

**MOBILE APPLICATION FOR DETECTING AUTISM SPECTRUM
DISORDER (ASD)**

**BY
LIM CHIA YOONG**

**A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)**

FEBRUARY 2025

COPYRIGHT STATEMENT

© 2025 Lim Chia Yoong. All rights reserved.

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor, Dr Muhammad Syaiful Amri Bin Suhaimi, for providing valuable guidance throughout this mobile application development project, which integrates deep learning models. As this is my first time working on such a project, his insightful advice has been instrumental in helping me identify areas for improvement and gain a deeper understanding of the development process.

Next, I want to express my gratitude to my parents and family for their unwavering encouragement and support during the project. Their motivation and belief in me have been a constant source of strength, helping me stay focused and determined to achieve my goals.

ABSTRACT

Autism Spectrum Disorder (ASD) is a neurodevelopmental disability that affects how humans interact, communicate, and behave. Autistic people often find it hard to socialise with others and may engage in self-injurious behaviours. Specifically, there is no cure for this disorder. Additionally, it is expensive to detect ASD as it requires long-term monitoring by experts. Some diagnostic methods even involve brain scanning. Thus, it is unaffordable for most families, especially those with limited financial resources, even if their children suffer from this disorder.

Early intervention is important, making early detection of ASD particularly significant. To address this challenge, face recognition technology powered by deep learning has emerged as a promising diagnostic tool. This project aims to implement transfer learning using facial recognition to detect ASD. While researchers have proven that questionnaires can be effective screening tools with good detection accuracy, this project seeks to further validate the accuracy of the detection process. The project provides two separate methods, which are facial detection and Q-Chat 10 approaches to streamline the ASD detection process.

Area of Study: Computer Vision, Mobile Application Development

Keywords: Transfer Learning, Autism Detection, Facial Recognition, Q-Chat 10, Mobile Application

TABLE OF CONTENTS

TITLE PAGE	I
COPYRIGHT STATEMENT	II
ACKNOWLEDGEMENTS	III
ABSTRACT	IV
TABLE OF CONTENTS	V
LIST OF FIGURES	VIII
LIST OF TABLES	XI
LIST OF ABBREVIATIONS	XIII
CHAPTER 1 INTRODUCTION	1
1.1 Background Information	1
1.2 Problem Statement and Motivation	2
1.3 Project Objectives	3
1.4 Project Scope and Direction.....	4
1.5 Contributions.....	5
1.6 Report Organisation	5
CHAPTER 2 LITERATURE REVIEW	7
2.1 Previous Works on ASD Detection through Facial Analysis	7
2.1.1 Detection of Autistic Individuals using Facial Images and Deep Learning	7
2.1.2 Empirical Study of Autism Spectrum Disorder Diagnosis Using Facial Images by Improved Transfer Learning Approach.....	10
2.1.3 Utilizing Deep Learning Models in an Intelligent Facial Expression Classification System for Autism Disorder Diagnosis	12
2.1.4 Efficient Net-Based Transfer Learning Technique for Facial Autism Detection	14
2.1.5 Strengths of Leveraging Pre-Trained Models in ASD Facial Recognition	15

2.1.6 Limitations of Leveraging Pre-Trained Models in ASD Facial Recognition	15
2.2 Previous Works on Checklist Approach	16
2.2.1 CNN Based ASD Early Screening for Chinese Children	16
2.2.2 Identification of Autism Spectrum Disorder using Deep Neural Network.....	17
2.2.3 Strengths of Implementing Checklist-Based Approach.....	18
2.2.4 Limitations of Implementing Checklist-Based Approach	18
2.3 Previous Works on ASD Mobile Application	19
2.3.1 ASDetect Mobile Application.....	19
2.3.2 ATEC (Autism Treatment Evaluation Checklist) Application.....	20
2.3.3 Strengths of ASD Detection Mobile Applications.....	21
2.3.4 Limitations of ASD Detection Mobile Applications	22
2.4 Summary	22
CHAPTER 3 PROPOSED METHOD/APPROACH	24
3.1 Overall Workflow for Whole Project	24
3.2 RAD Methodology.....	25
3.3 Model Design and Architecture	27
3.3.1 Facial Recognition Model.....	27
3.3.2 Checklist-Based Model	33
3.4 Mobile Application Design.....	36
3.4.1 System Architecture Diagram.....	36
3.4.2 Use Case Diagram for Mobile Application	37
3.4.3 Activity Diagram for Mobile Application	38
3.5 Timeline	41
CHAPTER 4 SYSTEM DESIGN	43
4.1 Flowchart	43
4.2 Database Design (Cloud Firestore).....	44
4.3 System Block Diagram	48
4.4 System Component Specifications	48
CHAPTER 5 SYSTEM IMPLEMENTATION	50
5.1 Hardware Setup.....	50
5.2 Software Setup.....	51

5.3 Setting and Configuration	54
5.4 System Operation.....	55
5.5 Implementation Issues and Challenges	72
5.6 Concluding Remark	72
CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	73
6.1 System Performance Metrics	73
6.1.1 Facial Recognition Model Performance Metric.....	73
6.1.2 Checklist-Based Model Performance Metric.....	73
6.1.3 Mobile Application Performance Metric	73
6.2 Testing Setup and Result	74
6.2.1 Facial Recognition Model Performance Result	74
6.2.2 Checklist-Based Model Performance Result	83
6.2.3 Mobile Application Setup and Performance Result.....	83
6.3 Project Challenges	91
6.4 Objectives Evaluation	92
6.5 Concluding Remark	93
CHAPTER 7 CONCLUSION AND RECOMMENDATION	94
7.1 Conclusion	94
7.2 Recommendations.....	95
REFERENCES	96
APPENDIX	99
POSTER	100

LIST OF FIGURES

Figure Number	Title	Page
Figure 2. 1. 1. 1	Pipeline of the proposed method [11]	8
Figure 2. 1. 1. 2	Block Diagram of the MobileNet Model Architecture [11]	9
Figure 2. 1. 2. 1	Pipeline of the proposed method [12]	10
Figure 2. 1. 2. 2	Architecture Diagram of modified Xception Model [12].....	12
Figure 2. 1. 3. 1	Pipeline of the proposed method [13]	13
Figure 2. 1. 3. 2	Architecture Diagram of modified VGG16 Model [13].....	14
Figure 2. 2. 1. 1	Proposed Model Architecture for Checklist Approach [15].....	17
Figure 2. 3. 1. 1	Main Page Showing Added Children's Details [17]	19
Figure 2. 3. 1. 2	Main Page Showing Additional Information [17].....	19
Figure 2. 3. 2. 1	Result Page Displaying Multiple Functions [18]	21
Figure 3. 1. 1	Overall Workflow for the Project.....	24
Figure 3. 2. 1	RAD Development Lifecycle [19]	25
Figure 3. 3. 1. 1. 1	Workflow for Training Facial Recognition Model	27
Figure 3. 3. 1. 2. 1	Customised layers added to the facial recognition model architecture.....	32
Figure 3. 3. 2. 1. 1	Workflow for Training Checklist-Based Model.....	33
Figure 3. 3. 2. 2. 1	Customised layers added to the checklist-based model architecture	35
Figure 3. 4. 1. 1	System Architecture Diagram	36
Figure 3. 4. 2. 1	Use Case Diagram	37
Figure 3. 4. 3. 1	Activity Diagram for Registering, Logging and Viewing Past Summarised Test Information.....	38
Figure 3. 4. 3. 2	Activity Diagram for Viewing Profile.....	40
Figure 3. 4. 3. 3	Activity Diagram for Performing Questionnaire or Facial Detection	41
Figure 3. 5. 1	Timeline for Project I	41
Figure 3. 5. 2	Timeline for Project II	42
Figure 4. 1. 1	Overall flowchart for mobile application	43

Figure 4. 2. 1 Images collection's database design.....	44
Figure 4. 2. 2 Questionnaire collection's database design	45
Figure 4. 2. 3 Responses collection's database design	46
Figure 4. 2. 4 User collection's database design	47
Figure 4. 3. 1 Block diagram for mobile application	48
Figure 5. 4. 1 Splash Screen.....	55
Figure 5. 4. 2 Register and Login Page.....	56
Figure 5. 4. 3 Sign Up Page	56
Figure 5. 4. 4 Dropdown Selection	57
Figure 5. 4. 5 Data Picker Selection	57
Figure 5. 4. 6 Password in plaintext (Sign Up Page)	57
Figure 5. 4. 7 Password in obscured text (Sign Up Page).....	57
Figure 5. 4. 8 Verifying email address page	58
Figure 5. 4. 9 Gmail inbox with verification link	58
Figure 5. 4. 10 Confirmation message	58
Figure 5. 4. 11 Conformation Page.....	59
Figure 5. 4. 12 Page for entering Gmail to reset password.....	60
Figure 5. 4. 13 Page showing password reset Gmail sent.....	60
Figure 5. 4. 14 Password reset link in Gmail.....	60
Figure 5. 4. 15 Reset new password	60
Figure 5. 4. 16 Message showing password changed	60
Figure 5. 4. 17 Password in plaintext (Sign in Page).....	61
Figure 5. 4. 18 Password in obscured text (Sign in Page)	61
Figure 5. 4. 19 Facial Recognition Page	62
Figure 5. 4. 20 Phone's Camera Interface.....	62
Figure 5. 4. 21 Confirming captured image by tapping “✓”	62
Figure 5. 4. 22 Result with predicted label and confidence score (Take Photo)	62
Figure 5. 4. 23 Device's Gallery Interface.....	63
Figure 5. 4. 24 Result with predicted label and confidence score (Upload from gallery)	63
Figure 5. 4. 25 Facial recognition's notification pop-up	64
Figure 5. 4. 26 Facial recognition's notification's expanded view.....	64
Figure 5. 4. 27 Questionnaire Page	65

Figure 5. 4. 28 Intermediate questions include “Previous” and “Next” buttons.....	65
Figure 5. 4. 29 Last questions include “Previous” and “Submit” buttons	65
Figure 5. 4. 30 Result with predicted label and confidence score (Questionnaire)	65
Figure 5. 4. 31 Questionnaire’s notification pop-up	66
Figure 5. 4. 32 Questionnaire’s notification’s expanded view	66
Figure 5. 4. 33 Home page with no previous records	67
Figure 5. 4. 34 Home page with previous records	67
Figure 5. 4. 35 Questionnaire Response Page.....	68
Figure 5. 4. 36 Questionnaire Response Page with Go Back button	68
Figure 5. 4. 37 Facial Recognition Response Page with Go Back button	68
Figure 5. 4. 38 Profile Page	69
Figure 5. 4. 39 Editing Name Page	69
Figure 5. 4. 40 Editing Phone Number Page	69
Figure 5. 4. 41 Logout Button in Profile Page	70
Figure 5. 4. 42 Delete account’s confirmation prompt	71
Figure 5. 4. 43 Password in plaintext (Re-authenticate Page)	71
Figure 5. 4. 44 Password in obscured text (Re-authenticate Page).....	71
Figure 6. 2. 1. 3. 1 Validation accuracy of not applying data augmentation	77
Figure 6. 2. 1. 3. 2 Validation accuracy of applying one data augmentation	77
Figure 6. 2. 1. 3. 3 Validation accuracy of applying twice data augmentation.....	77

LIST OF TABLES

Table Number	Title	Page
Table 2. 2. 1. 1	Q-Chat 10 questions that used in the paper [15].....	16
Table 2. 4. 1	Comparison of ASD Facial Recognition Accuracy Across Different Studies.....	22
Table 2. 4. 2	Comparison of Checklist-based models' accuracy	23
Table 2. 4. 3	Comparison of ASD Detection Mobile Applications	23
Table 3. 3. 1. 1. 1	Total Images before and after removing duplicates.....	28
Table 3. 3. 1. 1. 2	Breakdown of Autistic and Non-Autistic Images in Each Set.....	29
Table 3. 3. 1. 1. 3	Hyperparameters setting used to train facial recognition model	29
Table 3. 3. 2. 1. 1	Details of each attribute type in checklist.....	34
Table 3. 3. 2. 1. 2	Hyperparameters setting used to train checklist-based model.....	35
Table 5. 1. 1	Specifications of Laptop	50
Table 5. 1. 2	Specifications of Mobile Device.....	50
Table 5. 2. 1	Specifications of Main Software Tools	51
Table 5. 2. 2	Specifications of Software Languages.....	52
Table 5. 2. 3	Specifications of Libraries	52
Table 5. 2. 4	Specifications of Framework	54
Table 5. 2. 5	Specifications of Emulator.....	54
Table 6. 2. 1. 1. 1	Validation Accuracy Comparison of Model Architectures	74
Table 6. 2. 1. 2. 1	Validation Accuracy Comparison of Applying Cropping	75
Table 6. 2. 1. 2. 2	Validation Accuracy Comparison of Applying Gaussian Blur	75
Table 6. 2. 1. 2. 3	Validation Accuracy Comparison of Applying CLAHE.....	76
Table 6. 2. 1. 3. 1	Validation Accuracy Comparison of Applying Data Augmentation	76
Table 6. 2. 1. 4. 1	Validation Accuracy Comparison of Batch Size	78
Table 6. 2. 1. 4. 2	Validation Accuracy Comparison of Optimizer and Learning Rate	78
Table 6. 2. 1. 4. 3	Validation Accuracy Comparison of Applying “nesterov”	79
Table 6. 2. 1. 5. 1	Validation Accuracy Comparison of Applying Dropout.....	80

Table 6. 2. 1. 5. 2 Validation Accuracy Comparison of Applying Batch Normalization	80
Table 6. 2. 1. 6. 1 Validation Accuracy Comparison of Applying Fine Tuning.....	81
Table 6. 2. 1. 7. 1 Accuracy Comparison of Test Set	82
Table 6. 2. 1. 7. 2 Accuracy Comparison of External Images	82
Table 6. 2. 3. 1 App Lauch Test Case	83
Table 6. 2. 3. 2 User Sign In Test Case.....	84
Table 6. 2. 3. 3 User Sign Up Test Case	84
Table 6. 2. 3. 4 User Forget Password Test Case	85
Table 6. 2. 3. 5 Displaying Past Detection Details Test Case.....	86
Table 6. 2. 3. 6 Take a Photo Test Case.....	86
Table 6. 2. 3. 7 Pick from Gallery Test Case	87
Table 6. 2. 3. 8 Checklist-based Detection Test case.....	88
Table 6. 2. 3. 9 Update Profile Name Test Case.....	88
Table 6. 2. 3. 10 Update Phone Number Test Case	89
Table 6. 2. 3. 11 Logout Test Case	90
Table 6. 2. 3. 12 Delete Account Test Case.....	90
Table 6. 4. 1 Objectives Evaluation	92

LIST OF ABBREVIATIONS

<i>Adam</i>	Adaptive Moment Estimation
<i>AI</i>	Artificial Intelligence
<i>ASD</i>	Autism Spectrum Disorder
<i>CNN</i>	Convolutional Neural Network
<i>CNNS</i>	Convolutional Neural Networks
<i>FAQs</i>	Frequently Asked Questions Section
<i>MD5</i>	Message-Digest algorithm 5
<i>pHash</i>	Perceptual Hashing
<i>RAD</i>	Rapid Application Development
<i>t-SNE</i>	t-Distributed Stochastic Neighbour Embedding
<i>TFLite</i>	TensorFlow Lite

Chapter 1 Introduction

This chapter provides the background information of the project, together with the problem statements, motivations, objectives, project scopes, key contributions, and the organisation of the report.

1.1 Background Information

Autism Spectrum Disorder (ASD) is a neurodevelopmental disability that caused by genetic, biological, and environmental factors. It impairs brain functioning and influences how a person interacts, communicates, and behaves. Autistic people with significant symptoms often face challenges in daily living and require support from others. Common symptoms include avoiding eye contact, exhibiting repetitive behaviours, and experiencing difficulty communicating with others [1]. As a result, families often encounter the added strain of balancing work while providing the necessary support to autistic individuals.

Centers for Disease Control and Prevention (CDC) found that among 8-year-old children, 1 out of 36 is an autistic child [2]. According to the research conducted in 2021, the number of children with autism in Malaysia increased by 5% compared to previous year [3]. This rise underscores the growing prevalence of ASD and the need for early intervention strategies. Despite advances in understanding ASD, comprehensive guidelines for managing and mitigating its symptoms remain limited. Early detection is crucial to reduce symptom severity, enabling children with ASD to access expert support and fostering positive mindset earlier.

As artificial intelligence (AI) keeps evolving, it has brought advancements in deep learning. The progression in facial recognition driven by deep learning has enabled several models, especially Convolutional Neural Networks (CNNs), to achieve high accuracy in recognising faces. For instance, CNNs have achieved 91.0% accuracy in facial recognition for disease detection [4]. Similarly, a 92.31% accuracy has been achieved in detecting ASD using CNNs [5]. The high accuracy of CNN models has inspired this project to employ transfer learning with CNN models to detect ASD.

On the other hand, the use of Quantitative Checklist for Autism in Toddlers (Q-Chat) has demonstrated high accuracy, with a 95% detection rate using support vector machine (SVM) [6]. This highlights the potential of checklist-based methods as an effective screening tool for ASD.

With 90% of the world's population now owning smartphones, and this number continuously increasing [7], mobile technology presents an ideal platform for ASD detection. The project aims to provide facial recognition and checklist methods within a mobile application, offering a convenient tool to facilitate the ASD detection process.

1.2 Problem Statement and Motivation

Although ASD detection has been studied for over 80 years, it remains **an inefficient and resource-intensive process**. The detection process relies heavily on specialised expertise, requiring professionals to conduct in-depth developmental screenings and assessments. One of the most common diagnostic methods, which is development screening, requires regular interaction with non-schooling children and close collaboration with their parents to gather comprehensive information. This process often involves numerous assessments, which can take months or even years to obtain results [8].

Parents of children with potential ASD often face significant challenges in **accessing resources for autism detection**. There is a notable lack of publicly available tools designed to assist families in performing initial ASD screenings at home. Most existing resources are restricted to researchers or healthcare professionals, leaving common families with limited access to these valuable tools. This disparity disproportionately affects families with fewer financial resources, as they may struggle to afford the costly diagnostic procedures conducted by specialised professionals.

This issue arises when **autistic individuals present their behaviours differently** during assessments [9]. They may exhibit varying behaviours depending on the setting or the individuals they are interacting with, often without any intention. Some individuals may consciously hide their symptoms, which can lead to an

incomplete representation of their condition. This variability complicates the diagnostic process, as it heavily relies on observed behaviours during screenings, which can differ across assessments.

Despite advancements in ASD detection, the process remains resource-intensive, requiring multiple assessments and consultations. Additionally, available resources are often inaccessible to the general public, leaving many families without affordable options. Variability in how autistic individuals express behaviours further complicates the diagnostic process, making accurate and timely detection difficult. The primary motivation behind this project is to address these inefficiencies and challenges in the current ASD detection process, aiming to provide a more accessible and efficient solution for families.

1.3 Project Objectives

The aim of the project is to develop a user-friendly mobile application to effectively detect autistic children, providing a reliable, accessible, and efficient alternative to traditional diagnostic methods. This application will assist families in carrying out early screenings and reduce the burden on specialised healthcare systems.

(i) To streamline ASD detection process by training a facial detection model using transfer learning

This project will focus on employing facial recognition technology to detect ASD, aiming to reduce the lengthy and resource-intensive nature of traditional diagnostic methods. By leveraging transfer learning, the project will explore various pre-trained models to identify the most effective approach for accurate and efficient facial detection. This method will eliminate the need for expert intervention and minimise reliance on time-consuming assessments. It can provide real-time detection results, thereby reducing the long waiting times associated with traditional diagnostic methods.

(ii) To make ASD detection more accessible through a mobile application

This project aims to reduce the disparity in resource availability by integrating the facial recognition based ASD detection method into a mobile application. By doing

so, it will enable families to perform initial ASD screenings at home, without the need for professional assistance or expensive diagnostic tools. The mobile application will provide an accessible, user-friendly platform that can be used by parents, regardless of their financial resources or geographic location. This approach will facilitate early ASD detection, making it easier for families to access reliable results and take the necessary steps towards further evaluation and support.

(iii) To reduce diagnostic variability in ASD detection through facial recognition

Another objective is to tackle the challenges posed by the variability in autistic behaviours during assessments. Autistic individuals may exhibit different behaviours across various settings or interactions, which can complicate the diagnostic process. However, many autistic individuals share distinctive facial features that remain relatively constant, such as wider eyes, a broader upper face, a shorter midface, and a larger mouth [10]. By leveraging facial recognition technology, these consistent facial features can be captured regardless of external factors. This approach helps to minimise the impact of behavioural variations, resulting in a more consistent and reliable method for detecting ASD.

1.4 Project Scope and Direction

This project is divided into two main sub-tasks, which are training deep learning models and developing the mobile application. The first task involves training deep learning models for ASD detection using facial images and checklist-based data. These models will be trained using appropriate preprocessing techniques and deep learning to achieve high accuracy. The facial recognition model will be trained on facial image data, while the checklist model will be trained using the Q-Chat dataset combined with basic user information. Once trained, the best-performing models will be converted into formats compatible with mobile deployment.

The second task involves the development of a mobile application, which comprises several modules including facial recognition, checklist, home page, register and login, and profile modules. The primary modules, which are facial recognition and checklist, are responsible for assisting in ASD detection and providing inference results based on the input fed to the respective models. The remaining modules

CHAPTER 1

coordinate and support the overall functionality and usability of the application, ensuring a smooth user experience.

1.5 Contributions

The primary contribution of this project is its enhanced approach to ASD detection through a facial recognition mobile application, which goes beyond existing studies that primarily rely on checklist methods. By integrating facial recognition technology, this project offers a more efficient method for autism detection, aiming to reduce the time and complexity associated with traditional diagnostic processes. To further enhance the reliability of the detection, a simplified version of the checklist is also incorporated, offering two separate methods for validating results.

Additionally, the user-friendly design of the ASD detection mobile application ensures accessibility for families with children who may have autism. The app provides a convenient, real-time tool for users to perform the detection process without the involvement of professional expertise. This makes the application more accessible for a wider audience, supporting early autism detection and facilitating better outcomes for families.

1.6 Report Organisation

The details of this project are outlined in the following chapters. Chapter 1 introduces ASD and the detection process, covering both facial recognition and checklist-based approaches. It also includes the problem statement, motivation, objectives, project scope, and key contributions.

Chapter 2 provides a review of relevant background information, including the strengths and limitations of facial recognition for ASD detection, the checklist method, and existing ASD detection applications.

Chapter 3 discusses the overall workflow and methodology. It covers model training steps, the architectural diagram for the facial detection model, as well as the system architecture of the mobile application, including the use case and activity diagrams.

CHAPTER 1

Chapter 4 presents the system design, featuring the flowchart of the mobile application, database design, and detailed descriptions of each module.

Chapter 5 outlines the system implementation and operation. It describes the hardware and software used and demonstrates how the mobile application looks and functions in practice.

Chapter 6 presents the results and discussion for both the facial recognition and checklist-based models. It also includes test cases for the mobile application and a final evaluation of how well the project objectives have been achieved.

Finally, Chapter 7 concludes the report by summarising the overall project and providing recommendations for future work.

Chapter 2 Literature Review

In this chapter, the current techniques used to assist in ASD detection will be reviewed. Autism can be identified using various methods, such as facial recognition, checklists, physical interaction, brain imaging, and more. However, this chapter will focus on two feasible approaches, which are facial recognition and the checklist method, as they are more convenient and practical.

To address the challenges posed by limited expertise in customising models and to enable seamless integration into mobile applications, transfer learning is employed for ASD facial recognition. Additionally, to further enhance accuracy, the Q-Chat 10 checklist, consisting of 10 questions, was trained using a deep learning approach. While many methods for training the Q-Chat 10 checklist have traditionally relied on machine learning, deep learning was chosen in this study to facilitate seamless integration within mobile applications. This chapter will also examine existing mobile applications for ASD detection. Each of the three sections will provide a detailed analysis, including their strengths and weaknesses, to offer a comprehensive understanding of the topic.

2.1 Previous Works on ASD Detection through Facial Analysis

2.1.1 Detection of Autistic Individuals using Facial Images and Deep Learning

To classify autistic children, the paper proposes deep learning methods using transfer learning on pre-trained models, including MobileNet, InceptionV3, and InceptionResNetV2 [11]. Figure 2.1.1.1 shows the overview of the pipeline for detecting autistic children. It can be divided into few main steps, which are getting and pre-processing the data, using pre-trained models to extract features, and finally performing binary classification.

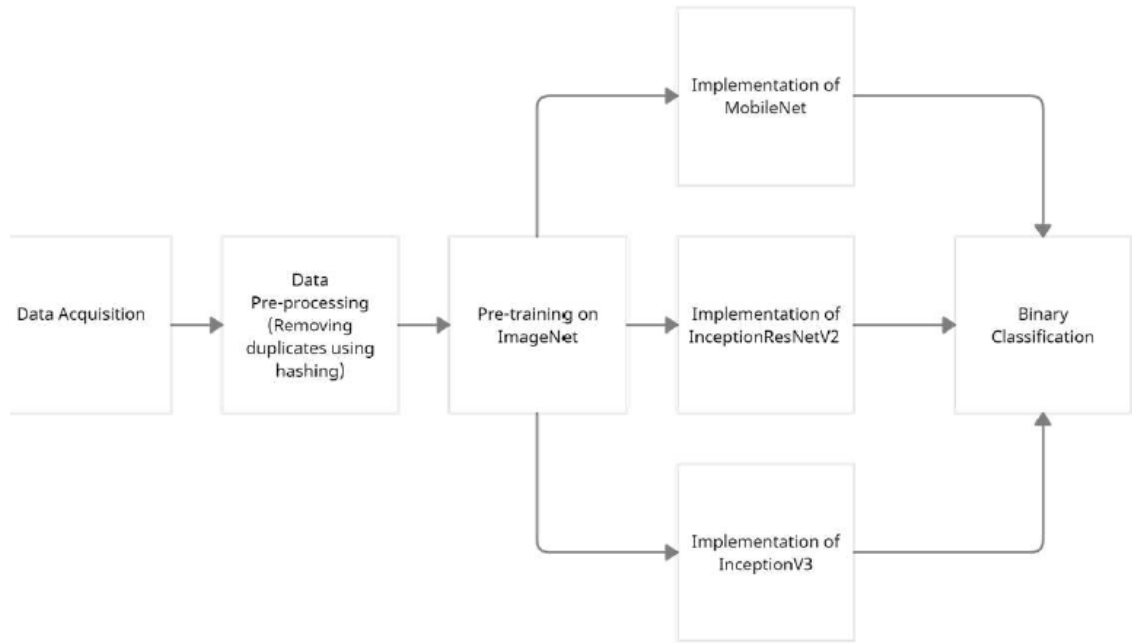


Figure 2. 1. 1 Pipeline of the proposed method [11]

In the initial step, this research paper acquired the dataset from Kaggle, which contained 1468 images in autistic and non-autistic categories respectively.

After that, the images were resized to 224x224 pixels. An interesting data cleaning step was used, which reduced the duplicate images using hashing. To remove exact duplicates, MD5 hash was used. It first produced a 128-bit hash value for each image. These hash values were then compared to check if any duplicates occurred. If the dataset contained matching hash values, it indicated the presence of identical images. Since no hash values matched, it could be concluded that the dataset did not contain identical images.

On the other hand, to handle near-exact duplicate images, pHash was used due to its locality-sensitivity. It grouped images that were almost identical, with only minor differences in colour, brightness, contrast, and aspect ratio. Similarly, it assigned a hash value to each image, and these values were listed to check for duplication. If any hash values matched, the corresponding images were considered near-exact duplicates. The images with the same hash value were then deleted, and only one instance of each was retained while the others were removed. In autistic images, there were 42 images with near-exact duplicates, while there were 94 for non-autistic images. After removing those images, the ratio for data splitting remained the

same as in the original dataset. The new dataset consists of 1,269 training images and 100 validation images for each category, respectively.

Then, MobileNet is applied because it consists of fewer parameters and is more efficient. The base model consisted of 28 layers. A zero-padding layer was applied to the images, which were resized to 224x224 pixels. It was used to maintain the feature maps, especially for those features within boundaries. Then, a convolutional layer was used for feature extraction, followed by a batch normalization layer to enhance performance. A ReLU layer was added to introduce non-linearity, ensuring that positive input retained its value while zero or negative values became zero. After these layers, depthwise convolutional layers were added to ensure each colour channel (RGB) was iterated by each kernel. This was followed by batch normalization and ReLU layers. Pointwise convolutional layers were then added to ensure every point was iterated, followed by additional batch normalization and ReLU layers. These layers were repeated 13 times to build up the MobileNet model.

The proposed network is illustrated in Figure 2.1.1.2. On top of the MobileNetV1 architecture, a dropout layer with a value of 0.5 was added to prevent overfitting. This was followed by a dense layer with a ReLU activation function. A final dense layer with a sigmoid activation function was used for binary classification. To train the proposed architecture, an Adam optimizer was used to minimise the loss with a learning rate of 0.0001. Binary cross-entropy was employed for binary classification.

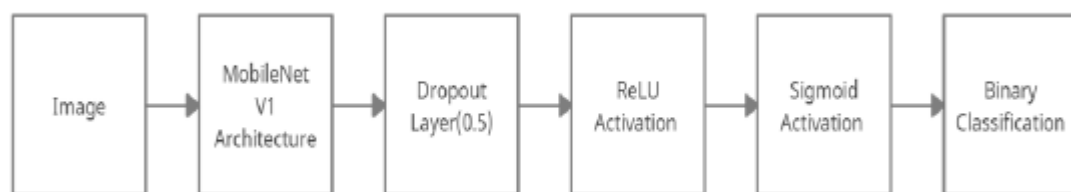


Figure 2. 1. 1. 2 Block Diagram of the MobileNet Model Architecture [11]

Each of the models ran for optimized values of epochs and patience. MobileNet used 42 epochs with a patience of 15, InceptionV3 used 100 epochs with a patience of 20, and InceptionResNetV2 used 23 epochs with a patience of 5. Among all of them, the MobileNetV1 architecture had the least runtime of 97 seconds due to

having fewer parameters. In terms of testing accuracy, this architecture achieved the highest accuracy with a value of 87%.

2.1.2 Empirical Study of Autism Spectrum Disorder Diagnosis Using Facial Images by Improved Transfer Learning Approach

This paper [12] focuses on finding a lightweight model for extracting facial features to detect ASD, experimenting with various hyperparameters. Transfer learning is employed, with the topmost layer adjusted to serve as a classification layer. The process begins with obtaining and preprocessing the facial dataset, followed by labelling and dividing it into training, validation, and test sets. To optimize accuracy, multiple training and testing cycles are conducted to identify the best hyperparameters. These optimal hyperparameters are then used to train the models for the ASD detection task. Finally, the model's performance is evaluated to determine the best-performing model. Figure 2.1.2.1 illustrates the overall workflow of the proposed method.

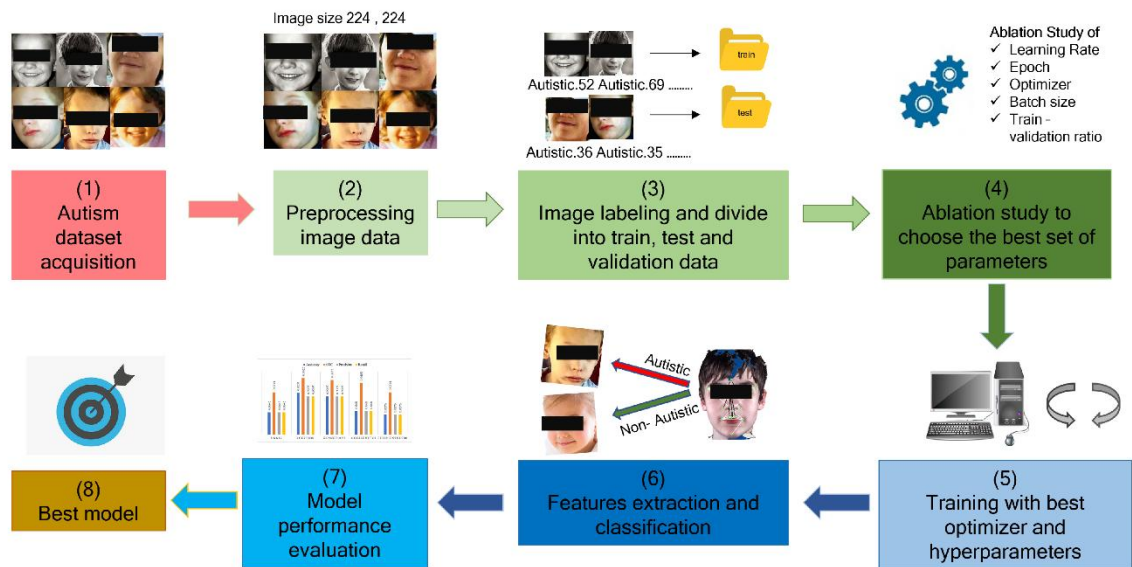


Figure 2. 1. 2. 1 Pipeline of the proposed method [12]

The author begins by obtaining the dataset from Kaggle, which contains 2,936 images. The dataset is split into training (2,540 images, 86.38%), validation (300 images, 10.22%), and test (100 images, 3.41%) sets, with an equal distribution of autistic and non-autistic classes. At the same time, the images are preprocessed by labelling and resizing them to 299x299x3 pixels to fit the model's input shape.

CHAPTER 2

Several models with different hyperparameters are tested to achieve the highest accuracy. Once the optimal hyperparameters are identified, they are fixed for further experimentation to determine the best-performing model. Notably, the author focuses on lightweight models suitable for mobile applications, including VGG19, Xception, ResNet50V2, MobileNetV2, and EfficientNetB0.

The Xception model architecture is reviewed as it achieves the highest accuracy in detecting autism. The base model consists of 42 layers, which can be divided into three parts, which are the entry, middle, and exit flows. In the entry flow, the model begins with a convolutional layer to accept images of size 299x299x3 pixels. This is followed by a convolutional layer with 32 filters and another with 64 filters. Subsequently, there are three residual blocks, each composed of two separable convolutional layers and one max-pooling layer. The number of filters in these blocks increases from 128 to 256 and then to 728 to capture more complex features.

In the middle flow, identical residual blocks are repeated eight times, each consisting of separable convolutional layers with 728 filters to extract increasingly complex features from the images.

The exit flow then contains separable convolutional layers with 728 filters, which are further increased to 1,024 filters, followed by a max-pooling layer. The number of filters is then increased to 1,536 and subsequently to 2,048, followed by a global average pooling layer, marking the end of the base architecture.

The modified topmost layers include a fully connected layer with 512 hidden units, a dropout layer (with a weight of 0.5) to prevent overfitting, and a dense layer for classifying ASD. Figure 2.1.2.2 illustrates the architecture of the modified Xception model.

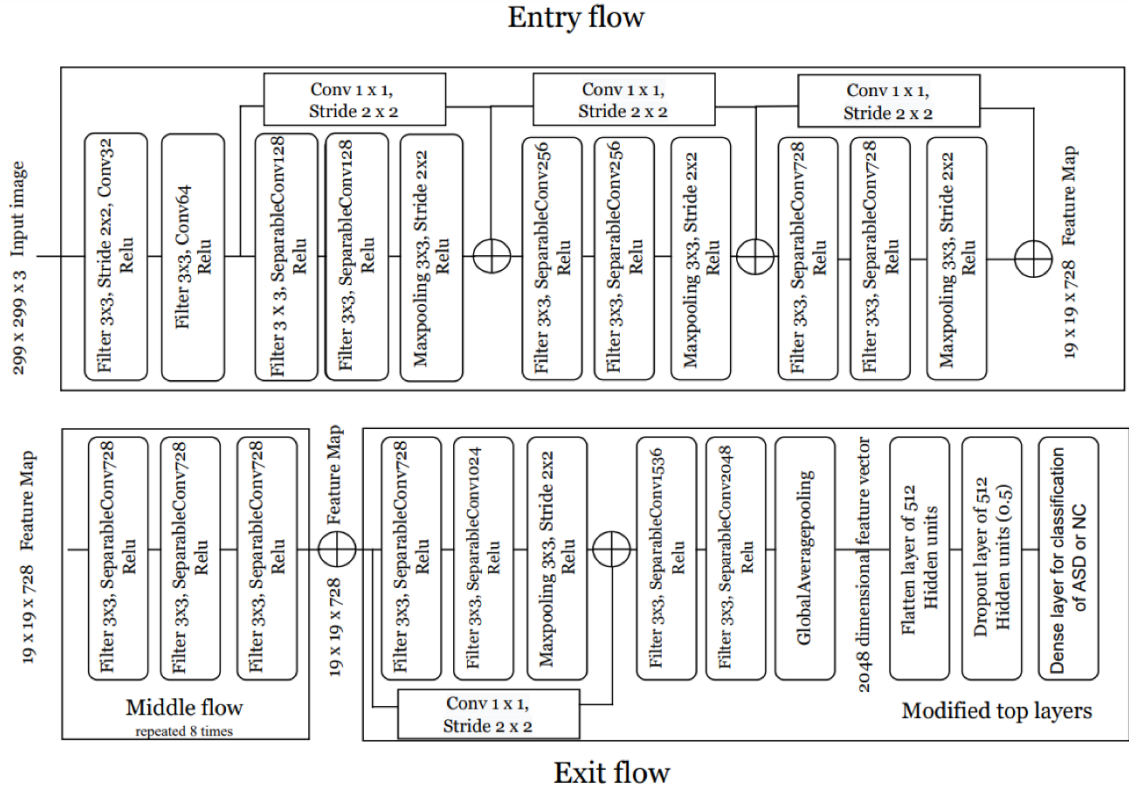


Figure 2. 1. 2. 2 Architecture Diagram of modified Xception Model [12]

After experimenting with different hyperparameters, the author determined that a learning rate of 0.001, the Adagrad optimizer, 50 epochs, and a batch size of 32 provided the best performance, achieving an accuracy of 92.01% with the Xception model.

2.1.3 Utilizing Deep Learning Models in an Intelligent Facial Expression Classification System for Autism Disorder Diagnosis

The paper [13] aims to apply transfer learning for facial recognition to detect autism using three models. Several preprocessing methods were employed, including normalization, augmentation, and label encoding. The preprocessed images were then fed into three pre-trained models, namely VGG16, InceptionV3, and EfficientNetB0, to evaluate their accuracy for classification. The author separately determined the optimal hyperparameters for each model and compared their best results. The overall workflow is illustrated in Figure 2.1.3.1.

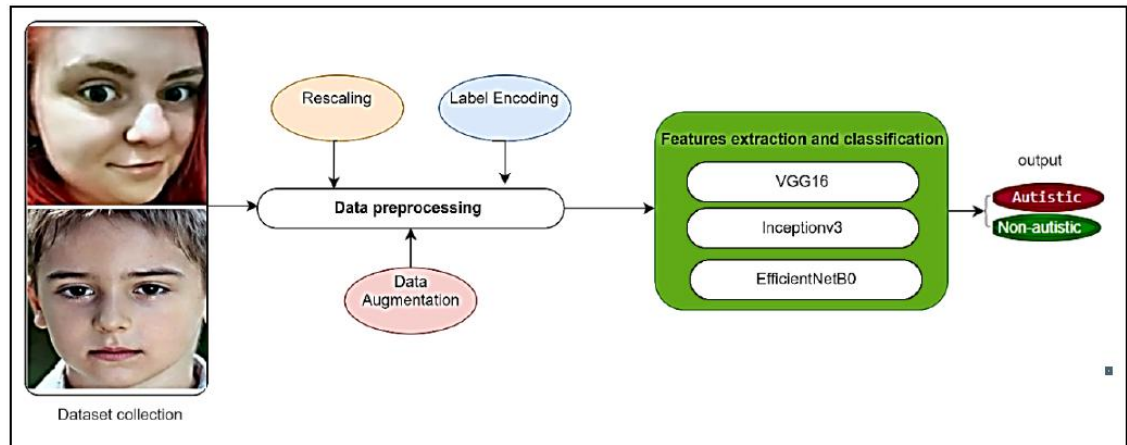


Figure 2. 1. 3. 1 Pipeline of the proposed method [13]

First, the author acquired the same dataset from Kaggle and divided it into consistent proportions: 86.38% for training, 10.22% for validation, and 3.41% for testing. The images were resized to 224x224x3 pixels and normalised to the range [0,1] to standardise the pixel values. Various augmentation techniques, such as rotation, flipping, and zooming, were applied to increase data diversity. The labels were encoded into two classes, namely autistic and non-autistic.

The preprocessed images were trained using the three pre-trained models. Among them, VGG16 demonstrated the best performance, as it efficiently captured intricate patterns. To elaborate, the base model consists of 19 layers structured into five blocks. The first two blocks contain two convolutional layers and a max-pooling layer, with the number of filters increasing from 64 to 128. The subsequent three blocks each include three convolutional layers and a max-pooling layer, with the number of filters increasing from 256 to 512, while the last block maintains 512 filters.

Custom layers were added to the architecture, including a flatten layer to convert the feature maps into a 1D feature vector. This was followed by three fully connected (FC) layers and a final dense layer with a softmax activation function for classification. The network architecture is illustrated in Figure 2.1.3.2. Furthermore, the author unfroze the last few convolutional layers to enable the model to learn additional features specific to the dataset.

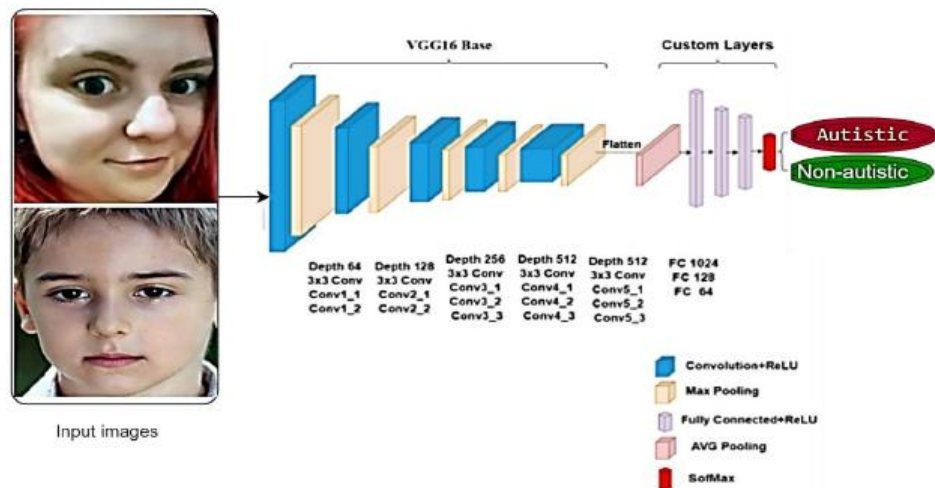


Figure 2. 1. 3. 2 Architecture Diagram of modified VGG16 Model [13]

The optimal hyperparameters were determined as a batch size of 128, 100 epochs, a learning rate of 0.001, and the Adam optimizer. The final model achieved an accuracy of 95%.

2.1.4 Efficient Net-Based Transfer Learning Technique for Facial Autism Detection

The paper [14] highlights the use of transfer learning with EfficientNetB0, a lightweight model suitable for mobile applications due to its lower number of parameters. The approach begins with acquiring and preprocessing the dataset, followed by adding custom layers for ASD classification and evaluation.

First, the dataset is divided into three partitions, which are 86.38% for training, 10.22% for validation, and 3.41% for testing. Duplicate images are removed, and the facial region is cropped to focus the model on relevant features. The images are resized to 224x224x3 pixels to match the model's input shape and normalised to a range of 0 to 1.

The preprocessed images are then fed into the EfficientNetB0 architecture, which consists of 20 layers in the base model. The model begins with a stem layer, which includes a 3x3 convolutional layer designed to downsample the input images. This is followed by a series of seven mobile inverted bottleneck convolution (MBConv) blocks, which use depthwise separable convolutions and squeeze-and-excitation layers to balance computational efficiency with robust feature extraction.

The blocks are structured as follows: a single MBConv1 layer with a 3x3 kernel, two MBConv6 layers with a 3x3 kernel, two MBConv6 layers with a 5x5 kernel, three MBConv6 layers with a 3x3 kernel, three MBConv6 layers with a 5x5 kernel, and four MBConv6 layers with a 5x5 kernel. The base architecture concludes with a projection layer, implemented as an MBConv6 layer with a 3x3 kernel, along with Conv2D, average pooling, and a dropout layer. For this study, the author uses the base model with a custom dense output layer for ASD classification.

In order to optimize the model, the author experimented with different hyperparameters, including learning rate, optimization function, and loss function. After testing various configurations, the author determined that a batch size of 24, 300 epochs, a learning rate of 0.001, and the Adam optimizer with a dropout value of 0.5 yielded the best results. Ultimately, the model achieved an accuracy of 85%.

2.1.5 Strengths of Leveraging Pre-Trained Models in ASD Facial Recognition

In previous studies [11], [12], and [14] lightweight models were utilised for ASD facial detection, aligning with our goal of developing a mobile application that requires minimal detection time. The authors in [11] employed MD5 hash and pHash techniques to account for exact and near-duplicate images, which helped enhance the robustness of the models and minimise the risk of overfitting. In [12], five lightweight models were tested, and generalised hyperparameters were identified to best suit them. The author in [13] augmented the images to improve model performance by adding more variation. Additionally, [14] not only removed duplicate images but also cropped the images to focus more on the face, helping the model capture more facial features.

2.1.6 Limitations of Leveraging Pre-Trained Models in ASD Facial Recognition

For ASD detection through facial recognition, previous studies focused on finding the highest accuracy model, while the actual implementation has not been considered. The next challenge is the limited dataset for ASD facial analysis. A smaller dataset may result in less accurate models. Next, most of the available facial datasets are well-prepared or even augmented with slight changes, which may not accurately reflect real-world scenarios. In real-world situations, images are often

taken from different angles and under varying conditions, which is not always captured in these datasets.

2.2 Previous Works on Checklist Approach

2.2.1 CNN Based ASD Early Screening for Chinese Children

This article focuses on using CNN and SVM classifiers separately for ASD detection in China to compare the performance of both methods [15]. The author utilised the Q-Chat 10 checklist along with personal information such as age, gender, jaundice history, and family history of ASD, resulting in a total of 14 questions. Each question is answered using a 5-point Likert scale: “always,” “usually,” “sometimes,” “rarely,” and “never.” Personal information, such as age, is collected as numeric data, while gender is recorded as male or female, and both jaundice history and family history of ASD are recorded as yes or no. Table 2.2.1.1 shows the Q-Chat 10 questions that were utilised in the study.

Table 2. 2. 1. 1 Q-Chat 10 questions that used in the paper [15]

Variable in Dataset	Corresponding Q-chat-10-Toddler Features
A1	Does your child look at you when you call his/her name?
A2	How easy is it for you to get eye contact with your child?
A3	Does your child point to indicate that s/he wants something? (e.g. a toy that is out of reach)
A4	Does your child point to share interest with you? (e.g. pointing at an interesting sight)
A5	Does your child pretend? (e.g. care for dolls, talk on a toy phone)
A6	Does your child follow where you're looking?
A7	If you or someone else in the family is visibly upset, does your child show signs of wanting to comfort them? (e.g. stroking hair, hugging them)
A8	Would you describe your child's first words as:
A9	Does your child use simple gestures? (e.g. wave goodbye)
A10	Does your child stare at nothing with no apparent purpose?

The samples were collected from two sites, each of which was divided into training (75%) and validation (25%) sets separately. Self-collected data from Chinese families was used as the test set. The model begins with a convolutional layer, followed by a max-pooling layer, and this structure is repeated three times to extract more detailed features. After the convolutional and pooling layers, a flatten layer is used to transform the 2D feature maps into a 1D vector. This is followed by a dense

layer and a dropout layer with a rate of 0.5 to prevent overfitting, which is repeated twice. The final output layer is a dense layer that performs binary classification. Figure 2.2.1.1 shows the model architecture diagram in detail.

Layer (type)	Output Shape	Param #
conv1d_3 (Conv1D)	(None, 14, 32)	128
max_pooling1d_3 (MaxPooling1D)	(None, 7, 32)	0
conv1d_4 (Conv1D)	(None, 7, 64)	6208
max_pooling1d_4 (MaxPooling1D)	(None, 3, 64)	0
conv1d_5 (Conv1D)	(None, 3, 64)	12352
max_pooling1d_5 (MaxPooling1D)	(None, 1, 64)	0
flatten_1 (Flatten)	(None, 64)	0
dense_3 (Dense)	(None, 64)	4160
dropout_2 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 64)	4160
dropout_3 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 1)	65

Figure 2. 2. 1. 1 Proposed Model Architecture for Checklist Approach [15]

For the first dataset, collected by others, the model achieved 100% accuracy on the validation set. For the second dataset, the accuracy was 88.39%, while the self-collected dataset achieved an accuracy of 91.84%. In comparison, the traditional SVM method, following the same process, achieved an accuracy of 86.50% on the self-collected dataset.

2.2.2 Identification of Autism Spectrum Disorder using Deep Neural Network

This study explores the application of deep learning methods to train the Q-Chat 10 questionnaire [16]. The process begins by acquiring a dataset from Kaggle and preprocessing it. The dataset comprises four categories: toddlers, children, adolescents, and adults. Initially, missing samples were removed, and standardisation was applied to normalise each column. To reduce the dimensionality of the questionnaire, Principal Component Analysis (PCA) was implemented, which eliminated less significant questions, leaving only six key attributes. Following this,

10-fold cross-validation was performed to split the dataset into 80% training and 20% testing sets.

After preprocessing, a customised architecture consisting of five layers was employed. The architecture includes an input layer, a Long Short-Term Memory (LSTM) layer with six attributes (derived from PCA) as input, a fully connected layer, a softmax layer, and an output layer for binary classification.

The model's performance was evaluated, showing that the adult dataset (18 years and above) achieved the highest accuracy at 89%, followed by the child (4–12 years) and toddler (1–3 years) datasets at 85%, and the adolescent (13–18 years) dataset at 84%.

2.2.3 Strengths of Implementing Checklist-Based Approach

Both reviewed articles use the Q-Chat 10 method, a concise checklist approach aimed at reducing inefficiencies in ASD detection. [15] utilised both external and self-collected datasets, which improved the generalisability of the trained models. It compared traditional machine learning methods with deep learning techniques, demonstrating the superior performance of deep learning. [16] incorporated several preprocessing steps, including standardisation and Principal Component Analysis (PCA), while also considering datasets from different age groups to enhance the model's adaptability and accuracy.

2.2.4 Limitations of Implementing Checklist-Based Approach

Both studies have used several sources of datasets but treat them differently when training the model. The architecture in [15] is quite complex, with many layers, and primarily focuses on an ethnic-based dataset from China, which limits its ability to generalise to other populations. In contrast, [16] uses PCA for dimensionality reduction, but this may eliminate some useful information and complicate the model architecture further. Additionally, by dropping rows with missing data, the study risks losing important features that could contribute to the model's performance.

2.3 Previous Works on ASD Mobile Application

2.3.1 ASDetect Mobile Application

ASDetect is a mobile application for parents to access the likelihood of children aged between 11 and 30 months [17]. Children beyond this interval will not be able to take the test.

First, parents need to register and log in to the application, then agree to a series of terms and conditions. After logging in, they can add their child's information, such as name and birth date, and confirm these details. The child's information will be displayed on the main page, as illustrated in Figure 2.3.1.1. Additional reminders about completing the assessment, as shown in Figure 2.3.1.2, are displayed at the bottom of the main page. Parents can also add multiple children to the application.



Figure 2. 3. 1. 1 Main Page Showing Added Children's Details [17]

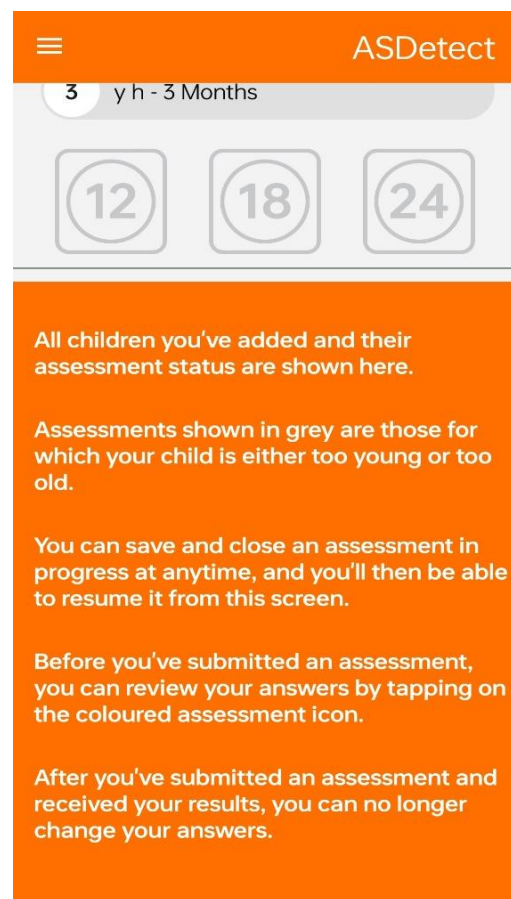


Figure 2. 3. 1. 2 Main Page Showing Additional Information [17]

After adding the child's information, parents will need to answer the questions. Upon answering, the application will show the example videos for describing the questions for both autism and non-autistic children. The videos demonstrate activities that can be done with the children. It is important to note that children aged 11 to 30 months are divided into three categories, each with its own set of questions and videos. Parents will then need to answer with "mostly" or "rarely," and the responses will be evaluated to provide a low or high likelihood of autism. Parents can review previously answered questions, and the main page will display the likelihood of autism in colour, indicating whether it is high or low. Additionally, parents will receive an email with the results for each question and a reminder to complete the next assessment as their child reaches the appropriate age.

2.3.2 ATEC (Autism Treatment Evaluation Checklist) Application

ATEC is another application used to detect ASD [18]. When users access the application, it displays four sections: the first section covers speech, language, and communication; the second section focuses on sociability; the third section addresses sensory and cognitive awareness; and the fourth section encompasses health, physical, and behavioural aspects. Each section contains between 14 and 25 questions, and each question will be provided with 3 to 4 selections. Users must complete the questions in the first section before they can proceed to the next section. Even after completing a section, users can go back to previous sections to change their answers. Users are allowed to modify their earlier responses within a section.

After answering the questions, users will receive a total score based on their responses. This score will prompt them to consider the likelihood of their child having autism. On another page, users can view the details of the result, including the total score and a line indicating whether the severity is severe, moderate, or mild. They can also check the individual scores for each of the four sections. Additionally, the result page will display the date and total score from when the test was taken.

In addition, users can add and review comments on the result page to note any special issues that occurred with their child on the day of the assessment. Figure 2.3.2.1 shows the result page of the application. Users can also view their questions and answers on this page. Moreover, users can add multiple children to the

application to track their records individually. Reminders can be set to alert parents or children when it is time to take the test again.

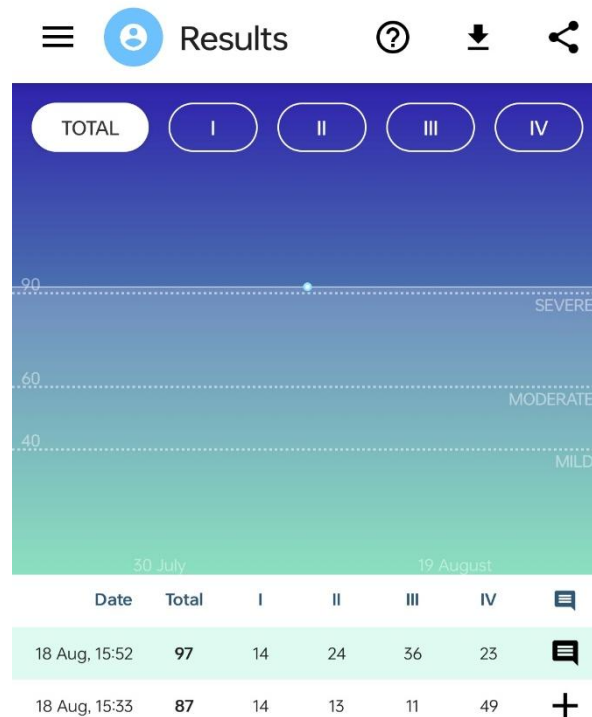


Figure 2. 3. 2. 1 Result Page Displaying Multiple Functions [18]

2.3.3 Strengths of ASD Detection Mobile Applications

Multiple children's profiles are allowed to add to both applications. The mobile application ASDetect [17] has several strengths. The first is the login feature, which enhances security by ensuring that only authorised users can access the application. It provides comprehensive guidelines on how to use the application, ensuring a smooth user experience. Not to mention, results are also sent via email to notify users. Additionally, the app owns a colourful and attractive user interface. The display of all added children on one page is particularly helpful, allowing users to manage multiple children's information without having to switch between different sections.

ATEC [18] is another mobile application with unique strengths. It allows users to retake the test, which is helpful if users accidentally chose the incorrect answer. It records the date for each test, providing an easy way for users to track their testing history. On the same page, it includes a comment session that allows users to insert

any necessary notes. Notably, it features a reminder function to notify users the time to retake the test.

2.3.4 Limitations of ASD Detection Mobile Applications

The number of existing mobile applications is limited, and two of them focus on implementing a questionnaire approach. This could be a time-consuming process, as respondents will need to answer numerous questions. Also, the length of the questions may be overwhelming, especially for children who complete the test by themselves. Their attention may be diverted, leading to inaccurate results.

2.4 Summary

For facial recognition to detect autism, the best-performing pre-trained models, considering the accuracy in each reviewed paper, are listed in Table 2.4.1. Notably, VGG16 achieved the highest accuracy among these models, as its ability to capture more detailed information contributes to its superior performance. This is followed by the Xception model, which achieved an accuracy of 92.01%. On the other hand, [11] achieved lower accuracy, possibly due to the removing duplicate images in the dataset, but still reached 87% accuracy with MobileNetV1, this followed by EfficientNetB0 achieved 85% accuracy.

Table 2. 4. 1 Comparison of ASD Facial Recognition Accuracy Across Different Studies

Model	MobileNetV1	Xception	VGG16	EfficientNetB0
Accuracy	87%	92.01%	95%	85%
Dataset Source	Kaggle Dataset	Kaggle Dataset	Kaggle Dataset	Kaggle Dataset
Total Images	2936	2936	2936	2936

For checklist-based to detect autism, the CNN-based model in [15] achieved remarkable accuracy across datasets, with 91.84% for self-collected data, significantly outperforming traditional SVM methods. Similarly, the LSTM-based model in [16] delivered consistent performance across all age groups, with each achieving an accuracy exceeding 84%. Table 2.4.2 provides the accuracy details for both models.

Table 2. 4. 2 Comparison of Checklist-based models' accuracy

Model	CNN-based model	LSTM-based model
Accuracy	91.84%	<ul style="list-style-type: none"> - Adult dataset: 89% - Child dataset: 85% - Toddler dataset: 85%, - Adolescent dataset: 84%
Dataset	Two Kaggle datasets and one self-collected dataset	Kaggle dataset, consisting of Adult, Child, Toddler, and Adolescent datasets

For mobile applications aimed at detecting autism, the similarities and differences between the two reviewed applications are summarised in Table 2.4.3.

Table 2. 4. 3 Comparison of ASD Detection Mobile Applications

	ASDetect	ATEC
Login Required	Yes	No
Guidelines Provided	Yes	No
Retake Allowed	No	Yes
Date Taken Recorded	No	Yes
User Interface (UI)	Attractive and colourful	Less attractive and less colourful
Result Comment Section Available	No	Yes
Email Notification	Yes	No
Reminder Feature	No	Yes
Support for Multiple Children	Yes	
Additional Children Displayed	Displays on a single page	Requires switching to another page to view additional children

Chapter 3 Proposed Method/Approach

This chapter outlines the overall workflow, methodology, model development process, and model architecture diagram. It also includes the system architecture, as well as the use case and activity diagrams related to the mobile application implementation in this project. Additionally, the project timeline is discussed in this chapter.

3.1 Overall Workflow for Whole Project

The overall workflow for this project is illustrated in Figure 3.1.1.

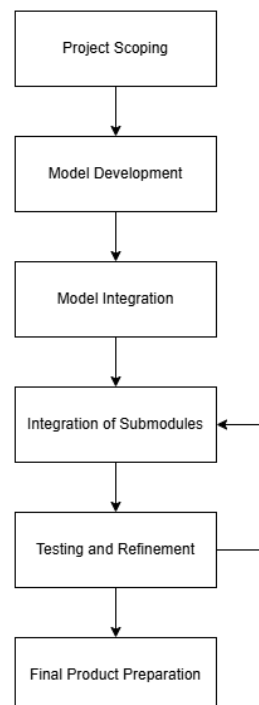


Figure 3. 1. 1 Overall Workflow for the Project

It begins with defining the project scope to establish a clear direction and set specific objectives. Once the scope is confirmed, development of the core models, namely facial recognition and the Q-Chat 10 checklist, will commence, ensuring they align with the defined goals. Following successful model development and training, these models will be integrated into a mobile application. A basic UI will be implemented to assess the models' functionality and accuracy within the mobile environment. Once confirmed, development of additional submodules will begin and later be integrated to form a complete system. The system will then be tested and

refined to ensure stability and usability. Issues identified during this stage will be resolved, with earlier stage revisited if needed. Once everything functions as expected, the application will be finalised and prepared for release. Section 3.2 explains how these activities align with the Rapid Application Development (RAD) methodology.

3.2 RAD Methodology

This project utilises RAD methodology, which closely aligns with the overall workflow. RAD breaks the system into modules, allowing for the creation of reusable components. This is particularly beneficial, as users may need to navigate back to the same module after performing certain actions. Additionally, RAD supports iterative design refinement based on user feedback, enabling easy adaptation during development. For example, functions that may not have been considered during the initial stages can be incorporated later without disrupting the overall timeline. This adaptability ensures timely delivery of necessary adjustments. Most importantly, such flexibility is crucial for this project, as ASD detection requires highly accurate results, which may require frequent revisions. Figure 3.2.1 illustrates the phases in the RAD process.

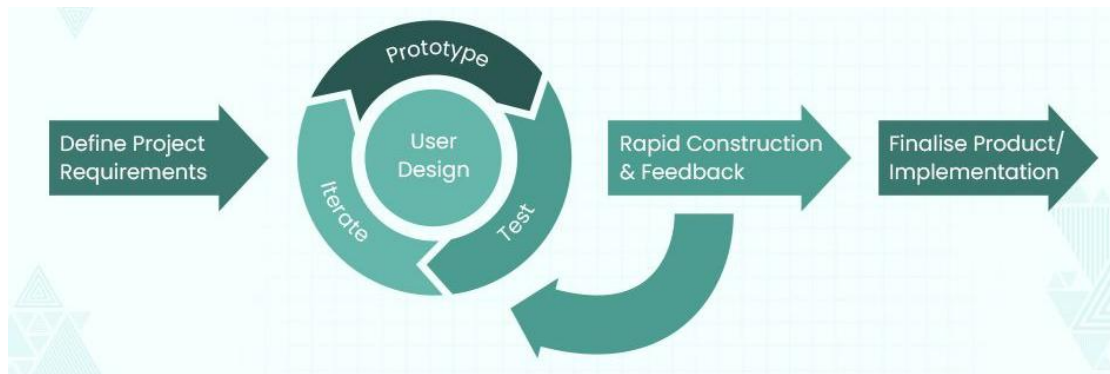


Figure 3. 2. 1 RAD Development Lifecycle [19]

Requirements Specification (Project Scoping)

This phase involves determining the project's scope and defining its core objectives. The primary goal is to develop a mobile application for early ASD detection using deep learning approaches for both facial recognition and Q-Chat 10 checklist modules. After defining project goals, research will be conducted on existing pre-trained facial recognition models, customisable architectures for checklist models, and

current ASD detection mobile applications. Apart from the two main modules, additional modules such as home page, register and login, and profile modules content will also be considered.

User Design (Model Development & Model Integration)

This phase begins with training the facial recognition model using transfer learning. Models are selected and fine-tuned through hyperparameter tuning and ablation studies, with experience working with different deep learning architectures. A lightweight custom model is then developed for the Q-Chat 10 checklist, tailored to the simpler task structure. Once trained, both models are converted for mobile compatibility. A throwaway UI prototype is created to test the functionality of the two models and evaluate how they perform within the application. The UI is then iteratively refined alongside the development of additional modules as mentioned above. Not to mention, some modules will require backend services, such as API integration, email verification, and data storage, while others may not. The interface adapts as features are added, ensuring consistency and ease of use.

Rapid Construction and Feedback (Submodule Integration & Initial Testing)

In this phase, the main and additional modules are integrated into a single working prototype. This prototype will be assessed holistically for UI flow and functionality, with any major issues or areas for improvement being promptly addressed. If necessary, the User Design phase will be revisited and refined to address these issues.

Product Finalisation and Implementation (Final Testing & Deployment)

In the final phase, all refined features undergo comprehensive testing to ensure the application operates as intended. This includes functionality checks and final bug fixes. Supporting documentation, such as reports and diagrams, will be prepared. Finally, the mobile application will be ready for public release, ensuring it is functional, user-friendly, and free from significant issues.

3.3 Model Design and Architecture

3.3.1 Facial Recognition Model

3.3.1.1 Block Diagram of Facial Recognition Model Training

To begin, Figure 3.3.1.1.1 illustrates the overall workflow for training the facial recognition model. It starts with image acquisition, duplicate removal, and dataset splitting. A baseline model is then established to serve as a reference point. This is followed by exploration of various model architectures. The development process progresses gradually, with each stage carefully evaluated before moving to the next. Systematic experimentation is conducted involving preprocessing comparisons, data augmentation techniques, hyperparameter tuning, architectural modifications, and fine-tuning strategies.

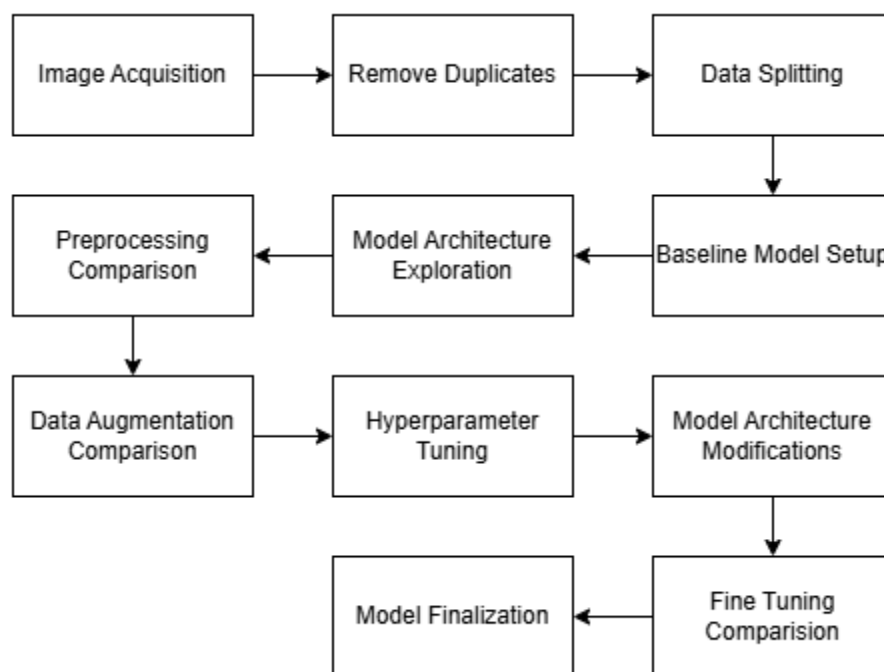


Figure 3. 3. 1. 1. 1 Workflow for Training Facial Recognition Model

Throughout the experiments, settings not involved in the specific tests will remain fixed as baseline settings to ensure consistent comparison. Through multiple rounds of testing and refinement, valuable insights are gained across different deep learning configurations. The final model is selected based on overall best performance, aiming to achieve the highest possible accuracy for ASD facial detection in a mobile application.

Image Acquisition

The workflow begins with acquiring images from two sources, which are Kaggle [20] and Google Drive [21]. Kaggle provides 1,463 images for both autistic and non-autistic classes, while Google Drive contains 1,470 images for each class. The images span an age range of 2 to 14 years, with a male-to-female ratio of 3:1, and come in varying sizes. Combining the two sources resulted in a total of 2,933 images for each class.

Remove Duplicates

Since the datasets from both sources had significant overlap, duplicate images were removed to prevent model overfitting. First, MD5 hashing was used to detect and eliminate exact duplicates, resulting in 2,933 autistic and 2,932 non-autistic images. To identify and remove near-duplicate images, further reducing the dataset to 1,481 autistic and 1,477 non-autistic images. Table 3.3.1.1.1 summarises the image counts before and after the duplicate removal process.

Table 3.3.1.1.1 Total Images before and after removing duplicates

	Initial	Applied MD5 hash	Applied MD5 hash and pHash
Autistic	2933	2933	1481
Non-Autistic	2933	2932	1477

Data Splitting

The dataset was then split into training, validation, and test sets. Initially, 90% of the dataset was allocated to the train+validation set and 10% to the test set. The train+validation set was further divided, with 90% for training and 10% for validation. The `train_test_split` function was used to ensure an equal distribution of autistic and non-autistic images in each subset. As a result, the final breakdown was 2,395 images for training, 267 for validation, and 296 for testing. Table 3.3.1.1.2 provides a detailed breakdown of the number of autistic and non-autistic images in each set.

Table 3. 3. 1. 1. 2 Breakdown of Autistic and Non-Autistic Images in Each Set

	Training Set	Validation Set	Test Set
Autistic	1199	134	148
Non-Autistic	1196	133	148

Baseline Model Setup

To ensure consistent results across training sessions, a random seed was set to 42. All input images were resized to 224×224 pixels to maintain uniformity and compatibility with the model architecture. Labels were converted into one-hot encoded format. Next, “preprocess_input” was applied based on the specific requirements of the chosen model architecture to avoid manually scaling and normalizing pixel values.

As the baseline model, MobileNetV1 was selected due to its proven balance between high accuracy and lightweight design, making it well-suited for mobile applications. The base model was frozen, meaning its pre-trained weights were not updated during this initial training phase. A GlobalAveragePooling2D layer was added to reduce the spatial dimensions of the feature maps, followed by a Dense output layer with softmax activation to adapt the model for binary classification.

Table 3.3.1.1.3 presents the initial hyperparameter settings used during the training process. Categorical cross-entropy was chosen as the loss function. Additionally, early stopping was implemented to prevent overfitting by halting training once the validation accuracy stopped improving, even if training accuracy continued to increase. The “ReduceLROnPlateau” callback was also used to reduce the learning rate when the validation loss plateaued, ensuring more efficient convergence without overshooting.

Table 3. 3. 1. 1. 3 Hyperparameters setting used to train facial recognition model

Hyperparameter	Setup
Optimizer	Adam
Initial Learning Rate	0.0001 (1e-4)
Minimum Learning Rate	0.000001 (1e-6)
Epochs	20

Batch Size	32
------------	----

Model Architecture Exploration

Exploring different model architectures is important, since it helps to identify the most suitable model for the specific task, ensuring optimal performance. MobileNetV1 (baseline model), MobileNetV2, Xception, VGG16, and EfficientNetB0 were selected for evaluation as they have shown good performance in earlier studies. The model that achieved the best overall performance was selected for subsequent tasks in the development process.

Preprocessing Comparison

The process began by comparing face-cropped images using the dlib library against uncropped images, as cropping can help focus the model on relevant facial features. The better-performing option was selected and used in the next step. Next, Gaussian Blur was applied and compared with the previous result to assess its impact on reducing noise and improving feature extraction, with the better configuration retained. Finally, CLAHE was evaluated to determine its effectiveness in improving image contrast and helping the model detect finer details, and the optimal setup was chosen based on performance.

Data Augmentation Comparison

Next, data augmentation techniques were compared to improve the model's ability to generalise and recognise diverse variations in the input images. The comparison involved three setups: no augmentation, single augmentation, and double augmentation. Augmentation was performed using ImageDataGenerator, with random transformations such as rotation within 20 degrees, width and height shifts of 0.2, shear range of 0.2, zoom of 0.2, and horizontal flip. By testing these augmentations, the best-performing approach will be carried forward for subsequent stages of the model development.

Hyperparameter Tuning

This process involves tuning the batch size, optimizer, and learning rate to enhance model performance. Early Stopping is applied, so the number of epochs is not a focus in this tuning stage.

First, batch size was tested with values of 16, 32 (default), and 64. Tuning batch size is important because it can influence model accuracy by affecting how the model updates its weights during training.

Next, different optimizers were compared, including Adam, AdamW, and SGD, each paired with learning rates suited to their behaviour. Selecting the right optimizer–learning rate combination is crucial, as it affects how effectively and reliably the model converges during training.

Model Architecture Modifications

On top of the selected base model architecture, additional modifications were explored to improve generalisation and reduce overfitting. Dropout layers with different values were tested, as dropout helps prevent over-reliance on specific neurons by randomly deactivating them during training. Furthermore, Batch Normalization was compared with and without inclusion, as it can stabilise and speed up training by normalising the input to each layer.

Fine Tuning Comparison

This process involves loading the previously best-performing model after completing all prior experimental settings. Fine tuning is carried out to further improve model performance by allowing the pretrained layers to adjust slightly to the new dataset, which can lead to better feature extraction for the specific task. The comparison includes three approaches: keeping the base model fully frozen, unfreezing certain blocks, and fully unfreezing the base model. It is important to note that the learning rate used here may differ from the previously optimised value, as fine-tuning typically benefits from a lower learning rate to avoid large updates that could disrupt the pretrained weights.

Model Finalization

During the fine-tuning stage, if multiple models produce similar results, they will all be evaluated on the test set and converted into a mobile-compatible format, which is TensorFlow Lite (TFLite), to observe any performance differences. The best-performing model will be selected based on evaluation metrics such as test accuracy, precision, recall, F1-score, and AUC-ROC, along with its accuracy in the mobile format. The model that performs the best overall will be integrated into the mobile application for ASD facial detection.

3.3.1.2 Architecture Diagram of Facial Recognition Model

As mentioned earlier, MobileNetV1, MobileNetV2, Xception, VGG16, and EfficientNetB0 models are used to evaluate their performance in detecting autism through facial recognition. Since the base architectures of these models have already been discussed in Chapter 2 (except for MobileNetV2), this section focuses solely on the custom layers added on top of each base model.

As shown in Figure 3.3.1.2.1, 2–3 custom layers are added after the base model. First, a Global Average Pooling layer is used to reduce the data dimensionality while retaining essential features. A Dropout layer may be added based on performance observations, as it helps prevent overfitting by randomly disabling neurons during training. Finally, a Dense output layer is used for classification.

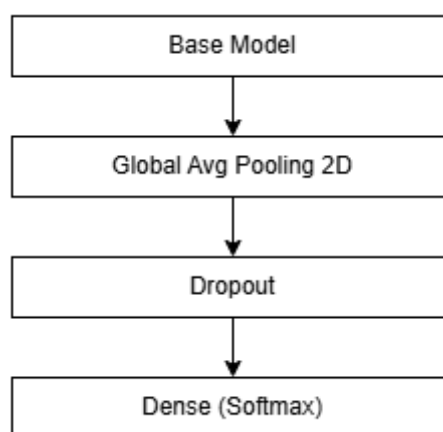


Figure 3.3.1.2.1 Customised layers added to the facial recognition model architecture

Although a Softmax activation function is typically used for multi-class classification, converting the model to TFLite for deployment may introduce compatibility issues when using Softmax for binary classification tasks. Therefore, for binary classification between autism and non-autism, a Sigmoid activation function is more appropriate and is used in the final model implementation.

3.3.2 Checklist-Based Model

3.3.2.1 Block Diagram of Facial Recognition Model Training

To begin, Figure 3.3.2.1.1 illustrates the overall workflow for training the Q-Chat 10 model. The objective is to provide an alternative method for ASD detection within a mobile application. A straightforward pipeline is employed without extensive experimentation, as this task is relatively simpler compared to facial recognition.

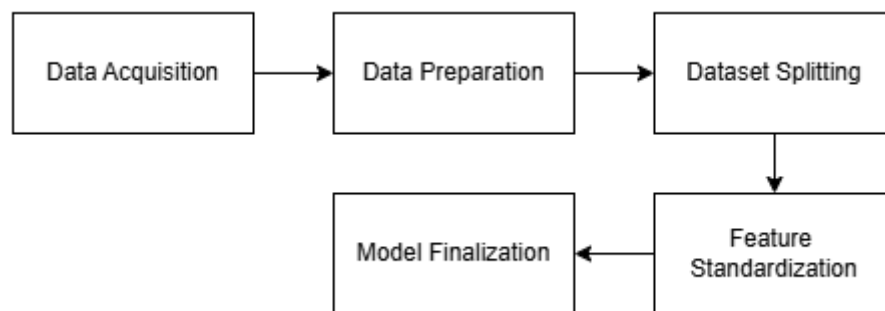


Figure 3. 3. 2. 1. 1 Workflow for Training Checklist-Based Model

Data Acquisition

The workflow begins with acquiring dataset from three sites of Kaggle, which are [22] containing 1985 records, [23] containing 704 records, and [24] containing 1054 records. These datasets were selected to increase the diversity of data and ensure robust model training. However, each dataset contained different numbers of columns and variations in the format of responses.

The next step involved identifying the common columns across all three datasets, resulting in 16 attributes: A1 to A10 (questionnaire items discussed in Chapter 2), age, gender, ethnicity, history of jaundice, family history of ASD, the person who completed the test, and the target variable indicating ASD classification.

CHAPTER 3

These datasets were then merged into a single DataFrame for further processing. Each attribute and its data type are listed in Table 3.3.2.1.1.

Table 3. 3. 2. 1. 1 Details of each attribute type in checklist

Question	Type
A1 to A10	Binary (0, 1)
Age	Number
Gender	Binary (0, 1)
Ethnicity	String
History of jaundice	Boolean (yes or no)
Family history of ASD	Boolean (yes or no)
Person who completed the test	Boolean (yes or no)

Data Preparation

After integration, the data was cleaned and standardised. Variable formatting inconsistencies were addressed by capitalising text, removing hyphens, and ensuring consistent spelling. Missing values were handled using the Simple Imputer with the most frequent strategy to retain important features. All variables, except the target variable, were encoded into numeric representations for ease of processing. The features (X variables) and target (Y variable) were then separated.

Data Splitting

The dataset was split into training, validation, and testing sets as follows: train + validation (90%) and test (10%). The train + validation set was further split into train (90%) and validation (10%), resulting in 3,031 records for training, 337 for validation, and 375 for testing.

Feature Standardization

The X variables in the train, validation, and test sets were standardised using StandardScaler, ensuring that the data was normalised (mean = 0, variance = 1). This normalization helps improve the performance of models sensitive to feature scales. The Y variable was encoded, with 1 representing the ASD class and 0 representing the non-ASD class.

Model Finalization

A custom deep learning model was built and compiled using the Adam optimizer with a default learning rate of 0.001. Although the project aimed for binary classification, categorical cross-entropy was chosen as the loss function for easier integration into the mobile application. The model was trained for 50 epochs with a batch size of 32 and evaluated on the validation dataset. The hyperparameters are summarised in Table 3.3.2.1.2.

Table 3. 3. 2. 1. 2 Hyperparameters setting used to train checklist-based model

Hyperparameter	Setup
Optimizer	Adam
Initial Learning Rate	0.001 (1e-3)
Epochs	50
Batch Size	32

Finally, the model's performance was assessed using the test dataset. Similarly, the key metric for evaluation was accuracy, especially after integrating the model into the mobile application. The model was evaluated using Python and subsequently converted to TFLite for deployment on the mobile application to perform evaluation.

3.3.2.2 Architecture Diagram of Checklist-Based Model

The customised model for the Q-Chat 10 approach is shown in Figure 3.3.2.2.1. This is a simple model consisting of three dense layers. The first two layers are designed to learn patterns by adding non-linearity, which is important for processing the questionnaire data. The final layer is used for binary classification, distinguishing between autism and non-autism. Notably, this model is intentionally kept lightweight, with different structure as described in Chapter 2. This approach helps allocate resources more efficiently, allowing greater focus on the facial recognition task.

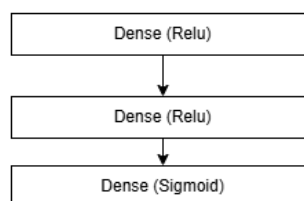


Figure 3. 3. 2. 2. 1 Customised layers added to the checklist-based model architecture

3.4 Mobile Application Design

3.4.1 System Architecture Diagram

Figure 3.4.1.1 illustrates the system architecture diagram outlining user interactions with the Flutter application and its integration with Firebase services.

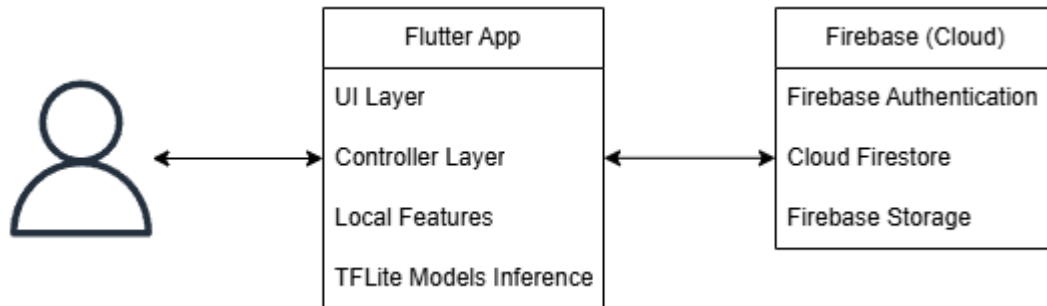


Figure 3. 4. 1. 1 System Architecture Diagram

The system architecture follows a structured flow where users interact with a Flutter application composed of four primary layers: the UI layer, controller layer, local device features, and TFLite model inference. The UI layer presents the visual interface to users, handling all frontend interactions and ensuring an intuitive experience. The controller layer manages application logic, including Firebase API calls for authentication, data fetching, and storage operations.

The local device features layer enables access to hardware functionalities such as the camera and local file storage for image uploads, while also handling local notifications to remind users to complete scheduled tests. This layer integrates with the TFLite model inference, which processes uploaded images or answered checklists to deliver real-time detection results back to users.

For backend services, the system leverages Firebase, which provides email/password authentication and automatic session persistence, eliminating the need for repeated logins. Firebase Cloud Firestore is used to store and retrieve critical data, including predefined questionnaire questions, information related to facial recognition, information related to checklist-based assessment, and user profile details. Firebase Storage manages the upload and retrieval of user-submitted images.

3.4.2 Use Case Diagram for Mobile Application

Figure 3.4.2.1 illustrates the use case diagram for the project. In this system, users are categorised as either new or registered. New users are required to register an account and verify their email address before gaining access to the login feature. Registered users can log in using their credentials, and if they forget their password, a reset option is available.

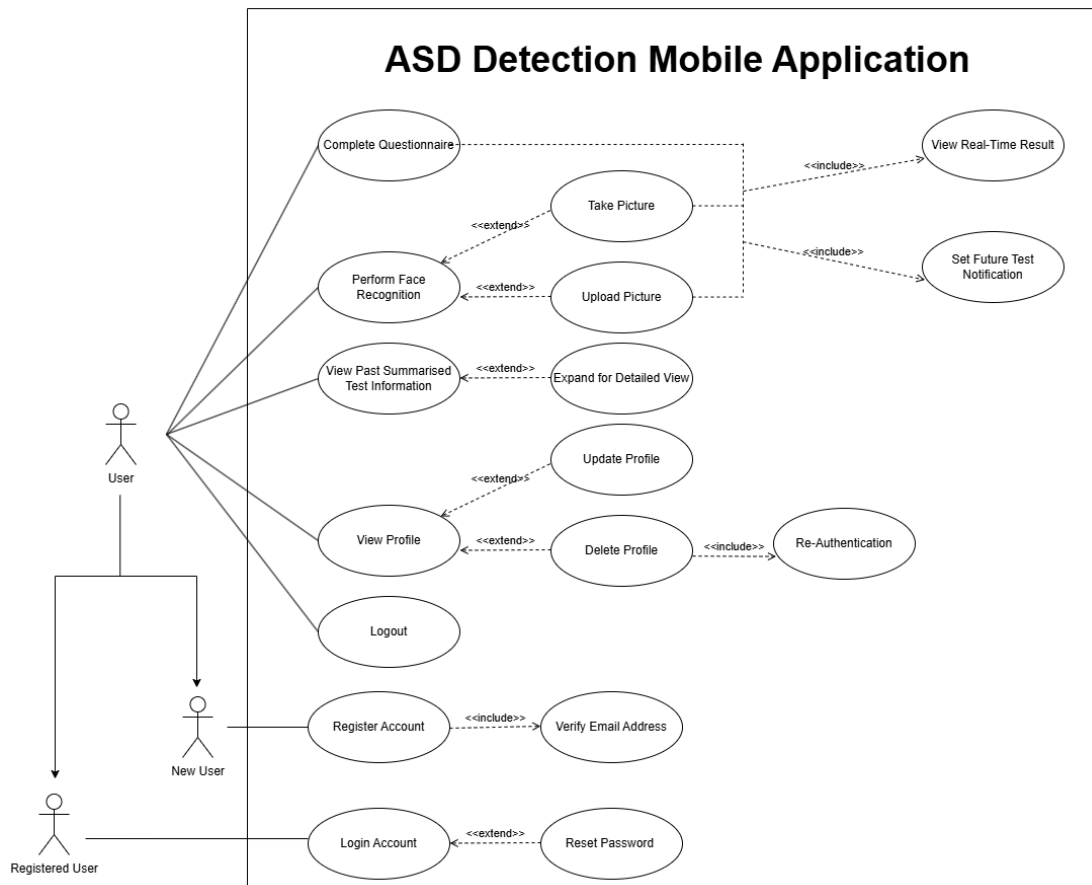


Figure 3. 4. 2. 1 Use Case Diagram

Once logged in, users can perform two primary actions: completing a questionnaire or using facial recognition to assist in autism detection. For the facial recognition feature, users may either upload an image from their gallery or capture one instantly. After submission, results are provided, and users will be reminded to retake the assessment after three months.

Additionally, users can view a summary of their past questionnaire responses and facial recognition results. These summaries are presented in collapsible sections, which can be expanded to display detailed answers or submitted images alongside

their respective outcomes. Users also have access to profile management features, including updating their information or permanently deleting their account. Account deletion requires reauthentication via email and password. Users can also log out of the application at any time.

3.4.3 Activity Diagram for Mobile Application

Register, Login, View Past Summarised Test Information

Figure 3.4.3.1 presents the activity diagram for user registration, login, and viewing previously summarised test information. The process begins when the user chooses either to register or to log in.

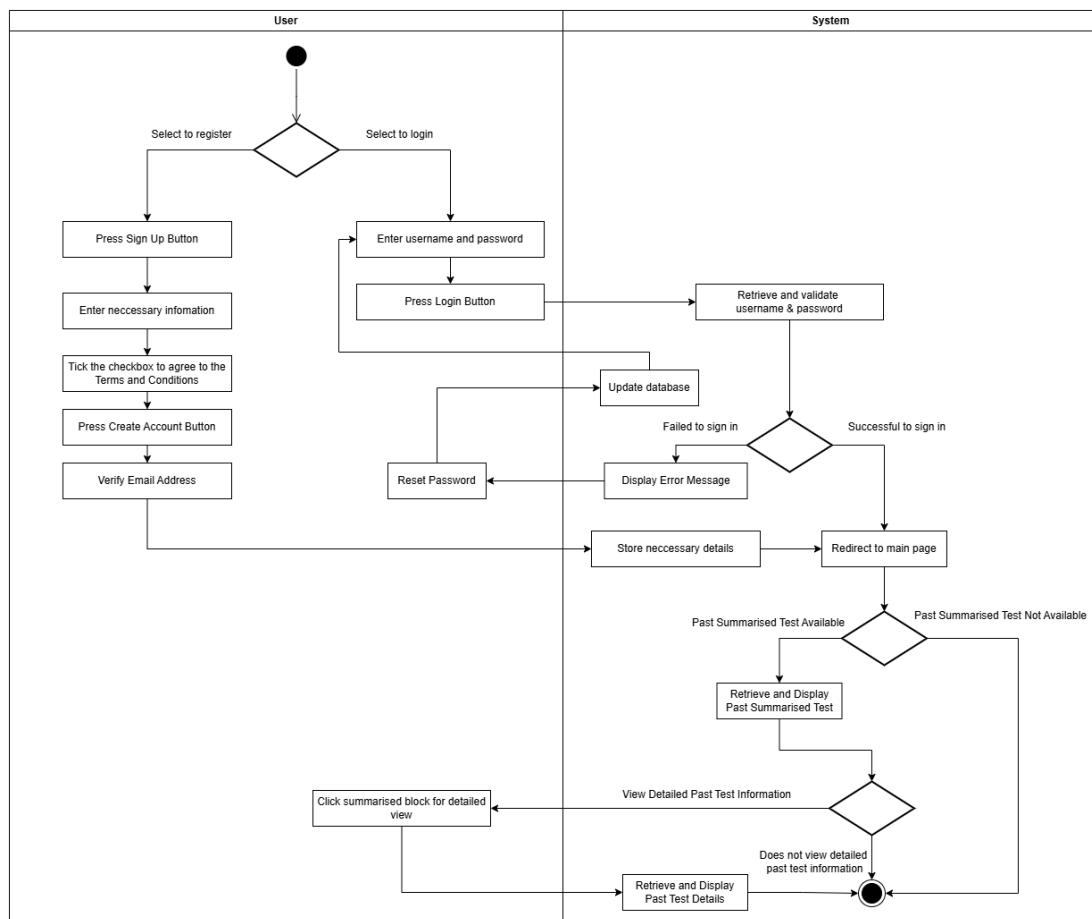


Figure 3. 4. 3. 1 Activity Diagram for Registering, Logging and Viewing Past Summarised Test Information

For registration, users must provide the required details and agree to the terms and conditions. Upon submission, an email verification is sent. Once verified, the user's information is stored in the database, and they are redirected to the main page.

For login, registered users enter their email address and password. The credentials are then retrieved and validated against the database. If the login fails, users are given the option to reset their password. Once reset, the updated password is saved to the database, and users can attempt to log in again.

Upon successful login, users are directed to the main page, where a summary of their past test results is retrieved and displayed. Users can also expand each summary entry to view the corresponding detailed information that is retrieved from the database.

View Profile

Figure 3.4.3.2 illustrates the activity diagram for viewing the user profile. Users can click “Profile” to view their account details, which are fetched from the database and displayed. If users wish to update or delete their account, they can select the corresponding option. To update their information, users must enter the new details they want to modify. To delete their profile, users must confirm the action and reauthenticate using their email address and password. The updated or deleted information is then reflected in the database. Users also have the option to log out from their account.

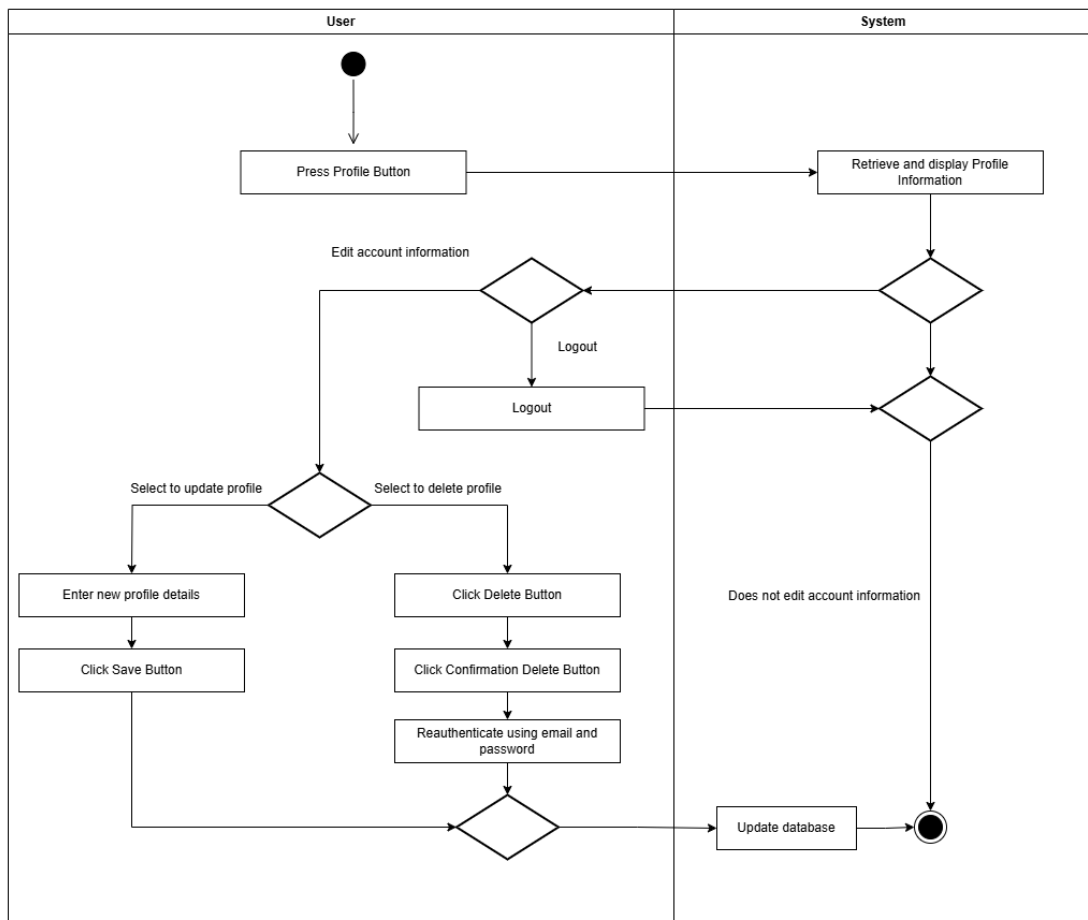


Figure 3. 4. 3. 2 Activity Diagram for Viewing Profile

Perform Questionnaire or Facial Detection

Figure 3.4.3.3 presents the activity diagram for performing either the questionnaire or facial detection task. Users can choose to perform one action at a time. For facial detection, users can either upload an image from their device or capture a real-time photo using the camera. Once the detection task is completed, the result is displayed and stored in the database. Additionally, a future notification is scheduled to remind users to take the test again.

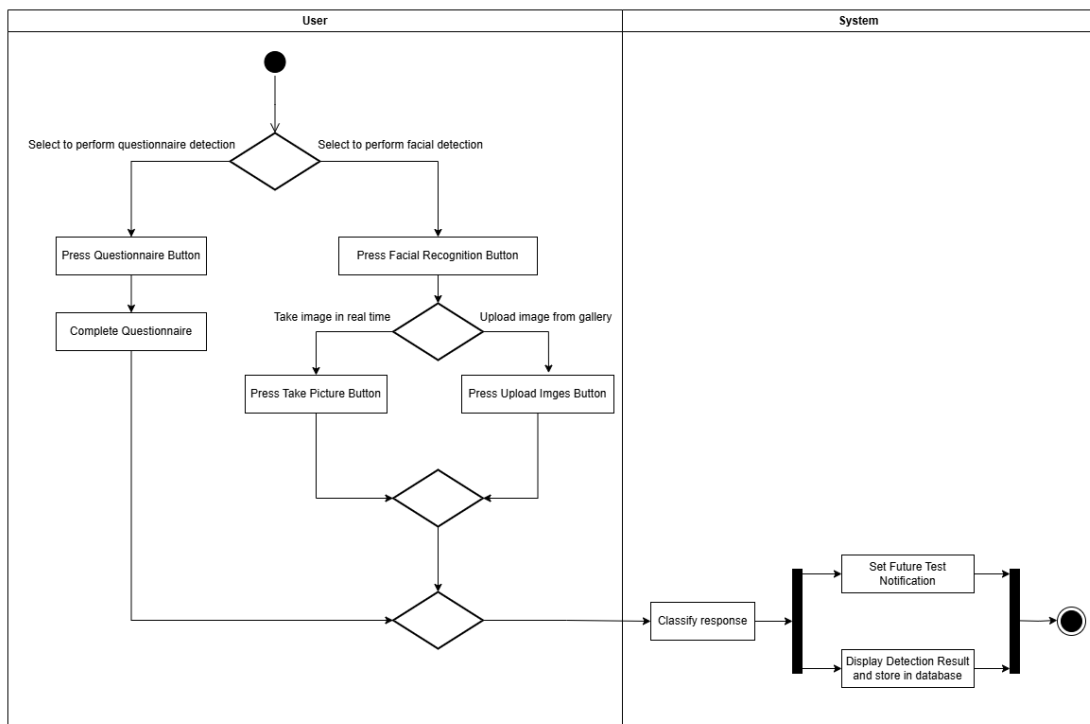


Figure 3. 4. 3. 3 Activity Diagram for Performing Questionnaire or Facial Detection

3.5 Timeline

Gantt chart will be used to represent the timeline for this project. It is often employed in project management to outline the predecessors, successors, and the duration needed to complete each task. This project is divided into two timelines, which are Project I and Project II.

Project I spans 7 weeks, which mainly focus on confirming the project requirements, designing diagrams, and developing important functions. Specifically, the main modules which are facial recognition and checklist-based, will be developed during this phase, together with simple interface design to provide a preliminary view of the application. Figure 3.5.1 illustrates the detailed breakdown for each task.

Task Name	Start Date	End Date	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
Project	2024-10-28	2024-12-15						
Refine Project Requirements	2024-10-28	2024-11-04						
Design Diagrams	2024-10-28	2024-11-04						
Design Main Modules UI	2024-11-04	2024-11-11						
Design Main Modules	2024-11-11	2024-11-25						
Test Main Modules	2024-11-25	2024-12-02						
Write Report	2024-11-25	2024-12-06						

Figure 3. 5. 1 Timeline for Project I

CHAPTER 3

Project II spans 13 weeks and focuses on finalizing the project and delivering the mobile application for ASD detection. This phase will continue the development of the main modules from Project I, making any necessary changes. After that, the core logic for the sub-modules and user interface will be designed. This will be followed by integrating the main and sub modules, making necessary adjustments, and validating the application. Not to mention, the facial recognition model is being retrained to get better performance in Project II. Figure 3.5.2 shows the detailed breakdown of each task in Project II.

Task Name	Start Date	End Date	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Project	2025-02-10	2025-05-09													
Continue Designing Main Modules	2025-02-10	2025-02-24													
Design Sub Modules' User Interface	2025-02-17	2025-02-24													
Design Sub Modules	2025-02-24	2025-03-10													
Integrate Main Sub Modules	2025-03-03	2025-03-17													
Refine All Modules	2025-03-10	2025-04-07													
Refine Home Page	2025-04-07	2025-04-21													
Write Report	2025-04-14	2025-05-09													
Refine Facial Recognition Model	2025-04-21	2025-04-28													
Validate Application	2025-04-21	2025-05-05													
Finalize Application	2025-04-28	2025-05-05													
Finalize Report	2025-04-28	2025-05-09													

Figure 3. 5. 2 Timeline for Project II

Chapter 4 System Design

This chapter outlines the overall system design of the mobile application. It includes the flowchart of the application, the database design using a NoSQL database (Firestore), the system block diagram, and the specifications of each component or module.

4.1 Flowchart

Figure 4.1.1 shows the overall flowchart when using the mobile application.

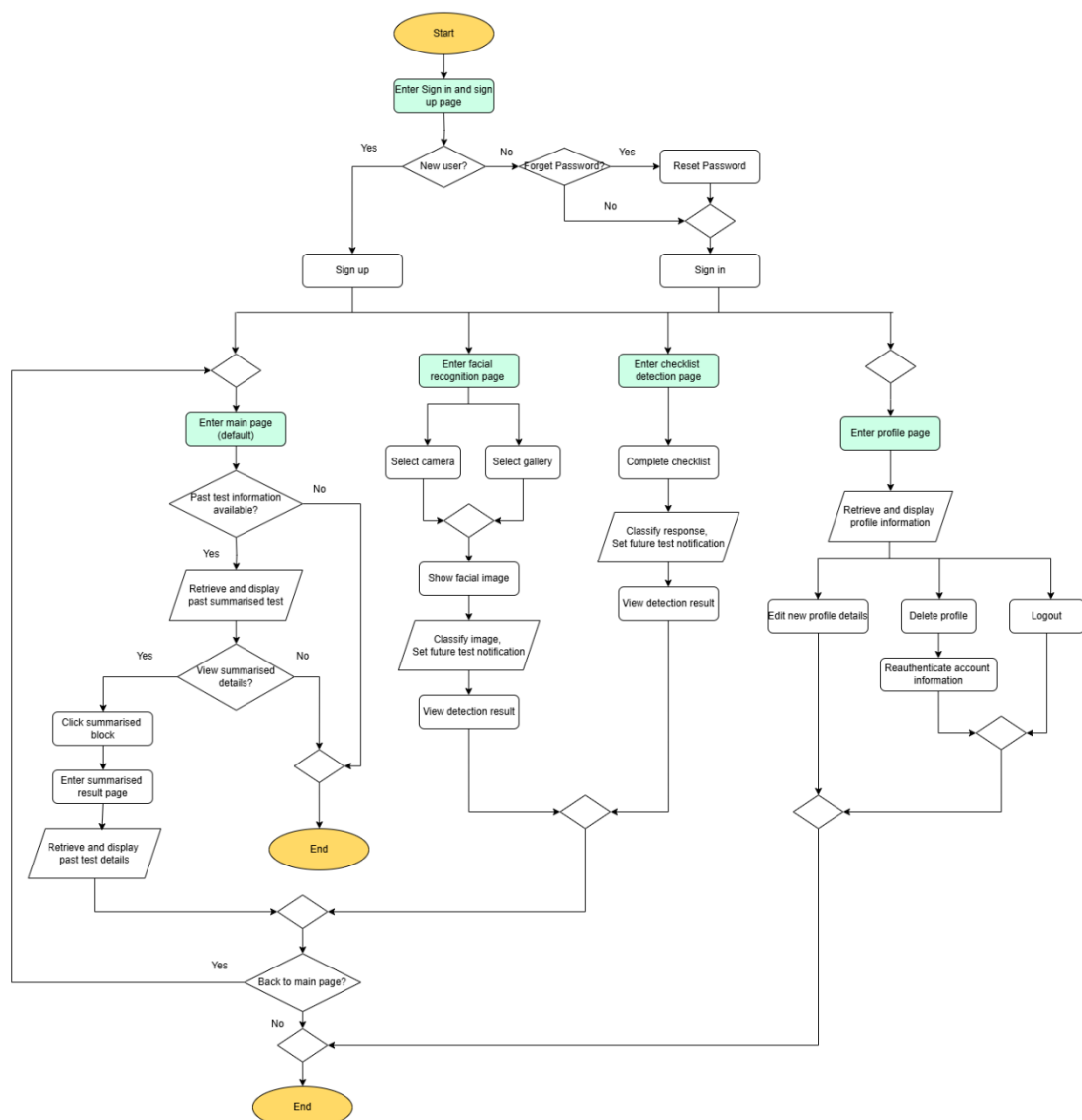


Figure 4. 1. 1 Overall flowchart for mobile application

4.2 Database Design (Cloud Firestore)

A NoSQL database design will be implemented with four main collections: Images, Questionnaire, Responses, and Users.

The database design for the Images collection is illustrated in Figure 4.2.1. This collection is used to store facial recognition-related information, including data about uploaded or captured images and their corresponding detection results. Within the Images collection, data is organized to support multiple users. Each user document can contain multiple responses. For each response, several subfields are stored, including result confidence, stored image URL, classification label, and timestamp of when the test was performed.

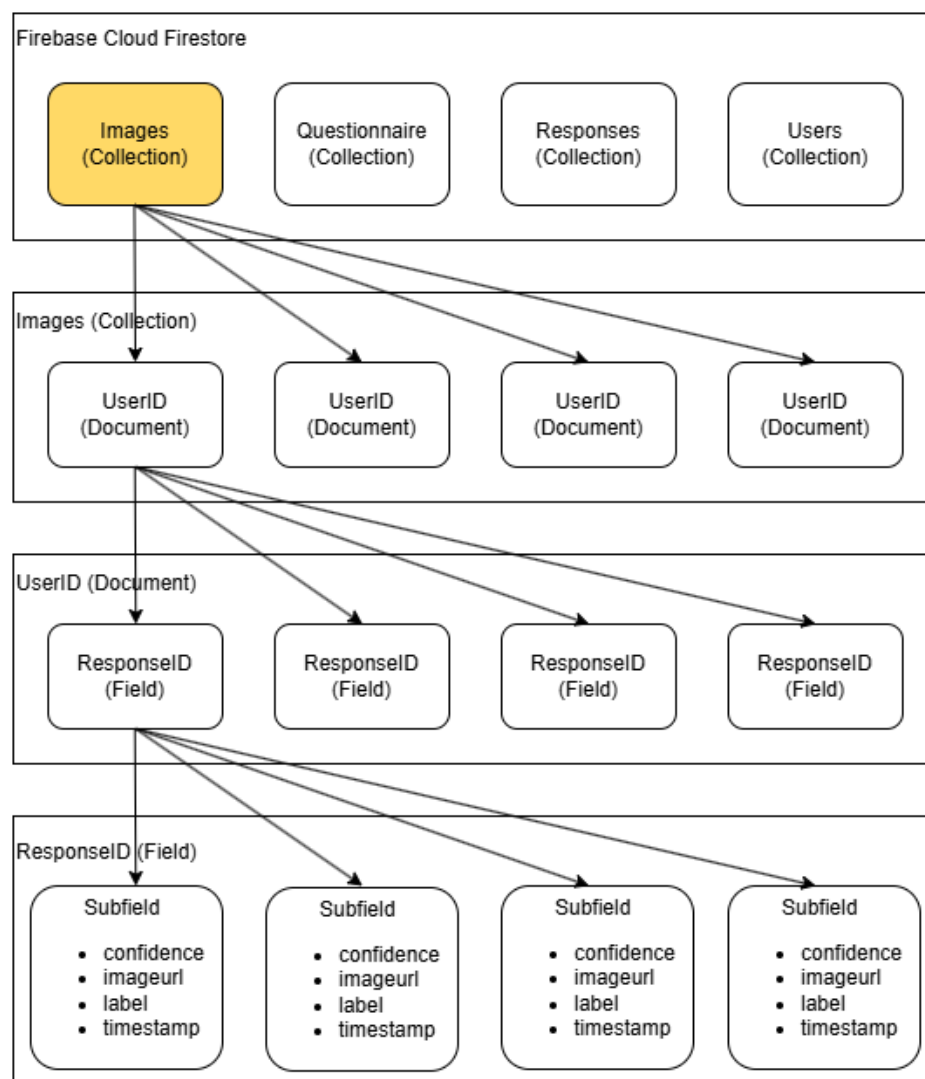


Figure 4. 2. 1 Images collection's database design

The database design for the Questionnaire collection is illustrated in Figure 4.2.2. This collection is used to store predefined checklist questions. Within the Questionnaire collection, each document represents a single question and includes two main fields: the question, which is stored as a String, and the option, which is stored as an array to represent the available answer choices.

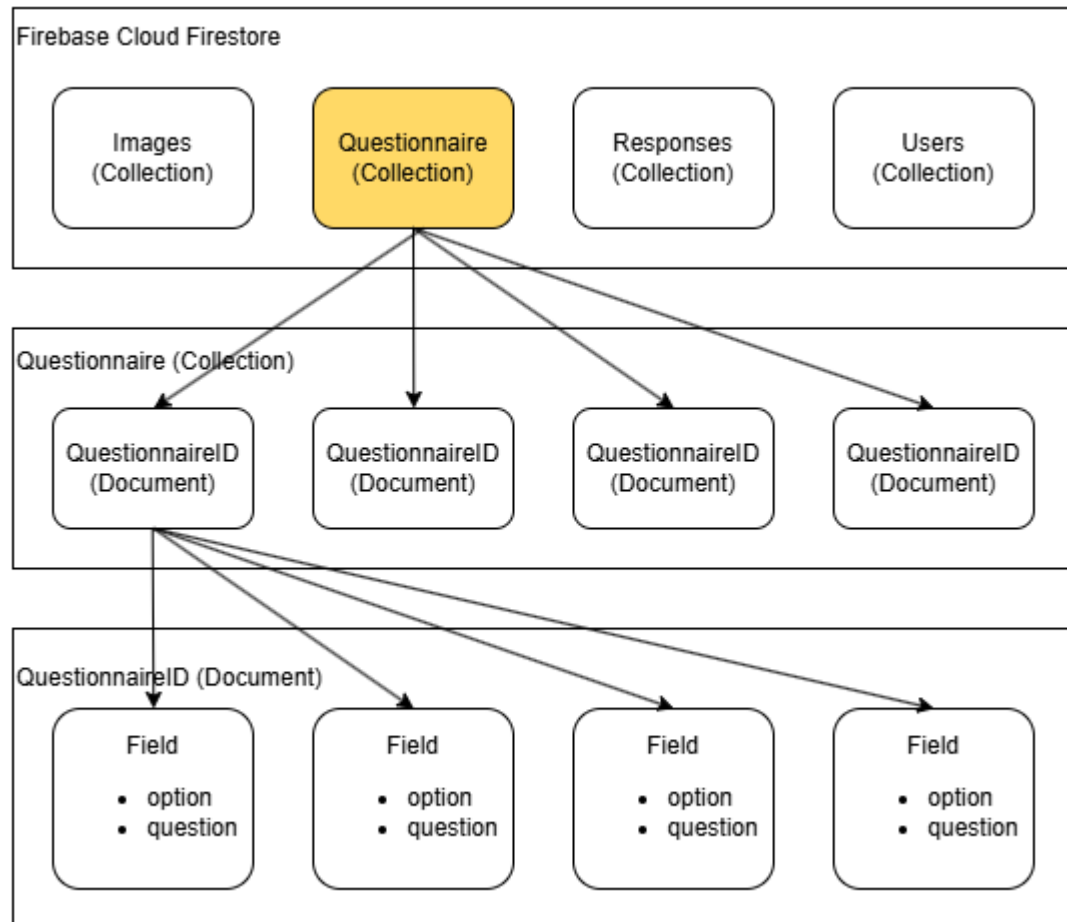


Figure 4. 2. 2 Questionnaire collection's database design

The database design for the Responses collection is illustrated in Figure 4.2.3. This collection is used to store the selected options from the checklist-based assessment. Within the Responses collection, data is organized to support multiple users, where each user document can contain multiple responses. For each response, several subfields are included, such as the selected options (stored in array format to represent answers for each questionnaire item), the result confidence, the classification label, and the timestamp indicating when the test was performed.

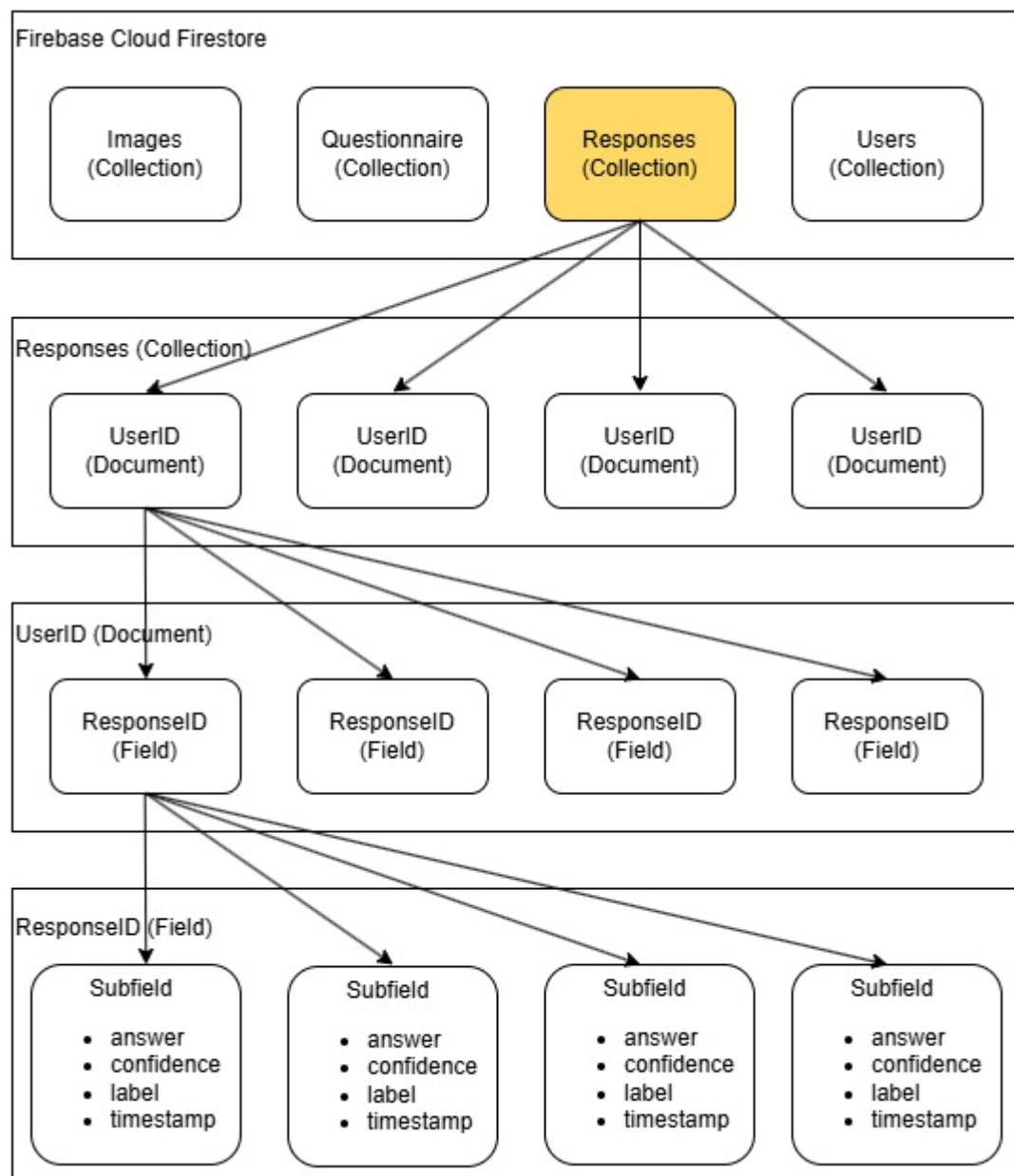


Figure 4. 2. 3 Responses collection's database design

The database design for the Users collection is illustrated in Figure 4.2.4. This collection is used to store user profile information. Within the Users collection, each document represents a single user and includes several key fields related to their personal data, such as name, email address, gender, and other relevant account details.

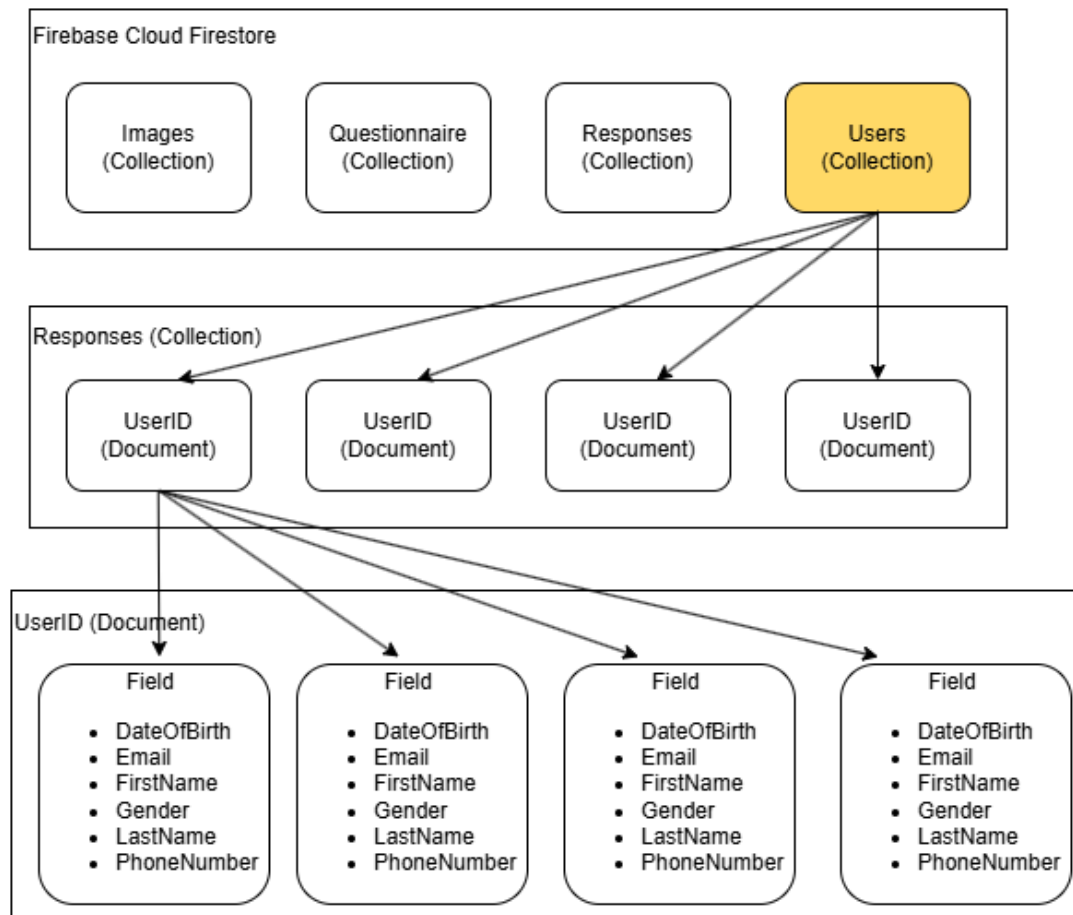


Figure 4. 2. 4 User collection's database design

4.3 System Block Diagram

Figure 4.3.1 represents the block diagram for the mobile application. The application is organised into five main modules: the register and login module, the main page module, the facial recognition module, the checklist module, and the profile module.

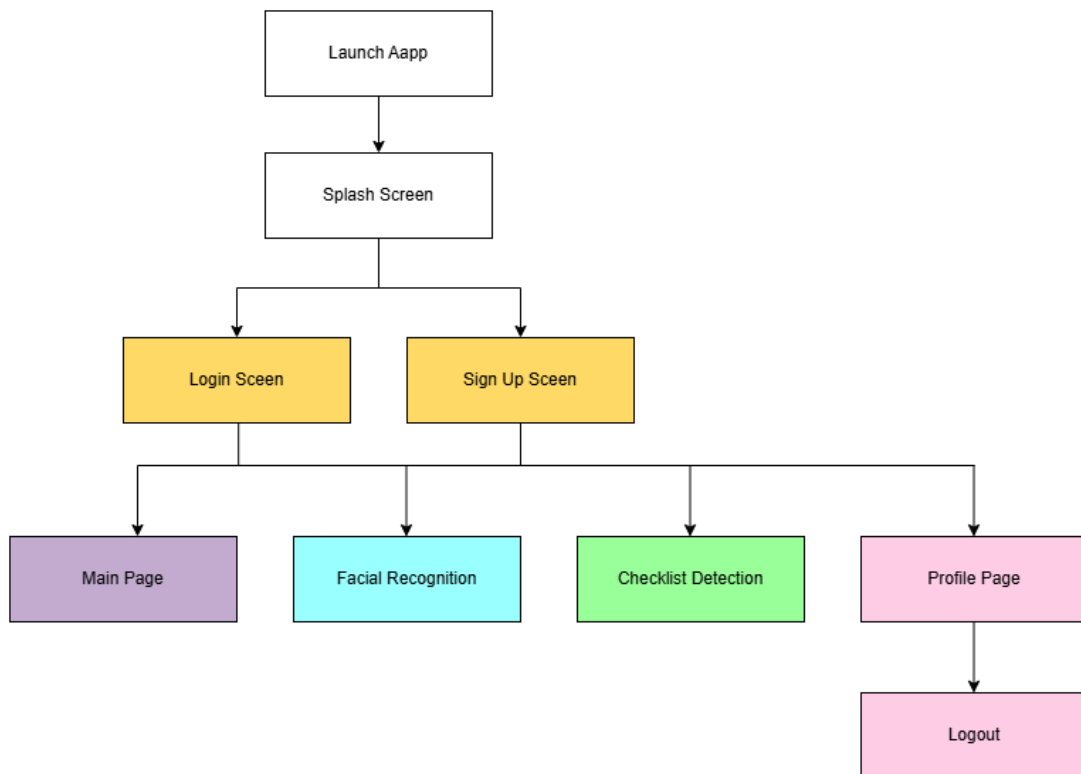


Figure 4. 3. 1 Block diagram for mobile application

4.4 System Component Specifications

The **register and login module**, which enables users to create an account and access the application. During registration, users will provide required personal information, including name, email, phone number, gender, date of birth, and a password. The module supports user login and includes a password recovery feature, with email verification triggered when needed.

The **main page module** serves as a centralised interface to display all test results from both the facial recognition and checklist modules. These results are shown in a compact format with options to view more details. It also includes a reminder system to prompt users to retake tests every three months.

CHAPTER 4

For the mobile application, the **facial recognition module** will use the trained model to analyse images. Users can either upload or capture a photo to receive a classification (autistic or non-autistic) along with a confidence score.

The **checklist module** will use the trained model to predict ASD based on user inputs. It will output a binary classification and confidence score, serving as a complementary method to validate results from the facial recognition module.

The **profile module** displays the information provided during registration, allowing users to review and update their details if necessary. Users will also have the option to delete and log out of their accounts if desired. It ensures that users have control over their data and account settings.

Chapter 5 System Implementation

This chapter outlines the hardware and software requirements used throughout the entire project. It also describes the necessary configurations that had to be completed before the development. In addition, this chapter presents how the mobile application looks, discusses the system's operation, and highlights the implementation challenges encountered during the project.

5.1 Hardware Setup

The hardware required for this project includes a laptop and a mobile device. The laptop is used for building and evaluating the models, developing the application, and running an emulator to test the application's features. The mobile device is primarily used to test the image capture functionality, while the remaining features are tested on the emulator for convenience. Tables 5.1.1 and 5.1.2 present the specifications of the laptop and the mobile device, respectively.

Table 5. 1. 1 Specifications of Laptop

Description	Specifications
Model	HP 15s-eq series
Processor	AMD Ryzen 3 4300U, 2700 MHz, 4 Cores, 4 Logical Processors
Operating System	Microsoft Windows 11 Home Single Language
Graphic	Radeon Graphics (integrated with the AMD Ryzen 3 4300U)
Memory	16 GB (15.4 GB usable)
System Type	64-bit operating system, x64-based processor

Table 5. 1. 2 Specifications of Mobile Device

Description	Specifications
Model	Oppo Reno 11F series
Processor	MediaTek Dimensity 7050
Operating System	Android 14, ColorOS
Memory	8 GB
Storage	256 GB
Camera	Front 32 MP, Rear 64 MP
GPU Clock Speed	800 MHz
Max CPU Clock Speed	2600 MHz

5.2 Software Setup

This section outlines the key tools, coding platforms, languages, libraries, and framework used in the project. Table 5.2.1 presents the main tools, Table 5.2.2 lists the languages used, Table 5.2.3 includes the libraries in Python and Dart, Table 5.2.4 shows the framework used for Dart development, and Table 5.2.5 shows the specifications of emulator used.

Table 5. 2. 1 Specifications of Main Software Tools

Description	Specifications
Coding Platform	<p><u>Kaggle</u></p> <p>It is a cloud-based platform widely used for machine learning tasks. It was chosen for this project due to its accessibility to powerful GPUs, which is especially beneficial for training and testing deep learning models efficiently.</p> <p><u>Android Studio</u></p> <p>specifically designed for Android development. It runs on Windows and serves as the main platform for writing and debugging Dart code, integrating machine learning models, and building the Android application. It also provides tools such as emulators and support for physical devices to test the application's features.</p>
Storage Service	<p><u>Firebase</u></p> <p>It provides a range of backend services, including Cloud Firestore, Firebase Storage, and Authentication, all of which are utilised in this project. Cloud Firestore and Firebase Storage are used to securely store and retrieve important data such as test results and user profiles. Firebase Authentication ensures that only authorised users can access the application, safeguarding login credentials and maintaining user privacy.</p>

Table 5. 2. 2 Specifications of Software Languages

Languages	<p><u>Python</u></p> <p>It is a widely used programming language that serves a variety of purposes. Here, it will be used to handle tasks related to training the facial recognition and checklist models for the ASD detection process.</p> <p><u>Dart</u></p> <p>A mobile application can be developed using Dart programming language. Due to its ability to support object-oriented programming, Dart is utilised to maintain to code readability. All the features stated for mobile application can be achieved with this coding.</p>
-----------	---

Table 5. 2. 3 Specifications of Libraries

Libraries & Packages (Python)	<p><u>OpenCV v4.10.0</u></p> <p>OpenCV is famous for its ability to handle computer vision tasks, namely resizing images. It will be used to assist with image processing in the detection of ASD.</p> <p><u>Tensorflow v2.17.0</u></p> <p>It helps in building, training, and deploying neural network models for a variety of tasks, including the detection of ASD. It provides tools for model conversion to optimize performance across different platforms.</p> <p><u>Keras v3.6.0</u></p> <p>It helps in quickly build and train deep learning models without needing to write complex code.</p> <p><u>Dlib v19.24.6</u></p> <p>It provides tools for detecting faces in images and finding key points (like eyes, nose) on the face, helps in detecting face region for cropping images.</p>
----------------------------------	--

	<p><u>Pandas v2.2.3</u></p> <p>It organises and manipulates data in tables, making it easy to prepare and visualise data. In the context of Q-Chat 10, it aids in managing datasets and preparing data for further analysis or model training.</p> <p><u>ImageHash v4.3.1</u></p> <p>It used to generate unique hashes for images, which helps in comparing images to check if they are similar.</p> <p><u>NumPy v1.26.4</u></p> <p>It can assist in combining or modifying data, which is useful when feeding data into models.</p>
Libraries & Packages (Dart)	<p><u>tflite_flutter v0.11.0</u></p> <p>It is a mobile library that enables pre-trained TensorFlow Lite models to run efficiently on various devices. In this project, it is used to convert and integrate both facial recognition and checklist-based models for detecting autism in children within a mobile application.</p> <p><u>Firebase Dart SDK</u></p> <p>It allows seamless integration of Firebase services into Flutter applications. It facilitates user authentication, data storage via Cloud Firestore, and file handling through Firebase Storage. These services are essential for managing user profiles, storing test results, and maintaining secure access.</p> <p><u>flutter_local_notifications v19.1.0</u></p> <p>This package is used to provide local notifications to users. In this project, it will issue alerts if the most recent response, whether from facial image detection or checklist input, is older than three months. This prompts users to retake the assessment to keep their results up to date.</p>

	<p><u>image_picker v0.8.9</u></p> <p>It allows users to either select an image from the gallery or capture one using the device camera. In this project, it is used to enable facial image input as part of the autism detection process.</p>
--	--

Table 5. 2. 4 Specifications of Framework

Frameworks (Dart)	<p><u>Flutter</u></p> <p>Flutter is a software development kit (SDK) that allows mobile applications to be compatible with both Android and iOS. Here, the project will use it to develop the user interface for the Android devices. All the modules' user interfaces can be achieved with Flutter.</p>
-------------------	---

Table 5. 2. 5 Specifications of Emulator

Description	Specifications
Model	Pixel 6 API 28
Processor	AMD Ryzen 3 4300U, 2700 MHz, 4 Cores, 4 Logical Processors (Emulator run on host machine's CPU)
Operating System	Android 11
Memory	4 GB
Storage	64 GB

5.3 Setting and Configuration

For model training, the Kaggle platform was utilised to take advantage of GPU resources, enabling faster processing. Access to GPU resources on Kaggle required account registration and phone verification. Additionally, datasets not originally available on the platform were manually uploaded to support the training process. Once the datasets were in place, training was conducted for both the facial recognition and checklist models.

For mobile application development, Android Studio was used as the primary IDE. An emulator was configured using the Pixel 6 profile with Android API level 28 (Android 11) to support debugging and testing. This emulator enabled the evaluation of features such as on-device model inference. To assess the facial recognition

model's performance using TFLite with external image sources, the images were inserted into the emulator environment beforehand.

5.4 System Operation

Model Training

This section outlines the operational flow of the system, including the model training process, integration into the mobile application, and the functioning of the mobile app itself.

The facial recognition model was trained using a variety of settings and configurations to achieve optimal performance. The training process followed the steps outlined in Section 3.3.1.1 and was based on the architecture described in Section 3.3.1.2. After thorough evaluation, the best-performing models were selected and converted into TFLite format for deployment on mobile devices.

The checklist-based model was trained once following the procedures in Section 3.3.2.1, and its design adhered to the architecture illustrated in Section 3.3.2.2. Upon completion, the model was also converted into TFLite format for integration. This concludes the model training and preparation phase.

Splash Screen

On the mobile application side, when users launch the app, a splash screen is displayed (as shown in Figure 5.4.1). Splash screen provides time for the application to initialise necessary services, such as checking authentication status, and fetching essential data from Firebase before navigating to the main interface.



Figure 5. 4. 1 Splash Screen

Register and Login Module

After the splash screen disappears, users are directed to the Register and Login page, as shown in Figure 5.4.2. When the “Sign Up” button is pressed, users are redirected to the Sign-Up page (Figure 5.4.3), where they are required to fill in all necessary details, including first name, last name, email address, gender, date of birth, phone number, and password, and agree to the terms and conditions.

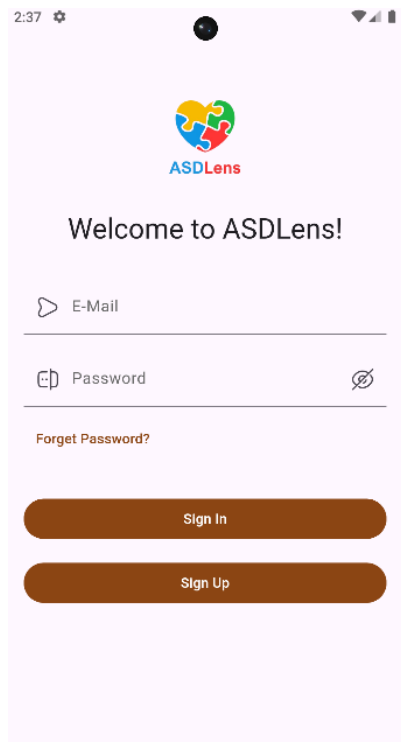


Figure 5. 4. 2 Register and Login Page

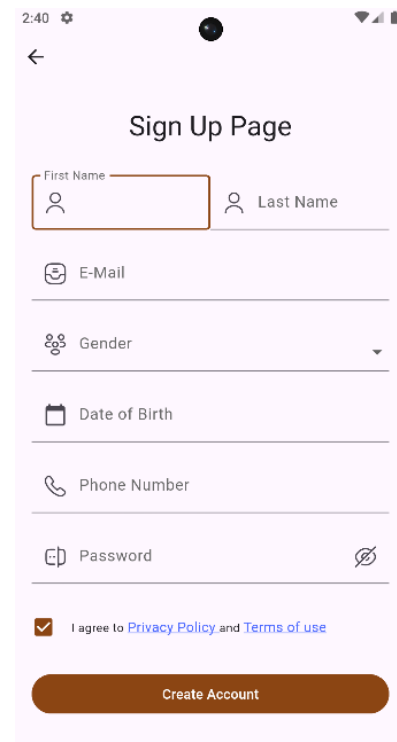


Figure 5. 4. 3 Sign Up Page

The gender field is presented as a drop-down selection (Figure 5.4.4), while the date of birth is selected using a date picker (Figure 5.4.5). The password field includes an eye icon that allows users to toggle between plain text and obscured text, as illustrated in Figure 5.4.6 and Figure 5.4.7 respectively.

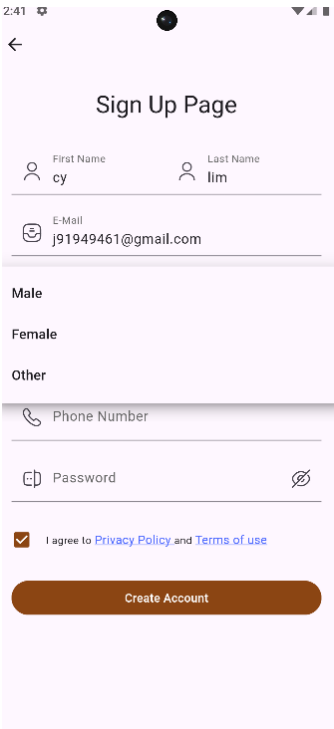


Figure 5. 4. 4 Dropdown Selection

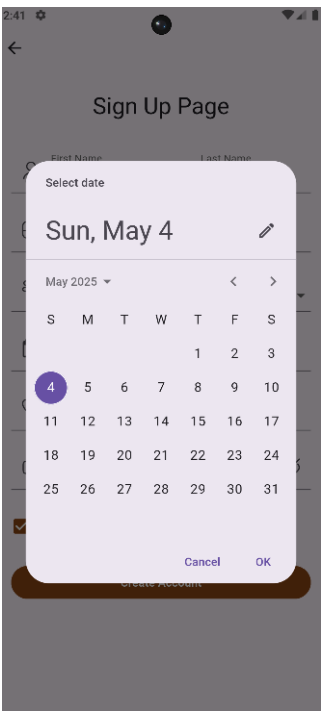


Figure 5. 4. 5 Data Picker Selection

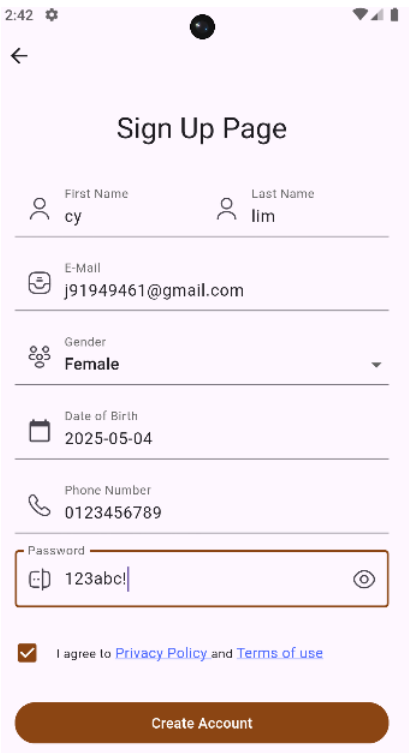


Figure 5. 4. 6 Password in plaintext (Sign Up Page)

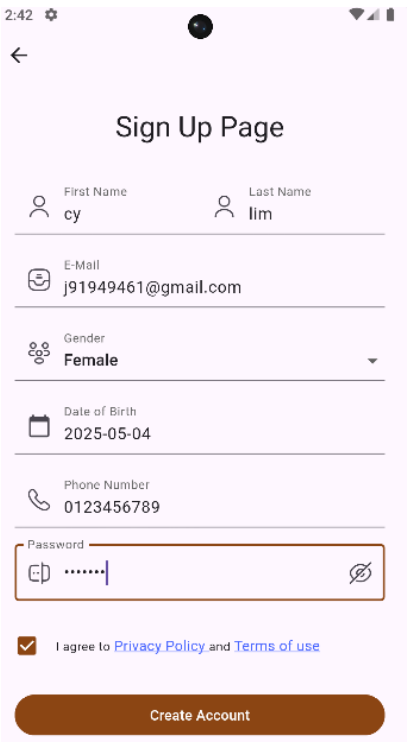


Figure 5. 4. 7 Password in obscured text (Sign Up Page)

Upon clicking the “Create Account” button, users are directed to a new page prompting them to verify their email address (Figure 5.4.8). They must check their Gmail inbox and click on the verification link (Figure 5.4.9). Once verified, a confirmation message will be shown (Figure 5.4.10).

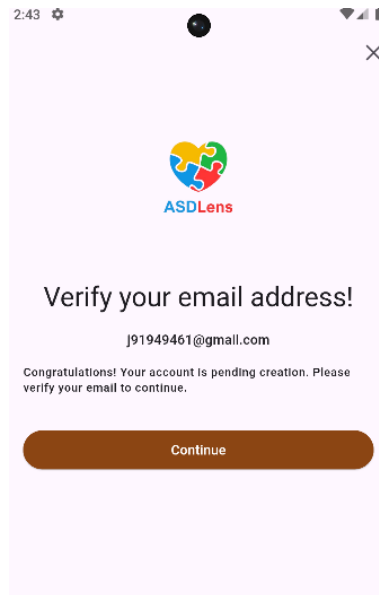


Figure 5. 4. 8 Verifying email address page

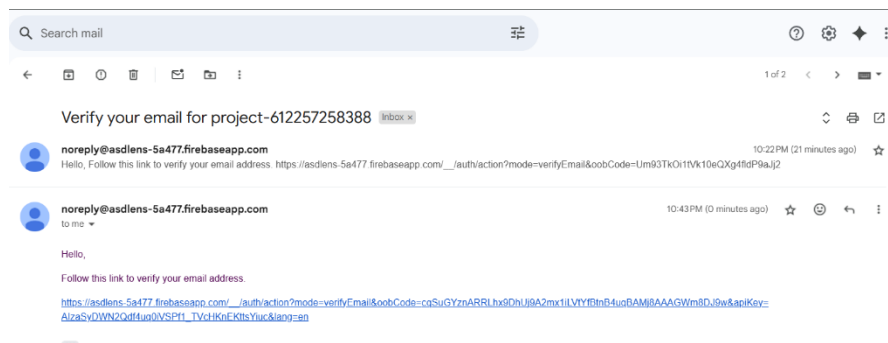


Figure 5. 4. 9 Gmail inbox with verification link

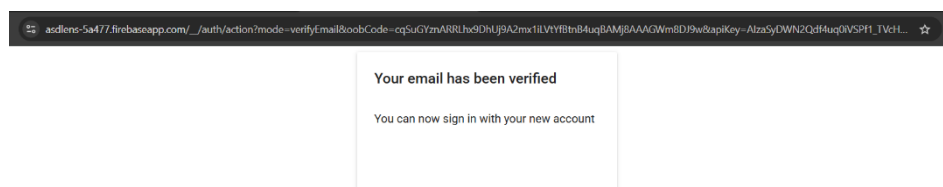


Figure 5. 4. 10 Confirmation message

Users can then return to the application, where a confirmation page stating that the account has been successfully created will be displayed (Figure 5.4.11). After clicking the “Continue” button, they will be redirected to the home page.

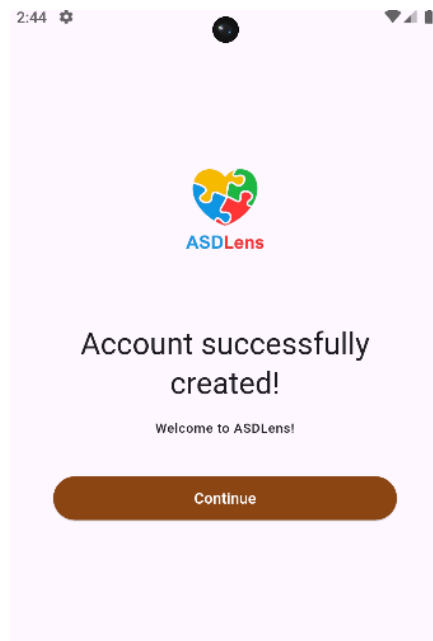


Figure 5. 4. 11 Conformation Page

When users click on “Forget Password”, they will be redirected to a page where they can enter their Gmail address to recover their account (Figure 5.4.12). After clicking the “Submit” button, a new page will show that a password reset email has been sent (Figure 5.4.13). Users can then go to their Gmail account and click on the link provided (Figure 5.4.14), after which they will be prompted to enter a new password (Figure 5.4.15). Once the password has been changed, a password changed message will be displayed (Figure 5.4.16). Users can now return to the mobile application, click on 'Done', and they will be directed to the Register and Login page, where they can sign in using their new password.

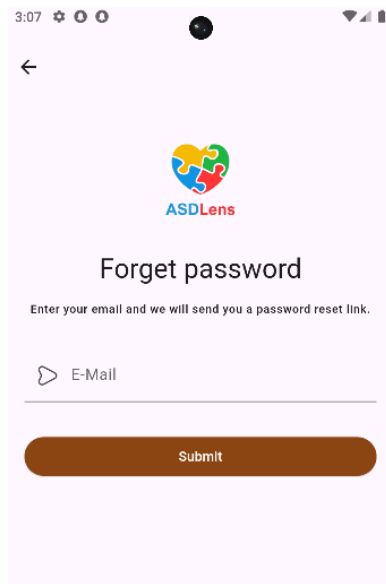


Figure 5.4.12 Page for entering Gmail to reset password

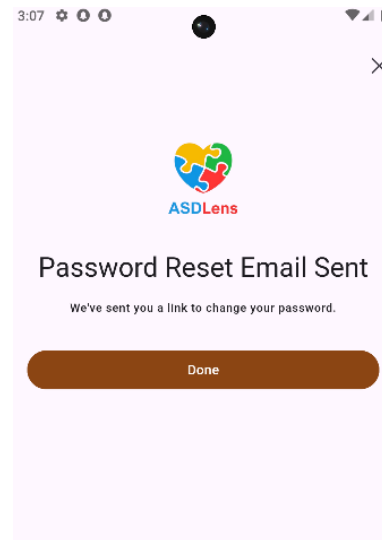


Figure 5.4.13 Page showing password reset Gmail sent

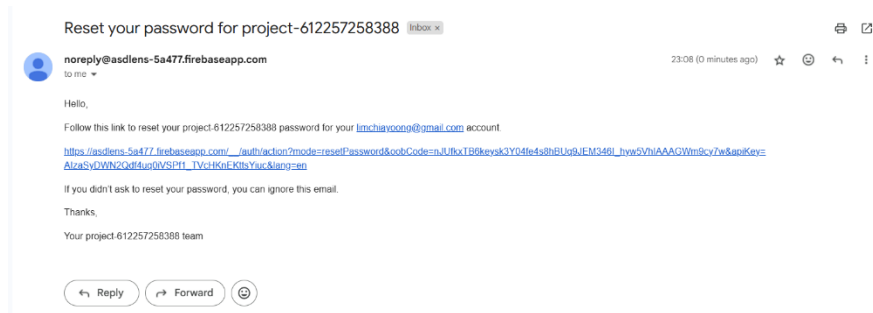


Figure 5.4.14 Password reset link in Gmail

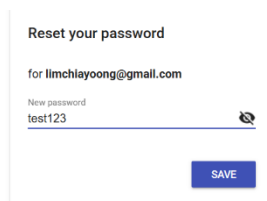


Figure 5.4.15 Reset new password

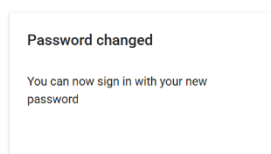
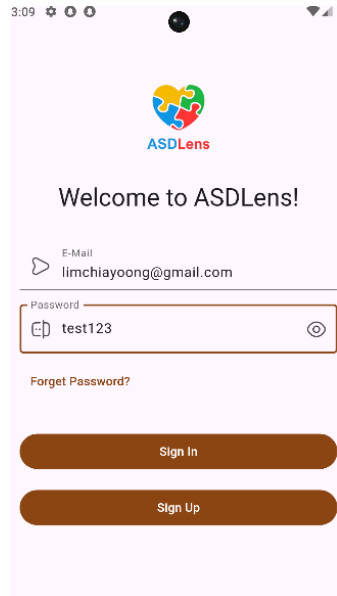
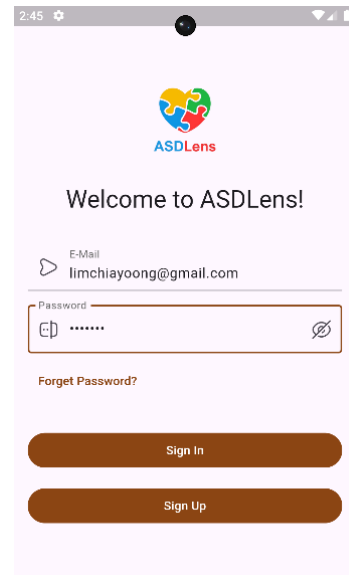


Figure 5.4.16 Message showing password changed

For the standard sign-in process, users can enter their Gmail address and password. The password field includes an eye icon that allows users to toggle between plain text and obscured text, as illustrated in Figure 5.4.17 and Figure 5.4.18. After entering the correct credentials, users will be redirected to the home page.



*Figure 5. 4. 17 Password in plaintext
(Sign in Page)*



*Figure 5. 4. 18 Password in obscured text
(Sign in Page)*

Facial Recognition Module

When the “Facial” button located on the bottom navigation bar is clicked, users are redirected to the Facial Recognition Page, as illustrated in Figure 5.4.19. By selecting “Take a Photo,” users are directed to their phone’s camera interface (Figure 5.4.20), where they can capture an image and confirm it by tapping the “✓” button (Figure 5.4.21). Once the image is processed, the system displays the result with the predicted label and confidence score (Figure 5.4.22). Clicking “Close” will return the user to the Home Page.

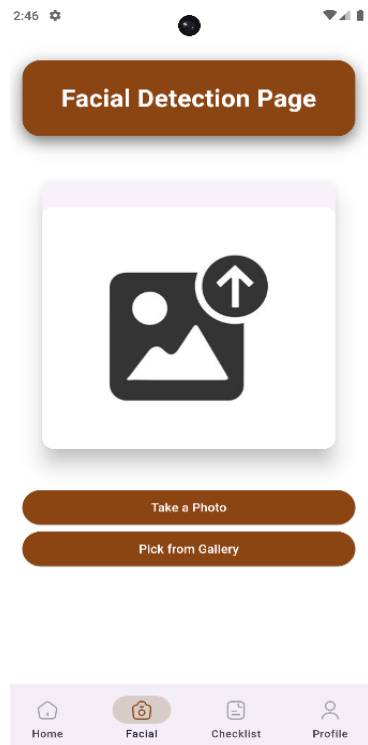


Figure 5. 4. 19 Facial Recognition Page

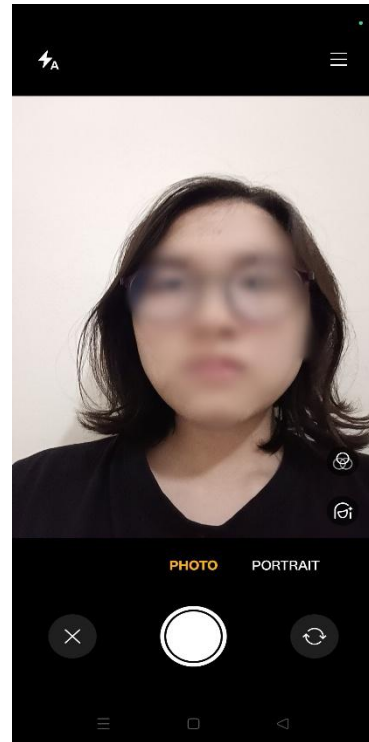


Figure 5. 4. 20 Phone's Camera Interface



Figure 5. 4. 21 Confirming captured image by tapping “✓”

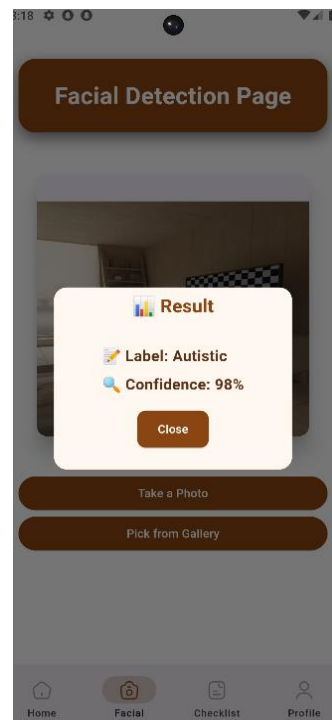


Figure 5. 4. 22 Result with predicted label and confidence score (Take Photo)

Alternatively, if users select “Pick from Gallery,” they will be redirected to their device’s gallery (Figure 5.4.23) to choose a photo. Once a photo is selected, the result will similarly be shown in terms of label and confidence score (Figure 5.4.24). Tapping “Close” again redirects users to the Home Page.

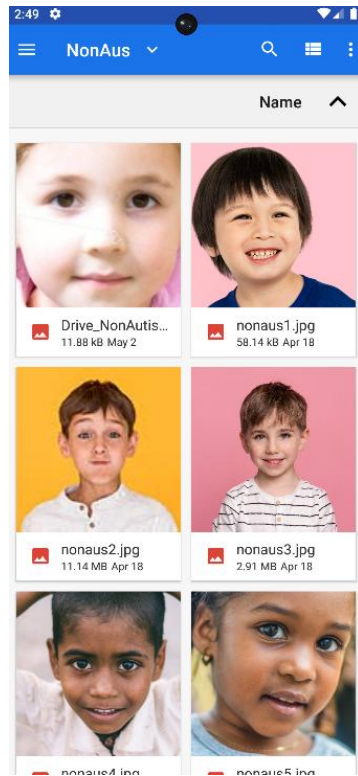


Figure 5. 4. 23 Device’s Gallery Interface

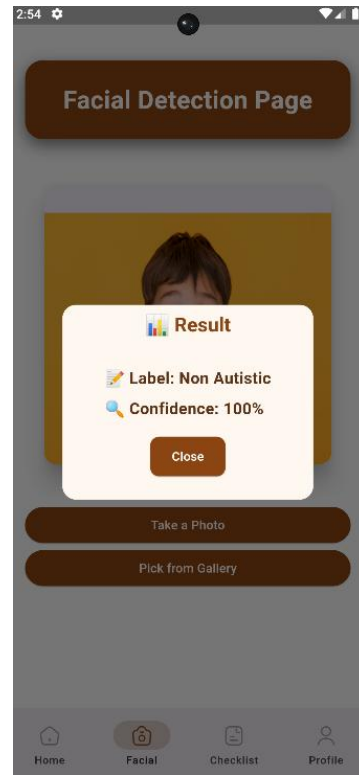


Figure 5. 4. 24 Result with predicted label and confidence score (Upload from gallery)

Regarding the notification system, users will be reminded to retake the test if their latest submission is older than three months. An example of the facial recognition's notification pop-up is shown in Figure 5.4.25, and its expanded view is shown in Figure 5.4.26.

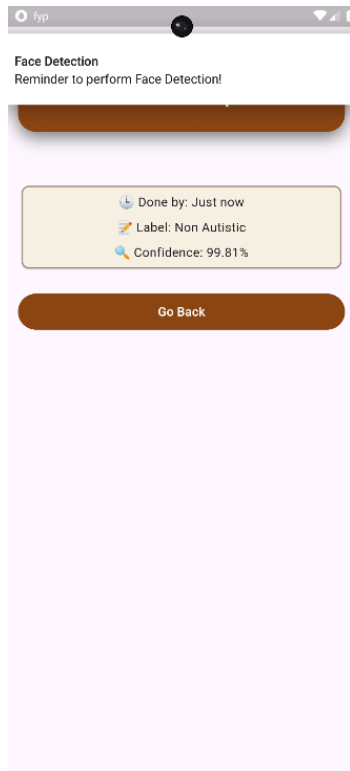


Figure 5. 4. 25 Facial recognition's notification pop-up

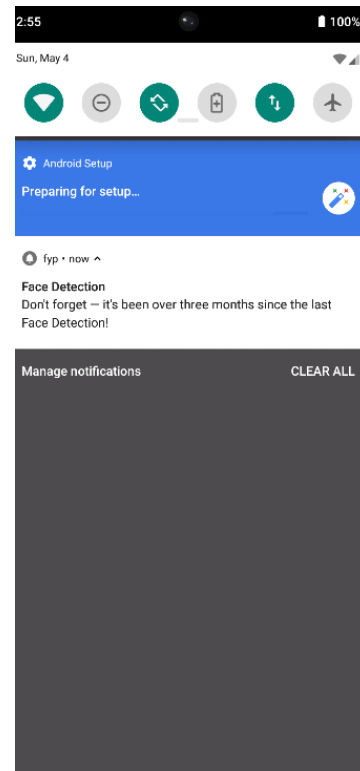


Figure 5. 4. 26 Facial recognition's notification's expanded view

Checklist Module

When the “Checklist” button located on the bottom navigation bar is clicked, users are redirected to the Questionnaire Page, as illustrated in Figure 5.4.27. Except for the first and last questions, all intermediate questions include “Previous” and “Next” buttons to allow users to navigate between questions (Figure 5.4.28). For the final question, users are presented with two options: return to the previous question to amend their response or submit the questionnaire (Figure 5.4.29). Upon clicking the “Submit” button, the result is displayed in terms of the predicted label and confidence score (Figure 5.4.30). Selecting “Close” will return the user to the Home Page.

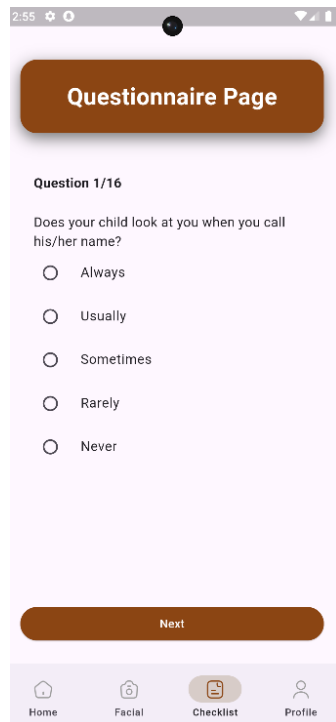


Figure 5. 4. 27 Questionnaire Page

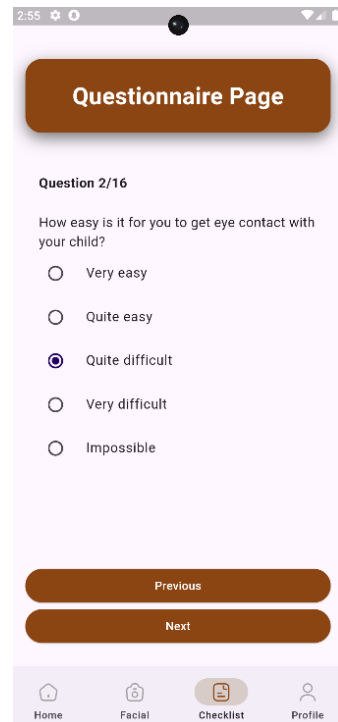


Figure 5. 4. 28 Intermediate questions include “Previous” and “Next” buttons

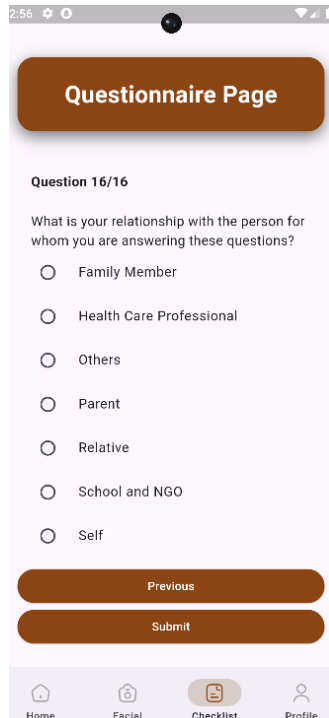


Figure 5. 4. 29 Last questions include “Previous” and “Submit” buttons

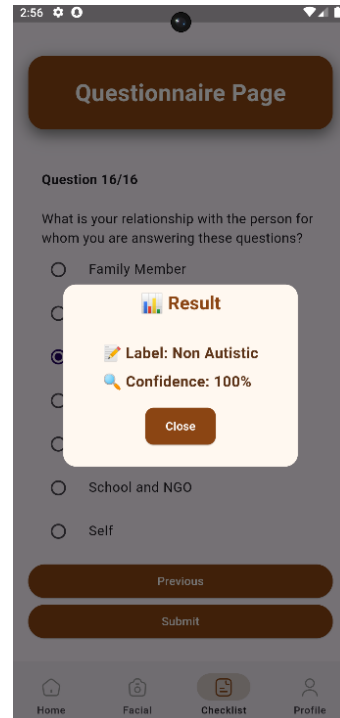


Figure 5. 4. 30 Result with predicted label and confidence score (Questionnaire)

Regarding the notification system, users will be reminded to retake the test if their latest submission is older than three months. For checklist-based reminders, the pop-up and expanded views are illustrated in Figure 5.4.31 and Figure 5.4.32, respectively.

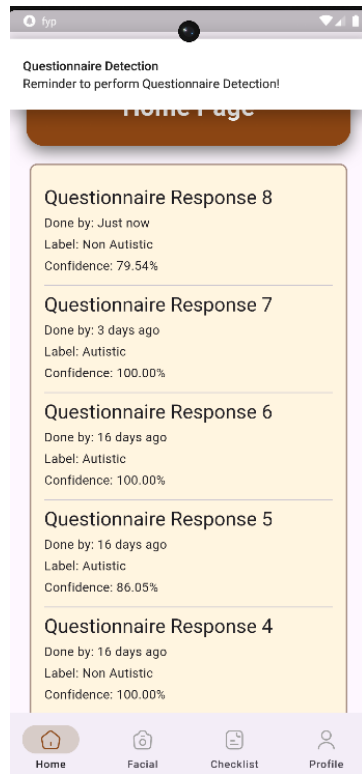


Figure 5. 4. 31 Questionnaire's notification pop-up

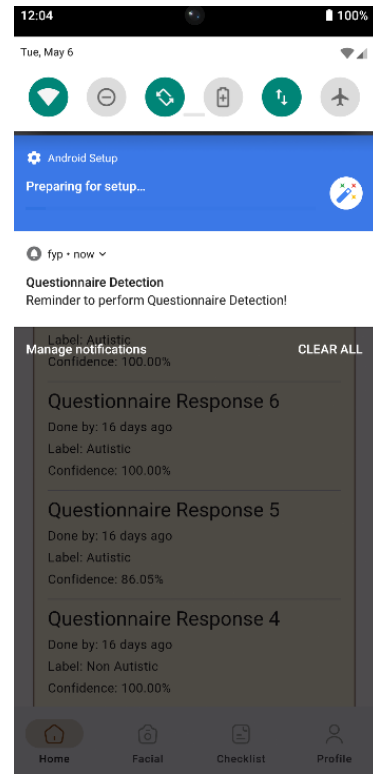


Figure 5. 4. 32 Questionnaire's notification's expanded view

Main Page Module

When users first access the Home Page, no data is displayed if no previous records exist, as shown in Figure 5.4.33. Once users complete either facial recognition or checklist detection, the corresponding results are immediately reflected on the Home Page. The upper section presents a summarised view of the checklist responses, while the lower section displays a summary of the image-based detections. These results are arranged in reverse chronological order, with the most recent entries at the top and the oldest at the bottom (Figure 5.4.34).

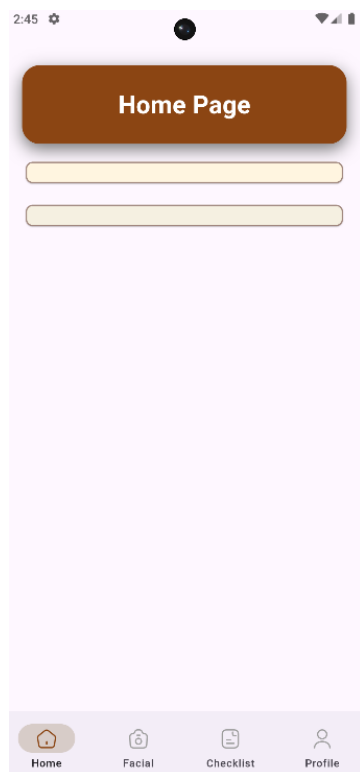


Figure 5. 4. 33 Home page with no previous records

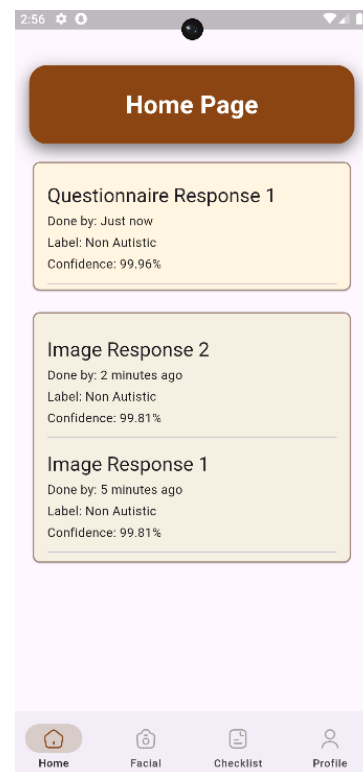


Figure 5. 4. 34 Home page with previous records

Each summarised section is interactive. Tapping on a questionnaire response navigates the user to a detailed view, where they can review the previously answered questions along with the selected options (Figure 5.4.35). The detection result is also displayed at the bottom of this view (Figure 5.4.36). Users can return to the Home Page by tapping the “Go Back” button.

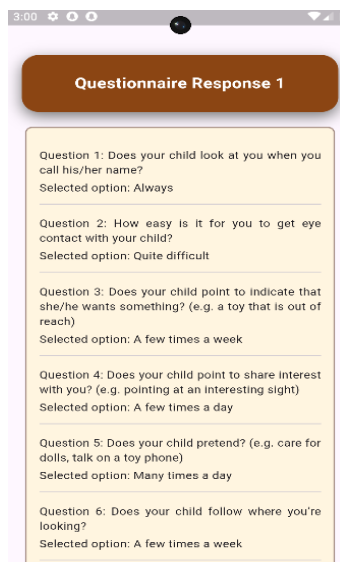


Figure 5. 4. 35 Questionnaire Response Page

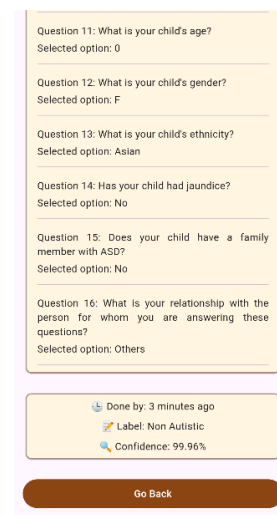


Figure 5. 4. 36 Questionnaire Response Page with Go Back button

Similarly, tapping on an image response opens a detailed view that includes the detection result (Figure 5.4.37). Users may also return to the Home Page by selecting the “Go Back” button.

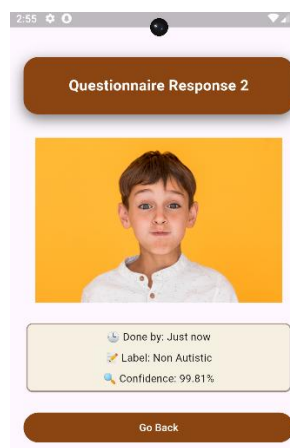


Figure 5. 4. 37 Facial Recognition Response Page with Go Back button

Profile Module

When users click the “Profile” button on the bottom navigation bar, they are redirected to the Profile Page, as illustrated in Figure 5.4.38. If they tap the “>” icon next to the name, they will be taken to a page where they can edit their first and last names (Figure 5.4.39). Similarly, tapping the “>” icon next to the phone number will open another page allowing them to update their phone number (Figure 5.4.40). After making any changes and clicking “Save”, users are redirected back to the Profile Page, where the updated information is reflected immediately.

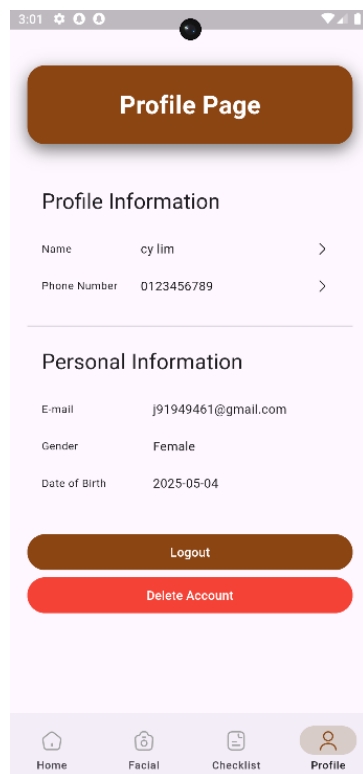


Figure 5. 4. 38 Profile Page

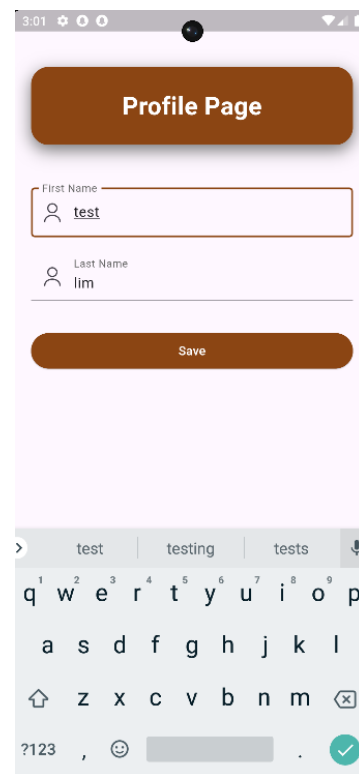


Figure 5. 4. 39 Editing Name Page

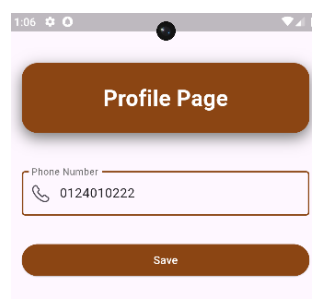


Figure 5. 4. 40 Editing Phone Number Page

Users may also click the “Logout” button (Figure 5.4.41), which redirects them to the Register and Login page.

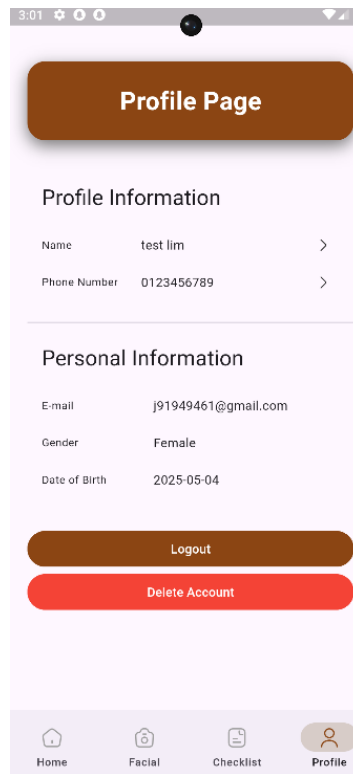


Figure 5. 4. 41 Logout Button in Profile Page

If users choose to delete their account by clicking “Delete Account”, a confirmation prompt will appear to ensure the user intends to proceed (Figure 5.4.42). If confirmed by clicking “Delete”, they will be asked to re-authenticate by entering their email and password (Figure 5.4.43). The password field includes an eye icon that allows users to toggle between plain text and obscured text (Figure 5.4.44). After verification, the account will be deleted, and the user will be redirected to the Register and Login page.

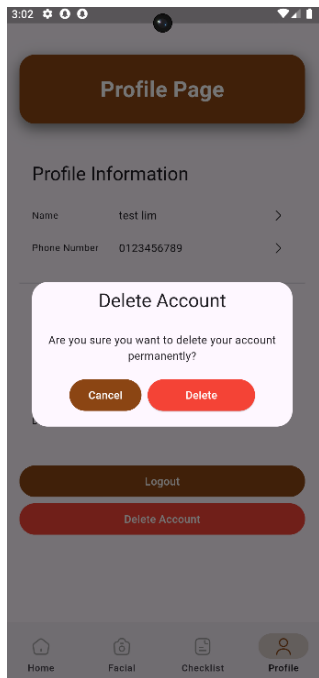


Figure 5. 4. 42 Delete account's confirmation prompt

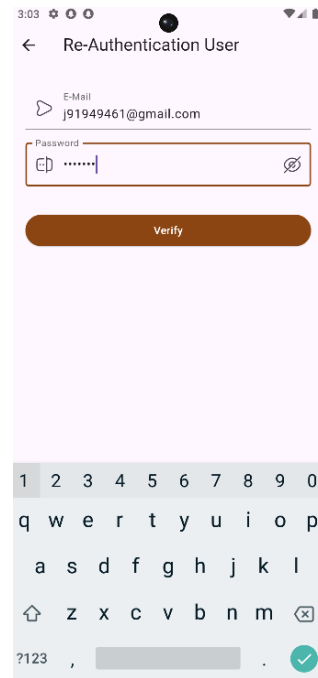


Figure 5. 4. 43 Password in plaintext (Re-authenticate Page)

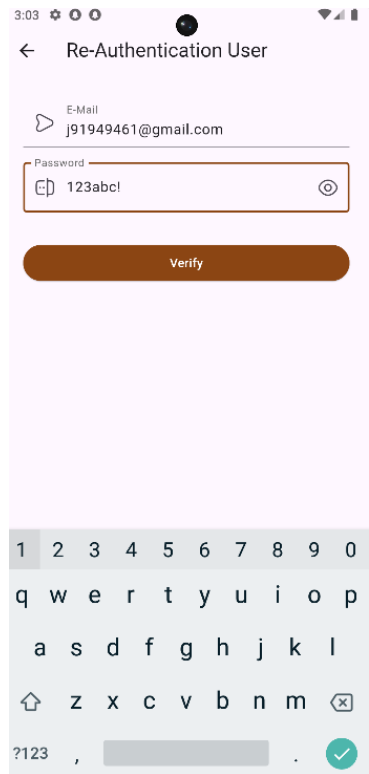


Figure 5. 4. 44 Password in obscured text (Re-authenticate Page)

5.5 Implementation Issues and Challenges

Integrating the trained model into the mobile application presented significant challenges. One major issue was ensuring that the preprocessing steps applied during model training, particularly input normalization using functions like `preprocess_input`, were consistently implemented on the mobile side. Any mismatch in preprocessing between training and deployment environments could lead to a notable drop in facial recognition accuracy. Furthermore, evaluating the model's performance directly on the mobile platform was time-intensive, as GPU acceleration is not available, making inference slower and extending overall development timeline.

Another challenge involved translating the conceptual design of the user interface into a functional implementation. While the initial UI ideas were clear at the design level, turning those ideas into actual code proved difficult. This required repeatedly referencing external resources, experimenting with different layout techniques, and adjusting for limitations within the development framework. Achieving a visually intuitive and user-friendly interface demanded significant trial and error and extended development time.

In addition, managing software version compatibility was an ongoing obstacle throughout the project. Variations in library and tool versions often led to conflicts, particularly when updating dependencies, which could unexpectedly break existing functionality. These issues were encountered in both the Python environment used for model development and the Dart environment used for mobile application development. Resolving such compatibility issues required careful version control, frequent debugging, and a deep understanding of the dependencies involved.

5.6 Concluding Remark

All in all, the actual implementation serves as the backbone of project development, transforming conceptual designs and theoretical plans into a functional solution. This phase involves careful consideration of various factors, including model integration, interface design, and software component compatibility. Although several challenges arose during the implementation process, they were effectively managed through persistence and problem-solving. These experiences not only helped in identifying appropriate solutions but also played a crucial role in enhancing the overall quality and success of the project.

Chapter 6 System Evaluation and Discussion

This chapter presents the performance evaluation of both the facial recognition and checklist-based models. It also includes the test case setup for the mobile application, a discussion of the challenges encountered during the project, and a re-evaluation of the project objectives.

6.1 System Performance Metrics

6.1.1 Facial Recognition Model Performance Metric

The primary performance metric used in this project is validation accuracy, which plays a key role in determining the best model configuration during the training phase. Validation accuracy helps assess how well the model generalises to unseen data and serves as the main reference for deciding which model settings to proceed with. Once the training is complete and the most suitable models are finalised, further evaluations are carried out. These include testing accuracy using the Keras framework, accuracy after conversion to the TFLite format, and TFLite inference accuracy using external images within the mobile application environment. These assessments ensure that the selected models maintain reliable performance not only in development but also after deployment on the mobile platform.

6.1.2 Checklist-Based Model Performance Metric

As this approach does not involve extensive tuning, the performance metrics are evaluated directly on the test set. The primary focus here is to explore the differences in model accuracy between the Keras and TFLite formats.

6.1.3 Mobile Application Performance Metric

Black box testing will be used to evaluate the performance of the mobile application. It will cover all the features and functionalities of the app without considering the internal workings, focusing instead on the output generated for various inputs and scenarios.

6.2 Testing Setup and Result

6.2.1 Facial Recognition Model Performance Result

6.2.1.1 Model Architecture Exploration Metrics

Following the steps mentioned in Section 3.3.3.1, the first evaluation was conducted through model architecture comparison. Various pre-trained models, namely MobileNetV1, MobileNetV2, Xception, VGG16, and EfficientNetB0, were assessed based on their validation accuracy.

From the results shown in the table 6.2.1.1.1, we observed that the Xception model achieved the highest validation accuracy at 80.15%, indicating its superior ability to extract and learn relevant features from the dataset. The second-best performer was MobileNetV1, followed closely by MobileNetV2, with accuracies of 79.40% and 79.03% respectively. These models performed similarly due to their lightweight yet efficient depthwise separable convolutions, which help maintain good performance with fewer parameters.

Table 6. 2. 1. 1 Validation Accuracy Comparison of Model Architectures

Model	Keras Validation Accuracy
Xception	80.15%
MobileNetV1	79.40%
MobileNetV2	79.03%
EfficientNetB0	76.78%
VGG16	72.66%

EfficientNetB0 achieved a validation accuracy of 76.78%, which is lower likely because its compound scaling approach may require more data to fully optimize its performance. Lastly, VGG16 had the lowest accuracy at 72.66%, possibly because of its large number of parameters and relatively outdated architecture, making it less efficient and prone to overfitting on smaller datasets.

Based on this comparison, the Xception model was selected to proceed for further evaluation and fine-tuning due to its superior performance.

6.2.1.2 Preprocessing Exploration Metrics

A After confirming the Xception model as the best-performing architecture, the first preprocessing technique examined was face cropping. A comparison was conducted between models trained with and without face cropping using the Dlib library, aiming to assess whether focusing solely on facial regions improves learning performance.

According to Table 6.2.1.2.1, the model trained on uncropped faces achieved a higher validation accuracy of 80.15%, whereas the cropped-face version showed a lower accuracy of 76.78%. These results suggest that the cropping process may have introduced noise or removed useful contextual information surrounding the face, thereby reducing the model's effectiveness. Therefore, using uncropped face images appears to provide more discriminative features for classification in this scenario.

Table 6. 2. 1. 2. 1 Validation Accuracy Comparison of Applying Cropping

Preprocessing Technique	Keras Validation Accuracy
Uncropped faces	80.15%
Cropped faces	76.78%

Following the use of uncropped face images, the Gaussian blur technique was explored as a preprocessing method. Based on Table 6.2.1.2.2, the model without Gaussian blur applied achieved a higher validation accuracy of 80.15%, while the application of Gaussian blur resulted in a lower accuracy of 76.78%. This suggests that applying Gaussian blur may have smoothed out important facial features and reduced edge sharpness, thereby limiting the model's ability to capture fine-grained details necessary for accurate classification.

Table 6. 2. 1. 2. 2 Validation Accuracy Comparison of Applying Gaussian Blur

Preprocessing Technique	Keras Validation Accuracy
Not applied Gaussian Blur	80.15%
Applied Gaussian Blur	76.78%

Next, CLAHE was applied to explore the impact of contrast enhancement on model performance. Based on Table 6.2.1.2.3, the model without CLAHE applied achieved a higher validation accuracy of 80.15%, while the model with CLAHE

applied showed a lower accuracy of 74.16%. This suggests that CLAHE may have introduced excessive contrast or amplified noise, which could interfere with the model's ability to focus on meaningful facial features.

Table 6. 2. 1. 2. 3 Validation Accuracy Comparison of Applying CLAHE

Preprocessing Technique	Keras Validation Accuracy
Not applied CLAHE	80.15%
Applied CLAHE	74.16%

Thus, the input images without applying cropping, Gaussian Blur, or CLAHE yielded the best results.

6.2.1.3 Data Augmentation Exploration Metrics

To proceed further, an exploration was conducted on the effect of data augmentation. Three configurations were evaluated: without applying data augmentation, applying data augmentation once, and applying data augmentation with a doubled dataset size.

An interesting observation emerged: since the model was saved based on the best accuracy achieved across epochs, both the single and double data augmentation settings reached a validation accuracy of 100%, as shown in Table 6.2.1.3.1. This prompted a deeper analysis of the training and validation accuracy curves across epochs.

Table 6. 2. 1. 3. 1 Validation Accuracy Comparison of Applying Data Augmentation

Data Augmentation	Keras Validation Accuracy
Not data augmentation	80.15%
Applied one data augmentation	100%
Applied double data augmentation	100%

As illustrated in Figure 6.2.1.3.1, the training and validation accuracy for the model without data augmentation increased smoothly and steadily. In contrast, Figures 6.2.1.3.2 and 6.2.1.3.3, representing models with single and double data augmentation, respectively, displayed high volatility, where accuracy fluctuated sharply between epochs, ranging from as low as 60% to as high as 100%. This

CHAPTER 6

instability, where some epochs achieved perfect accuracy and others dropped significantly, indicates potential overfitting or training inconsistency.

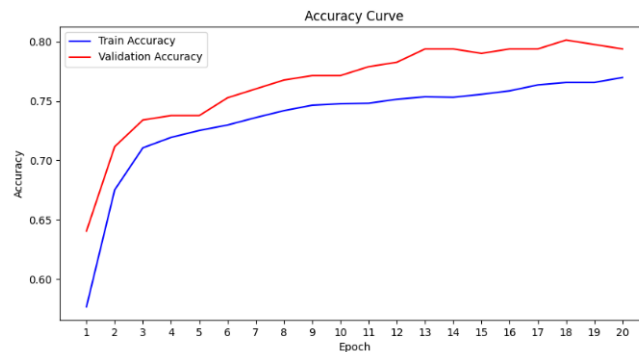


Figure 6. 2. 1. 3. 1 Validation accuracy of not applying data augmentation

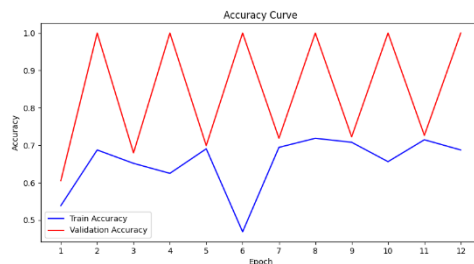


Figure 6. 2. 1. 3. 2 Validation accuracy of applying one data augmentation

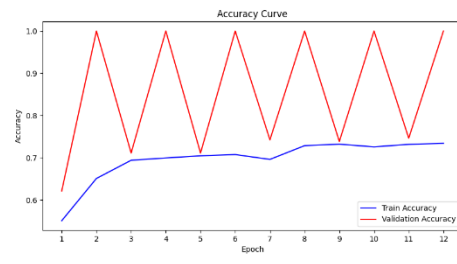


Figure 6. 2. 1. 3. 3 Validation accuracy of applying twice data augmentation

Therefore, despite the peak values, the configuration without data augmentation was considered more stable and reliable for further development.

6.2.1.4 Hyperparameter Tuning Exploration Metrics

The first tuning metric explored was batch size, as it directly influences model performance and convergence behaviour. Three batch sizes 16, 32, and 64, were evaluated.

According to Table 6.2.1.4.1, the batch size of 32 achieved the highest validation accuracy at 80.15%, followed by batch size 16 with 79.78%, while the largest batch size of 64 resulted in the lowest accuracy at 79.03%. This outcome may be explained by the fact that smaller batch sizes provide more frequent updates,

introducing some noise that can help the model generalise better. In contrast, larger batch sizes may reduce the model's generalisation capability due to smoother gradient updates and reduced variability during training. The batch size of 32 offers an effective balance between learning stability and generalisation performance.

Table 6. 2. 1. 4. 1 Validation Accuracy Comparison of Batch Size

Batch Size	Keras Validation Accuracy
16	79.78%
32	80.15%
64	79.03%

The next step in hyperparameter tuning involved adjusting both the learning rate and the optimizer simultaneously, as different optimizers typically perform best with different learning rates. From the experimental results shown in Table 6.2.1.4.2, the SGD optimizer with a learning rate of 1e-2 achieved the highest validation accuracy of 83.15%. This moderate learning rate outperformed both the higher (1e-1) and lower (1e-3) learning rates for SGD, possibly because it provided a balanced trade-off between convergence speed and stability, avoiding the overshooting of large steps and the slow learning of very small steps.

Table 6. 2. 1. 4. 2 Validation Accuracy Comparison of Optimizer and Learning Rate

Optimizer	Learning Rate	Keras Validation Accuracy
Adam	1e-3	82.77%
Adam	1e-4	80.15%
Adam	1e-5	70.41%
AdamW	1e-3	82.77%
AdamW	1e-4	80.15%
AdamW	1e-5	70.41%
SGD	1e-1	80.15%
SGD	1e-2	83.15%
SGD	1e-3	81.65%

When examining the Adam and AdamW optimizers, both exhibited identical performance across the tested learning rates, suggesting that the two optimizers behave similarly in this context. Their best performance was observed at a learning

rate of $1e-3$, achieving a validation accuracy of 82.77%. However, this still fell slightly below the peak performance of SGD, potentially due to Adam and AdamW adapting learning rates during training, which can sometimes make it harder for the model to settle on the best solution. In contrast, SGD uses a fixed learning rate, which can help the model learn more steadily and consistently.

The SGD optimizer includes a parameter called “Nesterov momentum”, which improves the update process by looking ahead at the future position of the parameters based on the current gradient. This feature was explored to evaluate its effect on validation accuracy.

According to Table 6.2.1.4.3, enabling Nesterov momentum (set to True) resulted in a higher validation accuracy of 83.52%, compared to 83.15% when it was disabled. This improvement is likely due to the way Nesterov momentum helps the optimizer make more accurate and stable updates by anticipating the direction of the gradient, thereby enhancing learning efficiency.

Table 6. 2. 1. 4. 3 Validation Accuracy Comparison of Applying “nesterov”

Optimizer	Learning Rate	“nesterov”	Keras Validation Accuracy
SGD	$1e-2$	False	83.15%
SGD	$1e-2$	True	83.52%

As a result, SGD with a learning rate of $1e-2$ and Nesterov momentum enabled is selected, as it provides the highest validation accuracy observed in this tuning process.

6.2.1.5 Model Architecture Modifications Exploration Metrics

The first architectural modification explored is the addition of a Dropout layer, which temporarily disables a fraction of neurons during training. This helps prevent overfitting by encouraging the model to learn more robust and generalisable features.

In this experiment, dropout rates of 0.2, 0.3, and 0.4 were evaluated and compared against a model with no Dropout layer. The results are presented in Table 6.2.1.5.1.

Table 6. 2. 1. 5. 1 Validation Accuracy Comparison of Applying Dropout

Dropout	Keras Validation Accuracy
No dropout	83.52%
Dropout (p=0.2)	83.15%
Dropout (p=0.3)	84.64%
Dropout (p=0.4)	82.77%

From the table, the Dropout rate of 0.3 produced the highest validation accuracy of 84.64%, suggesting that a moderate dropout rate can improve generalisation. This was followed closely by the model with no Dropout, which achieved 83.52%. Dropout rates of 0.2 and 0.4 resulted in slightly lower accuracies, possibly due to insufficient or excessive regularisation, which may hinder optimal learning.

The next architectural modification explored is the addition of a batch normalization layer. This technique is commonly used to stabilise and accelerate training by normalising the inputs to each layer, potentially allowing the model to converge faster and with improved performance.

However, based on the experimental results shown in Table 6.2.1.5.2, the model without batch normalization achieved the highest validation accuracy of 84.64%, compared to 82.02% when batch normalization was applied. This suggests that batch normalization may have disrupted the learning process or introduced unnecessary regularisation, leading to a slight drop in performance.

Table 6. 2. 1. 5. 2 Validation Accuracy Comparison of Applying Batch Normalization

Batch Normalization	Keras Validation Accuracy
Not batch normalization	84.64%
Applied batch normalization	82.02%

Therefore, Therefore, a Dropout layer with a rate of 0.3 and the configuration without batch normalization will be retained for the next stages of model refinement.

6.2.1.6 Fine Tuning Modifications Exploration Metrics

After applying all previously determined optimal settings, the model was loaded for further fine-tuning of layer weights. Four approaches were evaluated:

keeping all layers fully frozen, unfreezing only the exit flow (Blocks 13 and 14), unfreezing both the middle and exit flow (Blocks 11–14), and fully unfreezing all layers. Since fine-tuning generally benefits from a smaller learning rate, multiple learning rates were tested alongside each strategy.

As shown in Table 6.2.1.6.1, most fine-tuning strategies outperformed the fully frozen model. This suggests that unfreezing certain layers allows the model to learn more refined features, improving overall performance. Notably, fully unfreezing the model with a learning rate of 1e-3 achieved the highest validation accuracy of 87.26%, likely due to improved convergence and effective feature refinement across the entire network.

Table 6. 2. 1. 6. 1 Validation Accuracy Comparison of Applying Fine Tuning

Fine tuning approach	Learning Rate	Keras Validation Accuracy
Fully freeze	1e-2	84.64%
Unfreeze exit flow	1e-2	85.39%
Unfreeze exit flow	1e-3	86.14%
Unfreeze middle and exit flow	1e-2	86.14%
Unfreeze middle and exit flow	1e-3	86.14%
Fully unfreeze	1e-3	87.26%
Fully unfreeze	1e-4	86.14%
Fully unfreeze	5e-4	85.39%

When comparing partial fine-tuning strategies, both unfreezing only the exit flow and unfreezing the middle plus exit flow resulted in 86.14% accuracy with a learning rate of 1e-3, indicating minimal performance differences between the two.

As all fine-tuning approaches demonstrated competitive performance, including the fully frozen model, the corresponding models will be evaluated on the test set and converted into TFLite format to assess their performance on a mobile application.

6.2.1.7 Final Model Evaluation on Test Set and Mobile Deployment

These four models were evaluated on the test set in both Keras and TFLite formats. As shown in Table 6.2.1.7.1, converting models to TFLite generally results

in a decrease in accuracy. This is likely because TFLite is optimised for lower computational resources, prioritising faster inference on mobile devices. The ranking of the results is broadly consistent with the validation accuracy, with the fully unfrozen model achieving the highest test accuracy, which is 82.77% in Keras and 78.04% in TFLite. The second-highest performance was achieved by unfreezing only the exit flow, followed by the fully frozen model.

Table 6. 2. 1. 7. 1 Accuracy Comparison of Test Set

Fine-tuning	Keras accuracy	TFLite accuracy
Fully freeze	77.02%	68.92%
Unfreeze exit flow	80.41%	72.97%
Unfreeze middle and exit flow	79.73%	50%
Fully unfreeze	82.77%	78.04%

Notably, while the unfreeze middle and exit flow approach achieved slightly better accuracy than the fully frozen model in Keras, it performed poorly in the TFLite format, with only 50% accuracy. This may be due to the increased complexity of the model after partial unfreezing, which could lead to unstable behaviour or loss of precision during TFLite conversion.

In addition to the test set results, external images from “Freepik” and “Ready Kids” were used to further evaluate the model’s performance. These external sources consisted of 5 images of autistic children and 5 images of non-autistic children. The overall accuracy and the models’ ability to predict both autistic and non-autistic classes for these ten images are presented in Table 6.2.1.7.2. The fully unfrozen model achieved the highest accuracy on the external dataset, which is 90%, misclassifying only one image. The other three models performed identically, each achieving 70% accuracy.

Table 6. 2. 1. 7. 2 Accuracy Comparison of External Images

Fine-tuning	TFLite accuracy	Predicting Autistic Performance	Predicting Non-Autistic Performance
Fully freeze	70%	All predicted correctly	2 correctly predicted
Unfreeze exit flow	70%	All predicted correctly	2 correctly predicted

Unfreeze middle & exit flow	70%	All predicted correctly	2 correctly predicted
Fully unfreeze	90%	All predicted correctly	4 correctly predicted

An interesting observation is that all models successfully predicted all autistic images but struggled more with the non-autistic ones. This may be due to the models having learned noise or less distinctive features for the non-autistic class during training, which affected their ability to generalise to unseen non-autistic examples.

6.2.2 Checklist-Based Model Performance Result

The checklist-based model achieved the same accuracy of 97.60% for both Keras and TFLite versions. This is because the checklist approach is simple and does not rely on learning complex patterns like image-based models. Instead, it uses fixed rules or conditions based on consistent statistical calculations, such as mean and standard deviation, ensuring both models work the same way.

6.2.3 Mobile Application Setup and Performance Result

Table 6. 2. 3. 1 App Launch Test Case

Test Case Name: App Launch			Test Case ID: T001	
	Test Steps	Action	Expected Result	Result Status
Main Flows	1	System displays splash screen.	Splash screen is displayed.	PASS
	2	System detects user has previously logged in.	System proceeds to verify stored credentials.	PASS
	3	System verifies user's credentials.	System signs in the user and directs user to Home Page.	PASS
Alternative Flows	2a	System detects user has not previously logged in.	System directs user to Register and Log In Page.	PASS
	3a	System fails to verify user's credentials.		PASS

Table 6. 2. 3. 2 User Sign In Test Case

Test Case Name: User Sign In			Test Case ID: T002	
	Test Steps	Action	Expected Result	Result Status
Main Flows	1	User enters email and password.	Email and password fields are filled.	PASS
	2	User clicks Sign In button.	System begins credential verification.	PASS
	3	System verifies user's credentials.	System signs in the user and directs user to Home Page.	PASS
Alternative Flows	1a	Users enters incorrect email and/or password.	System displays error message.	PASS
	3a	System fails to verify user's credentials.		PASS

Table 6. 2. 3. 3 User Sign Up Test Case

Test Case Name: User Sign Up			Test Case ID: T003	
	Test Steps	Action	Expected Result	Result Status
Main Flows	1	User clicks Sign Up button.	System directs user to Sign Up Page.	PASS
	2	User enters all fields in Sign Up Page.	All required fields are filled correctly.	PASS
	3	User clicks Create Account button.	System sends a verification email to the provided address.	PASS
	4	User clicks on verification link.	System verifies user's email.	PASS
	5	System directs user to Account Successfully Created Page.	Confirmation of account creation is shown.	PASS

	6	User clicks on Continue button.	System directs user to Home Page.	PASS
Alternative Flows	2a	User does not enter all fields in Sign Up Page.	System displays error message.	PASS
	2b	User enters invalid email format.		PASS
	2c	User enters weak password format.		PASS
	2d	User enters a non-numeric phone number.		PASS
	2e	User does not accept the privacy policy.		PASS
	5a	User does not click on verification link.		PASS

Table 6. 2. 3. 4 User Forget Password Test Case

Test Case Name: User Forget Password			Test Case ID: T004	
	Test Steps	Action	Expected Result	Result Status
Main Flows	1	User clicks on “Forget Password?”.	System directs user to password recovery page.	PASS
	2	User enters email address.	Email is entered in the input field.	PASS
	3	System sends email verification link.	User receives a verification link at the entered email.	PASS
	4	User reset password.	System updates the password successfully.	PASS
	5	User clicks on “Done” button.	System directs user to Register and Login page.	PASS
Alternative Flow	2a	Users enters invalid email format.	System displays error message.	PASS

Table 6. 2. 3. 5 Displaying Past Detection Details Test Case

Test Case Name: Display past detection details			Test Case ID: T005	
	Test Steps	Action	Expected Result	Result Status
Main Flows	1	User clicks on Home button.	System directs user to Home Page.	PASS
	2	User clicks on past summarised result section.	System directs user to a detailed results page.	PASS

Table 6. 2. 3. 6 Take a Photo Test Case

Test Case Name: Take a photo to perform facial detection			Test Case ID: T006	
	Test Steps	Action	Expected Result	Result Status
Main Flows	1	User clicks on Facial button.	System directs user to Facial Page.	PASS
	2	User clicks on Take a Photo button.	System opens the device camera for photo capture.	PASS
	3	User captures a photo using the camera.	The photo is captured successfully and displayed on the screen.	PASS
	4	User clicks on “✓” button.	The photo is submitted and ready for facial detection processing.	PASS
	5	System schedules reminder.	System schedules a reminder for next attempt.	PASS
	6	System displays facial detection result.	System processes the captured photo and	PASS

			shows the facial detection result.	
	7	User clicks on Close button.	System closes the result and directs user to Home Page.	PASS

Table 6. 2. 3. 7 Pick from Gallery Test Case

Test Case Name: Pick from gallery to perform facial detection			Test Case ID: T007	
	Test Steps	Action	Expected Result	Result Status
Main Flows	1	User clicks on Facial button.	System directs user to Facial Page.	PASS
	2	User clicks on Pick from Gallery button.	System opens the device's photo gallery.	PASS
	3	User selects a photo to upload.	The selected photo is uploaded successfully.	PASS
	4	System schedules reminder.	System schedules a reminder for next attempt.	PASS
	5	System displays facial detection result.	System processes the selected photo and shows the facial detection result.	PASS
	6	User clicks on Close button.	System closes the result and directs user to Home Page.	PASS

Table 6. 2. 3. 8 Checklist-based Detection Test case

Test Case Name: Perform checklist-based detection			Test Case ID: T008	
	Test Steps	Action	Expected Result	Result Status
Main Flows	1	User clicks on Checklist button.	System directs user to Checklist Page.	PASS
	2	User selects the checklist option.	Option is selected successfully.	
	3	User clicks on Next button.	System displays the next question.	PASS
	4	User clicks on Submit button.	System submits the checklist and proceeds with the evaluation.	PASS
	5	System schedules reminder.	System schedules a reminder for next attempt.	PASS
	6	System displays checklist-based detection result.	System displays the evaluation result based on selected checklist options.	PASS
	7	User clicks on Close button.	System closes the result and directs user to Home Page.	PASS
Alternative Flow	3a	User clicks on Previous button.	System directs user to previous question.	PASS

Table 6. 2. 3. 9 Update Profile Name Test Case

Test Case Name: Update profile name			Test Case ID: T009	
	Test Steps	Action	Expected Result	Result Status
Main	1	User clicks on Profile button.	System directs user	PASS

Flows			to Profile Page.	
	2	User clicks on the “>” button next to the name.	System navigates to the name edit screen.	PASS
	3	User changes name field.	New name is entered successfully.	PASS
	4	User clicks on Save button.	System saves new name, and user is directed to the Profile Page to view updated name.	PASS

Table 6. 2. 3. 10 Update Phone Number Test Case

Test Case Name: Update phone number			Test Case ID: T010	
	Test Steps	Action	Expected Result	Result Status
Main Flows	1	User clicks on Profile button.	System directs user to Profile Page.	PASS
	2	User clicks on the “>” button next to the phone number.	System navigates to the phone number edit screen.	PASS
	3	User changes phone number field.	New phone number is entered successfully.	PASS
	4	User clicks on Save button.	System saves new phone number, and user is directed to the Profile Page to view updated phone number.	PASS

Table 6. 2. 3. 11 Logout Test Case

Test Case Name: Logout			Test Case ID: T011	
	Test Steps	Action	Expected Result	Result Status
Main Flows	1	User clicks on Profile button.	System directs user to Profile Page.	PASS
	2	User clicks on Logout button.	System logs out the user and directs to the Register and Login Page.	PASS

Table 6. 2. 3. 12 Delete Account Test Case

Test Case Name: Delete Account			Test Case ID: T012	
	Test Steps	Action	Expected Result	Result Status
Main Flows	1	User clicks on Profile button.	System directs user to Profile Page.	PASS
	2	User clicks on Delete Account button.	System prompts a confirmation alert.	PASS
	3	User clicks on Delete button.	System directs user to re-authentication screen.	PASS
	4	User enters email and password	Email and password fields are filled.	PASS
	5	User clicks on Verify button.	System verifies the credentials and deletes the account, redirecting user to Register and Login Page.	PASS
Alternative Flows	3a	User clicks on Cancel button.	System direct user to the previous page.	PASS

	4a	User does not enter all fields.	System displays error message.	PASS
	4b	User enters invalid email or/and password.		PASS

6.3 Project Challenges

This project encounters several critical challenges. One of the main issues is the limited availability of facial datasets, with fewer than 3,000 images available online. This small dataset restricts the model's ability to generalise effectively, reducing its accuracy and reliability. Furthermore, testing the model with real children diagnosed with autism is not feasible during mobile application deployment. The reliance on online datasets, which may not fully represent real-world scenarios, further limits the model's practical utility.

One significant challenge encountered during model training is accessing and using GPUs on platforms like Kaggle. The GPU settings often do not align with the expected configurations, and there are limited options for customising the environment on Kaggle. As a result, troubleshooting becomes a time-consuming process, often requiring online research or trial-and-error to identify and resolve underlying issues. This can cause delays and inefficiencies, affecting the speed and quality of the model's training.

During testing, repeated interactions with the same data or processes can cause complications, especially when using cloud services like Firebase. One major issue is the difficulty in managing and clearing stored data, as Firebase can accumulate outdated information. This leads to a situation where the application must be tested multiple times, and the data on Firebase needs to be deleted and reset before continuing. This process can be both tedious and time-consuming, especially when trying to ensure a fresh environment for each test.

6.4 Objectives Evaluation*Table 6. 4. 1 Objectives Evaluation*

Objective	Evaluation	Achievement
To streamline ASD detection process by training a facial detection model using transfer learning	The project has successfully achieved this objective by experimenting with different settings and tuning the model to improve detection accuracy. Using transfer learning, the model was trained to recognise key facial features, leading to higher detection rates. The model was then converted into a TFLite format for efficient deployment on mobile devices, facilitating autism detection.	Achieved
To make ASD detection more accessible through a mobile application	The project has developed a mobile application that integrates two ASD detection methods: facial recognition and a checklist-based approach. This mobile application provides a portable solution, making it accessible to everyone, not just a limited group of people. It enables families and caregivers to perform initial screenings conveniently. The app is designed to be user-friendly, ensuring broad accessibility regardless of users' resources or location.	Achieved
To reduce diagnostic variability in ASD detection through facial recognition	By using the mobile application for repeated ASD screenings, the project minimises diagnostic variability caused by inconsistent behaviours in autistic children. As facial features are stable and detectable regardless of external factors, the system ensures more consistent results. Additionally, if one attempt is not accurate,	Achieved

	subsequent trials can improve the detection, further reducing variability.	
--	---	--

6.5 Concluding Remark

This project provides flexibility in autism detection, enabling early identification without the need for professional involvement, which can facilitate early intervention. The project leverages an available dataset to train the model and experiments with different settings, aiming to deliver reliable detection results. It not only integrates two ASD detection methods but also incorporates essential mobile application features such as registration, login, and user profiles. Additionally, the app allows users to review their previous results, a critical feature as users may sometimes provide incorrect responses, leading to inaccurate detection. By offering this functionality, the app ensures a more consistent, reliable, and comprehensive detection process.

Chapter 7 Conclusion and Recommendation

This chapter presents the overall conclusion of the project, summarising the key outcomes. It also provides recommendations for future improvements and further development.

7.1 Conclusion

In conclusion, this project focuses on the development of a mobile application for detecting autism using facial recognition, complemented by a checklist-based method to further validate results. The project follows the RAD approach, prioritising rapid development and iteration to efficiently address user needs. The aim is to streamline the ASD detection process, making it more accessible to the general public and reducing the complexity and time involved in traditional diagnostic methods. This also addresses the challenge of children with ASD exhibiting different behaviours in various social contexts. Importantly, there are no existing mobile applications currently using facial recognition for ASD detection, making this project especially significant.

For facial recognition, transfer learning was employed to test various models, and several techniques were assessed to optimise performance. The Xception model in Keras, with full unfreezing, achieved an accuracy of 82.77%, while the TFLite version reached 78.04%. External sources for comparison suggest that the model generalises well with an accuracy of 90%. For the questionnaire-based approach, both the Keras and TFLite models achieved an impressive accuracy of 97.60%.

The best-performing facial recognition model was adapted and integrated into the mobile application, along with the checklist-based model. These two methods form the core of the ASD detection process within the app. In addition to these detection methods, essential mobile application features such as registration, login, user profiles, and the home page modules have been incorporated to ensure a comprehensive user experience.

This project lays the foundation for a mobile solution that enables families and caregivers to perform early ASD screenings, reducing the reliance on professionals and improving accessibility.

7.2 Recommendations

Firstly, future training datasets should be sourced from more diverse locations, possibly through collaborations with specialised centres. This would provide a broader range of real-world images, enhancing the dataset. By acquiring a more diverse set of images, the model can be better fine-tuned to improve its ability to generalise across various conditions. Additionally, involving autistic children in the testing phase would be valuable to assess the performance of both the facial recognition and checklist-based detection methods. In the future, adopting more state-of-the-art models and performing further tuning on each could optimise performance.

Secondly, with regard to the mobile application, the current project does not focus heavily on mobile performance. In the future, the well-developed model could be deployed on a server to reduce the computational burden on the mobile device. This would not only improve the mobile application's performance but also streamline the detection process.

Finally, beyond the two existing detection methods, future versions of the mobile application could incorporate additional diagnostic techniques, such as handwriting analysis. This could serve as another key method for distinguishing autism, potentially increasing the app's accuracy and reliability.

REFERENCES

- [1] Government of Canada, “Signs and symptoms of autism spectrum disorder (ASD) - Canada.ca,” *Canada.ca*, Apr. 07, 2022. <https://www.canada.ca/en/public-health/services/diseases/autism-spectrum-disorder-asd/signs-symptoms-autism-spectrum-disorder-asd.html>
- [2] M. J. Maenner, “Prevalence and Characteristics of Autism Spectrum Disorder Among Children Aged 8 Years — Autism and Developmental Disabilities Monitoring Network, 11 Sites, United States, 2020,” *MMWR. Surveillance Summaries*, vol. 72, no. 2, pp. 1–14, Mar. 2023, doi: <https://doi.org/10.15585/mmwr.ss7202a1>.
- [3] “Khairy Moots National Autism Council As Autism Rates Rise,” *CodeBlue*, Jul. 15, 2022. Available: <https://codeblue.galencentre.org/2022/07/15/khairy-moots-national-autism-council-as-autism-rates-rise/>
- [4] X. Kong, Z. Wang, J. Sun, X. Qi, Q. Qiu, and X. Ding, “Facial recognition for disease diagnosis using a deep learning convolutional neural network: a systematic review and meta-analysis,” *Postgraduate Medical Journal*, vol. 100, no. 1189, pp. 796–810, Aug. 2024, doi: <https://doi.org/10.1093/postmj/qgae061>.
- [5] Md. F. Rabbi, S. M. M. Hasan, A. I. Champa, and Md. A. Zaman, “A Convolutional Neural Network Model for Early-Stage Detection of Autism Spectrum Disorder,” presented at the 2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD, IEEE, Feb. 2021, pp. 110–114. doi: <https://doi.org/10.1109/ICICT4SD50815.2021.9397020>.
- [6] G. Tartarisco *et al.*, “Use of Machine Learning to Investigate the Quantitative Checklist for Autism in Toddlers (Q-CHAT) towards Early Autism Screening,” *Diagnostics*, vol. 11, no. 3, p. 574, Mar. 2021, doi: <https://doi.org/10.3390/diagnostics11030574>.
- [7] J. Howarth, “How Many People Own Smartphones? 80+ Smartphone Stats,” *Exploding Topics*, Jan. 26, 2023. <https://explodingtopics.com/blog/smartphone-stats>
- [8] C. Kraft *et al.*, “Wait Times and Processes for Autism Diagnostic Evaluations: A First Report Survey of Autism Centers in the U.S,” 2023. Available:

REFERENCES

<https://www.cms.gov/files/document/wait-times-and-processes-autism-diagnostic-evaluations-first-report-survey-autism-centers-us.pdf>

[9] C. Pratt, “Characteristics of Individuals with an ASD: Articles: Indiana Resource Center for Autism: Indiana University Bloomington,” *Indiana Resource Center for Autism*, 2017. <https://www.iidc.indiana.edu/irca/articles/characteristics-of-individuals-with-an-asd.html>

[10] K. K. Mujeeb Rahman and M. M. Subashini, “Identification of Autism in Children Using Static Facial Features and Deep Neural Networks,” *Brain Sciences*, vol. 12, no. 1, p. 94, Jan. 2022, doi: <https://doi.org/10.3390/brainsci12010094>.

[11] Y. Khosla, P. Ramachandra, and N Chaitra, “Detection of autistic individuals using facial images and deep learning,” presented at the 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), IEEE, Dec. 2021, pp. 1–5. doi: <https://doi.org/10.1109/csitss54238.2021.9683205>.

[12] M. S. Alam, M. M. Rashid, R. Roy, A. R. Faizabadi, K. D. Gupta, and M. M. Ahsan, “Empirical Study of Autism Spectrum Disorder Diagnosis Using Facial Images by Improved Transfer Learning Approach,” *Bioengineering*, vol. 9, no. 11, p. 710, Nov. 2022, doi: <https://doi.org/10.3390/bioengineering9110710>.

[13] N. Farhah, “Utilizing Deep Learning Models in an Intelligent Facial Expression Classification System for Autism Disorder Diagnosis,” *Int. J. Advance Soft Compu. Appl*, vol. 16, no. 2, pp. 2074–8523, 2024, doi: <https://doi.org/10.15849/IJASCA.240730.15>.

[14] T. Saeed Mian, “Efficient Net-based Transfer Learning Technique for Facial Autism Detection,” *Scalable Computing: Practice and Experience*, vol. 24, no. 3, pp. 551–560, Sep. 2023, doi: <https://doi.org/10.12694/scpe.v24i3.2233>.

[15] L. Sun and Y. Mao, “CNN Based ASD Early Screening for Chinese Children,” *Advances in Social Science, Education and Humanities Research/Advances in social science, education and humanities research*, pp. 211–221, Jan. 2024, doi: https://doi.org/10.2991/978-2-38476-230-9_26.

REFERENCES

- [16] A. S. Mohanty, P. Parida, and K. C. Patra, "Identification of Autism Spectrum Disorder using Deep Neural Network," *Journal of Physics: Conference Series*, vol. 1921, p. 012006, May 2021, doi: <https://doi.org/10.1088/1742-6596/1921/1/012006>.
- [17] La Trobe University, "ASDetect," *Google Play Store*, 2009. https://play.google.com/store/apps/details?id=au.edu.latrobe.asdetect&hl=en_US (accessed Oct. 23, 2019).
- [18] Zakhar Lobanov, "Autism Evaluation Checklist," *Google Play Store*, 2021. <https://play.google.com/store/apps/details?id=ru.atec&hl=en> (accessed Sep. 09, 2024).
- [19] M. Gadhavi, "Rapid Application Development Guide 2023 - Phases, Tools & Benefits," *Radixweb*, Sep. 22, 2022. <https://radixweb.com/blog/introduction-to-rapid-application-development>
- [20] Imran Khan, "Autistic Children Facial Dataset," *www.kaggle.com*, 2021. <https://www.kaggle.com/datasets/imrankhan77/autistic-children-facial-data-set>
- [21] Gerry, "autism - Google Drive," *drive.google.com*, 2022. <https://drive.google.com/drive/folders/1XQU0pluL0m3TIIxqntano12d68peMb8A>
- [22] L. M. Uppuluri, "ASD children traits," *www.kaggle.com*, 2022. <https://www.kaggle.com/datasets/uppulurimadhuri/dataset>
- [23] Larxel, "Autism Screening on Adults," *www.kaggle.com*, 2020. <https://www.kaggle.com/datasets/andrewmvd/autism-screening-on-adults>
- [24] Fadi, "Autism screening data for toddlers," *www.kaggle.com*, 2018. <https://www.kaggle.com/datasets/fabdelja/autism-screening-for-toddlers>
- [25] "Q-CHAT-10 Quantitative Checklist for Autism in Toddlers A B C D E 1." Available: <https://www.autismalert.org/uploads/PDF/SCREENING--AUTISM--QCHAT-10%20Question%20Autism%20Survey%20for%20Toddlers.pdf>


APPENDIX

A.1 Questionnaire Sample [25]

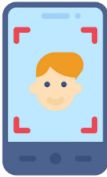
For each item, please circle the response which best applies to your child:

		A	B	C	D	E
1	Does your child look at you when you call his/her name?	Always	Usually	Sometimes	Rarely	Never
2	How easy is it for you to get eye contact with your child?	Very easy	Quite easy	Quite difficult	Very difficult	Impossible
3	Does your child point to indicate that s/he wants something? (e.g. a toy that is out of reach)	Many times a day	A few times a day	A few times a week	Less than once a week	Never
4	Does your child point to share interest with you? (e.g. pointing at an interesting sight)	Many times a day	A few times a day	A few times a week	Less than once a week	Never
5	Does your child pretend? (e.g. care for dolls, talk on a toy phone)	Many times a day	A few times a day	A few times a week	Less than once a week	Never
6	Does your child follow where you're looking?	Many times a day	A few times a day	A few times a week	Less than once a week	Never
7	If you or someone else in the family is visibly upset, does your child show signs of wanting to comfort them? (e.g. stroking hair, hugging them)	Always	Usually	Sometimes	Rarely	Never
8	Would you describe your child's first words as:	Very typical	Quite typical	Slightly unusual	Very unusual	My child doesn't speak
9	Does your child use simple gestures? (e.g. wave goodbye)	Many times a day	A few times a day	A few times a week	Less than once a week	Never
10	Does your child stare at nothing with no apparent purpose?	Many times a day	A few times a day	A few times a week	Less than once a week	Never

POSTER





MOBILE APPLICATION FOR DETECTING ASD



A mobile application providing facial recognition and questionnaire-based approaches


INTRO

Autism Spectrum Disorder (ASD) is a neurodevelopmental condition that impacts communication, behavior, and social interaction. With the rising number of ASD cases, early detection becomes crucial. Identifying ASD early can help reduce the severity of symptoms and provide timely support.

METHODOLOGY

Facial Recognition Model



```

graph TD
    A[Image Acquisition] --> B[Remove Duplicates  
(MD5 and pHash)]
    B --> C[Train with Pre-Trained Model]
    C --> D[Fine Tuning]
    D --> E[Integrate models into mobile application]
            
```


Checklist-Based Model

```

graph TD
    A[Data Acquisition] --> B[Data Preprocessing]
    B --> C[Train with Customised model]
    C --> E[Integrate models into mobile application]
            
```



IMPORTANCE

- Reduces costs and minimizes reliance on experts
- Provides an accessible and convenient tool for families
- Ensures consistent ASD detection despite behavioral variability




ENHANCED

- Lack of ASD detection mobile app
- Available detection apps rely on questionnaire-based approaches

Done By: Lim Chia Yoong
Supervised by: Dr Muhammad Syaiful Amri Bin Suhaimi



Bachelor of Computer Science (Honours)
Universiti Tunku Abdul Rahman

Bachelor of Computer Science (Honours)
Faculty of Information and Communication Technology (Kampar Campus), UTAR

100