	Medium	Medium	Mitigation measures include a defined conflict resolution policy and comprehensive integration testing.
Delayed ethics/Ministry approval	Project	High	High	Mitigation measures include early submission and an alternative interim evaluation pathway.

Limited therapist/parent availability	Project	Medium	Medium	Mitigation measures include flexible scheduling and remote participation where appropriate.
Consent or data breach	Ethical	Low	Very High	Mitigation measures include strict access controls, encryption, and a documented incident response plan.
Sinhala/Tamil TTS quality insufficient	Technical	Medium	Medium	Mitigation measures include multiple TTS engines and a recorded-audio fallback approach.
Scope creep	Project	Medium	Medium	Mitigation measures include defined scope boundaries and regular

				supervisor review meetings.

The critical path runs through ethics approval, dataset collection, model training, Platform B integration, pilot execution, and the final report. The primary contingency is to proceed with model training on public data and complete Platform A independently.

2.11 Limitations and Scope

The scope of this work includes the design, development, and pilot evaluation of the dual-platform AAC system with FER in the Sri Lankan context. What falls outside the scope includes full randomised controlled trials, longitudinal studies, nationwide deployment, support for languages beyond Sinhala, Tamil, and English, and integration of modalities other than FER.

Known limitations are as follows.

1. Dataset representativeness: The model may not fully represent the diversity of Sri Lankan children, and performance for children with ASD may differ from neurotypical populations.

2. Pilot constraints: Sample size and duration will be constrained by ethics approval timelines and participant availability. Findings should be interpreted as preliminary and formative.

3. Atypical expressiveness in ASD: The model's ability to recognise emotions in children with ASD is an open question.

4. TTS quality: Text-to-speech quality for Sinhala and Tamil may vary across engines.

5. Single-developer constraints: As a solo project, the Scrum framework was adapted with all roles combined into one. This limits the breadth of testing, peer review, and formal usability evaluation.

6. Device variability: Performance may vary across devices; testing focuses on representative mid-range devices.

3. Work Completed

3.1 Summary of Progress

At the time of this interim submission, Sprint 1 has been completed in full and Sprint 2 is substantially complete. The literature review, system architecture, technology stack selection, and the initial Flutter project setup are all in place. A considerable amount of early effort went into understanding the problem domain, not only through published literature but also through informal conversations with parents and therapists. Those conversations surfaced practical realities that the literature alone did not fully convey, and they directly influenced several design decisions described in the sections that follow.

The system was implemented as a dual-platform solution consisting of Platform A and Platform B, each targeting different ASD severity levels. Platform A, designed for Level 1-2 users, was developed as a customisable symbol-based AAC interface with multilingual support and caregiver-controlled configuration. Core features such as the symbol grid, category navigation, and text-to-speech functionality were successfully implemented and tested on real devices.

Platform B, designed for Level 3 and above, extends this functionality by integrating an on-device facial expression recognition (FER) pipeline. The FER component includes face detection, image preprocessing, model inference using an EfficientNetB0-based architecture with CBAM, and emotion-adaptive response generation. Initial integration of the TensorFlow Lite model into the Flutter application was completed, and preliminary testing confirmed real-time inference capability within acceptable performance limits.

The decision to implement two separate platforms was influenced by clinical observations and initial field exposure at Karapitiya Teaching Hospital. Feedback from clinicians indicated that children at different ASD severity levels require significantly different interaction designs, particularly in terms of interface complexity and assistive support. As a result, Platform A focuses on structured and customisable communication, while Platform B introduces intelligent assistance through emotion recognition.

Some work originally planned for Sprint 3, particularly the AI model pipeline setup and preliminary training experiments, was started ahead of schedule during Sprint 2. This provides a useful head start on model development for the next phase.

The larger tasks remain ahead. Full dataset curation using purpose-collected data, complete model training and evaluation, Platform B integration with the FER pipeline, and the clinical pilot study are all planned for subsequent sprints. There have been some delays in specific areas, notably symbol set licensing, ethics coordination, and TTS quality for Sinhala and Tamil, but these are accounted for in the revised project plan (Section 5.4). Overall, the project is in a reasonable position for this stage of the academic timeline.

3.2 Requirements, Design, and Produced Artefacts

A comprehensive literature review covering ASD, AAC, facial expression recognition, and the Sri Lankan context was completed and is presented in Sections 1 and 2. Requirements for both platforms were gathered through published literature analysis, a review of existing AAC systems, and informal discussions with a speech-language therapist and two parents of children with ASD. The architecture, data flow, and technology choices have been documented, together with user roles (child, parent/caregiver, therapist) and use case mapping.

Key design decisions made during this phase include the dual-platform approach to address different ASD severity levels, the selection of Flutter for cross-platform development, the initial evaluation of MobileNetV2 for on-device FER with subsequent adoption of EfficientNetB0 with CBAM for improved feature extraction, and the offline-first architecture to accommodate areas with unreliable connectivity. Firebase remains a planned option for future synchronisation, while the current approach uses local storage with manual export backup.

To support the design phase of the project, several modelling artefacts were produced, including a system architecture diagram, an entity relationship diagram, a use case diagram, and a class diagram. These artefacts were used to clarify user interactions, define the data structure, and guide the system architecture prior to and during implementation. All figures presented in this section include appropriate captions in accordance with academic reporting standards.

Figure 2 presents the overall system architecture, showing the mobile AAC applications, the local storage layer, the on-device FER module, and the planned cloud components.

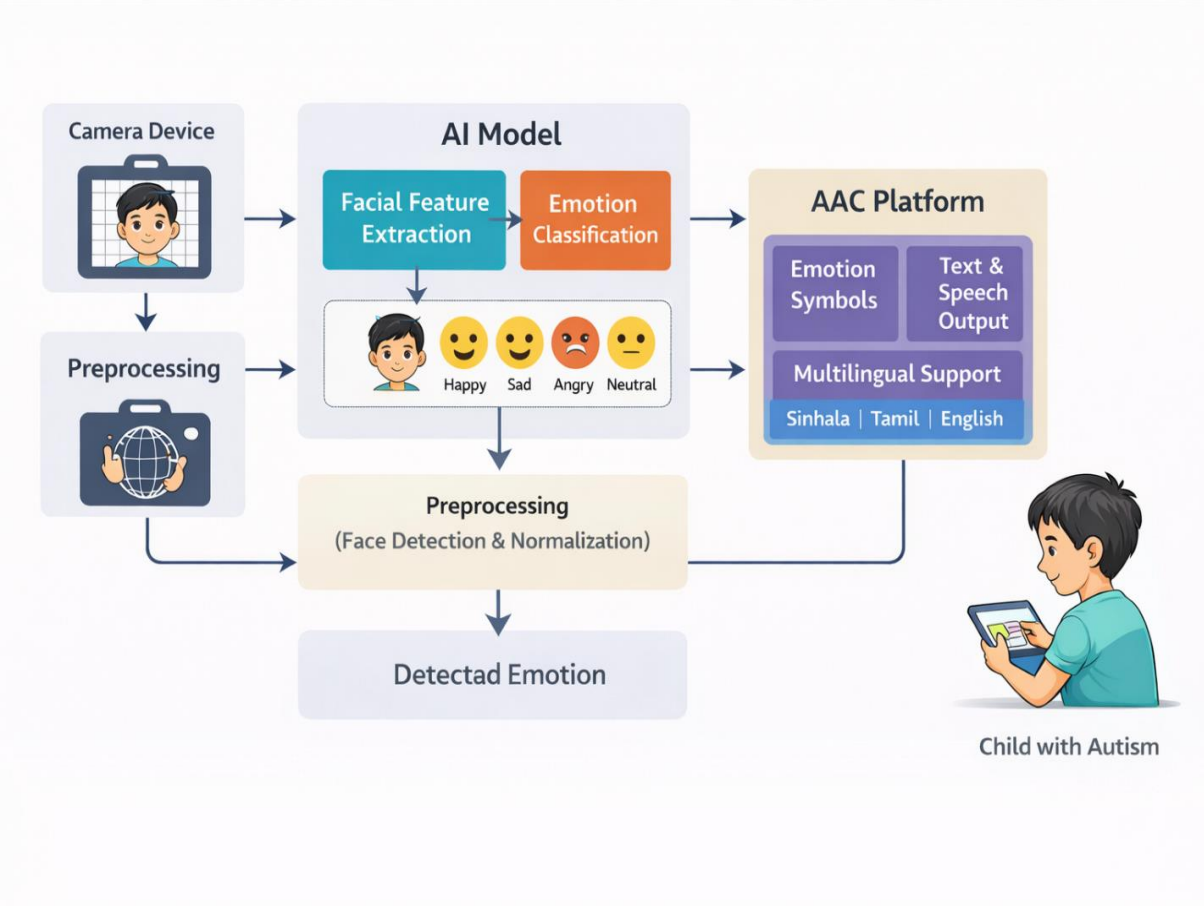


Figure 2: System Architecture Diagram

The diagram highlights the end-to-end flow from user interaction to system response. Core AAC functions are delivered locally through Platform A and Platform B, while emotion inference is processed on device through the FER pipeline in Platform B before adaptive prompts are presented. This structure supports low latency communication assistance and remains operational in environments with limited internet connectivity.

3.2.1 Entity Relationship Diagram

The entity relationship diagram represents the logical data structure of the AAC system at the interim stage. It models the key entities required to support communication, user management, and monitoring features within the application.

The system should store data related to users, child profiles, categories, symbols, sessions, usage logs, and emotion records in a structured manner. Child profiles are linked to sessions, symbol selections, and emotion records, enabling the system to maintain a consistent history of AAC interactions and detected emotional states.

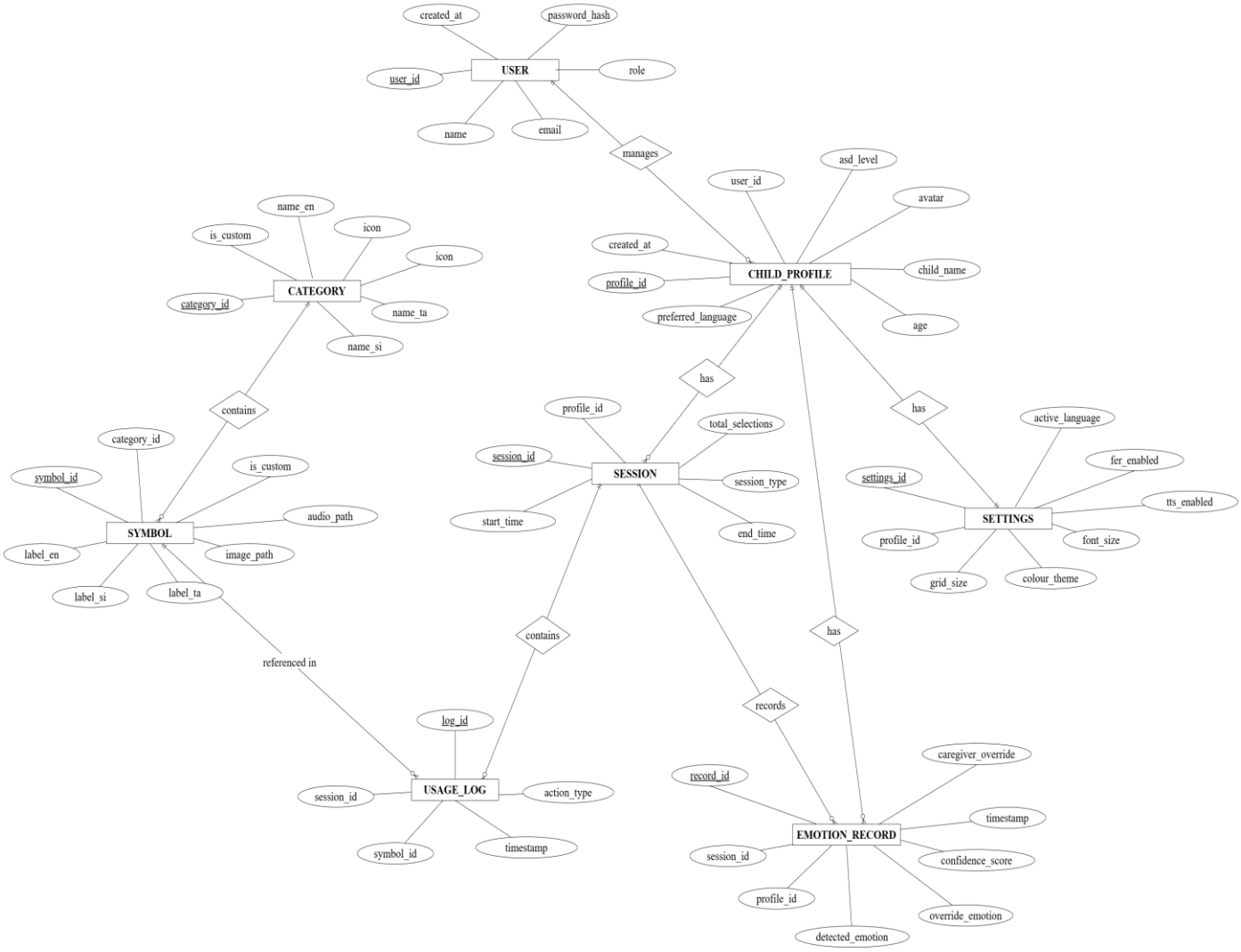
The system should support category and symbol entities to manage vocabulary in a scalable and customisable way. The system should also allow session based logging to capture user interactions over time, which can later be reviewed through the caregiver and therapist dashboard.

At this stage, the data model is implemented using local storage technologies such as SQLite and JSON files, in line with the offline first design approach. The ER diagram also provides a foundation for potential future integration with a cloud based database if synchronisation features are implemented.

Figure 3: Entity Relationship Diagram

Download Link : https://drive.google.com/file/d/1kRURwi8_up3HiGxyBgYpFTq-omLEjHHB/view?usp=sharing

ERDiagram of the AI-Enhanced AAC System



3.2.2 Use Case Diagram

The use case diagram models the primary interactions between the system and its three main user roles, namely the child, the parent or caregiver, and the therapist. It provides a high level representation of how each actor engages with the AAC system.

The child primarily interacts with communication focused features. The system should allow the child to select symbols from a grid based interface, listen to text to speech output, browse categories, view a visual schedule, and use the facial expression recognition functionality. These interactions represent the core AAC usage flow of the application.

The parent or caregiver plays both a support and management role. The system should allow caregivers to create and manage user profiles, switch between Sinhala, Tamil, and English, view detected emotions, override emotion classifications, manage symbols and categories, configure the grid size and theme, record custom audio, export and import backups, and manage child profiles.

The therapist interacts mainly with monitoring and administrative features. The system should allow therapists to access the dashboard, review usage logs and progress, view emotion history, manage vocabulary, and manage child profiles.

This diagram was used to define the functional scope of the system and to verify that all user roles and interactions had been accounted for.

Download Link :
https://drive.google.com/file/d/1iq6jvoSSK03f0fsV_psegGihgB3PPITE/view?usp=sharing

Figure 4: Use Case Diagram of the AAC System

3.2.3 Class Diagram

The class diagram describes the structural design of the AAC system at the software level by identifying the main classes and their relationships.

The system should include core classes such as User, ChildProfile, Category, Symbol, and SymbolGrid to support profile management and symbol based communication. The system should also include supporting classes such as TTSEngine, Session, and UsageLog to handle speech output and interaction tracking.

For the AI enhanced component, the system should include classes such as FERModule, FaceDetector, EmotionClassifier, AdaptationEngine, and CaregiverOverride. These classes work together to process facial input, classify emotions, and adapt the AAC interface accordingly.

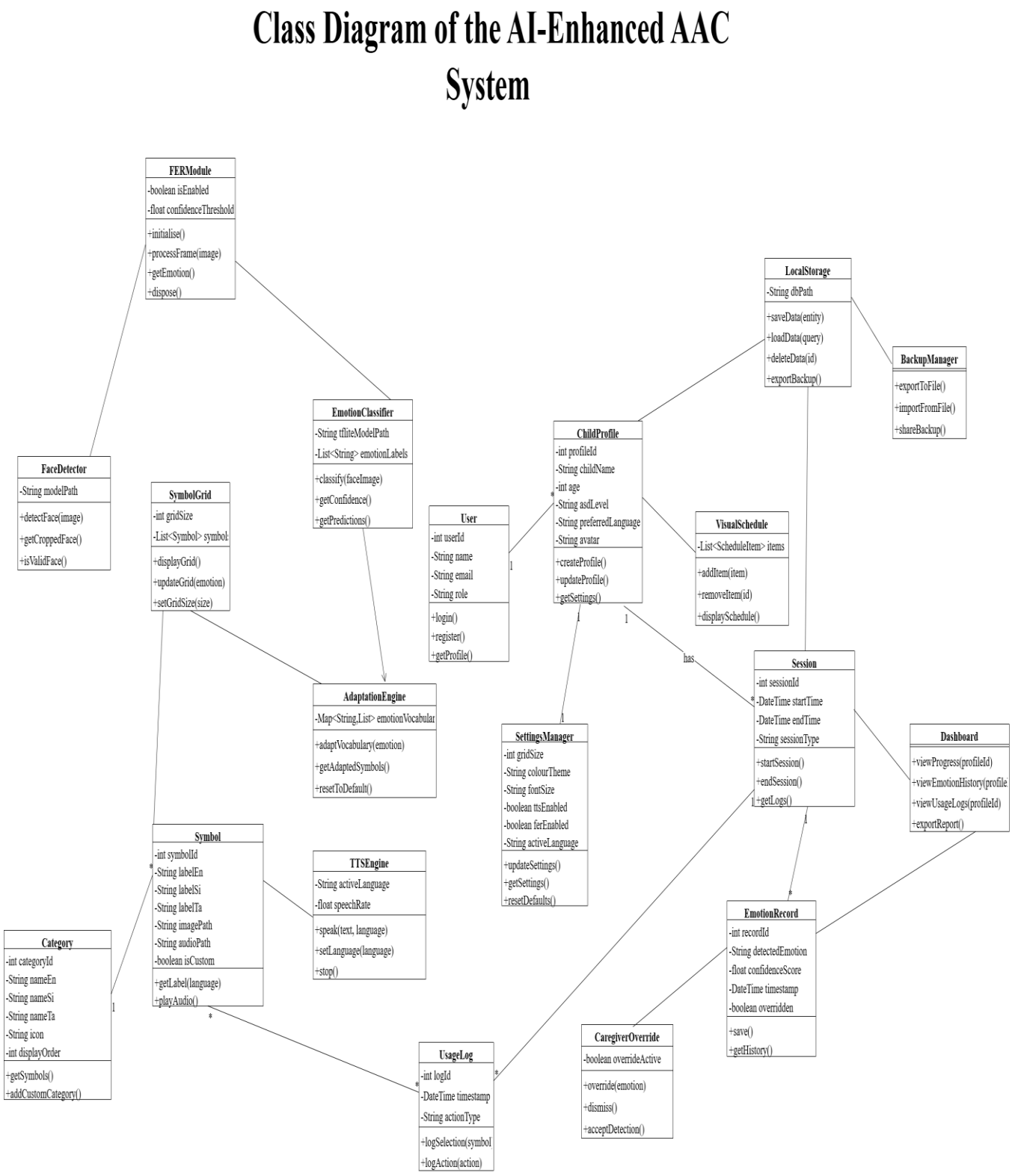
Additional classes such as LocalStorage, BackupManager, Dashboard, VisualSchedule, and SettingsManager are included to support data storage, backup handling, monitoring, scheduling, and personalisation features.

The relationships between these classes illustrate how system components interact. For example, a child profile is associated with sessions and settings, while a session contains usage logs and emotion records. The FER module depends on face detection and emotion classification, and the adaptation engine modifies the symbol grid based on the detected emotional state.

Structuring the system at this level of detail was intended to keep it modular and maintainable as development progresses.

Download Link :
https://drive.google.com/file/d/1lsY4kLsz48pxip9bFGNWLEc7vAcfNSny/view?usp=sharing

Figure 5: Class Diagram of the AAC System



3.3 Development Environment and Core AAC

The application was developed using Visual Studio Code as the primary IDE, with Flutter SDK (Dart SDK >=2.19.0) as the development framework. The project follows a standard Flutter folder structure with separate directories for models, services, screens, widgets, and utilities. It targets Android 8.0+ and iOS 14+ from a single Dart codebase.

The following table lists the key development libraries and packages used in the project, as specified in the project dependency file (`pubspec.yaml`). These packages were selected based on their relevance to the project requirements, community support, and compatibility with the offline-first architecture.

Table 9: Development Libraries and Packages

Package	**Version**	**Purpose**	**Justification for Selection**
flutter (SDK)	>=2.19.0	Cross-platform mobile framework	Single codebase for both Android and iOS, large community support, faster iteration through hot reload, and native performance through platform channels
cupertino_icons	^1.0.6	iOS-style icon set	Provides platform-consistent iconography for iOS users without requiring separate asset bundles

flutter_tts	^3.8.3	Text-to-speech output	Most widely adopted Flutter TTS package, supports multiple languages including Sinhala and Tamil engine access, and works offline
shared_preferences	^2.2.2	Local key-value storage	Lightweight and simple for storing user settings and preferences, has no server dependency, and supports the offline-first architecture
google_mobile_ads	^4.0.0	Advertisement integration	Planned for a future sustainability model to keep the application free for end users, and is the official Google package with strong documentation
http	^1.1.0	HTTP client	Standard Dart HTTP package for planned payment gateway and optional API calls, with a minimal footprint

connectivity_plus	^5.0.0	Network connectivity detection	Required for offline-first behaviour, and detects connection state changes so the application can switch between offline and online modes
camera	^0.11.0+2	Camera access	Official Flutter camera plugin that provides access to the front-facing camera required for facial expression capture in Platform B
tflite_flutter	^0.12.1	On-device ML inference	Viable option for running TensorFlow Lite models within Flutter via platform channels, and enables on-device emotion classification without cloud dependency
image	^4.1.7	Image processing	Needed for preprocessing camera frames

			(cropping, resizing to 224x224, normalisation) before passing them to the TFLite model
flutter_test (dev)	SDK	Unit and widget testing	Built-in Flutter testing framework with no additional dependency required, and supports mock-based testing of AAC and FER logic
flutter_launcher_icons (dev)	^0.13.1	App icon generation	Automates icon generation for both Android and iOS from a single source image, and reduces manual configuration
flutter_lints (dev)	^6.0.0	Code quality	Enforces recommended Dart coding standards and best practices through static analysis

Custom assets bundled with the application include the TensorFlow Lite emotion model (`emotion_model.tflite`), emotion class labels (`labels.txt`), and the NotoColorEmoji font for consistent emoji rendering across devices.

For the AI model training component, the following Python libraries were used within Google Colab.

Table 10: AI Model Training Libraries (Google Colab)

Library	Purpose	Justification for Selection
TensorFlow / Keras	Model training and evaluation	Industry-standard deep learning framework that supports transfer learning with EfficientNetB0, and direct TFLite export for mobile deployment
NumPy	Array manipulation and computation	Required dependency for TensorFlow, and used for data preprocessing and numerical operations
Matplotlib	Training curve visualisation	Standard plotting library used to visualise loss and accuracy curves across training epochs for model evaluation
Pillow (PIL)	Image loading and preprocessing	Widely used image library that handles loading, resizing, and format conversion of facial expression images
scikit-learn	Classification metrics and confusion matrix	Provides precision, recall, F1-score, and confusion matrix functions required for model evaluation against the 80% accuracy target

tf.data API	Optimised data pipeline	Built into TensorFlow, enables efficient batching, shuffling, and augmentation during training, and reduces GPU idle time

Core AAC features for Platform A have been partially implemented.

- **Navigation and routing:** Screen navigation uses Flutter's Navigator 2.0 pattern, covering the main AAC grid view, category selection, settings, and profile screens.
- **Symbol grid interface:** A basic symbol grid is functional, displaying images with text labels in preliminary categories (needs, feelings, common objects). Grid size is currently fixed; configurable sizes (2x2 to 6x6) are planned for the next sprint.
- **Multilingual support:** The UI framework supports switching between Sinhala, Tamil, and English. Labels and interface text are stored in separate localisation files. The English vocabulary is mostly populated; Sinhala and Tamil vocabularies require further work.
- **Local data layer:** SQLite is used for structured data and local JSON files store vocabulary definitions, enabling full offline operation for vocabulary access and settings.
- **Text-to-speech:** Basic TTS integration using the device's built-in engine is functional for English. Preliminary testing indicated that the default platform TTS for Sinhala and Tamil produces unnatural output, and third-party TTS services will need to be evaluated.

Getting the interface right took considerably longer than the original timeline anticipated. Children with autism can disengage entirely when a screen feels cluttered or visually overwhelming (Fletcher-Watson and Happe, 2019), so every colour choice, every icon, and every tap target required deliberate consideration. Several versions were tested before the current soft palette and rounded component style was settled upon.

The symbol set proved more challenging than expected. Most established AAC symbol libraries are designed for Western contexts, and the food symbols typically show sandwiches

and hamburgers, not rice, kottu, or string hoppers. A significant amount of manual sourcing and adaptation was required to build a culturally appropriate vocabulary, and licensing terms added further complexity. This work remains ongoing and has been prioritised for the next sprint.

3.4 AI Model Pipeline

Model training was conducted on Google Colab with GPU support, which proved workable but not without friction. The free tier imposes session time limits, and on several occasions training runs were interrupted before completion. In some early instances, unsaved progress was lost entirely. After those setbacks, checkpoint saving was adopted far more aggressively, which added overhead but prevented further losses.

The model architecture went through a deliberate process of iteration. MobileNetV2 was evaluated first as a baseline, being lightweight, well documented for TFLite conversion, and a reasonable starting point for mobile deployment. However, validation performance did not reach the required level. After further experimentation, EfficientNetB0 combined with a CBAM (Convolutional Block Attention Module) was selected, producing noticeably better feature representation.

The training configuration is summarised below.

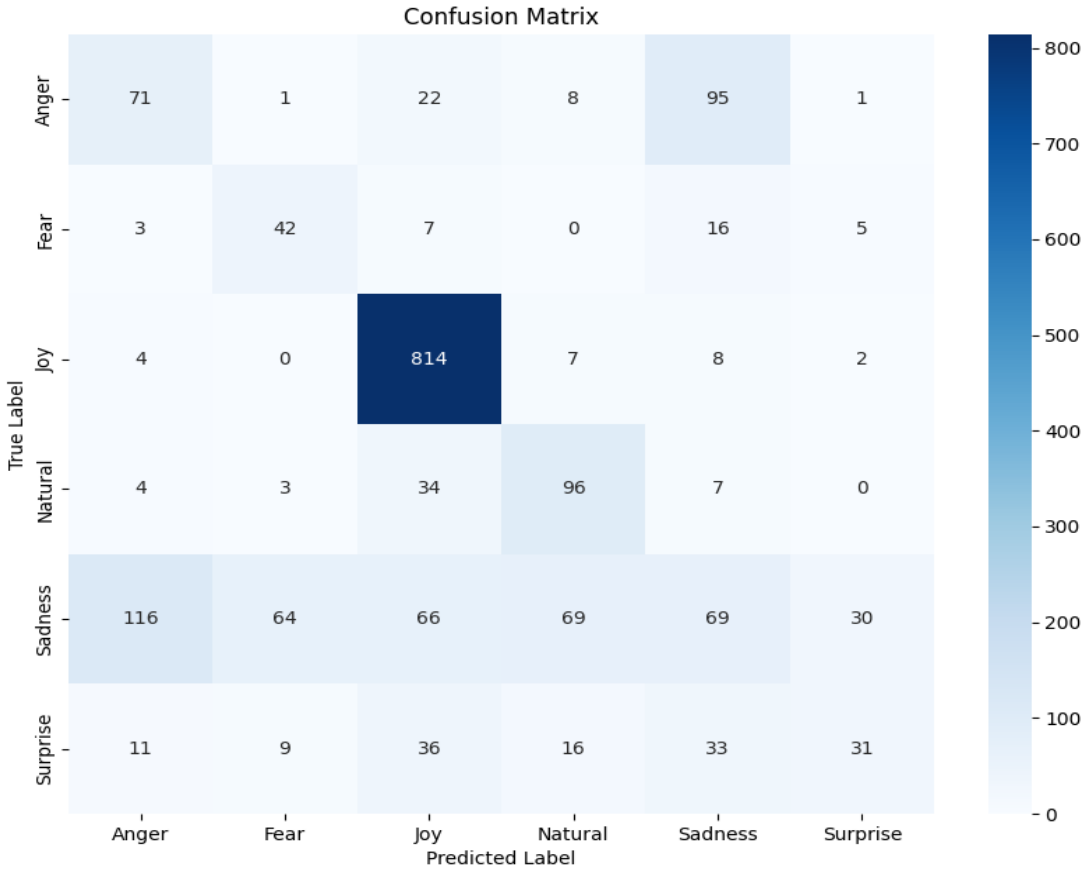
- Input size: 224x224
- Batch size: 16 (constrained by GPU memory limitations)
- Mixed precision training: float16 for speed and memory efficiency, with float32 retained in the final layers for numerical stability
- Dataset pipeline: tf.data API with optimised loading
- Class balancing: sample_from_datasets to address class imbalance
- Data augmentation: random flip, rotation, zoom, and contrast adjustments
- MixUp regularisation to improve generalisation
- Regularisation: dropout (0.4), L2 regularisation, and label smoothing
- Optimiser: AdamW

- Learning rate scheduling: ReduceLROnPlateau
- Training control: EarlyStopping

Current results show training accuracy of approximately 85–87% and validation accuracy of 82–84%, though test accuracy sits lower at around 66%. Closing that generalisation gap is the primary model objective for the next phase. The model learns the training and validation distributions reasonably well, but still struggles with unseen real world samples, which is a recognised challenge in FER when datasets are limited or imbalanced (Li and Deng, 2020). Per class analysis shows strong performance on the "happy" class, with weaker results for "fear", "sad", and "angry", most likely attributable to class imbalance and environmental variability such as lighting and head pose. Improving dataset balance and adding more diverse samples for the weaker classes will be the focus going forward.

Figure 6 presents the confusion matrix for the current FER model, illustrating which emotion classes are classified correctly and where misclassifications are concentrated.

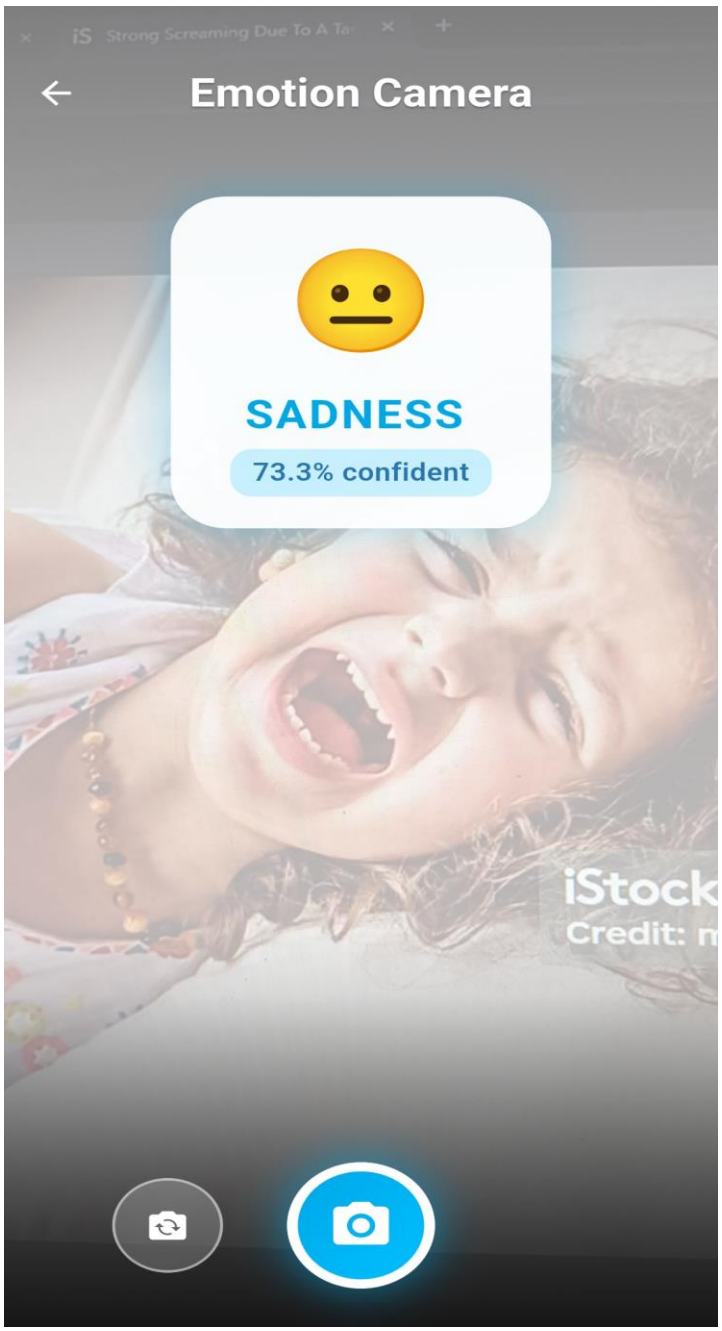
Figure 6: Confusion Matrix



Multiple model variants were trained by adjusting epoch counts and precision settings (float32, float16, and mixed). The final model was selected based on validation performance and stability rather than training accuracy alone. Full training on the complete dataset (2,000–5,000 images) will be conducted once the dataset is fully prepared, including purpose-collected data following ethics approval.

Figure 7 shows the facial expression recognition interface in Platform B, demonstrating on-device emotion detection using the front-facing camera and TFLite model. The detected emotion is used to adapt the AAC vocabulary accordingly, supporting contextually appropriate communication.

Figure 7: Facial Expression Recognition Screen (On-Device Detection)



3.5 Backend and Dashboard

Backend work is at an early, prototype level. A Firebase project has been created and basic configuration has been explored, but full end-to-end integration has not been completed. The current data store is local (SQLite/JSON), and backup is handled through manual export and

import. A basic therapist-parent dashboard prototype has been started, including login UI and navigation screens, placeholder progress views, and an early vocabulary management interface. Full account linking and Firebase synchronisation are planned for a later stage once the offline core is stable.

3.6 Ethics and Partnership

Draft consent forms and participant information sheets have been prepared in English (included in the appendices). Sinhala and Tamil translations are planned for the next phase. Initial contact with Karapitiya Teaching Hospital has been established, and the formal ethics application is in advanced preparation. Ministry of Health approval requirements have been researched and documented.

3.7 Testing

Testing at the interim stage has focused on unit tests for the most critical parts of the codebase. The data layer functions, symbol handling logic in the AAC module, and the emotion-based decision logic in the FER integration have all been covered with automated tests. The AI model itself runs as a TensorFlow Lite component on the device, which makes direct unit testing of the full inference pipeline impractical at this stage. Instead, mock-based tests were written to validate the output handling, for example confirming that the correct dominant emotion is selected from a set of prediction probabilities.

All implemented tests were executed using Flutter's built-in testing framework and passed without failure. That said, full test coverage has not been achieved. During this phase, development priority was given to getting core features functional, with the understanding that test coverage would be expanded in subsequent sprints.

Although the application is designed for both Android and iOS platforms using Flutter, all testing conducted at the interim stage has been limited to the Android platform. This is primarily due to the unavailability of an Apple Developer account at this stage. Manual testing of the basic AAC communication flow, including symbol selection, category navigation, and text-to-speech output, has been carried out on an Android emulator and one physical Android

device. Therefore, current testing results and evaluation findings are based solely on Android devices.

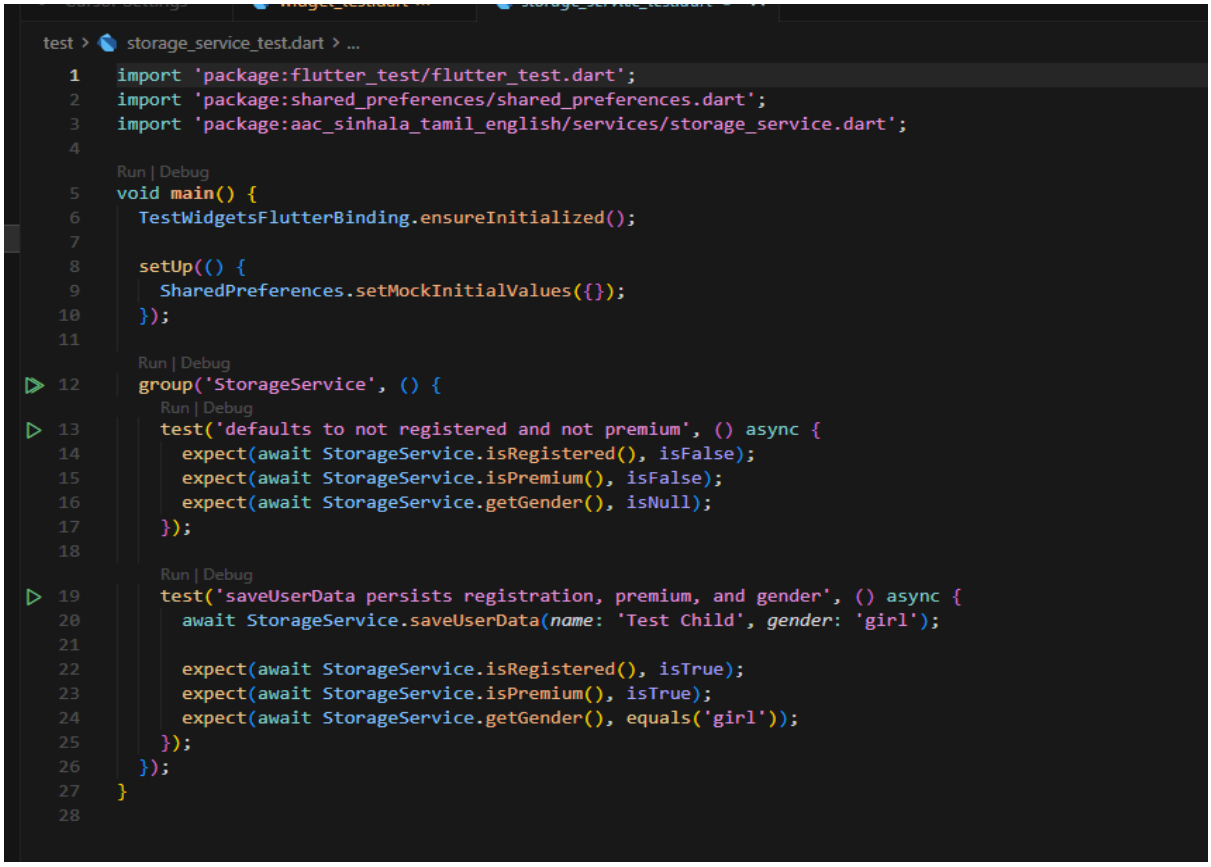
iOS deployment is being explored through a partnership discussion with IdeaHub (Pvt) Ltd, with the possibility of releasing the application as a charitable initiative following Ministry of Health approval. iOS-specific testing, including device compatibility, TTS behaviour, and performance validation, is planned for a later sprint once the developer account and distribution arrangements are finalised.

Formal user testing with children and caregivers will only take place after ethics approval is obtained. As described in Section 2.7, the evaluation will follow a mixed-methods approach. Quantitative evaluation will include structured questionnaires administered to caregivers and therapists, alongside model performance metrics. Qualitative evaluation will include semi-structured interviews and observational notes from pilot sessions. All evaluation evidence will be documented in the final report.

Figure 8 presents the output from the Flutter unit test suite at the interim stage.

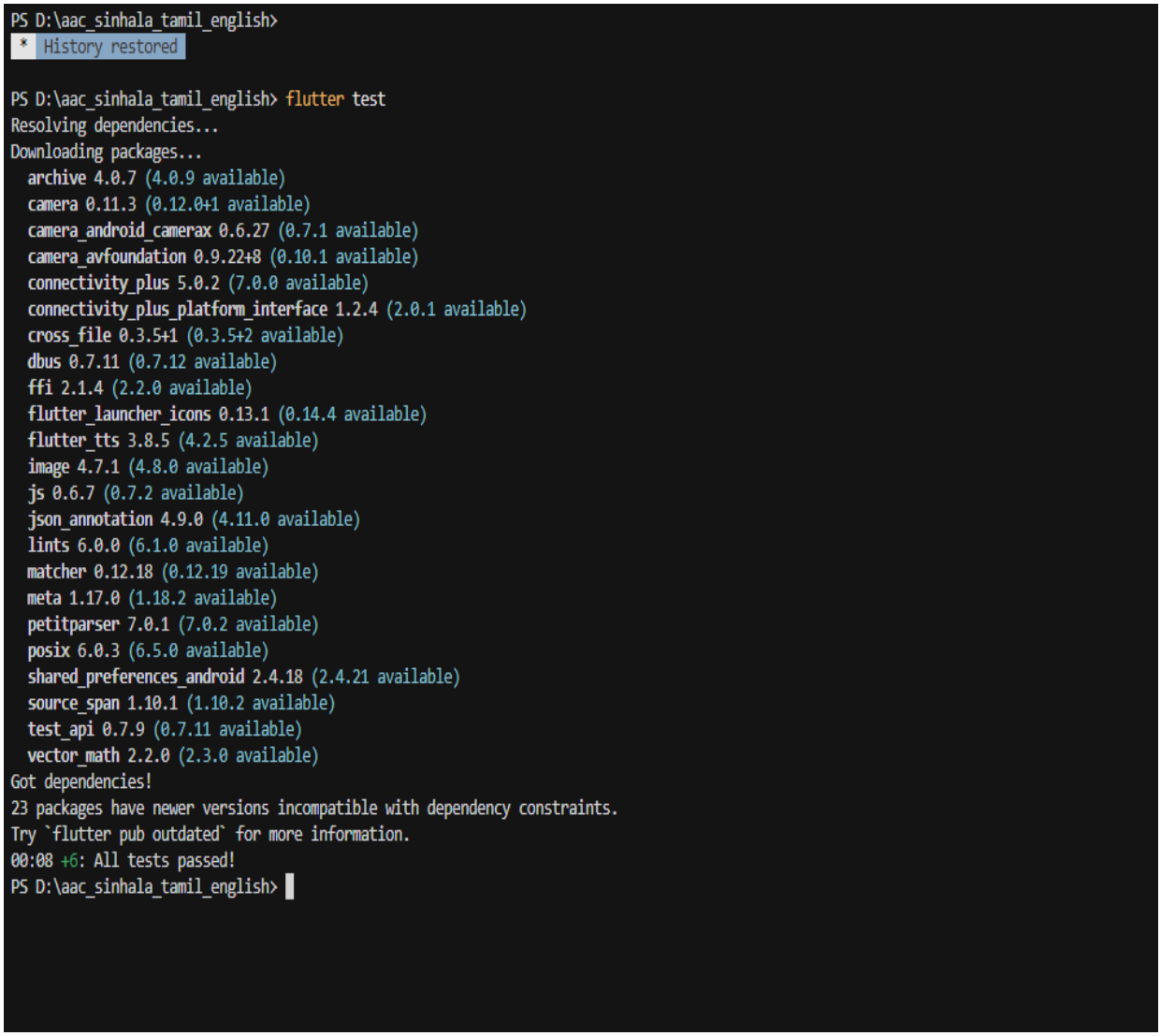
Figure 8: Flutter Unit Test Output

```
)
You, 5 minutes ago |
import 'package:flutter_test/flutter_test.dart';
import 'package:aac_sinhala_tamil_english/widgets/emoji_text.dart';
import 'package:flutter/material.dart';

Run | Debug
void main() {
  Run | Debug
  testWidgets('EmojiText renders with NotoColorEmoji font',
      (WidgetTester tester) async {
    await tester.pumpWidget(
      const MaterialApp(
        home: Scaffold(
          body: EmojiText('😊', fontSize: 42),
        ), // Scaffold
      ), // MaterialApp
    );

    expect(find.text('😊'), findsOneWidget);

    final textWidget = tester.widget<Text>(find.text('😊'));
    expect(textWidget.style?.fontFamily, equals('NotoColorEmoji'));
    expect(textWidget.style?.fontSize, equals(42));
  });      YasasPasinduFernando, 3 months ago • init
}
```



```
test › storage_service_test.dart › ...
1   import 'package:flutter_test/flutter_test.dart';
2   import 'package:shared_preferences/shared_preferences.dart';
3   import 'package:aac_sinhala_tamil_english/services/storage_service.dart';
4
    Run | Debug
5   void main() {
6     TestWidgetsFlutterBinding.ensureInitialized();
7
8     setUp(() {
9       SharedPreferences.setMockInitialValues({});
10    });
11
      Run | Debug
12    group('StorageService', () {
        Run | Debug
13      test('defaults to not registered and not premium', () async {
14        expect(await StorageService.isRegistered(), isFalse);
15        expect(await StorageService.isPremium(), isFalse);
16        expect(await StorageService.getGender(), isNull);
17      });
18
          Run | Debug
19      test('saveUserData persists registration, premium, and gender', () async {
20        await StorageService.saveUserData(name: 'Test Child', gender: 'girl');
21
22        expect(await StorageService.isRegistered(), isTrue);
23        expect(await StorageService.isPremium(), isTrue);
24        expect(await StorageService.getGender(), equals('girl'));
25      });
26    });
27  }
28
```

```
test › app_theme_test.dart › ...
1   import 'package:flutter/material.dart';
2   import 'package:flutter_test/flutter_test.dart';
3   import 'package:aac_sinhala_tamil_english/screens/theme/app_theme.dart';
4
    Run | Debug
5   void main() {
      Run | Debug
6     group('AppTheme', () {
        Run | Debug
7       test('returns girl colors when isGirl is true', () {
8         final colors = AppTheme.getThemeColors(true);
9         expect(colors['primary'], equals(const Color(0xFFFF69B4)));
10      });
11
        Run | Debug
12      test('returns boy colors when isGirl is false', () {
13        final colors = AppTheme.getThemeColors(false);
14        expect(colors['primary'], equals(const Color(0xFF00A8E8)));
15      });
16
        Run | Debug
17      test('builds ThemeData with expected app bar color', () {
18        final theme = AppTheme.getThemeData(true);
19        expect(theme.appBarTheme.backgroundColor, equals(const Color(0xFFFF69B4)));
20      });
21    });
22  }
23
```



Given that this is an interim-stage submission and pilot activities are still pending formal completion, testing has been limited to initial automated checks. Comprehensive end-to-end, performance, and user-based testing is scheduled for the final phase.

Table 11: Work Completed Summary

Work Package	Status	Notes
Literature review	Complete	Documented in Sections 1-2
Requirements and design	Substantially complete	Documented in Section 2
Flutter project setup	Complete	Single codebase for Android/iOS
Platform A (core AAC)	Partially complete	Symbol grid, navigation, basic TTS, multilingual placeholders
Platform B (AI + FER)	In progress	Pipeline set up, with initial training on public data
Firebase backend	Early / exploratory	Project created, but not fully implemented or synced
Therapist-parent dashboard	Early / prototype	Basic UI and placeholders available, with integration pending
Ethics and partnership	In progress	Draft consent forms completed, and initial hospital contact established
AI model training	In progress	Initial experiments completed, with full training still pending
TFLite export and benchmarking	Complete (preliminary)	Inference time acceptable

Testing	In progress	Unit tests and manual testing completed, with no formal user testing yet

3.8 Current UI Screens (Interim)

The following figures present the current state of the mobile application's user interface. The interface follows a child-friendly design approach using soft colours, rounded components, and clear icon-based navigation, intended to minimise cognitive load while supporting intuitive symbol selection.

Figure 9: Register Screen (UI)

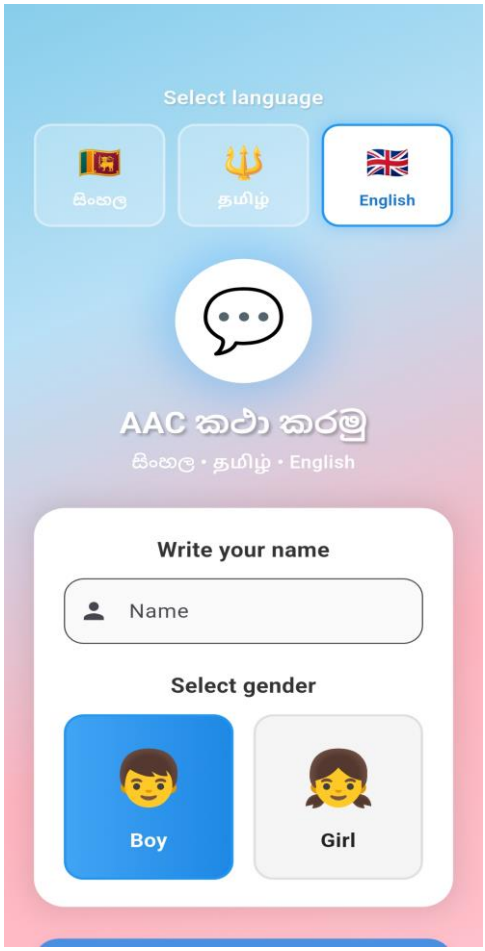
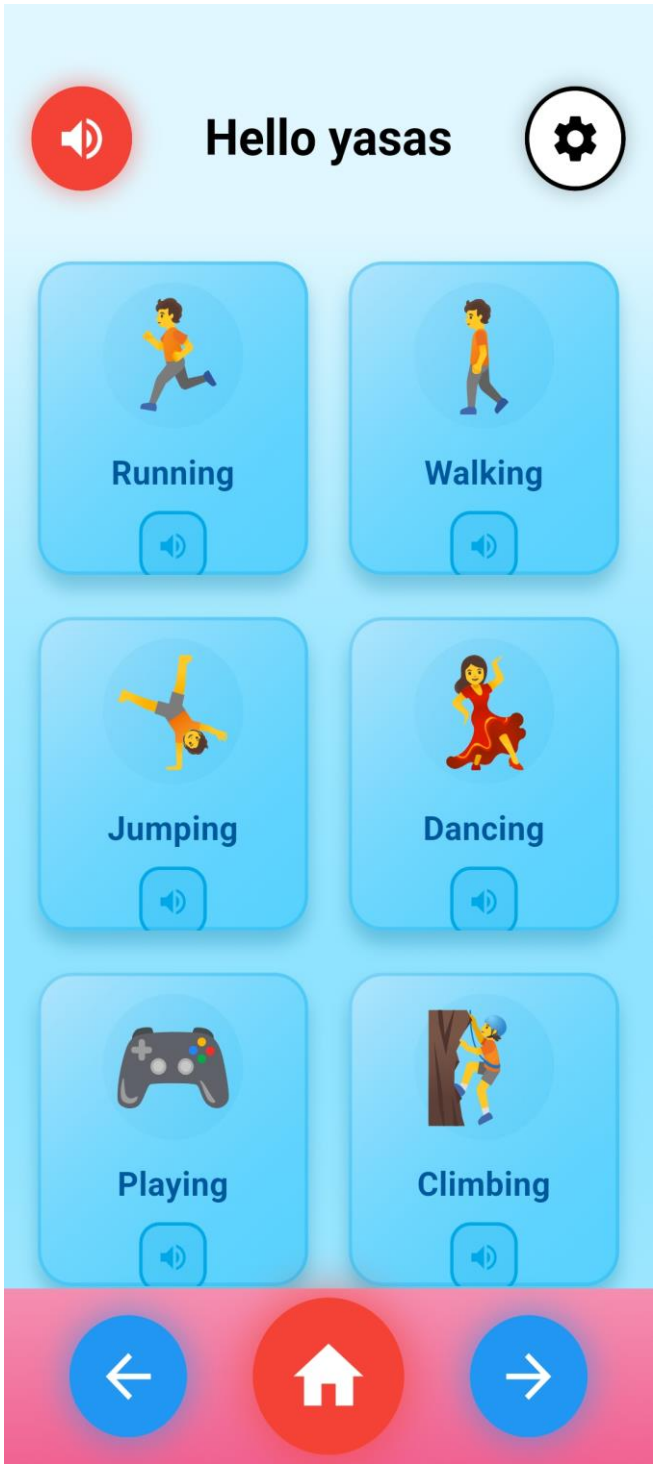


Figure 10: Home Screen (UI)

Platform B: -

Platform A: -

Figure 11: Categories Screen (UI)


Hello yasas
Running
Walking
Jumping
Dancing
Playing
Climbing

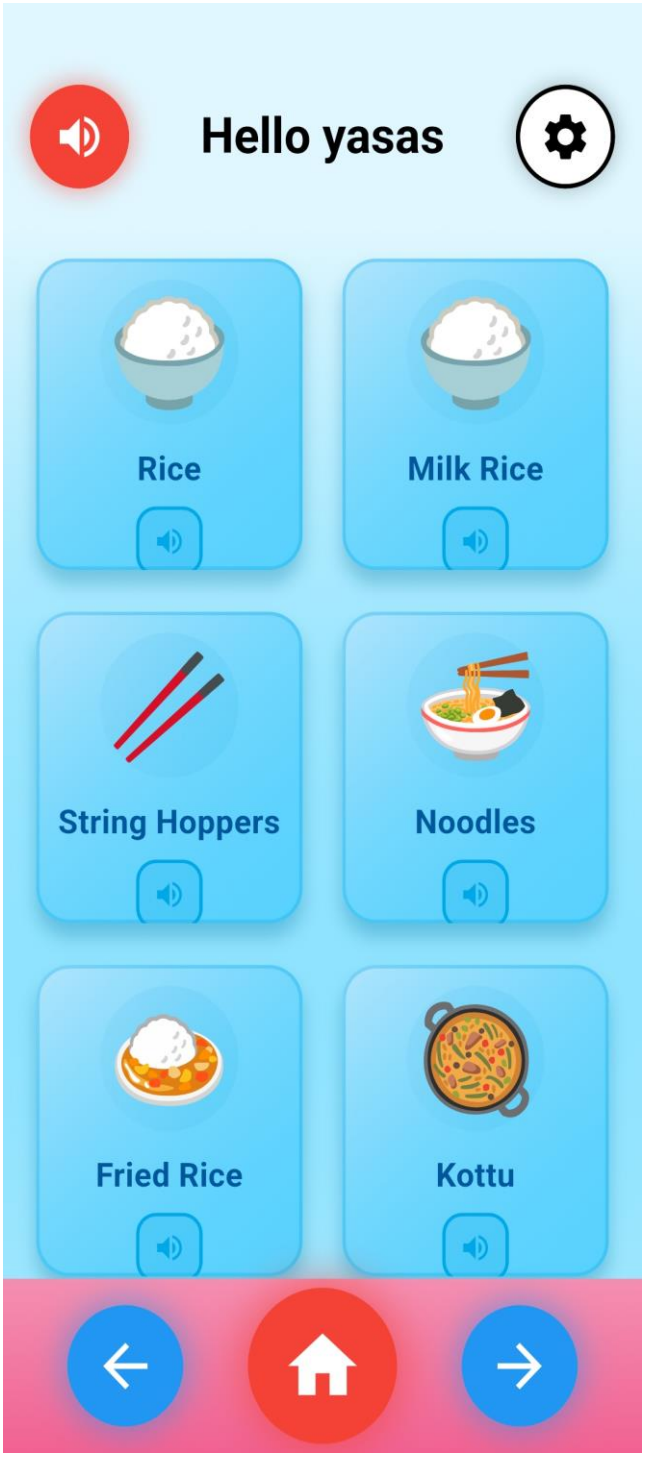

Hello yasas
Rice
Milk Rice
String Hoppers
Noodles
Fried Rice
Kottu

Figure 12: Settings Screen (UI)

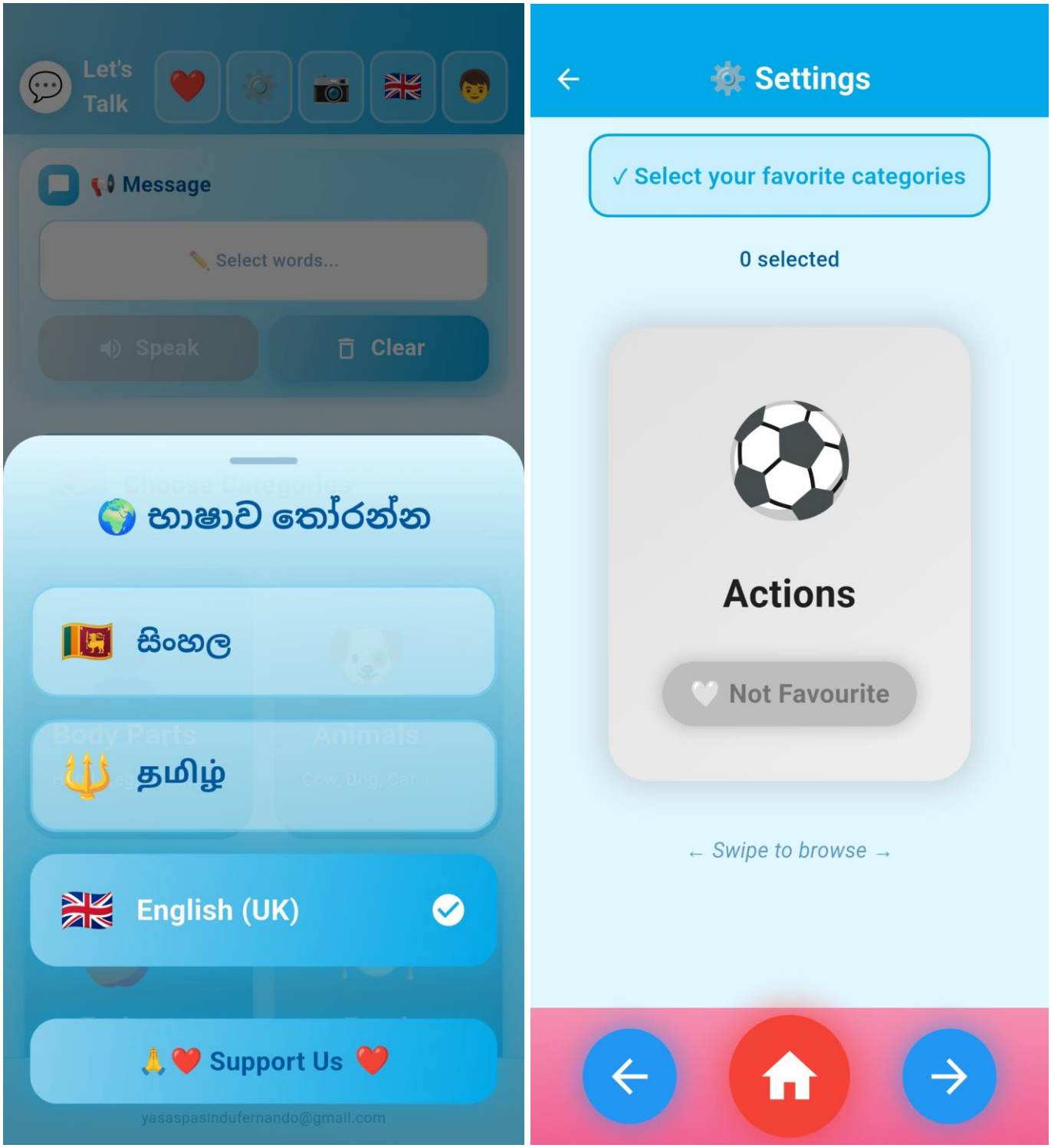
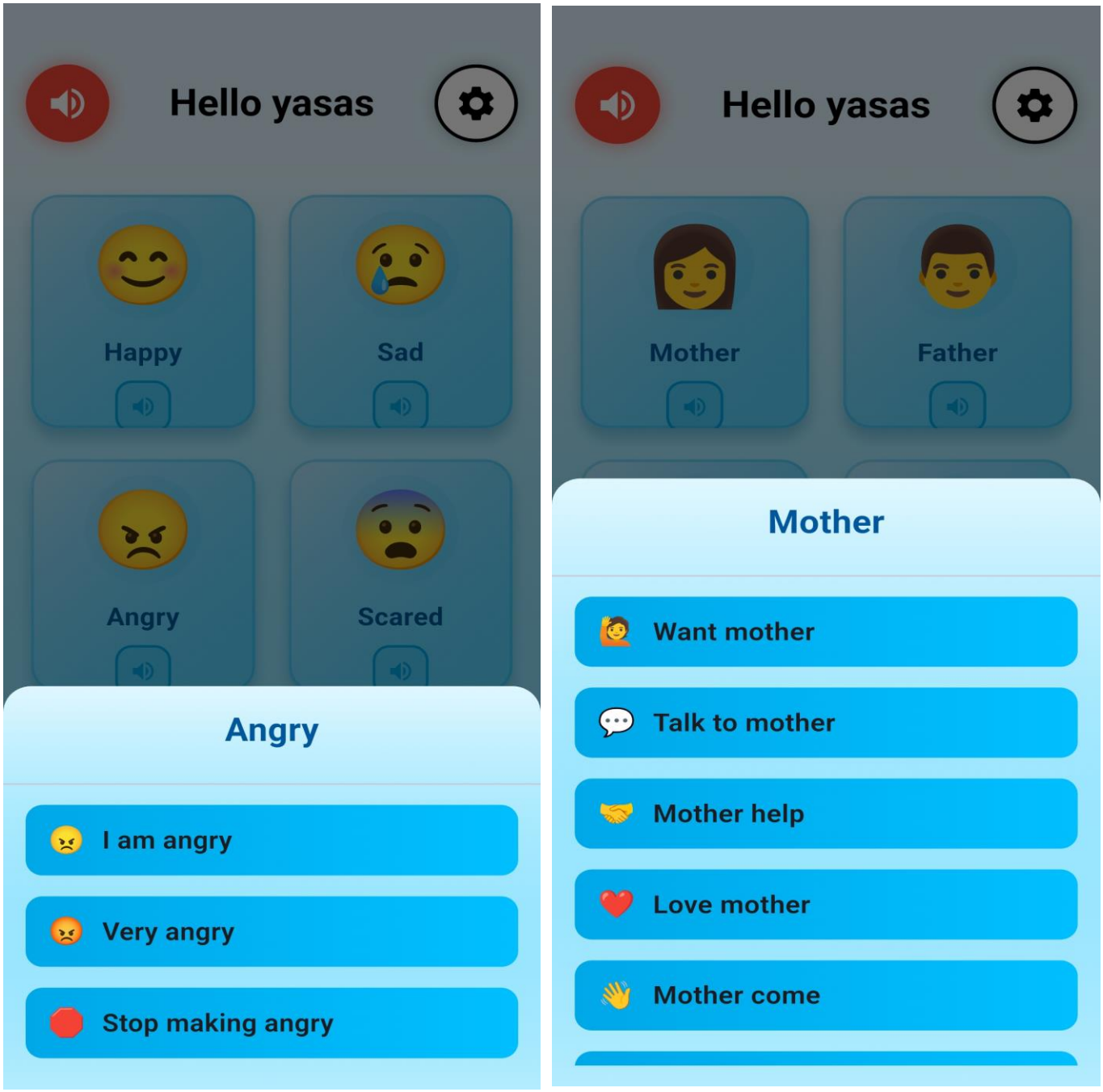


Figure 13 demonstrates a typical AAC interaction, showing symbol selection and the resulting text-to-speech output.

Figure 13: AAC Interaction Example (Symbol Selection and Output)



This figure demonstrates how a child interacts with the AAC grid, selects a symbol, and receives immediate audio feedback. This flow is designed to be simple and consistent, supporting faster learning and communication.

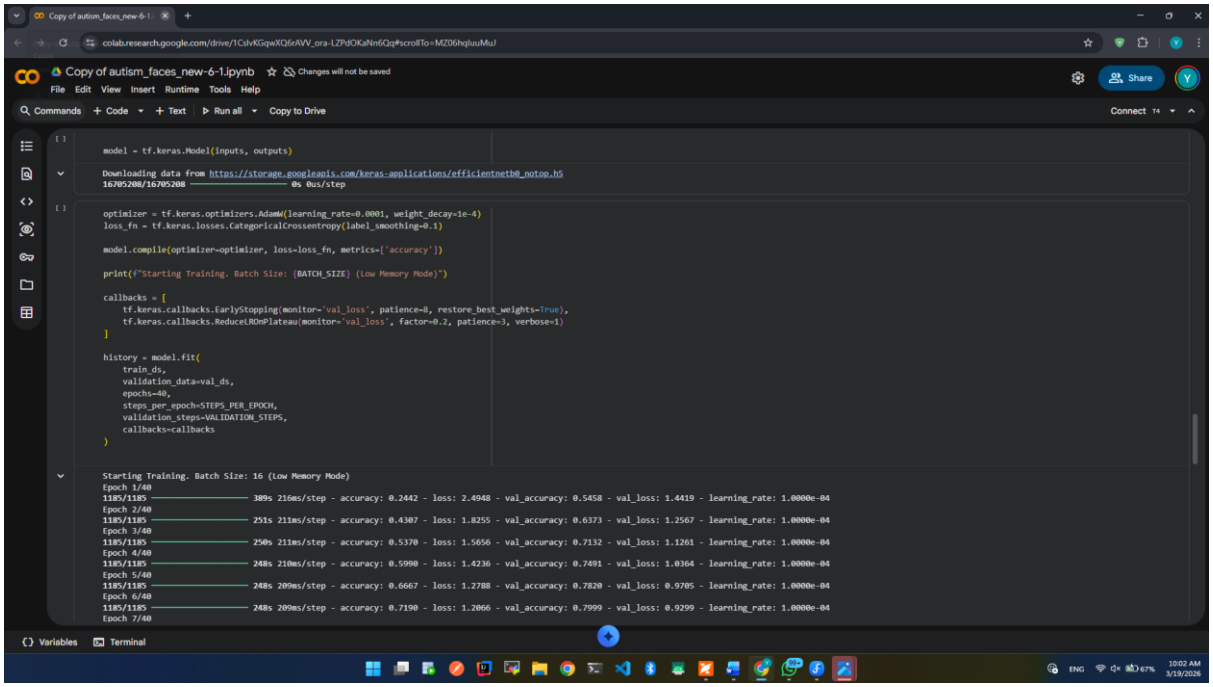
Additional user interface screens related to Platform A (Level 1-2) are provided in Appendix A. The main report emphasises Platform B due to its central role in the research, particularly the integration and evaluation of facial expression recognition.

3.9 Model Training Evidence

Model training for the FER component was conducted on Google Colab with GPU support. Multiple training runs were carried out with different combinations of epochs, learning rates, and regularisation settings. Each run was logged, and training/validation loss and accuracy curves were monitored to identify overfitting and guide parameter adjustments.

Figure 14 presents a representative Colab training session showing the loss and accuracy logs across epochs.

Figure 14: Model Training in Google Colab



Link :- https://colab.research.google.com/drive/1CslvKGqwXQ6rAVV_ora-LZPdOKaNn6Qq#scrollTo=MZ06hqIuuMuJ

The Colab notebook shows the training and validation loss/accuracy curves for each run. Mixed precision training was enabled to speed up training and reduce memory usage, while keeping key parts in float32 for stability. Across experiments, accuracy improved gradually as the number of epochs and other parameters were tuned.

During testing, monitoring these logs during runs was helpful for spotting when the model started to overfit and when to stop or adjust the epochs.

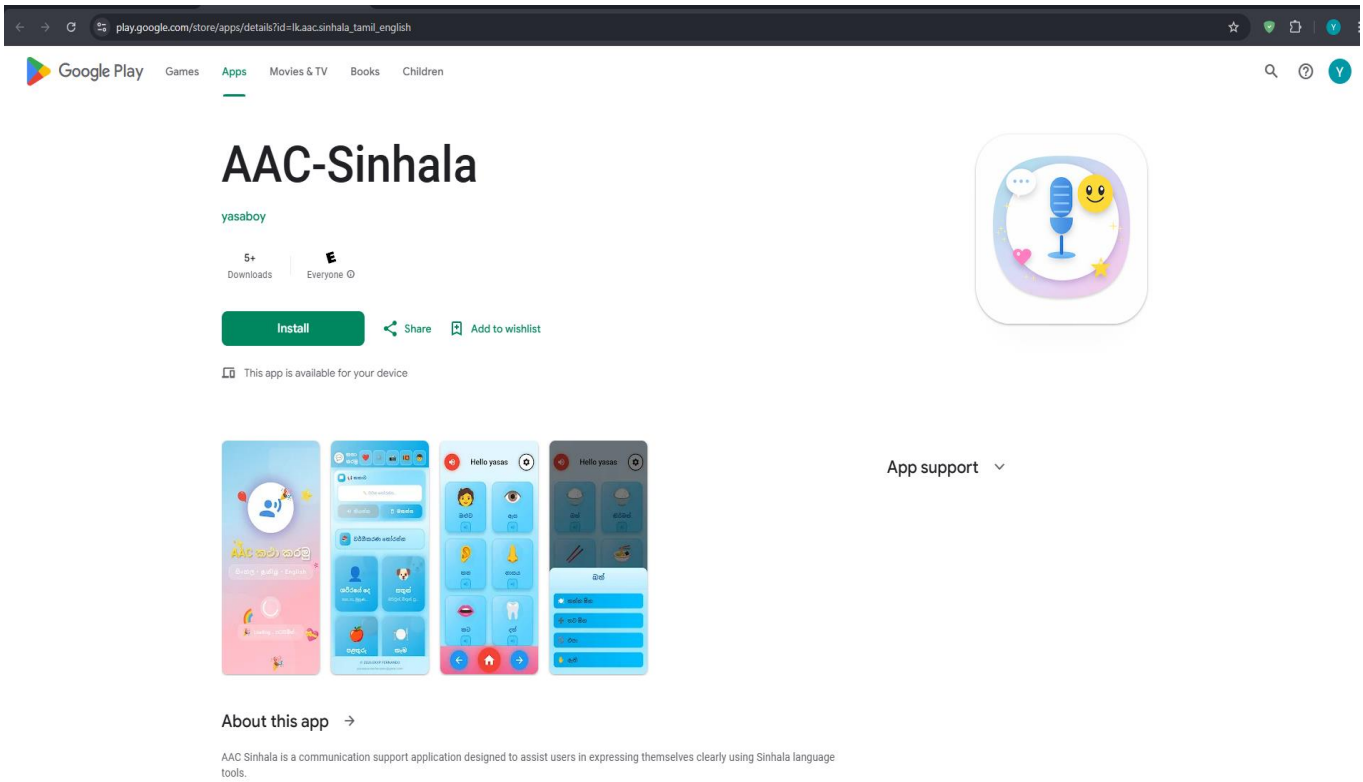
Mixed precision training (float16 for computational layers, float32 for final layers) reduced memory usage and enabled training with reasonable batch sizes on the free-tier GPU. Accuracy improved gradually across experiments as hyperparameters were refined.

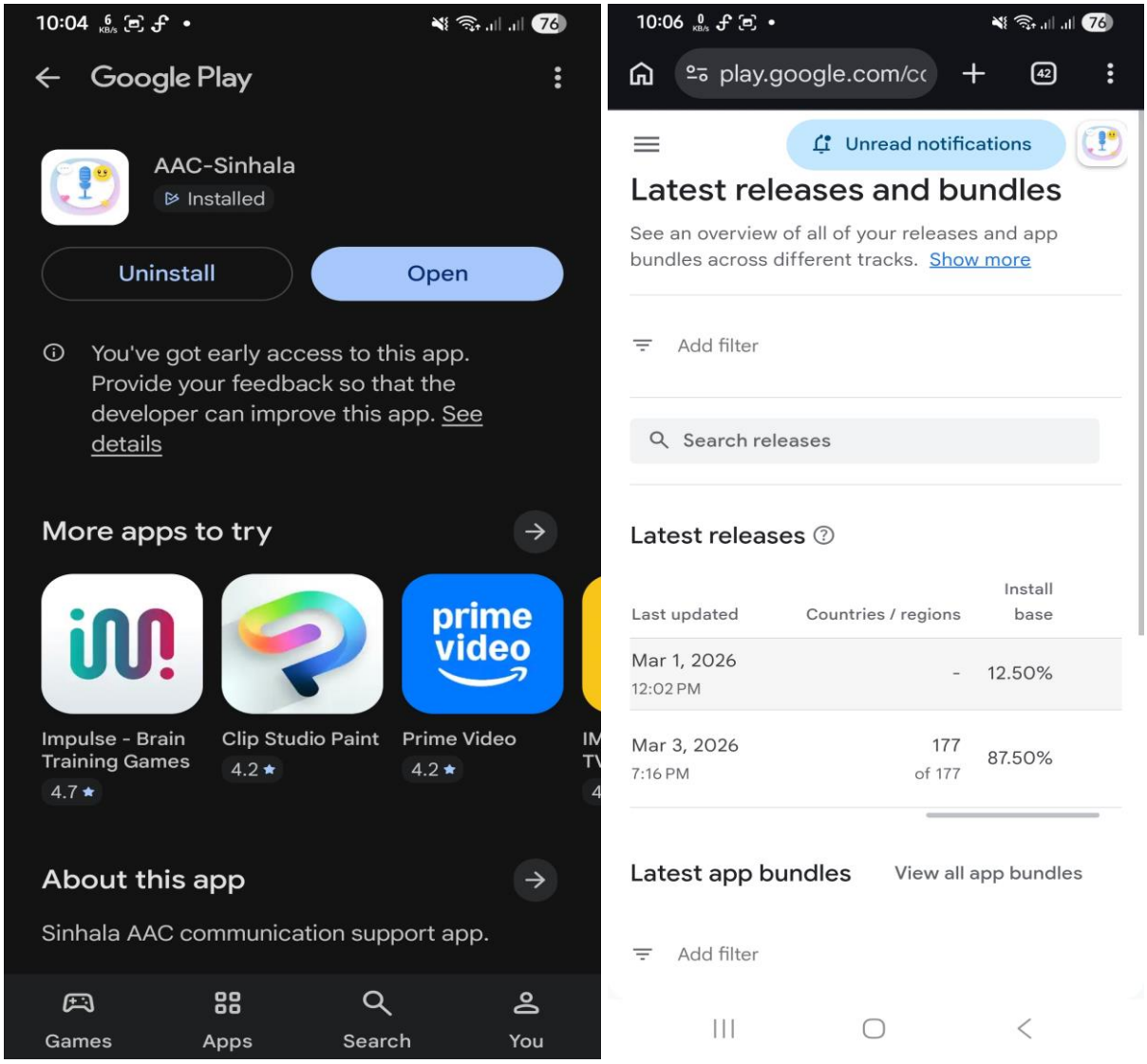
3.10 Deployment and Play Store Testing

The application was packaged and uploaded to the Google Play Console under the Internal Testing track. An Android App Bundle was generated from the Flutter project and distributed to internal testers. Internal testers installed the application on their own Android devices and verified the main flows, including the AAC grid, navigation, and basic settings. This confirmed that deployment through the Play Store is technically feasible at this stage.

Figure 15 presents the Google Play Console internal testing dashboard.

Figure 15: Google Play Console Internal Testing





Play Store Link :- https://play.google.com/store/apps/details?id=lk.aac.sinhala_tamil_english

Internal testers installed the app on their own Android devices and checked the main flows, especially the AAC grid, navigation, and basic settings. This helped to confirm that the app can be installed and launched through a normal Play Store flow. A proper Play Store link will be added in the final version after further testing and polishing.

These internal tests mainly showed that, at this stage, deployment through the Play Store route is technically feasible even though the app is still in a pilot state.

iOS deployment has not yet been undertaken, as an Apple Developer account is not currently available. Discussions are ongoing with IdeaHub (Pvt) Ltd regarding the possibility of providing the developer account and distribution infrastructure for iOS, potentially as part of a charitable initiative following Ministry of Health approval.

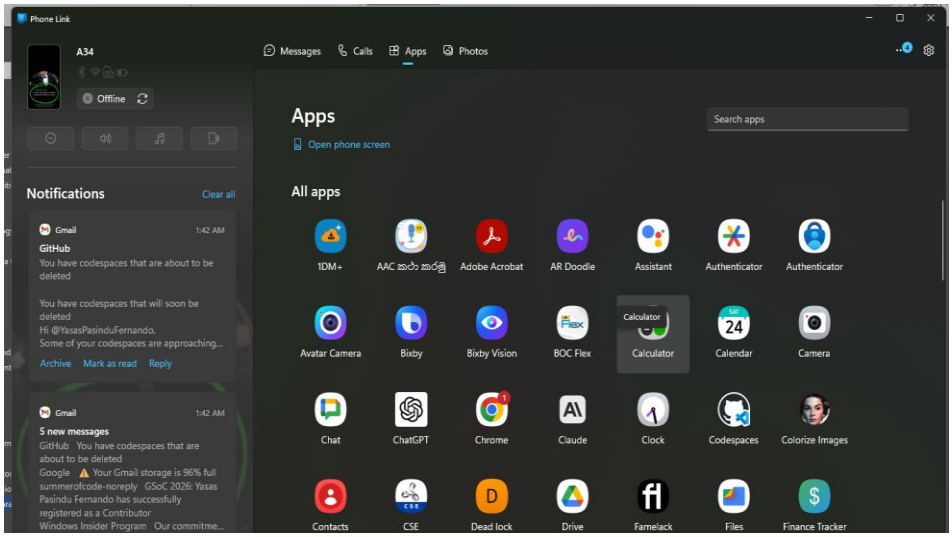
3.11 Real Device Testing Sessions

In addition to emulator testing, the application was tested on two physical Android devices under everyday conditions. These sessions focused on basic AAC communication and general stability rather than formal user studies. Symbol selection, text-to-speech output, and screen navigation were tested repeatedly to check for crashes or performance issues. Initial checks of the FER pipeline confirmed that the camera feed and on-device inference could run without freezing. Compatibility checks across different display sizes were also carried out to confirm that key interface elements remained readable and usable without layout breakage.

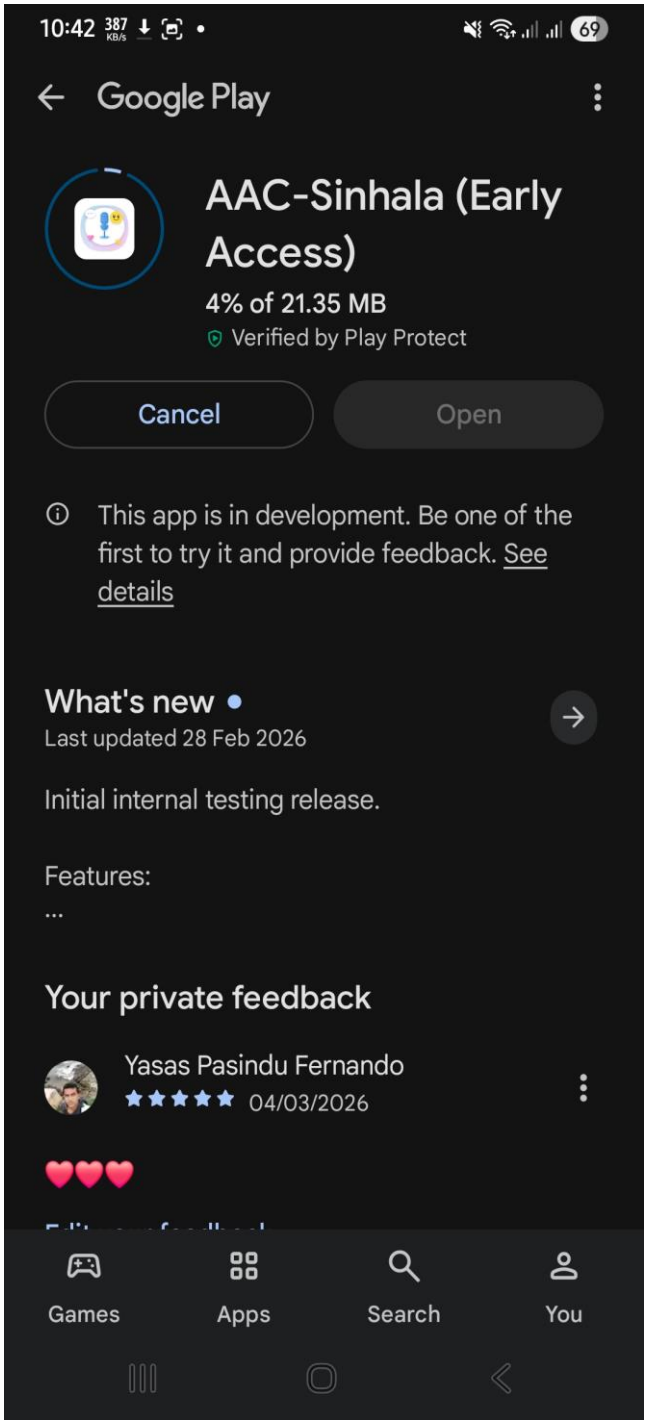
Figure 16 shows the application running on a real Android device.

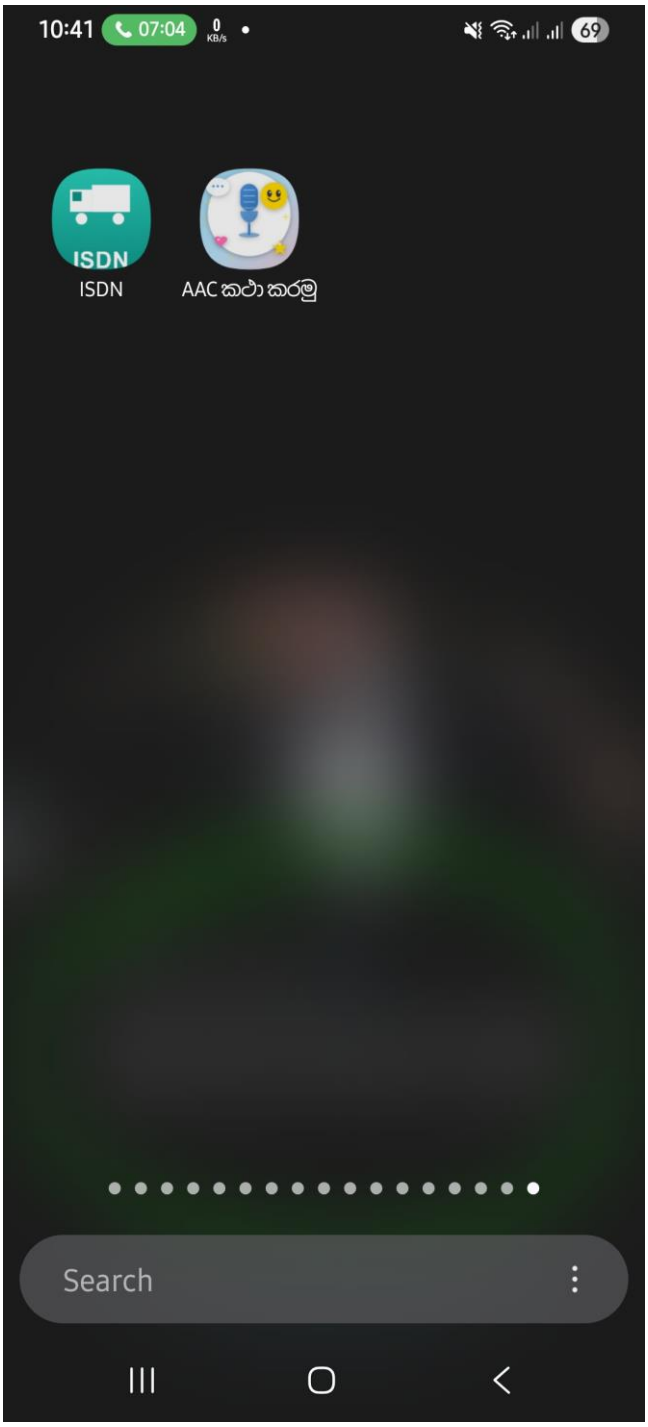
Figure 16: Application Running on Real Device

Tablet:



Phone:





The application performed acceptably on both devices, providing confidence that the prototype can function on real hardware before any wider pilot deployment.

3.12 Initial Field Exposure (Karapitiya Context)

An initial field exposure was conducted at Karapitiya Teaching Hospital in Galle to gain a practical understanding of the real-world context in which the proposed system would be used. This visit was exploratory in nature and did not involve formal data collection, as ethical approval had not yet been obtained at this stage.

Observations from this setting highlighted significant communication challenges faced by children with autism spectrum disorder. Many of the children present did not have reliable means of expressing basic needs such as hunger, discomfort, or emotional distress. Caregivers reported relying on continuous interpretation and guesswork, often without access to suitable digital tools. The absence of AAC applications supporting the Sinhala language was also identified as a critical gap, reinforcing the motivation for this project.

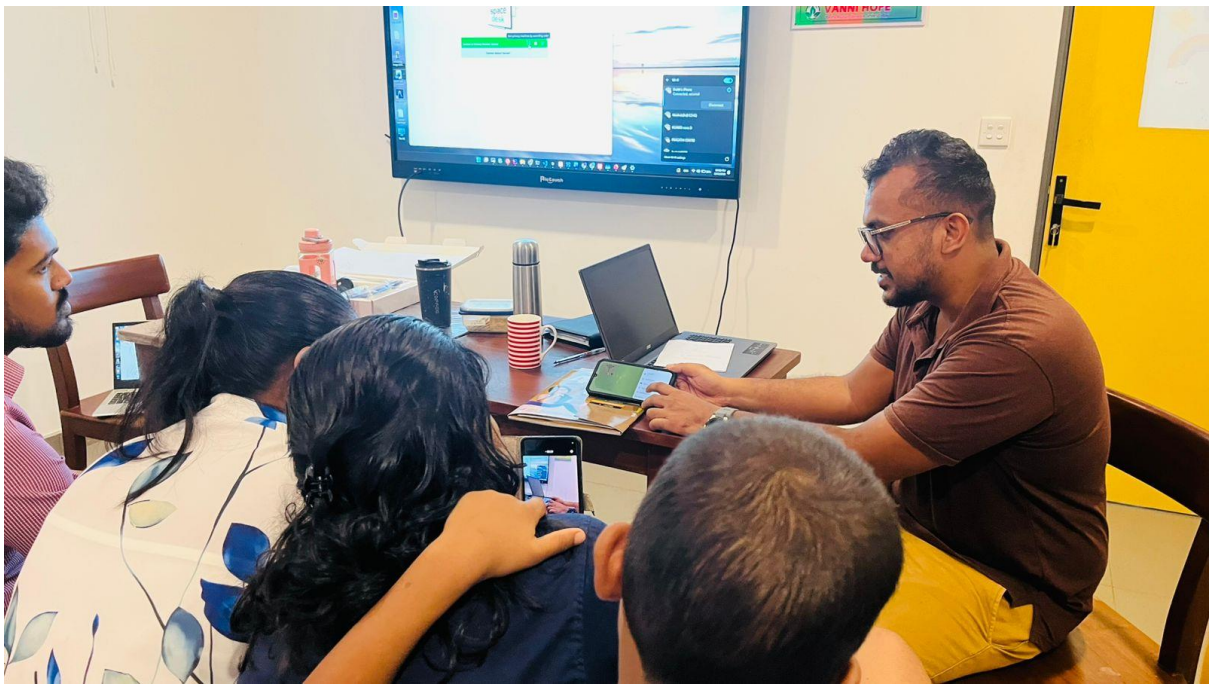
From an environmental perspective, the clinical setting presented several practical constraints. The ward environment was relatively noisy, attention spans among the children were limited, and many families did not have consistent access to tablets or similar devices. Internet connectivity in home environments was also reported as unreliable, even among families who owned compatible devices. These constraints were directly observed and had a clear influence on subsequent design decisions.

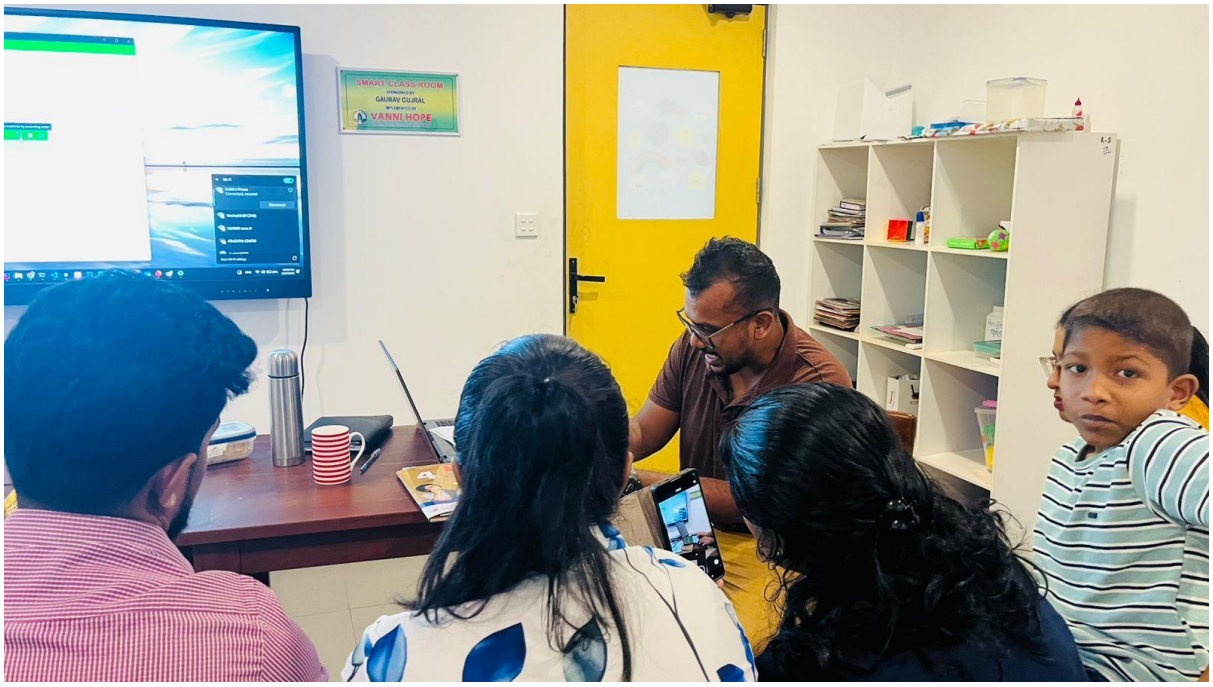
In particular, the adoption of an offline-first architecture, the implementation of a simplified user interface, and the decision to maintain a minimal and uncluttered symbol grid were all informed by insights gained during this field exposure.

Figure 17 illustrates the field exposure session conducted at Karapitiya Teaching Hospital, where the AAC application was demonstrated within a real-world context.

Drive Link Photos videos:- https://drive.google.com/drive/folders/1CoWy-is5cbDUR7BOddoymTZy5umPKJYY?usp=sharing

Figure 17: Field Exposure at Karapitiya Teaching Hospital




SMART CLASS ROOM
GAURAV OJURAL
VANNI HOPE

The formal pilot testing phase will be conducted following ethical approval. However, this initial exposure provided practical insights that could not be fully captured through the literature review alone and contributed significantly to shaping the overall system design.

The formal pilot testing phase will be conducted following ethical approval. However, this initial exposure provided practical insights that the literature review alone could not have fully captured, and it had a noticeable influence on the overall system design.

3.13 Supporting Links

The following links will be included in the final submission.

- **Google Colab (Model Training):** This item was pending at the interim stage
- **Google Play Testing Link:** This item was pending at the interim stage
- **User Guidance (User Manual):** This item was pending at the interim stage.

3.14 External Interest

There has been some informal interest in the project concept outside the academic context. Preliminary discussions have taken place with IdeaHub (Pvt) Ltd regarding the potential for releasing the application as a charitable initiative, particularly for the iOS platform where the

developer account and distribution infrastructure would be provided through the company. However, no formal collaboration has been established at this stage. The project remains at the prototype stage, and Ministry of Health approval has not yet been obtained. Any external collaboration, charitable distribution, or outreach will only be pursued after system completion, pilot validation, and all necessary approvals are in place.

4. Further Work

The following subsections outline the remaining tasks required to complete the project. These tasks are aligned with the revised project plan presented in Section 5.6.

4.1 Platform A Completion

The remaining work for Platform A includes finalising the symbol set and resolving licensing issues, completing the full vocabulary in Sinhala, Tamil, and English with culturally appropriate symbols, integrating and evaluating text-to-speech for all three languages, implementing full customisation features (configurable grid size, user-added symbols, custom categories, colour themes, and font size settings), implementing visual scheduling functionality, and conducting internal usability testing with at least one therapist and one parent or caregiver.

Additional personalisation features are planned, including the ability for caregivers to record custom audio for symbols, edit existing cards, and add new symbols using photos from the device camera. These features are intended to improve engagement and communication effectiveness for children at ASD Levels 1–2.

4.2 Platform B and AI Integration

The remaining work for Platform B includes completing the curated dataset (2,000–5,000 images) combining public data with purpose-collected data, conducting full model training with hyperparameter tuning, achieving and documenting the 80% accuracy target with full confusion matrix analysis and per-class precision, recall, and F1 scores, integrating the TFLite model into the Flutter application with the face detection and preprocessing pipeline,

implementing configurable emotion-adaptive logic and caregiver override, and testing emotion detection on multiple devices (Android and iOS).

4.3 Backend and Synchronisation

The remaining backend work includes completing manual export and import backup and restore flows, designing and implementing optional Firebase synchronisation with conflict handling (if pursued), and hardening security through role-based access, encryption, and audit logging.

4.4 Dashboard

The dashboard requires completion with progress visualisations, vocabulary management, and data export options. Feedback sessions with at least one therapist and one parent are also planned.

4.5 Pilot and Evaluation

The pilot phase requires obtaining ethics approval from Karapitiya Teaching Hospital and completing Ministry of Health processes, recruiting pilot participants (target of 5 to 15 children with ASD and their caregivers and therapists), conducting supervised pilot use over 4 to 8 weeks, collecting quantitative data (usage logs, emotion detection accuracy, task completion rates) and qualitative data (caregiver and therapist feedback), and analysing findings with documentation of limitations and recommendations.

4.6 Final Deliverables

Final deliverables include the final report incorporating pilot findings and full model evaluation, user documentation such as an installation guide and user manual, a deployment package containing the APK or IPA, model files, and backend configuration, and a presentation or demonstration for the examining panel.

Table 12: Remaining Work Plan

Task	Target Period	Dependency	Status

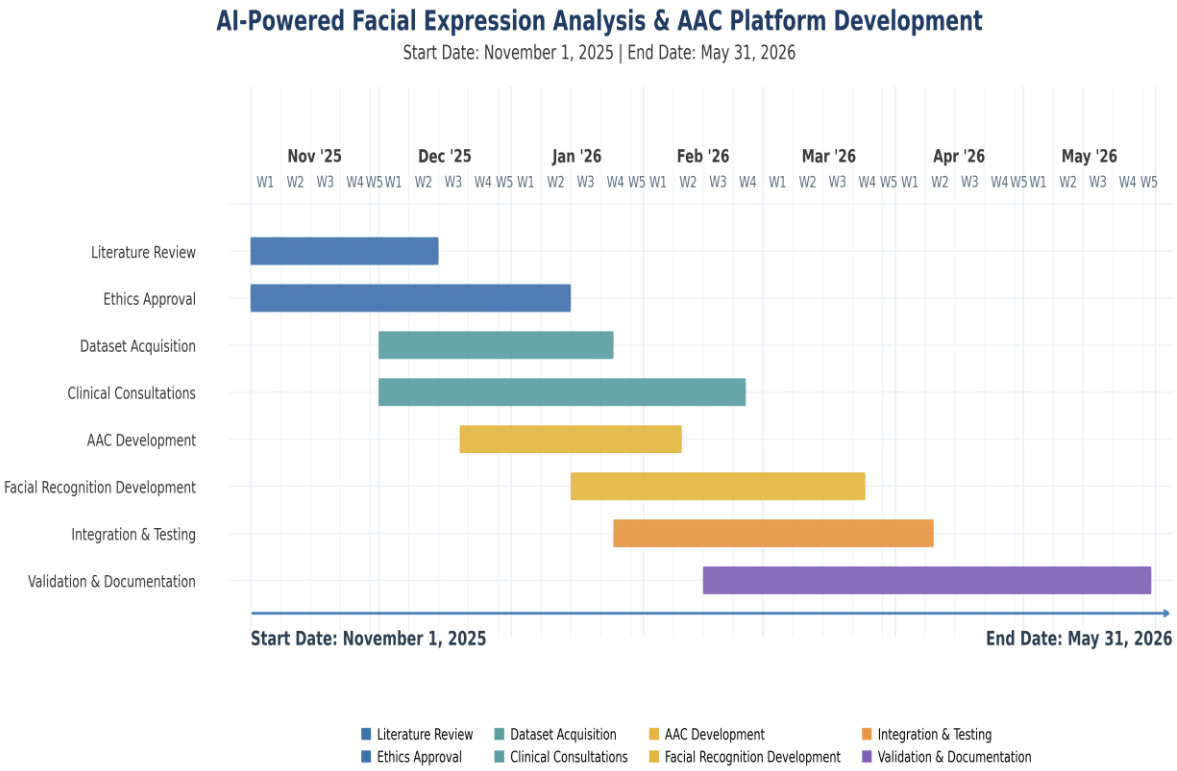
Finalise symbol set and licensing	March 2026	None	Pending
Complete Platform A (full trilingual AAC)	March 2026	Symbol set	Pending
Complete curated dataset	March to April 2026	Ethics approval	Pending
Full model training and evaluation	March to April 2026	Dataset	Pending
Platform B integration (FER + adaptation)	April 2026	Model	Pending
Full dashboard	April 2026	None	Pending
Full offline-first sync	April 2026	None	Pending
Ethics approval (hospital + MoH)	March to April 2026	Application	In preparation
Pilot recruitment	April 2026	Ethics approval	Pending
Pilot execution	May 2026	Recruitment	Pending
Final report	May 2026	All above	Pending

5. Progress Review

5.1 Original Project Plan

At the outset of the project, a Gantt chart was prepared as part of the project proposal to establish the planned schedule across the academic year. This chart divided the work into five sprints spanning approximately seven months, with each sprint building upon the deliverables of the previous one. Figure 18 presents the original project Gantt chart.

Figure 18: Initial Project Gantt Chart



The five sprints in the original plan were defined as follows.

Sprint 1 (November to December 2025) covered requirements analysis, architecture design, literature review, and the initial Flutter project setup. The expected deliverable was a documented system architecture and a basic application skeleton.

Sprint 2 (January to February 2026) covered Platform A core AAC features, including the symbol grid, trilingual support, basic TTS, and navigation. It also included Firebase backend configuration and initial dashboard implementation. The expected deliverable was a functional, though incomplete, AAC application and backend.

Sprint 3 (March 2026) covered dataset assembly, full model training and evaluation, TFLite export, and symbol set finalisation. The expected deliverable was a trained model ready for integration.

Sprint 4 (April 2026) covered Platform B FER integration, dashboard completion, ethics approval, pilot design, and participant recruitment. The expected deliverable was a complete system ready for pilot deployment.

Sprint 5 (May 2026) covered pilot execution, data collection and analysis, final report writing, and preparation of deliverables. The expected deliverable was the final report and all supporting documentation.

At the interim submission point, which falls at the end of February 2026, Sprints 1 and 2 were expected to be fully complete.

5.2 Progress Against the Original Plan

Sprint 1 (November to December 2025) was completed on schedule. The literature review, requirements analysis, system architecture, and technology stack selection were all delivered within the planned timeframe. The Flutter project was initialised with the intended folder structure and build targets for both Android and iOS. No significant issues arose during this phase.

Sprint 2 (January to February 2026) is substantially complete, though progress was uneven across different work packages. The core AAC features for Platform A, including the symbol grid, screen navigation, basic text-to-speech, and the multilingual switching framework, have been partially implemented and are functional at a prototype level. The AI model pipeline was set up ahead of schedule, with preliminary training experiments on public datasets completed during this sprint rather than in Sprint 3 as originally planned. This early start on the model work provides a useful buffer for the next phase.

However, four areas did not progress as quickly as the original plan anticipated.

The symbol set and licensing process proved more time-consuming than expected. Identifying symbols that are culturally appropriate for Sri Lankan users, and then navigating the licensing terms of various symbol libraries, required correspondence and research that had not been fully accounted for in the original estimate. This work remains in progress.

The Firebase backend is at an early prototype level. The decision was taken to prioritise the offline-first architecture and local storage layer over cloud integration. Firebase configuration

has been started, but full synchronisation and authentication have been deliberately deferred. This was a conscious scope prioritisation rather than an unplanned delay.

Text-to-speech quality for Sinhala and Tamil fell short of expectations. The assumption that the default platform TTS engines would produce acceptable output proved incorrect during initial testing. Alternative TTS services will need to be evaluated in the next sprint.

The ethics application for Karapitiya Teaching Hospital has taken longer to coordinate than originally anticipated. The process involves both the hospital's institutional ethics committee and Ministry of Health governance requirements. The application is in advanced preparation but has not yet been formally submitted.

5.3 Justification for Delays

The delays identified in Section 5.2 are minor and have specific, identifiable causes. Each is explained below with its impact and the corrective action taken.

Delay 1: Symbol Set Licensing

- **Problem.** Most established AAC symbol libraries (such as PCS and Widgit) are designed for Western contexts. The food, clothing, and cultural symbols in these libraries do not represent Sri Lankan daily life.
- **Cause.** Finding and licensing culturally appropriate symbols required correspondence with multiple symbol library providers and manual adaptation of existing assets. The time required for this process was significantly underestimated in the original plan.
- **Impact.** The full trilingual vocabulary for Platform A is not yet complete.
- **Action taken.** Manual sourcing of culturally relevant symbols is ongoing. This task has been carried forward to Sprint 3 as a priority item.

Delay 2: Ethics Coordination

- **Problem.** The formal ethics application for the pilot study at Karapitiya Teaching Hospital has not yet been submitted.

- **Cause.** The approval process involves coordination with both the hospital's institutional ethics committee and the Ministry of Health. The administrative timelines and procedural requirements for these bodies were not fully known at the planning stage and have proved longer than initially assumed.
- **Impact.** Purpose-collected data from children with ASD cannot yet be used for model training, and the formal pilot study cannot begin until approval is obtained.
- **Action taken.** The ethics application is in advanced preparation. An exploratory field exposure was conducted in the meantime to inform design decisions without formal data collection.

Delay 3: Text-to-Speech Quality

- **Problem.** The default platform TTS engines on Android do not produce acceptable speech output for Sinhala and Tamil.
- **Cause.** The original plan assumed that the built-in TTS engines would provide adequate quality for all three languages. This assumption was only disproved during implementation when the Sinhala and Tamil output was tested and found to be unnatural and difficult to understand.
- **Impact.** Trilingual TTS is not yet functional. English TTS works as expected.
- **Action taken.** Evaluation of third-party TTS services and recorded audio alternatives has been added to the Sprint 3 backlog.

Delay 4: AI Model Training on Google Colab

- **Problem.** The pace of model experimentation was slower than originally anticipated.

- **Cause.** The training pipeline was executed on Google Colab using the free-tier GPU allocation. The free tier imposes session time limits, which meant that longer training runs were sometimes interrupted before completion, resulting in lost checkpoints and the need to repeat experiments.
- **Impact.** Fewer training iterations were completed than planned during Sprint 2, though preliminary results were still obtained.
- **Action taken.** More frequent checkpoint saving was adopted to prevent further data loss. The backlog for Sprint 3 includes completing full model training with the final dataset.

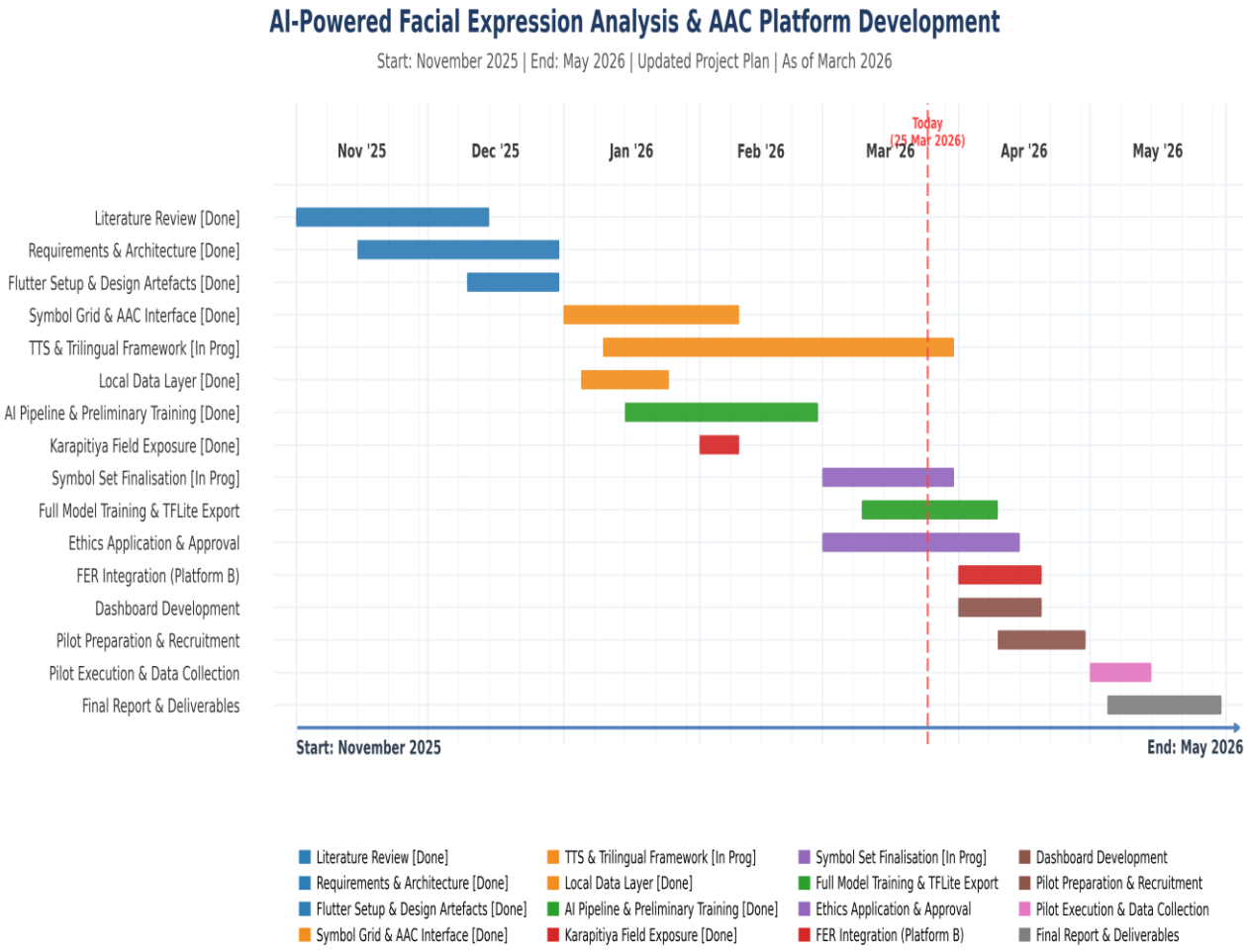
Overall Impact Assessment

None of these delays affect the critical path in a way that compromises the final deadline, provided the revised plan outlined in Section 5.4 is followed. The critical path runs through ethics approval, dataset collection, model training, Platform B integration, pilot execution, and the final report. Platform A development and model training on public data can proceed independently of the ethics timeline.

5.4 Revised Plan for Remaining Work

Based on the progress to date and the delays identified, the project plan has been revised for the remaining sprints. The key adjustment is the introduction of an overlap between Sprints 3 and 4, where Platform A completion, model training, and ethics preparation proceed in parallel. This is feasible because these tasks have limited interdependencies and Scrum allows the backlog to be reprioritised between sprints as needed. Figure 19 presents the updated project Gantt chart reflecting these changes.

Figure 19: Updated Project Gantt Chart



The revised schedule is as follows.

March 2026 (Revised Sprint 3). The focus during this period will be on finalising the symbol set, completing Platform A vocabulary and TTS integration, submitting the ethics application, continuing model training on public datasets, and beginning dataset curation.

April 2026 (Revised Sprint 4). This period will focus on full model training completion, Platform B FER integration and testing, dashboard completion, and ethics follow-up. Pilot protocol design and participant recruitment will also begin during this sprint.

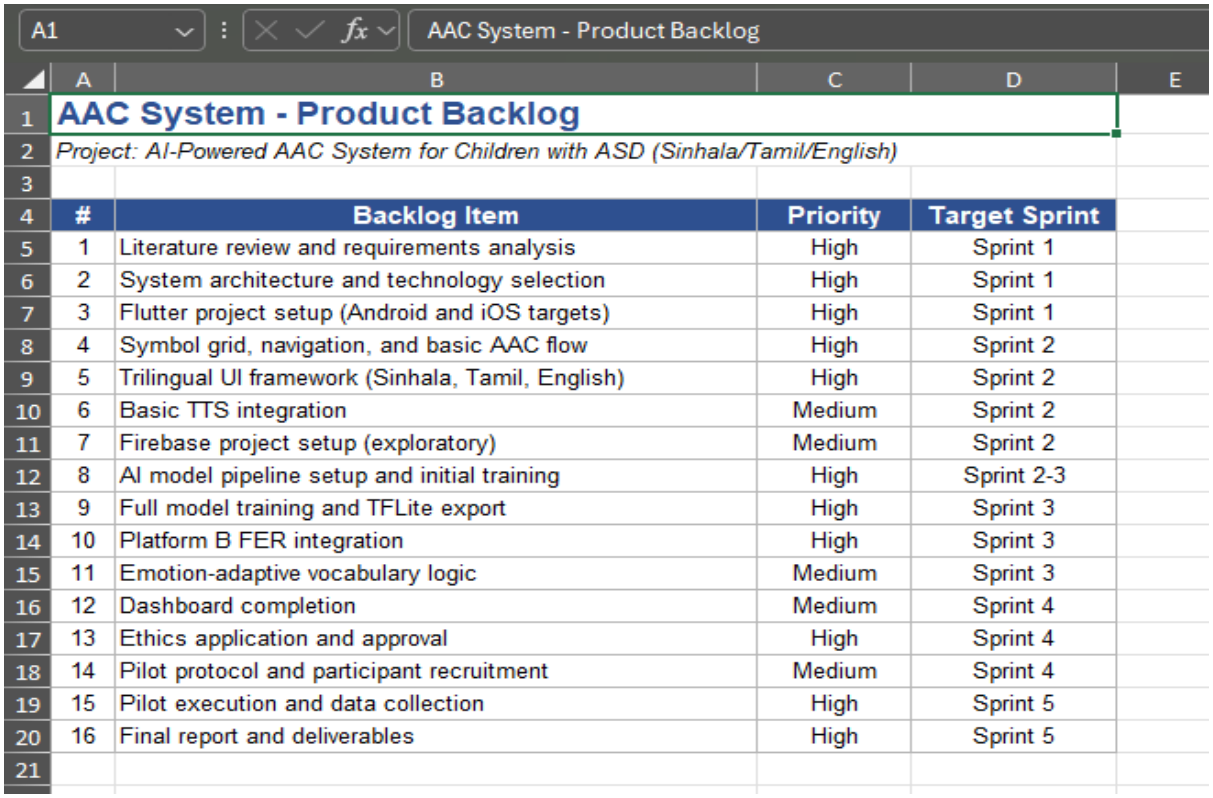
May 2026 (Revised Sprint 5 and Final Phase). This final sprint covers pilot execution at Karapitiya Teaching Hospital, data collection and analysis, final report writing, preparation of deliverables, and the project presentation.

The main difference between the original and revised plans is the parallel scheduling of previously sequential tasks during March to May 2026 and the compression of Sprints 3 to 5 into one-month cycles. This approach absorbs the minor delays without extending the overall project timeline.

Throughout the project, sprint progress and task status were tracked using a Google Sheets spreadsheet, which served as both the product backlog and the sprint backlog tracker. Figure 20 presents the sprint tracking spreadsheet used during the project.

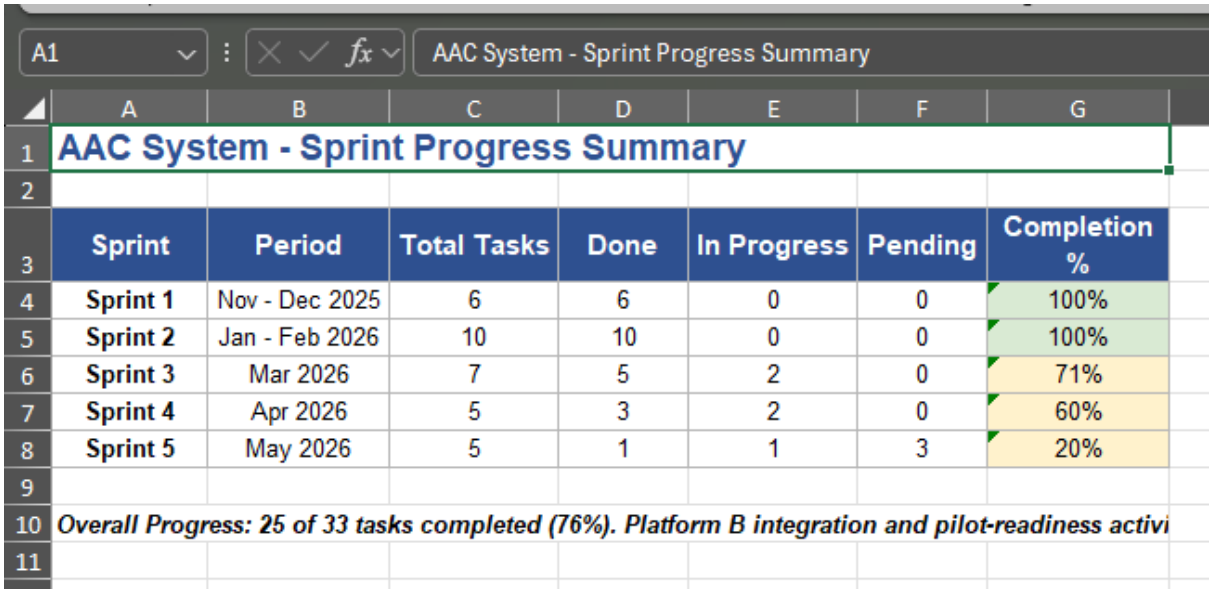
Figure 20: Google Sheets Sprint Tracking Spreadsheet

Link:https://docs.google.com/spreadsheets/d/1m2-U_Ga_bLSauQyvLuXoFvqTGAIyuhJ-/edit?usp=sharing&ouid=106973302403748897349&rtpof=true&sd=true



A1 | AAC System - Product Backlog

	A	B	C	D	E
1	**AAC System - Product Backlog**				
2	*Project: AI-Powered AAC System for Children with ASD (Sinhala/Tamil/English)*				
3					
4	**#**	**Backlog Item**	**Priority**	**Target Sprint**	
5	1	Literature review and requirements analysis	High	Sprint 1	
6	2	System architecture and technology selection	High	Sprint 1	
7	3	Flutter project setup (Android and iOS targets)	High	Sprint 1	
8	4	Symbol grid, navigation, and basic AAC flow	High	Sprint 2	
9	5	Trilingual UI framework (Sinhala, Tamil, English)	High	Sprint 2	
10	6	Basic TTS integration	Medium	Sprint 2	
11	7	Firebase project setup (exploratory)	Medium	Sprint 2	
12	8	AI model pipeline setup and initial training	High	Sprint 2-3	
13	9	Full model training and TFLite export	High	Sprint 3	
14	10	Platform B FER integration	High	Sprint 3	
15	11	Emotion-adaptive vocabulary logic	Medium	Sprint 3	
16	12	Dashboard completion	Medium	Sprint 4	
17	13	Ethics application and approval	High	Sprint 4	
18	14	Pilot protocol and participant recruitment	Medium	Sprint 4	
19	15	Pilot execution and data collection	High	Sprint 5	
20	16	Final report and deliverables	High	Sprint 5	
21					



A1 | AAC System - Sprint Progress Summary

	A	B	C	D	E	F	G
1	AAC System - Sprint Progress Summary						
2							
3	Sprint	Period	Total Tasks	Done	In Progress	Pending	Completion %
4	Sprint 1	Nov - Dec 2025	6	6	0	0	100%
5	Sprint 2	Jan - Feb 2026	10	10	0	0	100%
6	Sprint 3	Mar 2026	7	5	2	0	71%
7	Sprint 4	Apr 2026	5	3	2	0	60%
8	Sprint 5	May 2026	5	1	1	3	20%
9							
10	*Overall Progress: 25 of 33 tasks completed (76%). Platform B integration and pilot-readiness activi*						
11							

the workbook used for sprint tracking was organised into several sheets so that backlog management, day-to-day progress, and governance could be recorded in one place. The Product Backlog sheet listed prioritised features and epics. The Sprint Backlogs sheet broke work down by sprint. The Sprint Tracker sheet followed a Kanban-style layout for task status. Additional sheets supported project oversight, namely Sprint Summary for high-level progress, Milestones for key dates, Meeting Log for supervisor and stakeholder reviews, Issue Tracker for defects and blockers, Decision Log for agreed changes, Retrospectives for sprint reflections, and Time Log for effort records. The same structure was maintained when the file was prepared for upload to Google Drive and reviewed as a screenshot for Figure 20.

The Sprint Summary sheet reports aggregate progress across the full product backlog (for example, 25 of 33 tasks completed, approximately 76 per cent). This figure reflects backlog completion across all planned sprints and includes completed early-start items brought forward from later sprint plans. Tasks are marked as complete only where implementation evidence is available, while partially completed activities are retained as in progress to avoid overstating delivery. The remaining items are concentrated in pilot execution, final analysis, and end-stage documentation tasks scheduled toward the final submission window.

5.5 Risk Assessment for Remaining Work

The most significant risk to the revised plan is a delay in ethics approval, which would prevent purpose collected data from being included in the training dataset and would delay the pilot study. The primary contingency is to complete model training and evaluation using public datasets only, and to conduct Platform A usability testing with informal participants rather than a formal pilot. This would still produce a valid proof of concept, though with acknowledged limitations.

Other risks, including model accuracy below the 80 per cent target, insufficient TTS quality for Sinhala and Tamil, and limited therapist or parent availability for the pilot, are mitigated through the strategies documented in Section 2.10.

5.6 Work Breakdown Structure

The project is divided into the following main work packages.

Requirements and Design covers the literature review, stakeholder analysis, requirements gathering, system architecture design, and technology selection.

Platform A Development covers Flutter setup, symbol grid implementation, trilingual support, text to speech integration, customisation features, visual scheduling, and usage logging.

AI Model covers data sourcing, preprocessing, transfer learning setup, model training, evaluation, quantisation, and benchmarking.

Platform B Integration covers face detection, TFLite inference, emotion adaptive logic, caregiver override, and end to end testing.

Backend and Dashboard covers Firebase configuration, schema design, security considerations, offline synchronisation handling, dashboard development, and progress visualisation.

Ethics and Pilot covers consent forms, translations, the ethics application, Ministry coordination, participant recruitment, pilot execution, and analysis.

Documentation covers the interim report, final report, user documentation, presentation materials, and code archival.

5.7 Conclusion

The project has completed Sprint 1 in full and the majority of Sprint 2. This section summarises the progress made against each of the six project objectives and outlines the evaluation work that remains before the objectives can be fully achieved.

Objective 1 was to design a dual-platform system architecture with trilingual support, offline-first operation, and secure data management. This objective has been substantially achieved. The system architecture, including the dual-platform design (Platform A for ASD Level 1-2 and Platform B for Level 3+), the offline-first data layer, and the technology stack, has been fully designed and documented. The architecture diagram, ER diagram, use case diagram, and class diagram have all been produced as design artefacts. What remains is the implementation of optional Firebase synchronisation and the completion of security hardening, both of which are planned for later sprints.

Objective 2 was to implement a cross-platform mobile application using Flutter. This objective is partially achieved. The Flutter project has been set up with build targets for both Android and iOS. Core AAC features, including the symbol grid, category navigation, multilingual switching framework, and basic text-to-speech, are functional at a prototype level. The application has been deployed to the Google Play Console under the Internal Testing track and tested on physical Android devices. Platform A completion (full trilingual vocabulary, customisation features, visual scheduling) and iOS testing remain outstanding.

Objective 3 was to train and deploy an EfficientNetB0-based FER model targeting at least 80% accuracy across six emotion classes. This objective is in progress. The AI model pipeline has been set up on Google Colab, preliminary training experiments have been completed using public datasets, and the model architecture has been finalised (EfficientNetB0 with CBAM, after MobileNetV2 was evaluated and replaced). Current results show training accuracy of 85-87% and validation accuracy of 82-84%, though test accuracy remains lower at approximately 66%. Full model training on the complete dataset, TFLite export, and on-device integration are

planned for Sprint 3. Closing the generalisation gap between validation and test accuracy is the primary model objective going forward.

Objective 4 was to develop a therapist-parent dashboard. This objective is at an early stage. A basic dashboard prototype has been started with login UI, navigation, and placeholder progress views. Full dashboard development, including progress visualisations, vocabulary management, and data export, is planned for Sprint 4.

Objective 5 was to establish ethical approval protocols and a clinical partnership for a pilot study at Karapitiya Teaching Hospital. This objective is in progress. Initial contact with Karapitiya Teaching Hospital has been established, and an exploratory field exposure was conducted to understand the clinical context. Draft consent forms and participant information sheets have been prepared in English (included in Appendices A and B). The formal ethics application is in advanced preparation but has not yet been submitted. Ministry of Health approval requirements have been researched and documented.

Objective 6 was to document the project in accordance with FC6P01ES module requirements. This objective is on track. The interim report has been prepared using Harvard referencing, with all sections structured according to the module template.

Remaining Work and Evaluation

The core development work for Sprints 1 and 2 has established the foundation of the system, but the most critical phase of the project lies ahead. The primary remaining task is the formal evaluation of the system, which is essential for determining whether the project objectives have been fully met. The evaluation will follow the mixed-methods approach described in Section 2.7. Quantitatively, the FER model will be evaluated against the 80% accuracy target using precision, recall, F1-score, and a confusion matrix, and system usability will be measured through structured questionnaires and the System Usability Scale (SUS) administered to caregivers and therapists. Qualitatively, semi-structured interviews and observational notes from the pilot study at Karapitiya Teaching Hospital will capture the practical experiences of users and identify areas for improvement.

It is through this evaluation that the project objectives will ultimately be validated or refined. The quantitative metrics will determine whether the system meets its defined performance and

usability thresholds, while the qualitative findings will reveal whether the system is genuinely useful and appropriate in the Sri Lankan clinical and home context. The conclusions in the final report will be drawn by triangulating these two streams of evidence, as described in Section 2.7.5.

Risks and Contingency

The most significant risk is a delay in ethics approval, which would prevent purpose-collected data from being included in the training set and would push back the pilot study. A contingency plan exists for this scenario. Model training and evaluation would proceed using public datasets only, and Platform A usability testing would be conducted with informal participants. This would still produce a valid proof of concept, though with acknowledged limitations in the evaluation.

Some tasks have taken longer than the original plan allowed for, particularly symbol set licensing, ethics coordination, and the discovery that platform TTS engines do not produce acceptable Sinhala and Tamil output. These delays are acknowledged but none are severe enough to compromise the final deadline, provided the revised plan (Section 5.4) is followed.

Summary

At this stage, the project is in a reasonable position relative to the academic timeline. The foundational architecture, the development environment, the preliminary AI model experiments, and the clinical partnership groundwork are all in place. The remaining sprints focus on completing the implementation, conducting the formal evaluation, and drawing evidence-based conclusions on whether the system effectively addresses the identified gap in AAC provision for children with autism in Sri Lanka.

6. References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Mane, R., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. and Zheng, X. (2016) 'TensorFlow: a system for large-scale machine learning', in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*. Berkeley, CA: USENIX Association, pp. 265-283.

Alant, E. and Bornman, J. (2021) *Augmentative and alternative communication: engagement and participation*. San Diego, CA: Plural Publishing.

American Psychiatric Association (2013) *Diagnostic and statistical manual of mental disorders*. 5th edn. Washington, DC: American Psychiatric Publishing.

American Speech-Language-Hearing Association (2022) *Augmentative and alternative communication (AAC)*. Available at: https://www.asha.org/public/speech/disorders/aac/ (Accessed: 22 February 2025).

Barrett, L.F., Adolphs, R., Marsella, S., Martinez, A.M. and Pollak, S.D. (2019) 'Emotional expressions reconsidered: challenges to inferring emotion from human facial movements', *Psychological Science in the Public Interest*, 20(1), pp. 1-68.

Beukelman, D.R. and Light, J.C. (2020) *Augmentative and alternative communication: supporting children and adults with complex communication needs*. 5th edn. Baltimore, MD: Paul H. Brookes.

Boehm, B.W. (1988) 'A spiral model of software development and enhancement', *Computer*, 21(5), pp. 61-72.

Brooke, J. (1996) 'SUS: a "quick and dirty" usability scale', in Jordan, P.W., Thomas, B., Weerdmeester, B.A. and McClelland, I.L. (eds.) *Usability Evaluation in Industry*. London: Taylor and Francis, pp. 189-194.

Creswell, J.W. and Creswell, J.D. (2018) *Research design: qualitative, quantitative, and mixed methods approaches*. 5th edn. Thousand Oaks, CA: SAGE Publications.

Dawe, M. (2006) 'Desperately seeking simplicity: how young adults with cognitive disabilities and their families adopt assistive technologies', in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York: ACM, pp. 1143-1152.

Department of Census and Statistics, Sri Lanka (2012) *Census of population and housing - 2012*. Colombo: Department of Census and Statistics.

Divan, G., Vajaratkar, V., Desai, M.U., Strik-Lievers, L. and Patel, V. (2021) 'Prevalence and risk factors for autism spectrum disorder in low- and middle-income countries: a systematic review and meta-analysis', *Global Mental Health*, 8, e30.

Ekman, P. and Friesen, W.V. (1971) 'Constants across cultures in the face and emotion', *Journal of Personality and Social Psychology*, 17(2), pp. 124-129.

Elsabbagh, M., Divan, G., Koh, Y.J., Kim, Y.S., Kauchali, S., Marcin, C., Montiel-Nava, C., Patel, V., Paula, C.S., Wang, C., Yasamy, M.T. and Fombonne, E. (2012) 'Global prevalence of autism and other pervasive developmental disorders', *Autism Research*, 5(3), pp. 160-179.

Firebase (2023) *Firebase documentation*. Available at: https://firebase.google.com/docs (Accessed: 15 January 2025).

Fletcher-Watson, S. and Happe, F. (2019) *Autism: a new introduction to psychological theory and current debate*. 2nd edn. Abingdon: Routledge.

Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Luetge, C., Madelin, R., Pagallo, U., Rossi, F., Schafer, B., Valcke, P. and Vayena, E. (2018) 'AI4People: an ethical framework for a good AI society', *Minds and Machines*, 28(4), pp. 689-707.

Flutter (2023) *Flutter documentation*. Available at: https://flutter.dev/docs (Accessed: 15 January 2025).

Ganz, J.B. (2015) *AAC for individuals with autism spectrum disorders*. New York: Springer.

Ganz, J.B., Davis, J.L., Lund, E.M., Goodwyn, F.D. and Simpson, R.L. (2012) 'A meta-analysis of single case research studies on aided augmentative and alternative communication systems with individuals with autism spectrum disorders', *Journal of Autism and Developmental Disorders*, 42(1), pp. 60-74.

Goodfellow, I., Bengio, Y. and Courville, A. (2015) *Deep learning*. Cambridge, MA: MIT Press.

Grossard, C., Dapogny, A., Cohen, D., Bernheim, S., Martinerie, J., Janvier, M., Grynszpan, O., Chaby, L., Bailly, K. and Dubuisson, S. (2020) 'Children with autism spectrum disorder produce more ambiguous and less socially meaningful facial expressions', *Molecular Autism*, 11(1), p. 5.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H. (2017) 'MobileNets: efficient convolutional neural networks for mobile vision applications', *arXiv preprint arXiv:1704.04861*.

Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H. and Le, Q. (2019) 'Searching for MobileNetV3', in *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Piscataway, NJ: IEEE, pp. 1314-1324.

International Telecommunication Union (2022) *Measuring digital development: facts and figures 2022*. Geneva: ITU.

Ioffe, S. and Szegedy, C. (2015) 'Batch normalization: accelerating deep network training by reducing internal covariate shift', in *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: JMLR, pp. 448-456.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. and Kalenichenko, D. (2018) 'Quantization and training of neural networks for efficient integer-arithmetic-only inference', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE, pp. 2704-2713.

Jobin, A., Ienca, M. and Vayena, E. (2019) 'The global landscape of AI ethics guidelines', *Nature Machine Intelligence*, 1(9), pp. 389-399.

Kothari, C.R. (2004) *Research methodology: methods and techniques*. 2nd edn. New Delhi: New Age International Publishers.

LeCun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning', *Nature*, 521(7553), pp. 436-444.

Li, S. and Deng, W. (2020) 'Deep facial expression recognition: a survey', *IEEE Transactions on Affective Computing*, 13(3), pp. 1195-1215.

Light, J.C. and McNaughton, D. (2012) 'The changing face of augmentative and alternative communication: past, present, and future challenges', *Augmentative and Alternative Communication*, 28(4), pp. 197-204.

Light, J.C. and McNaughton, D. (2015) 'Designing AAC research and intervention to improve outcomes for individuals with complex communication needs', *Augmentative and Alternative Communication*, 31(2), pp. 85-96.

Lord, C., Brugha, T.S., Charman, T., Cusack, J., Dumas, G., Frazier, T., Jones, E.J.H., Jones, R.M., Pickles, A., State, M.W., Taylor, J.L. and Veenstra-VanderWeele, J. (2020) 'Autism spectrum disorder', *Nature Reviews Disease Primers*, 6(1), p. 5.

Lorah, E.R., Parnell, A., Whitby, P.S. and Hantula, D. (2015) 'A systematic review of tablet computers and portable media players as speech generating devices for individuals with autism spectrum disorder', *Journal of Autism and Developmental Disorders*, 45(12), pp. 3792-3804.

Maenner, M.J., Warren, Z., Williams, A.R. et al. (2023) 'Prevalence and characteristics of autism spectrum disorder among children aged 8 years', *MMWR Surveillance Summaries*, 72(2), pp. 1-14.

Mazefsky, C.A., Herrington, J., Siegel, M., Scarpa, A., Maddox, B.B., Scahill, L. and White, S.W. (2013) 'The role of emotion regulation in autism spectrum disorder', *Journal of the American Academy of Child and Adolescent Psychiatry*, 52(7), pp. 679-688.

McNaughton, D. and Light, J. (2013) 'The iPad and mobile technology revolution: benefits and challenges for individuals who require augmentative and alternative communication', *Augmentative and Alternative Communication*, 29(2), pp. 107-116.

Millar, D.C., Light, J.C. and Schlosser, R.W. (2006) 'The impact of augmentative and alternative communication intervention on the speech production of individuals with developmental disabilities', *Journal of Speech, Language, and Hearing Research*, 49(2), pp. 248-264.

Ministry of Health, Sri Lanka (2020) *National guideline for ethics review of health research in Sri Lanka*. Colombo: Ministry of Health.

Nair, V. and Hinton, G.E. (2010) 'Rectified linear units improve restricted Boltzmann machines', in *Proceedings of the 27th International Conference on Machine Learning*. Madison, WI: Omnipress, pp. 807-814.

Perera, H., Wijewardena, K., Aluthwelage, R., Seneviratne, S. and Buddhika, K. (2019) 'Prevalence of autism spectrum disorder in Sri Lanka: a population-based study', *Sri Lanka Journal of Child Health*, 48(2), pp. 133-138.

Picard, R.W. (2000) *Affective computing*. Cambridge, MA: MIT Press.

Romski, M. and Sevcik, R.A. (2005) 'Augmentative communication and early intervention: myths and realities', *Infants and Young Children*, 18(3), pp. 174-185.

Samad, A., Razick, S., De Silva, M.V.C. and Hettiarachchi, S. (2020) 'Challenges and opportunities for autism services in Sri Lanka', *Journal of Autism and Developmental Disorders*, 50(8), pp. 3023-3029.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C. (2018) 'MobileNetV2: inverted residuals and linear bottlenecks', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE, pp. 4510-4520.

Shorten, C. and Khoshgoftaar, T.M. (2019) 'A survey on image data augmentation for deep learning', *Journal of Big Data*, 6(1), p. 60.

Sommerville, I. (2016) *Software engineering*. 10th edn. Harlow: Pearson Education.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) 'Dropout: a simple way to prevent neural networks from overfitting', *Journal of Machine Learning Research*, 15(1), pp. 1929-1958.

TensorFlow (2023) *TensorFlow Lite documentation*. Available at: https://www.tensorflow.org/lite (Accessed: 15 January 2025).

Trevisan, D.A., Hoskyn, M. and Birmingham, E. (2018) 'Facial expression production in autism: a meta-analysis', *Autism Research*, 11(12), pp. 1586-1601.

Wickramasinghe, N., Dissanayake, A. and Samarasinghe, D. (2021) 'Parent training and support programmes for autism in low-resource settings: a scoping review', *Global Health Action*, 14(1), 1910556.

World Health Organization (2021) *Autism spectrum disorders*. Available at: https://www.who.int/news-room/fact-sheets/detail/autism-spectrum-disorders (Accessed: 22 February 2025).

World Medical Association (2013) 'World Medical Association Declaration of Helsinki: ethical principles for medical research involving human subjects', *JAMA*, 310(20), pp. 2191-2194.

Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. (2014) 'How transferable are features in deep neural networks?', in *Advances in Neural Information Processing Systems*, 27. Red Hook, NY: Curran Associates, pp. 3320-3328.

7. Bibliography

The following sources were consulted during preparation of this report and have informed the background, methodology, or discussion.

Bishop, C.M. (2006) *Pattern recognition and machine learning*. New York: Springer.

Chollet, F. (2017) *Deep learning with Python*. Shelter Island, NY: Manning Publications.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L. (2009) 'ImageNet: a large-scale hierarchical image database', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE, pp. 248-255.

Ekman, P. and Friesen, W.V. (1978) *Facial Action Coding System: a technique for the measurement of facial movement*. Palo Alto, CA: Consulting Psychologists Press.

He, K., Zhang, X., Ren, S. and Sun, J. (2016) 'Deep residual learning for image recognition', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE, pp. 770-778.

Kaggle (2013) *FER-2013: Facial Expression Recognition 2013 Dataset*. Available at: https://www.kaggle.com/datasets/msambare/fer2013 (Accessed: 15 January 2025).

Mollahosseini, A., Hasani, B. and Mahoor, M.H. (2019) 'AffectNet: a database for facial expression, valence, and arousal computing in the wild', *IEEE Transactions on Affective Computing*, 10(1), pp. 18-31.

Pressman, R.S. and Maxim, B.R. (2019) *Software engineering: a practitioner's approach*. 9th edn. New York: McGraw-Hill Education.

Simonyan, K. and Zisserman, A. (2015) 'Very deep convolutional networks for large-scale image recognition', in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. San Diego, CA: ICLR.

End of Interim Report

Appendix A: Platform A (Level 1-2) User Interface and Features

This appendix presents additional user interface screens and implementation details related to Platform A, which is designed for children at Autism Spectrum Disorder (ASD) Level 1-2.

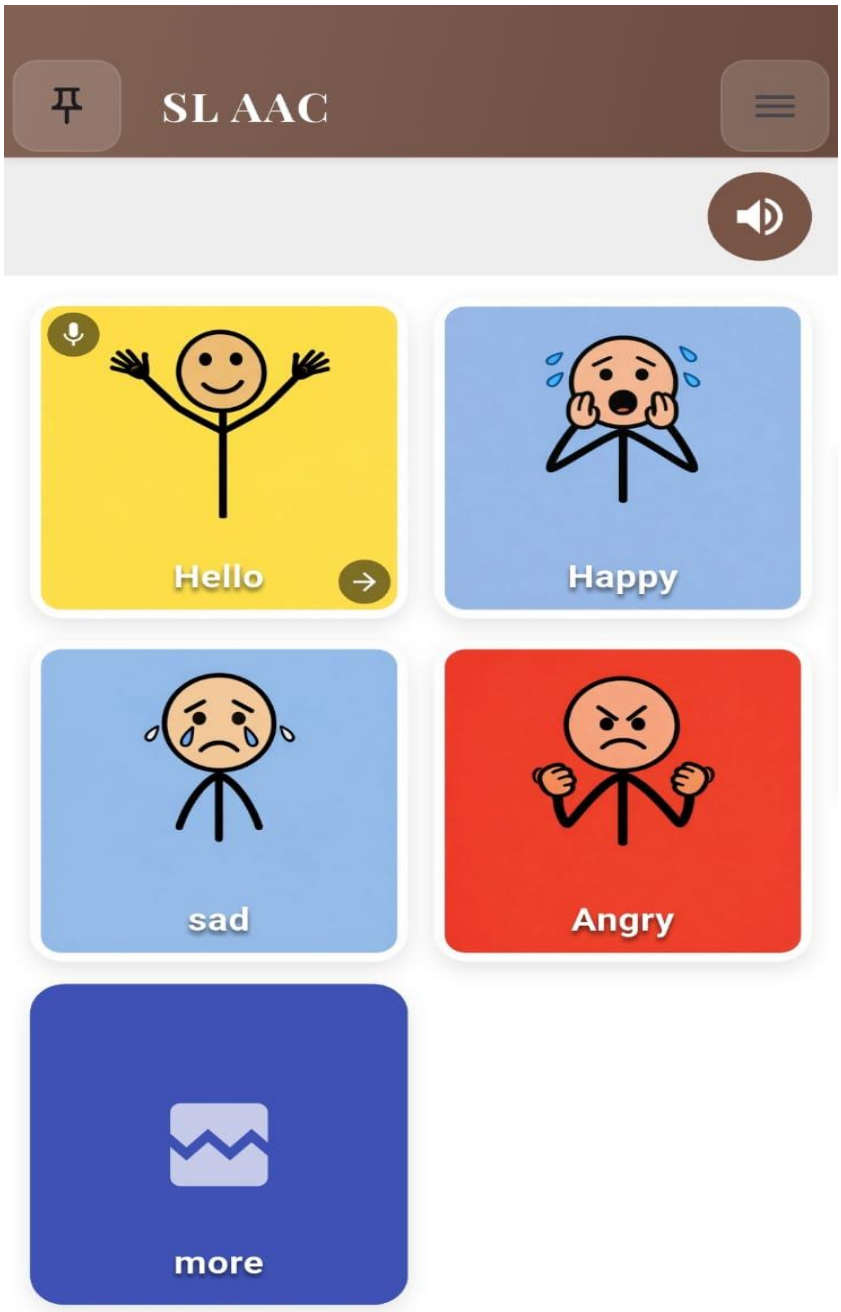
Platform A provides a customisable, symbol-based/Image Augmentative and Alternative Communication (AAC) interface with trilingual support (Sinhala, Tamil, and English). It is intended for children who possess some level of functional communication but benefit from structured visual support to enhance interaction.

The separation of the system into Platform A and Platform B was informed by clinical observations and initial field exposure at Karapitiya Teaching Hospital. Feedback from clinicians indicated that children at lower ASD severity levels require a simpler, highly customisable interface, while children at higher severity levels (Level 3 and above) require additional assistive mechanisms such as emotion recognition.

Although Platform A is an important component of the overall system, the primary research focus of this study is on Platform B, which integrates facial expression recognition (FER) to support children with severe communication impairments. Therefore, detailed figures and experimental results related to Platform B are presented in the main body of the report, while supporting visual evidence for Platform A is included in this appendix to maintain clarity and focus.

The following figures illustrate the key user interface components and interaction flow of Platform A.

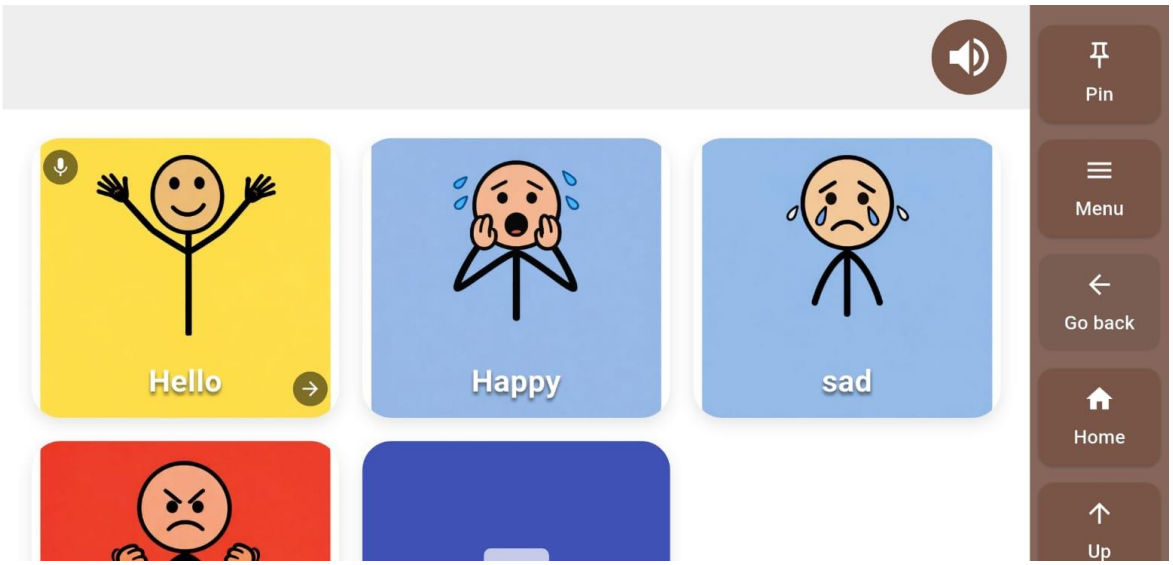
Figure A1: Platform A - Home Screen

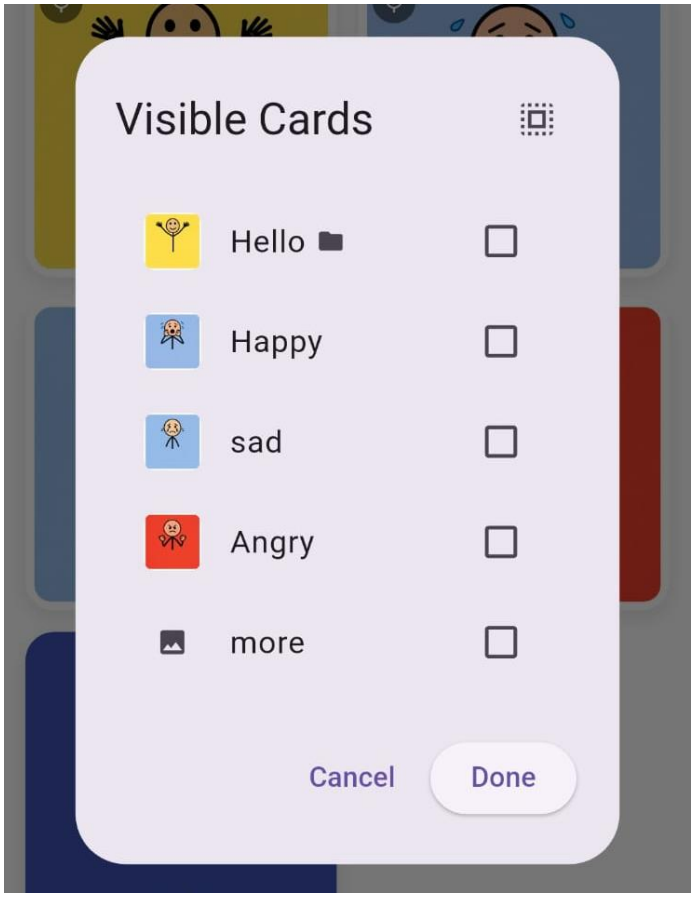


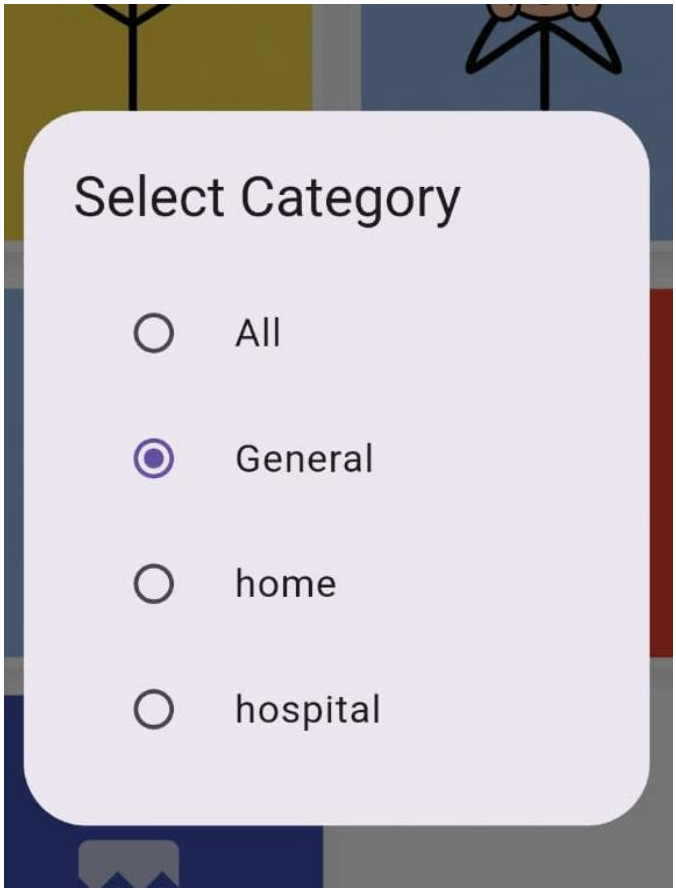
- This screen provides the home interface used for everyday symbol-based communication in Platform A.

- This screen allows the child to select common symbols (for example, greetings and emotion cards) with minimal navigation steps.
- This is important because a consistent, low-friction home layout supports reliable engagement and faster AAC access for ASD users.

Figure A2: Platform A - Category Selection Interface

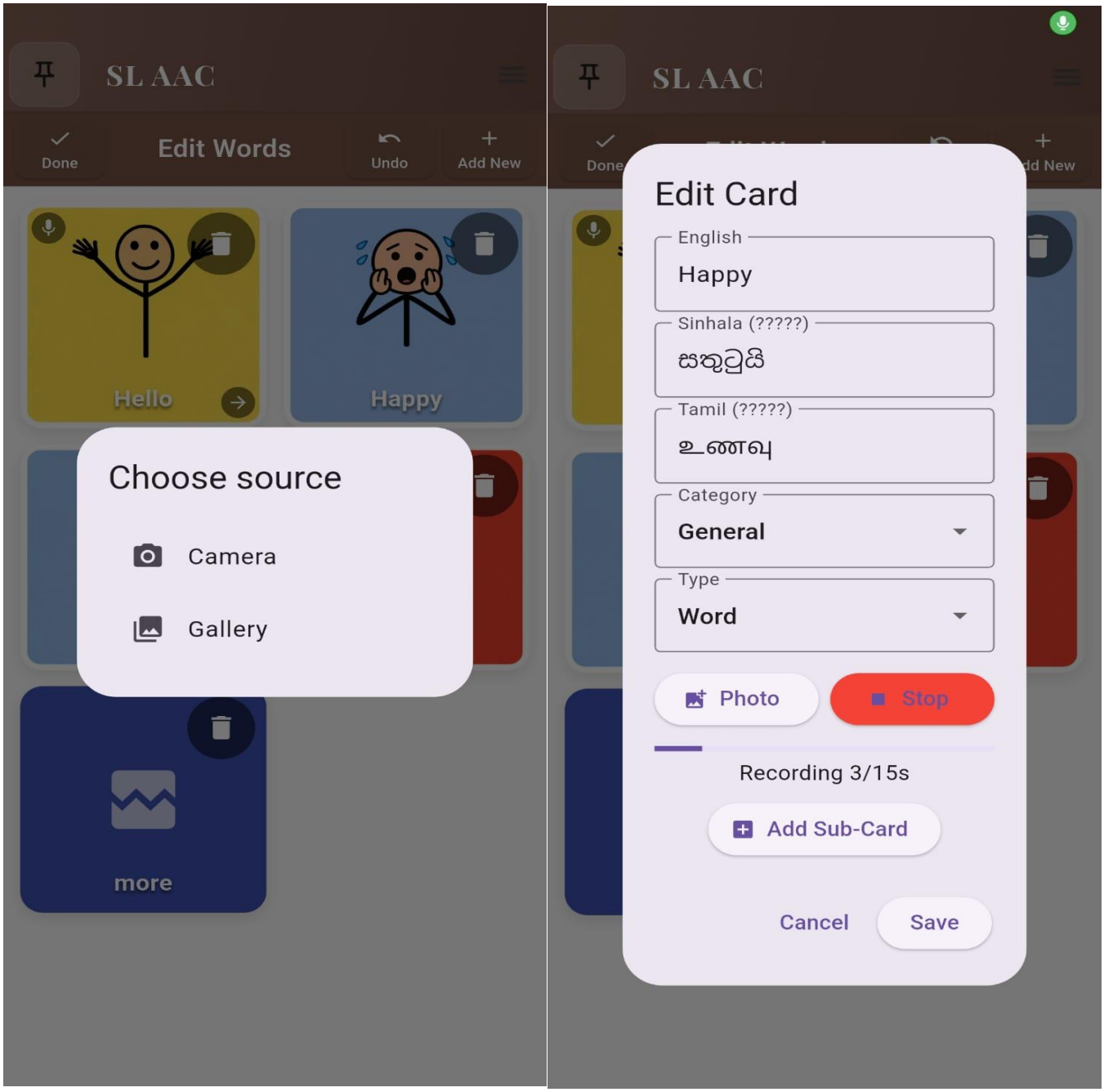






- This screen provides the category selection interface used to organise symbols in Platform A.
- This screen allows the user to switch between category filters (for example, All, General, home, and hospital) before selecting visible cards.
- This is important because category-based navigation improves symbol discoverability and reduces the effort required to locate appropriate communication options.

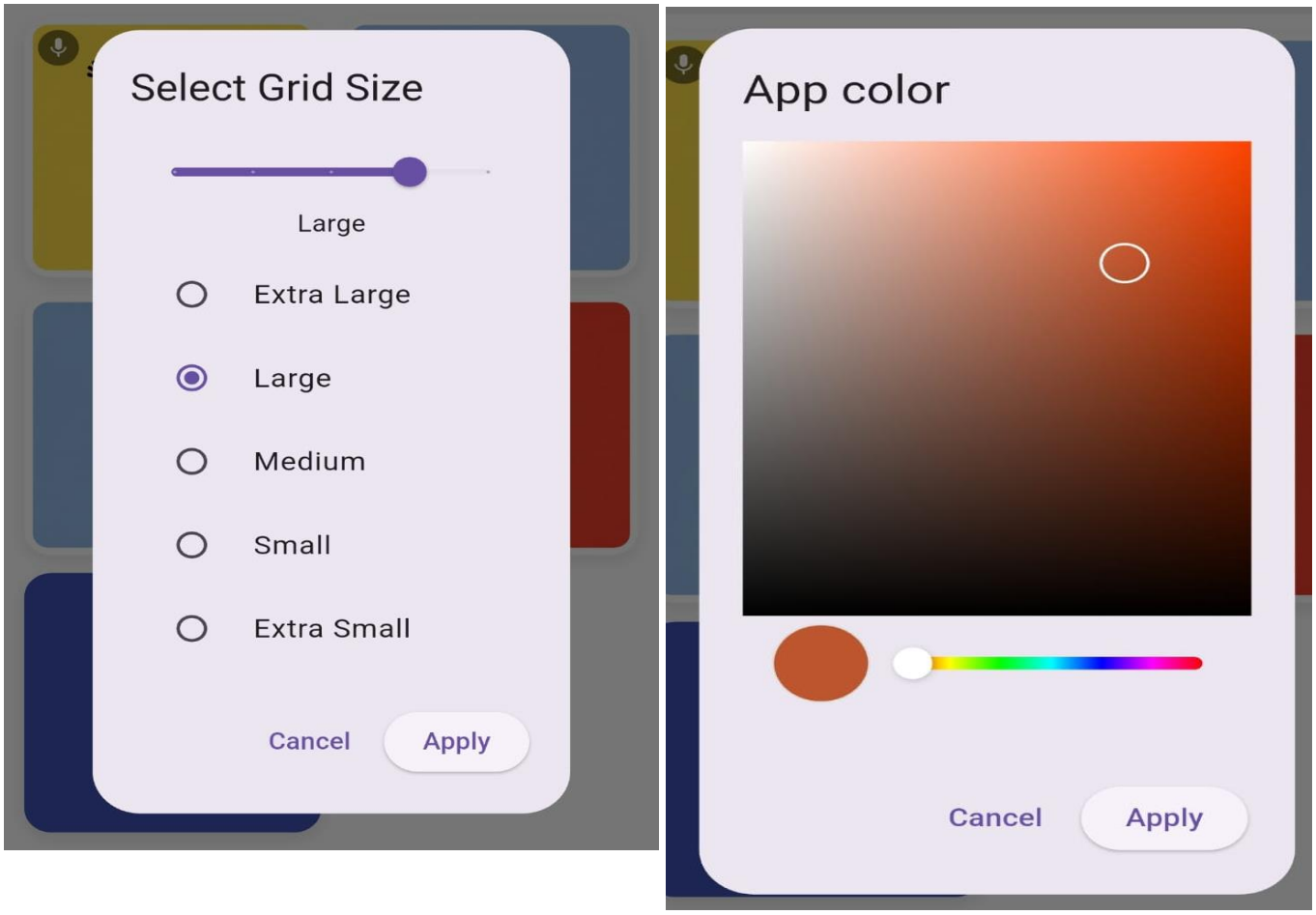
Figure A3: Platform A - Capture photo and record parent voices

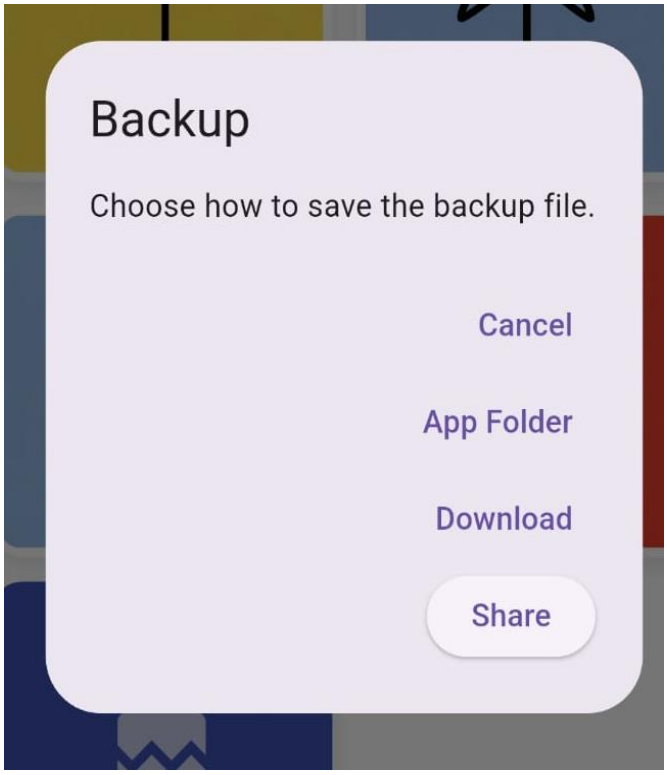


- This feature supports the creation of personalised cards by capturing photos or selecting images and recording parent voice samples.
- This screen allows caregivers to choose the image source (camera or gallery), edit bilingual text fields, select categories, and save recorded audio.

- This is important because personalised symbols and caregiver voice cues strengthen relevance and improve communication engagement in daily interaction.

Figure A4: Platform A - Settings and Customisation Screen





- This screen provides configuration controls for Platform A, including grid size selection, app colour theme, and backup handling.
- This screen allows caregivers to adjust visual layout (for example, large to extra-small grid sizes) and manage local backup options such as download and sharing.
- This is important because adaptive visual presentation and reliable offline backup support sustained usability in low-connectivity environments.

This appendix provides a compact user interface reference for Platform A (Level 1-2). The main body of the report concentrates on Platform B because it represents the primary research contribution, while the included Platform A evidence supports the overall system completeness without overloading the central narrative.

Appendix C, Architecture and design diagrams

This appendix provides supporting design evidence for the diagrams used in the main report. The main report contains the final readable versions of the diagrams, while this appendix records the editable source links for traceability.

The supporting design evidence includes:

- System architecture diagram (Figure 5.1 in the main body).
- Use case diagram (Figure 5.2 in the main body).
- Data model diagram (Figure 5.3 in the main body).
- Entity Relationship diagram (Figure 5.4 in the main body).
- Emotion detection pipeline diagram (Figure 5.5 in the main body).
- Supporting class structure view (Figure 5.6 in the main body).
- Trilingual content flow diagram (Figure 5.7 in the main body).
- Preprocessing pipeline diagram (Figure 6.10 in the main body).
- EfficientNetB0 preprocessing correction diagram (Figure 6.14 in the main body).
- Supplementary detailed class diagram (Figure C.6 in this appendix).

Table C.1: Editable diagram source links.

Diagram	**Related figure**	**Editable source link**	**Note**
System architecture diagram	Figure 5.1	Download editable system architecture diagram	Main Chapter 5 architecture view
Feasibility summary diagram	Figure 4.1	Download editable feasibility summary diagram	Chapter 4 feasibility overview

Diagram	Related figure	Editable source link	Note
ER diagram	Figure 5.4	Download editable ER diagram	Final ER/data relationship view used in Chapter 5.
Emotion detection pipeline	Figure 5.5	Download editable emotion detection pipeline diagram	Final supportive AI workflow
Supporting class structure view	Figure 5.6	Download editable supporting class structure diagram	Main Chapter 5 readable class-structure view
Preprocessing pipeline	Figure 6.10	Download editable preprocessing pipeline diagram	Data preprocessing workflow
EfficientNetB0 preprocessing correction diagram	Figure 6.14	Download editable EfficientNetB0 preprocessing correction diagram	Final preprocessing correction evidence
Supplementary detailed class diagram	Figure C.6	Download editable supplementary detailed class diagram	Detailed class-level appendix evidence.

The editable links are provided as supporting design evidence. They are included to show diagram traceability and do not replace the final figures presented in the main report.

High-resolution exported versions of the final diagrams are retained in the project evidence folder where available. Editable diagram source links are provided for the relevant figures and in Appendix C as supporting design evidence.

Project Evidence Folder

Supplementary detailed class diagram

Detailed class diagram evidence.

The detailed class diagram below is included as supplementary design evidence. It expands the cleaner class-structure view shown in Figure 5.6 by showing additional relationships between the main screens, services, local storage, vocabulary data, audio assets and emotion-support components. It is placed in Appendix C rather than the main Chapter 5 discussion so that the main design chapter remains readable while the more detailed class-level evidence remains available for inspection.

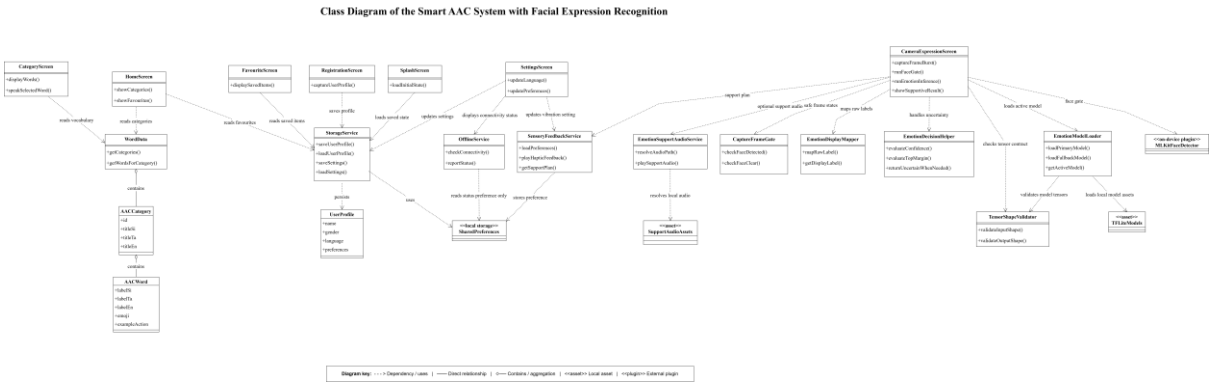


Figure C.6: Supplementary detailed class diagram for the Smart AAC system.

Editable diagram source: Download editable supplementary detailed class diagram

Dashed arrows indicate dependency-style relationships, such as a screen using a service or a service accessing a local asset. Solid relationships and containment notation are used where the diagram shows stronger structural grouping.

Appendix D, Model training evidence

This appendix collects the model training evidence referenced in Chapter 6, Section 6.4.

- Google Colab notebook used for model training and TensorFlow Lite export. Link: Final Colab notebook.
- Google Drive folder containing the exported model artefacts (emotion_efficientnetb0_finetuned.tflite, emotion_mobilenetv2_fallback.tflite, labels.txt). Link: Model output folder.
- The following figures provide notebook, model architecture and training-session records used to support the model implementation discussion.

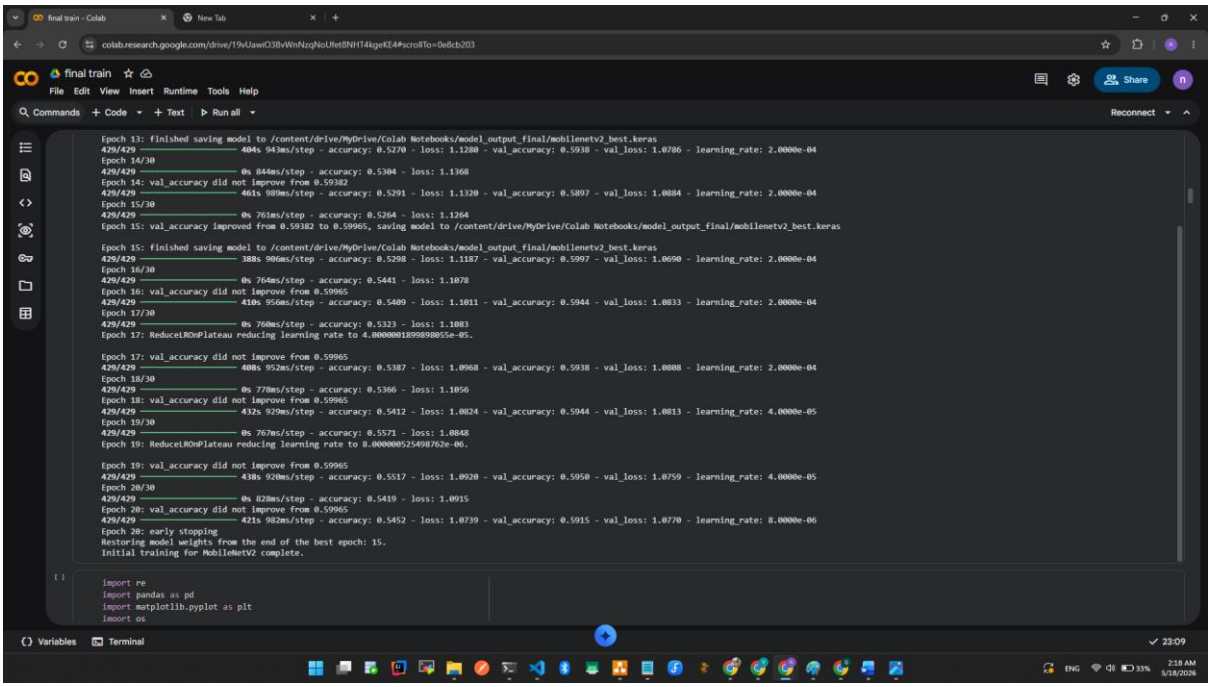
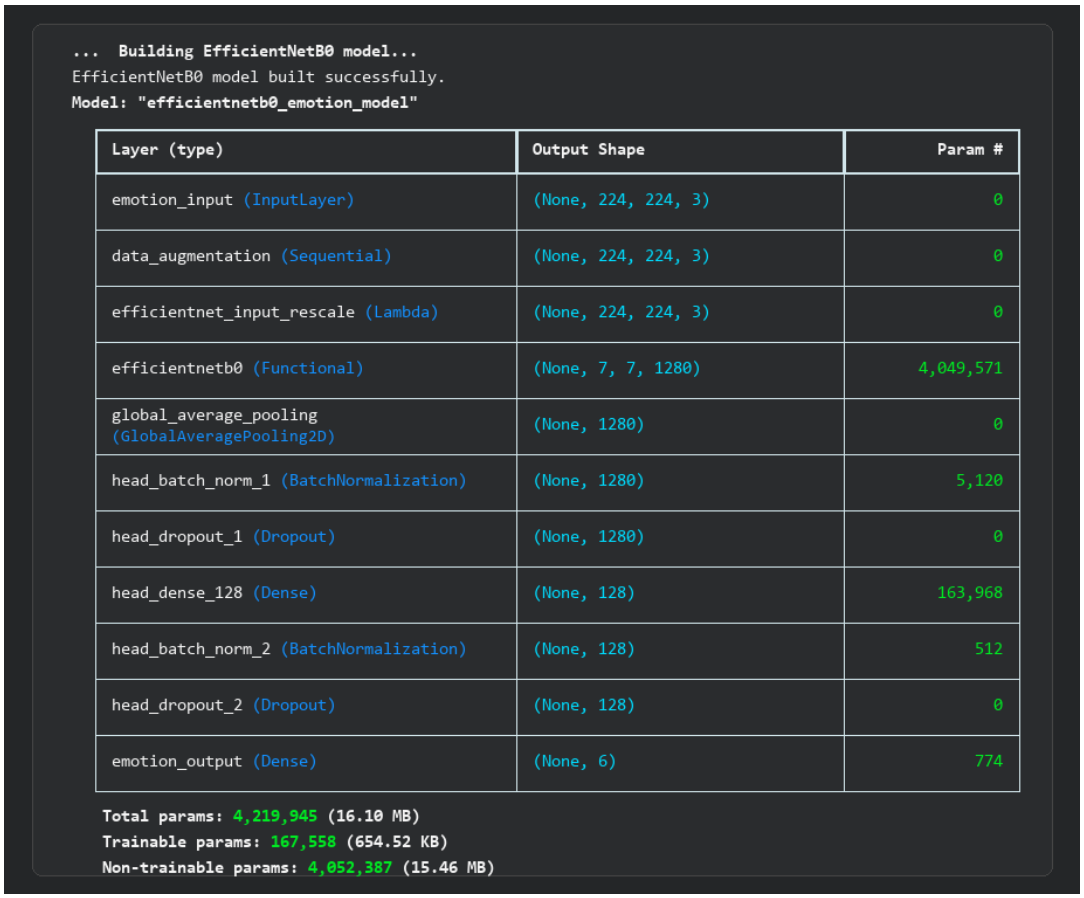


Figure D.1: Google Colab training notebook.

The notebook header confirms project ownership and the model output package saved to Google Drive.



... Building EfficientNetB0 model...
EfficientNetB0 model built successfully.
Model: "efficientnetb0_emotion_model"

Layer (type)	Output Shape	Param #
emotion_input (InputLayer)	(None, 224, 224, 3)	0
data_augmentation (Sequential)	(None, 224, 224, 3)	0
efficientnet_input_rescale (Lambda)	(None, 224, 224, 3)	0
efficientnetb0 (Functional)	(None, 7, 7, 1280)	4,049,571
global_average_pooling (GlobalAveragePooling2D)	(None, 1280)	0
head_batch_norm_1 (BatchNormalization)	(None, 1280)	5,120
head_dropout_1 (Dropout)	(None, 1280)	0
head_dense_128 (Dense)	(None, 128)	163,968
head_batch_norm_2 (BatchNormalization)	(None, 128)	512
head_dropout_2 (Dropout)	(None, 128)	0
emotion_output (Dense)	(None, 6)	774

Total params: 4,219,945 (16.10 MB)
Trainable params: 167,558 (654.52 KB)
Non-trainable params: 4,052,387 (15.46 MB)

Figure D.2: EfficientNetB0 model summary.

The summary relates to the primary emotion-recognition model.

The model summary confirms the deployed EfficientNetB0-based architecture used for six-class emotion recognition. The base network produces a 7 × 7 × 1280 feature map, followed by global average pooling and dense classification layers for the six output classes. The summary is included as supplementary implementation evidence rather than as a separate evaluation result.

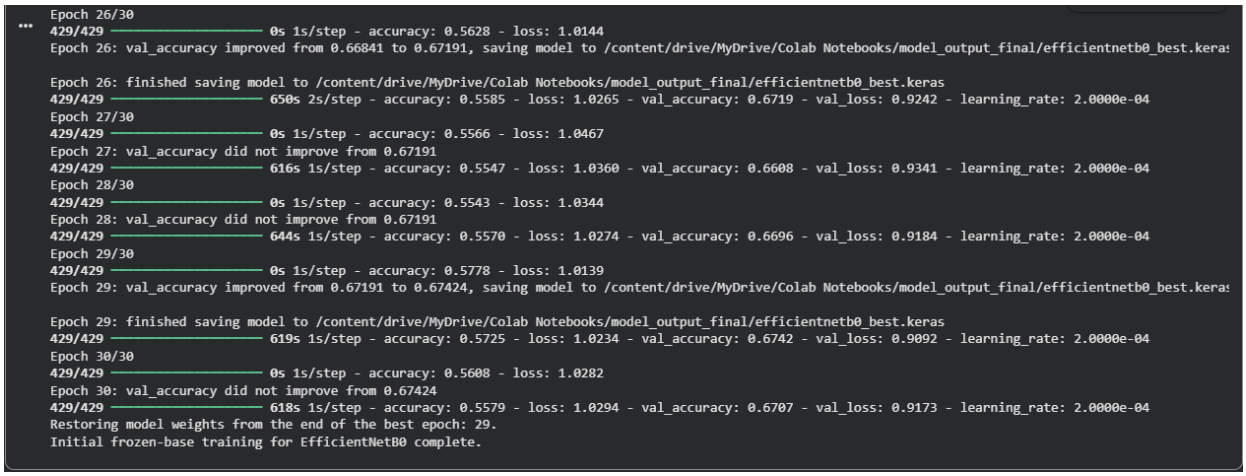


Figure D.3: EfficientNetB0 checkpoint recovery evidence.

The figure records training continuation and checkpoint recovery from the final Google Colab training session.

The final EfficientNetB0 training session completed 30 epochs in Google Colab with checkpoint-based recovery enabled. A temporary notebook reconnection and synchronisation issue occurred after epoch 28. The saved checkpoint state and training logs show that training continued to epoch 30 and that the best-performing weights were retained in the exported model artefacts.

Appendix E, TFLite and model package evidence

This appendix records the files actually shipped inside the APK and the verification artefacts.

Table E.1: Library and package list.

The table is carried over from the interim report and updated for the current build.

Package / dependency	Version (current pubspec.yaml)	Role
flutter	SDK	Cross-platform framework
flutter_tts	^3.8.3	Text-to-speech
shared_preferences	^2.2.2	Local key-value persistence
http	^1.1.0	(Reserved for future use)
connectivity_plus	^5.0.0	Connectivity information for offline-first behaviour
camera	^0.11.0+2	Front / back camera abstraction
tflite_flutter	^0.12.1	On-device TensorFlow Lite inference
image	^4.1.7	Image decoding / resize / crop
google_mlkit_face_detection	^0.13.0	Face-gate using ML Kit
audioplayers	^5.2.1	Supportive WAV playback
flutter_test	SDK	Unit / widget testing
flutter_launcher_icons	^0.13.1	Launcher icon generation
flutter_lints	^6.0.0	Recommended lint rules

Model package (verified against assets/models/):

- assets/models/emotion_efficientnetb0_finetuned.tflite (4.7 MB)
- assets/models/emotion_mobilenetv2_fallback.tflite (2.6 MB)
- assets/models/labels.txt (six raw labels: Anger, Fear, Joy, Natural, Sadness, Surprise)

Tensor contract for both models: input [1, 224, 224, 3] float32, output [1, 6] float32, and no Flex / SELECT_TF_OPS.

Appendix F, Flutter application screenshots and demonstration evidence

Screenshots used in Chapter 6, Section 6.3 and Section 6.5 are archived here in higher resolution where available. The evidence folder also includes supporting demonstration material for the final Flutter application.

• F.1 Splash and registration screen (Figure 6.2).

• F.2 Home screen (Figure 6.3).

• F.3 Categories screen (Figure 6.4).

• F.4 AAC interaction example with TTS (Figure 6.5).

• F.5 Favourites screen (Figure 6.6).

• F.6 Settings screen (Figure 6.7).

• F.7 Trilingual UI side-by-side (Figure 6.8).

• F.8 Active model debug screen (Figure 6.22).

• F.9 Live camera emotion detection screen (Figure 6.23).

• F.10 Optional support audio folder layout (Figure 6.24).

• F.11 Personal card capture and customisation flow (Figure 6.25).

Application screenshot and demonstration evidence folder

Status: the final app screenshots are included in the main implementation chapter and archived in the supporting evidence folder where available.

Appendix G, Google Play testing and production access evidence

This appendix collects the verified release evidence currently present in the project tree, internal-testing evidence, and the production access request evidence shown in the Play Console screenshot.

G.1 Release APK build evidence (verified)

- **Output file (Flutter):** build/app/outputs/flutter-apk/app-release.apk, 126,196,920 bytes (126.2 MB).
- **Output file (Gradle mirror):** build/app/outputs/apk/release/app-release.apk, same file produced by the same build.
- **APK SHA-1:** 43eb07192f0b2eb58393c903d583e79ab1068a67 (from app-release.apk.sha1).

- **Application ID:** lk.aac.sinhala_tamil_english.
- **Version name:** 1.1.0. **Version code:** 2.
- **Variant:** release. **Baseline profiles** included for minApi: 28-30 and minApi: 31+.
- **Native debug symbols:** build/app/outputs/native-debug-symbols/release/native-debug-symbols.zip (31 MB), available for future crash-report symbolication.
- **Source of these values:** build/app/outputs/apk/release/output-metadata.json (parsed during the writing pass).

G.2 Google Play internal testing (per interim report)

The Android application was distributed through Google Play internal testing for controlled review and real-device testing. The interim report (Fernando, 2026, Section 3.10) records the upload of an Android App Bundle to the Google Play Console under the Internal Testing track, the distribution to invited internal testers and their verification of AAC navigation, basic settings and on-device functionality. The corresponding Play Store listing URL pattern is https://play.google.com/store/apps/details?id=lk.aac.sinhala_tamil_english.

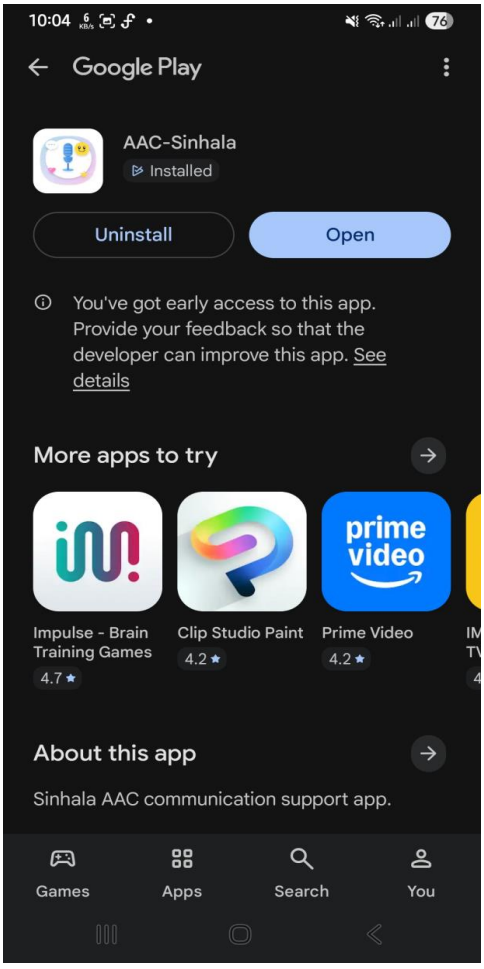


Figure G.1: Google Play internal testing evidence.

This appendix copy records controlled internal-testing evidence.

G.3 Google Play production access request (under review)

The visible Play Console evidence for AAC-Sinhala (package lk.aac.sinhala_tamil_english) shows production status as inactive and the production access request as submitted / under review. At the time this update was prepared (16 May 2026), the screenshot showed the applied time as "Today, 1:01 AM" in the Play Console interface. Google's message stated that the application for production access had been received and was being reviewed, with an email update to the account owner expected usually within seven days or less. This is evidence of a submitted request only. Public production release approval is not claimed.

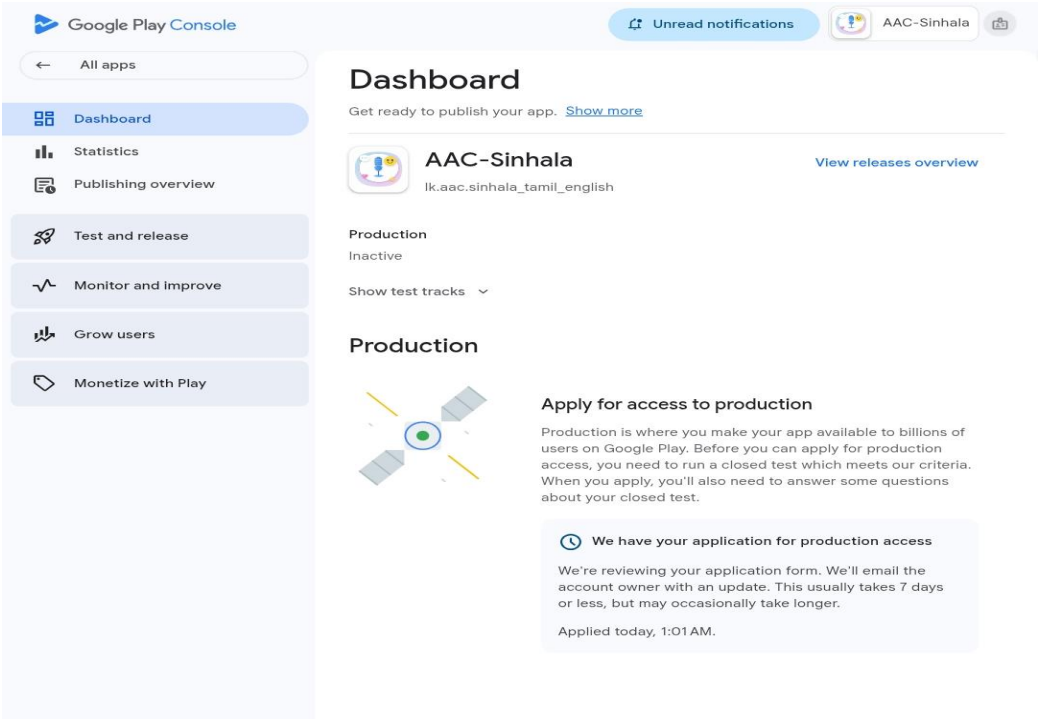


Figure G.2: Google Play production access request.

The figure shows the production access request submitted for the AAC-Sinhala application.

G.4 iOS pathway

iOS deployment has not yet been undertaken. There is currently no Apple Developer account in place. As recorded in the interim report (Fernando, 2026, Section 3.10), discussions are ongoing with Ideahub (Pvt) Ltd regarding the possibility of providing the developer account and distribution infrastructure for iOS, potentially as part of a charitable initiative following the broader health-sector approval pathway. The report does not claim a confirmed Apple App Store release. No separate iOS evidence is claimed in this submission.

G.5 Supporting links

Table G.1: Supporting links.

The table records internal testing, source repository and model training notebook links.

Item	Link	Status
Google Play Console internal testing track	Private Play Console evidence. Screenshots are included in Figure 6.26, Figure 6.27, Figure G.1 and Figure G.2.	Private, confirmed via interim report 3.10
Google Play Store listing URL (by application ID)	https://play.google.com/store/apps/details?id=lk.aac.sinhala_tamil_english	Reachable through the Play Console internal-testing flow
Apple App Store TestFlight (via Ideahub)	Not available in current submission	Planned future pathway only. No Apple developer account or App Store release is claimed.
GitHub repository	Private repository retained by the author	Private
Google Drive project folder	Private project folder retained by the author	Private
Colab notebook (model training)	Private Colab notebook retained by the author	Private
Google Play Console production access request	See Figure 6.27 and Figure G.2	Submitted / under Google review. Production status inactive
Smart AAC App User Feedback Form	https://docs.google.com/forms/d/e/1FAIpQLSefThBhhQx15VgER3vzelNmLjO5ilkppCfFMuPfTy6UaIKUPA/viewform	Anonymous academic feedback form for usability, language support, AAC features,

Item	Link	Status
		support audio and camera observation
Google Form response summary	See Figure H.2	Anonymous response summary evidence is included in Appendix H. It is not treated as clinical validation.

Appendix H, Real-device testing evidence

- Primary test device: real-device testing was carried out on the Android device shown in Figure 7.3. The exact model and Android version are not separately stated in this appendix.
- Test photos: real-device testing evidence is included in Figure 7.3. Any photographs used as supporting evidence must avoid identifiable children.
- Test artefacts: the scripted walkthrough in Chapter 7 was executed manually on the primary test device. Supporting evidence is summarised in Chapter 7, Section 7.3 and Section 7.5.
- **Important constraint:** photos must not contain identifiable children. Any photographs of real users will be redacted before inclusion.

H.2 Google Form user feedback evidence

Form title: Smart AAC App User Feedback Form.

Form link:
https://docs.google.com/forms/d/e/1FAIpQLSefThBhhQx15VgER3vzelNmLjO5ilkppCfFMuPfTy6UaIKUPA/viewform

Purpose: to collect short anonymous feedback from invited testers about the Smart AAC mobile application, including usability, language support, AAC card usefulness, support audio, camera-based emotion observation and suggestions for improvement before wider release.

Questions covered: tester role; device used; language tested (Sinhala, Tamil, English, or more than one language); ease of opening and using the app; clarity of the home screen; usefulness of categories and communication cards; understandability of Sinhala/Tamil/English labels; app issues such as crash, slow loading, language issue, sound issue or camera issue; whether the app can help a child express basic needs or feelings; useful features such as visual cards, TTS, favourites, support audio, camera emotion support and offline use; whether the camera/emotion

support feature was tested; whether the predicted emotion felt reasonable; whether the supportive-not-diagnostic boundary was clear; overall rating; suggestions for improvement before wider release; and anonymous consent to use feedback in the academic report.

Ethical boundary: the form was not a clinical evaluation, did not measure formal model accuracy, did not collect clinical diagnosis data, and did not treat AI output as clinical evidence. Feedback is used anonymously and treated as informal early usability feedback only.

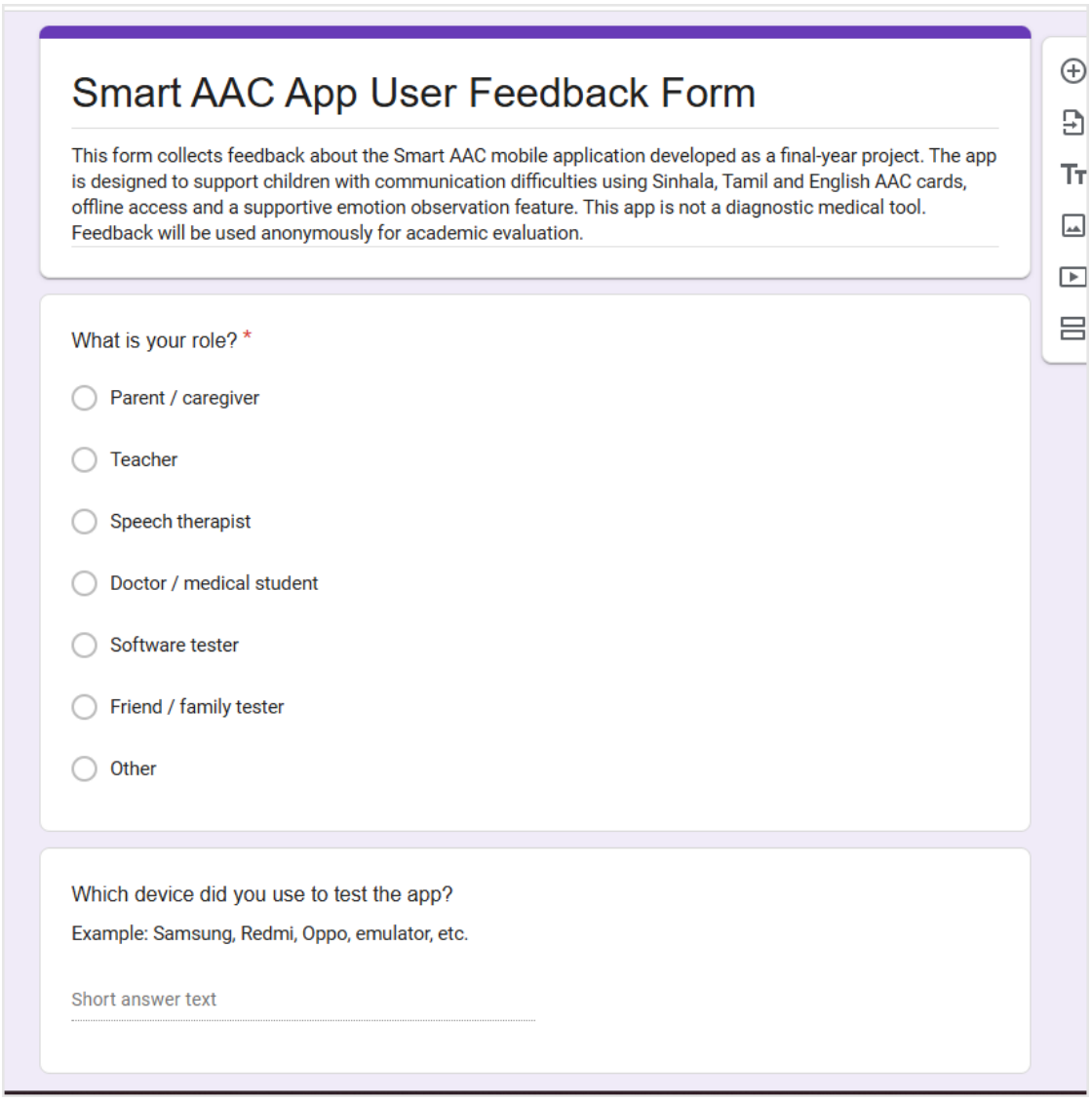


Figure H.1: Google Form feedback collection.

The form was used to collect anonymous tester feedback for the Smart AAC application.

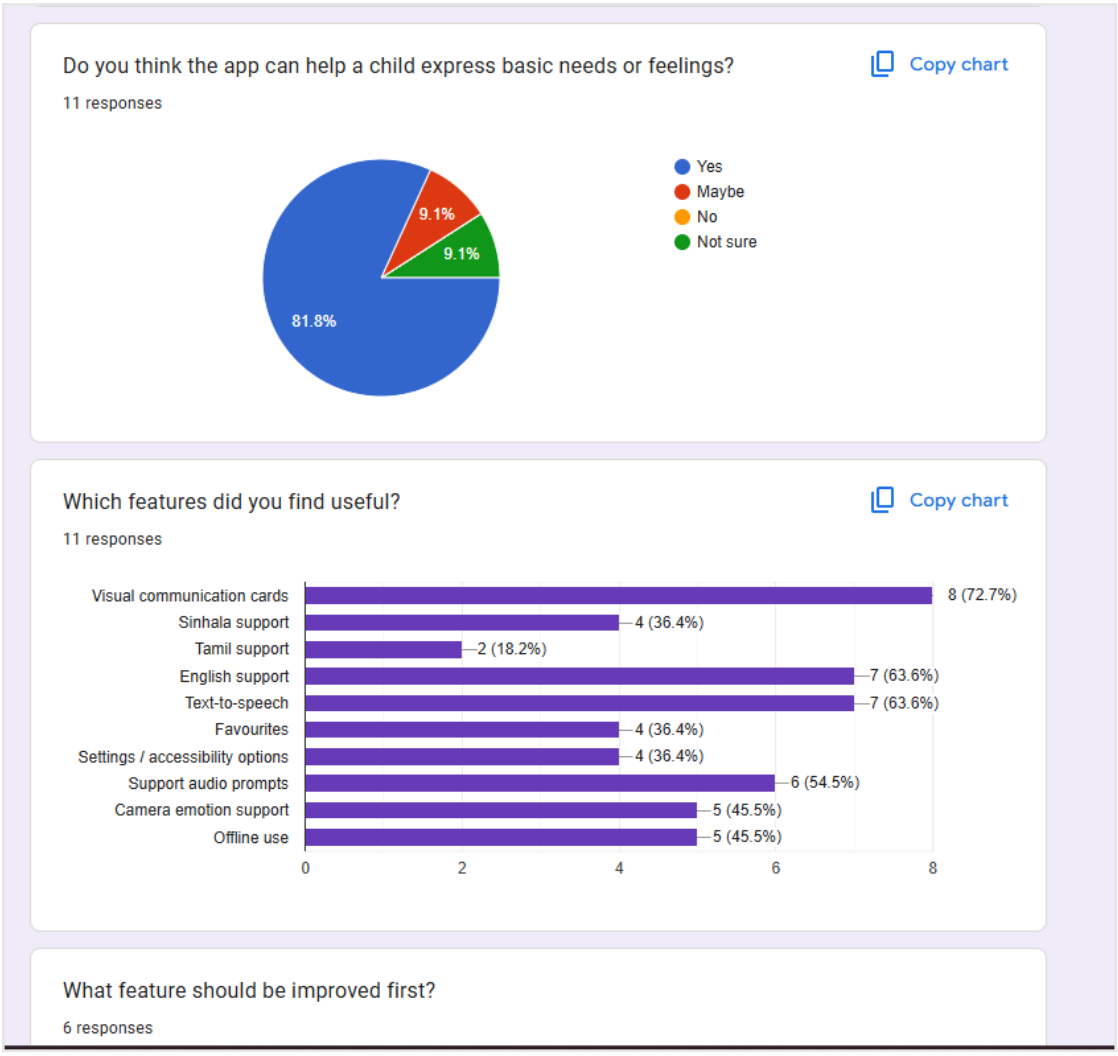


Figure H.2: Anonymous tester feedback summary.

The figure summarises anonymous tester feedback collected through the Google Form.

Appendix I, Ethical and data protection notes

This appendix records the ethical and data-protection boundary used while building the application, and the **stricter** boundary that would apply to any **future** local child face-data project.

Current build (what this thesis actually did):

- No identifiable child data was collected during development, and the deployed application does not store or transmit raw camera frames.
- Caregiver supervision is assumed for any camera-based interaction. The application provides explicit *No face detected*, *Face not clear*, *Uncertain* and *Reliable* states rather than forcing a label on every frame. The AI component is **not** a diagnostic tool and does **not** diagnose autism or clinical emotion categories for medical purposes.
- Optional supportive audio prompts are pre-recorded by the author and a Tamil-speaking collaborator. They contain no clinical claims.
- The Android manifest declares only three permissions: CAMERA, INTERNET and ACCESS_NETWORK_STATE.
- No Firebase, cloud database or analytics service is enabled in the current build.

Future ethics pathway (pilot and deployment):

- Pragathi Centre / National Hospital Galle ethical clearance is required before any pilot involving children.
- **Ministry or health-sector approval would be required before wider data collection or clinical deployment** at scale. This appendix does **not** claim that such approval is already in place.

Future dataset pathway (hypothetical, not started):

- A Sri Lankan child facial-expression dataset could only be explored as a separate governed study, not as an informal extension of coursework. Face images from minors are high-risk personal / biometric data.
- Any protocol would need **formal ethical clearance**, **parental or guardian consent**, consideration of **child assent** where appropriate, **secure storage**, **role-based access**, defined **retention and deletion** (or **pseudonymisation** where compatible with training goals), and alignment with **Sri Lankan data-protection law**, including the Personal Data Protection Act, No. 9 of 2022, and institutional ethics rules.
- Collection should be coordinated through **formal collaboration** with the **relevant hospital, ethics committee and health-sector authorities**. Raw images must not be pooled casually on personal drives or public clouds.

- Model training on any future local data would still need to respect the same objective framing as this thesis: the system remains a supportive AAC and observation tool, not a clinical decision engine.

Appendix J, Source and reproducibility evidence

- **Application source repositories:**
- **• Main Smart AAC application with emotion detection: aac_sinhala_tamil_english**
- **• Companion customisable AAC application: simple_aac_app**
- **The main repository contains the structured trilingual Smart AAC application with the on-device facial expression recognition workflow. The companion repository contains the simpler caregiver-customisable AAC application used for the Platform A / Level 1-2 customisation path.**
- **Branch note: thesis writing branch: final-thesis; development branch: upgrade-app.**
- Release APK location (local): build/app/outputs/flutter-apk/app-release.apk (approximately 126.2 MB based on the current build folder).
- **Release signing:** keystore wired through android/app/build.gradle.kts and android/key.properties (not committed).
- Model training source: Final Colab notebook.
- Supporting project records are retained in the private project repository and supporting documents folder. These include source code, model-notebook evidence, release APK metadata and implementation notes used by the author for traceability.

Representative unit-test code evidence is presented in Chapter 7.2, where it is discussed alongside the final Flutter test result and unit-test summary table. Appendix J is retained for source and reproducibility traceability rather than duplicating the same code excerpts.

Appendix K, User Manual

This appendix provides the user manual for the Smart AAC System with Facial Expression Recognition for Autism. The manual has been prepared to guide users through the main functions of the application, including launching the system, using the communication interface, selecting AAC cards, and understanding the facial expression recognition features. It also explains the basic steps needed to operate the application safely and effectively. The complete user manual can be accessed through the following link: https://drive.google.com/drive/folders/179F_UpW9nuJ8eYSDecl7wVowU6GE10hN?usp=sharing.

Appendix L, Supervisor Log Sheet

This appendix includes the supervisor log sheet for the Smart AAC System with Facial Expression Recognition for Autism project. The log sheet records the main supervision discussions, feedback received, project direction changes, and progress updates completed during the development period. It is attached as a separate PDF document for reference.

Attachment: Supervisor Log Sheet PDF