REAL - TIME FACE RECOGNITION MOBILE APPLICATION FOR CLASS ATTENDANCE

BY

JACYNTH THAM MING QUAN

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)

JAN 2022

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Real-Time Face Recognition Mobile Application For Class Attendance

Academic Session: Jan 2022

I JACYNTH THAM MING QUAN

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:
33, Jalan Desa 7/13a,
Bandar Country Homes,
48000 Rawang, Selangor

Mr. Tou Jing Yi
Supervisor's name

Date: 17th March 2022

Date: 21st April 2022

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 23th March 2022

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Jacynth Tham Ming Quan (ID No: 18ACB01600) has completed this final year project entitled "Real-Time Face Recognition Mobile Application For Class Attendance" under the supervision of Mr. Tou Jing Yi (Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology, and Dr Manoranjitham a/p Muniandy (Co-Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

_____

Jacynth Tham Ming Quan

DECLARATION OF ORIGINALITY

I declare that this report entitled "**REAL - TIME FACE RECOGNITION MOBILE APPLICATION FOR CLASS ATTENDANCE**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : 

Name : JACYNTH THAM MING QUAN

Date : 17th March 2022

ACKNOWLEDGEMENTS

I would like to convey my heartfelt appreciation and gratitude to my supervisor, Mr Tou Jing Yi and my moderator, Dr Manoranjitham a/p Muniandy. Both of them had inspired me and presented me with this once-in-a-lifetime opportunity to engage in an Artificial Intelligence application project that I have been passionate about for a long time. Mr Tou supported me profusely when I had new ideas and helped me solve doubts and issues when needed with their constructive feedback. This project marks my very first step into paving a triumphant career in the Information Technology field. A billion thanks to both of you.

To my family, who has been extremely supportive of me, words cannot express my gratitude for all of you. Thank you for always supporting me through thick and thin. Thank you for always being there when I need to lend an ear. Thank you for constantly believing me and encouraging me to achieve my goals and chase my dreams. This project would not be possible without your mental support.

To two very special people in my life who shall remain anonymous, you two have helped me through hardships and stressful times. You two picked me up when I was down, encouraged me when I felt hesitant, stood by my side when I needed support, both physically and mentally. You two are like shimmering stars in the sky that I look to when I am lost in the dark. Thank you for always being by my side.

Here, I would also like to show my gratefulness to all the people who have assisted me in putting my imaginative ideas, which are well above the level of simplicity, into something concrete. Thank you to all those who have supported me throughout my university life.

ABSTRACT

In every education setting in Malaysia, there are growing concerns regarding the student attendance-taking process. Currently, the paper-and-pen attendance-taking method is not only time-consuming and inaccurate but also susceptible to impersonation. Thus, the tediousness of manual attendance-taking not only burdens teachers with extra workloads, but also indirectly deteriorates the quality of the lesson delivered. Moreover, the existing face recognition systems developed to conquer this issue are futile due to their sluggish recognition and inability to distinguish between identical siblings effectively.

Hence, this paper describes a novel implementation of a face recognition mobile application named FaceIt, that automates the attendance-taking process in classrooms altogether. The implemented setup requires a basic Android smartphone and a tripod to have a real-time video stream fed automatically into the detection and recognition pipeline within the mobile application itself. The face detection process is then executed using Firebase ML Kit Face Detection API and the face recognition process after that is realized with the use of an enhanced mobile application deep-learning TensorFlow Lite model called MobileFaceNet. The application implemented in this paper is unique compared to other face recognition class attendance applications in the market due to this application's ability to provide a fallback flow to handle identical siblings, something that most, if not all face recognition models in the market are having difficulties with. Furthermore, Firebase Cloud Database is used to sync all the application's data across multiple devices within the same educational institution.

The novelty of the developed application is to provide a fully automated attendance taking process in classrooms, with minimal human intervention required only to counteract the vulnerabilities of the face recognition model when presented with identical siblings. The developed application aims to replace the current attendance-taking systems in educational institutions with improved recognition, faster performance time and added convenience for both students and teachers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
API	Application Programming Interface
AT & T	American Telephone & Telegraph
CLAHE	Contrast Limited Adaptive Histogram Equalization
CNN	Convolutional Neural Network
GPS	Global Positioning System
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
LBP	Local Binary Pattern
LBPH	Local Binary Pattern Histogram
LDA	Linear Dependent Analysis
MATLAB	Matrix Laboratory
PCA	Principal Component Analysis
PHP	Hypertext Preprocessor
QR	Quick Response
RFID	Radio Frequency Identification
SQL	Structured Query Language
SVM	Support Vector Machine
XML	eXtensible Markup Language
YOLO V3	You Only Look Once Version 3

Chapter 1

Introduction

The aspects presented in this chapter include an introduction to the project's background, the problem statement and motivation of the project, the aims and contributions of this project to the relevant fields of studies, and the outline of this project's report.

1.1 Background Information

Ever since the dawn of the 21st century, technology has successfully melded into the daily lives of humans. Undeniably, the introduction of information and communication technologies had been a crucial milestone in the advancement of human knowledge. Till today, humans have been continuously finding ingenious methods to automate daily tasks and solve complex problems. Among the many different fields of technology-related research, computer vision and image analysis have always been a significant focus due to their wonderous capabilities.

Computer vision is commonly interpreted as the field of study that looks at techniques to let machines "see" the world and intelligently interpret digital media contents such as images and videos. The digitalised world is full of pictures, videos, and texts. As humans, searching for specific visual details may be a trivial problem. A person is capable of describing visual content, summarising it, and recognising specific details effortlessly. However, when it comes to computers and artificial intelligence, complex backend algorithms are needed to give them the vision they need to mimic the capabilities of human vision. This is where computer vision comes into play. In order for the process of visual perception to be automated, computer vision systems make use of various image processing algorithms.

Image processing, a subset of computer vision, is defined as the process of manipulating digitalised images to extract useful information. Examples of simple image processing include saturation fix, brightness adjustment, colour contrast, and etcetera. These image processing functions can be experienced in any ordinary picture editing application. On the other hand, complex image processing algorithms such as

noise reduction and image restoration are used in computer vision to complete specific tasks. Undeniably, one of the most actively researched applications of image processing and computer vision is none other than face recognition.

Face recognition is defined as a biometric recognition technique in which the faces of different individuals are identified based on the face pictures saved in a dataset. This technology had become increasingly popular in recent years due to its multitudinous applications as a non-intrusive identity confirmer. The phrase "non-intrusive" describes face recognition as a verification process that does not require human contact with the biometric input device, a camera, in this case. Traditionally, face recognition has been heavily associated with the security sector [1]. For example, biometric identification was crucial in highly secured places such as bank vaults and office headquarters. On a more global level, face recognition is integrated into CCTV systems scattered in major cities to identify the whereabouts of certain people such as burglars or even highly-dangerous serial killers who are being sought out by policemen.

Today, face recognition technology can even be experienced right at one's fingertips. Many smartphones utilise face recognition technologies for device-unlocking purposes. Moreover, Google Photos and Apple Photos are able to categorise one's photo gallery by face. This process is done by applying face recognition to each picture and categorising them based on the face matches found. Without a doubt, the problems to which face recognition can solve are endless. One of the possible applications of face recognition is in school attendance management. Figure 1.1 below shows an example of face recognition used for attendance taking in a desktop application.

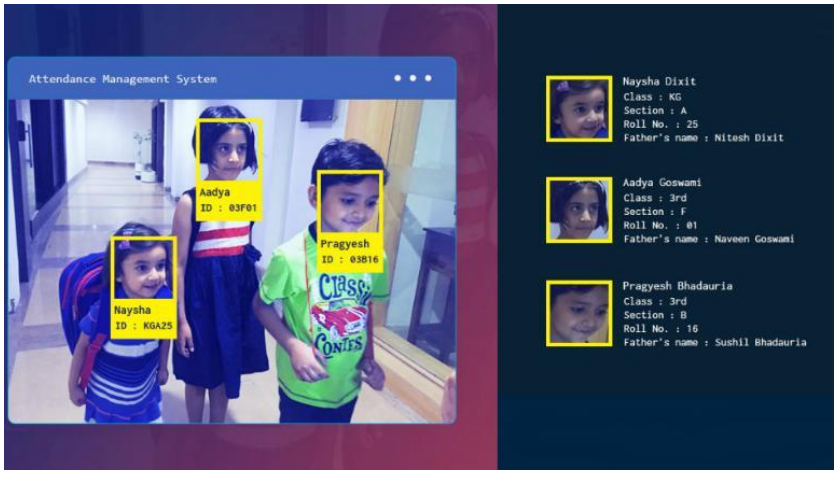


Figure 1.1 Face Recognition Attendance Management System [2]

Attendance management is paramount in every educational institution. From primary education to tertiary education and beyond, teachers keep track of the students' attendance for various purposes. These include progress assessment of students, consistency checking of class attendance, and fundamental record keeping. The most common methods of attendance taking include calling the names of students, passing around attendance sheets for signatures, manually comparing student card numbers with a list, and etcetera. The failure to meet the minimum required attendance percentage may lead to adverse consequences for students. These consequences may be as minor as disciplinary penalties or as severe as terminations of studies. Thus, to assert the credibility of the attendance taken, the introduction of face recognition as an attendance-taking method has been studied and implemented throughout the years till this day.

1.2 Problem Statement and Motivation

The two main problems of manual attendance taking are that it is time-consuming and susceptible to impersonation and forgery. The implemented application, FaceIt, uses face recognition to conquer this issue by making the attendance-taking process quick and efficient as human intervention is mostly omitted. Unlike manual attendance taking methods which may take up to an hour to cover a large lecture class, automated attendance taking only requires a few seconds per student in order for the attendance to be taken. Moreover, the use of face recognition in this project works by identifying students based on their unique facial features, which creates a sound barrier against fake attendance issues such as signature forging.

Furthermore, this project integrates face recognition technologies seamlessly into a mobile application platform to conquer the problems in existing face recognition applications. Unlike the majority of face recognition applications which are computer-based, the use of a mobile-based application in this project makes the application portable and convenient to use. Moreover, this project emphasises on a more user-friendly mobile interface, in opposed to existing face recognition applications, which have complicated user interfaces. The complexity of the user interface is concerning to senior lecturers and non-tech-savvy teachers and causes them to adhere to manual attendance-taking methods instead. Furthermore, this project uses efficient client-server

connections with the cloud database, Firebase, to sync information between lectures within the same educational institution. Hence, this project focuses on the integration of face recognition into an easy-to-use mobile application so that the attendance-taking process can be automated and made to be convenient and secure for teachers to use.

The problems mentioned above must be conquered as attendance taking plays a vital role in the education sector. Teachers need help in conquering manual attending-taking processes to reduce their workloads in schools so that they can focus more on delivering effective lessons to students. With the stake of quality education at risk, FaceIt was developed with the use of face recognition in a mobile application as an automated attendance-taking medium.

Next, the surge in mobile devices' computational capabilities also motivates the development of a mobile application that is capable of integrating face recognition technologies for attendance taking. With the simplicity of mobile-based platforms, advanced computer vision techniques can be more straightforward and more feasible for daily use. In FaceIt, the implementation of face recognition in a mobile application is used to overcome the problems faced by time-consuming and laborious manual attendance-taking methods. The use of a mobile application to house face recognition technologies also makes the entire attendance-taking system portable and cost-efficient. Therefore, the final mobile application delivered in this project not only improves the efficiency of attendance-taking in schools but also saves valuable human resources as well as enhances the attending-taking process for both teachers and students.

1.3 Project Scope and Direction

To solve the drawbacks of previous face recognition applications, the outcome of this project is a novel model of a more efficient, portable, and time-saving face recognition mobile application for attendance-taking in classrooms. The developed mobile application is able to acquire individual attendance automatically through a real-time video stream, perform face recognition to ensure the credibility of the attendance taken, and record the attendance into a cloud database. The coverage of this project includes all levels of the education sector, which include kindergartens, primary schools, secondary schools, and tertiary education institutions such as universities and colleges. As a result, this project aims to tackle the problems faced by manual attendance

methods in classes of varying sizes. Moreover, the targeted groups of this face recognition mobile application are students from the age of 3 to 22. The works in this project are based on the assumption that all teachers have Android phones, and that all school institutions are Wi-Fi covered as the developed application needs to continually communicate with a cloud database to run the face recognition module and update the attendance into the school's database. Furthermore, it is assumed that when a new school wishes to use the developed application, the head administrator has to contact the developer of the application to add in the school name and obtain a school authentication code, which is unique to each school and is required for new users to register under a particular school.

1.4 Project Objectives

The primary aim of this project is to implement the use of face detection and face recognition technologies into a mobile application for convenient class attendance taking. Listed below are the three main objectives that have been achieved in this project:

- To develop a face recognition mobile application that is able to automate the attendance-taking process in schools.
- To develop a user-friendly mobile application interface for teachers to manage students' attendance.
- To implement a face recognition algorithm with more than 85% accuracy and a processing time of fewer than 3 seconds per student.

1.5 Impact, Significance, and Contribution

The contributions of this project are significant to the students, teachers, and the education sector in general. With this face recognition mobile application, schooling institutions are able to save precious physical and human resources. With the use of an automated mobile application to take attendance, schools can save up on the physical resources needed for manual attendance taking, such as pens and papers. Moreover, schools do not have to burden teachers and administration clerks to take student

attendance and enter the attendance records into the school's database manually. This frees up human resources to perform other more significant tasks, such as planning more comprehensive subject lessons. As a result, teachers have more time to prepare for their classes and are able to bring more impactful lessons to their students. Hence, the contributions of this project indirectly boost the current education standards in terms of lesson quality. Next, the use of face recognition to ensure attendance credibility makes it more challenging for students to impersonate their attendance records. As a result, students will be warier about their attendance, which causes them to be more punctual for classes. Additionally, students will avoid immoral ethics in schools, such as faking attendance on behalf of a friend. Furthermore, with the quick and easy attendance-taking method provided by this project, students can focus more on the lesson delivered and thus, perform better in assessments and examinations.

The developed face recognition mobile application in this project is unique compared to existing face recognition applications because of two main reasons. Firstly, the developed application implements the attendance-taking process as an automated, live-streaming video. To begin the face detection and face recognition process, teachers only have to click the start button once, and the application will recognise students as they walk past and glance into the mobile phone's camera for a brief moment. Secondly, the developed face recognition application is able to handle the presence of identical siblings. Almost all existing face recognition attendance systems are incapable of doing so because face recognition algorithms, in general, have hard times differentiating between identical siblings due to their similar, if not indistinguishable, facial features. However, with a quick workaround, the developed application is able to prompt human intervention when needed in order to handle identical siblings in classrooms.

In short, this project emphasises heavily on fast computation time and high recognition accuracy to increase its overall effectiveness as an attendance-taking system. Coupled with a highly easy-to-use mobile app user interface, this project also caters to senior teachers who may not be very well-versed with technology. All in all, this project aims to aid schools in conquering the problems associated with manual attendance taking.

1.6 Report Organization

The details of this project are presented as described in this section. In Chapter 2, the face recognition process, existing systems, and other related background information are compared and reviewed. Next, Chapter 3 describes the system methodology and the general flow of each of the project modules briefly before diving deep into the implementation and testing results in the following 5 chapters, accounting for 1 chapter per project module. Chapter 4 explains the Login and Registration module, Chapter 5 describes the Class module, Chapter 6 discusses the Student module, Chapter 7 explains the Enrolment module and Chapter 8 presents the Attendance module. The following chapter, Chapter 9, summarizes the module testing results, discusses the testing results in a stimulated classroom environment as well as evaluate the post-project objectives. Last but not least, Chapter 10 concludes the entire report with a compendious summary.

Chapter 2

Literature Review

In this chapter, the background of the project is presented, including the steps and algorithms involved in the face recognition process and the comparison between them. Next, the existing desktop-based and mobile-based face recognition class attendance applications are reviewed, compared, and contrasted.

2.1 The Face Recognition Technology

Face recognition is formally defined by Martinez [3] as the technology behind the use of biological systems, such as sensors, to detect and identify specific faces using computer systems. Similarly, Rouse [4] defined face recognition as a class of biometric software that has the ability to map a person's unique facial features mathematically and store the resulting data as a digital faceprint.

The first-ever research done on face recognition started as early as the 1960s and has advanced immensely since then. The initial works of face recognition revolved around manual marking of facial "landmarks" such as the pupils, lips, and nose and then rotating them mathematically using a computer to make up for pose variations when compared with the original data. Conducted by three pioneers by the names of Woody Bledson, Charles Bisson, and Helen Chan Wolf, this experiment highlighted face recognition as a potentially viable biometric. This sparked an interest in many other researchers, who soon up took projects in this field in an attempt to unravel the concepts behind face recognition [5]. As of the 21st century, different face recognition and face detection techniques can be seen throughout the Internet. The main aim of face recognition research is to improve the overall recognition accuracy and recognition rate.

The processing flow of face recognition consists of four ordered steps – face detection, face alignment, feature extraction, and feature matching. Figure 2.1 illustrates the connection between the four face recognition steps, as mentioned above. The in-depth details of each step are discussed in the following sub-sections.

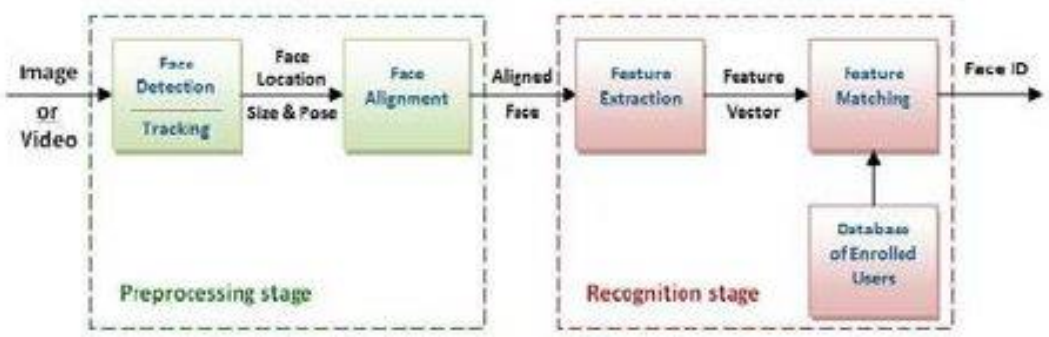


Figure 2.1 Summary of Face Recognition Process [6]

2.1.1 Face Detection

Face detection marks the first step of the face recognition process and is part of the image processing stage, as seen in the figure above. The term "face detection" is often confused with the term "face recognition". However, the former actually refers to the process of locating the face regions in images, while the latter refers to the process of identifying the actual face region itself. In most scenarios, the input image consists of non-face objects such as buildings, flora, fauna, and even other human body parts. Hence, face detection aims at using various machine learning algorithms to locate human faces within images.

Generally, face detection algorithms begin the process by searching for the human eye first before proceeding to detect the rest of the facial features – eyebrows, nose, mouth, and etcetera. When the face region is detected, extra tests may be performed by the algorithm to confirm that the face region is valid. Existing face detection methods can be segregated into four different categories – rule-based, template matching, feature-based and appearance-based.

Knowledge-based face detection methods detect a face based on a predetermined set of rules. These methods depend solely on human knowledge to define the characteristics of a human face. However, if the rules defined are too detailed or too general, the model might produce many false positives. As there are many possibilities for the rules to be defined, it is difficult for developers to come up with a set of well-defined rules for the face detection algorithm.

Next, feature-based face detection methods detect faces by extracting distinct facial features such as ears, eyes, mouths, and noses. The differences between knowledge-based methods and feature-based methods are that the latter detects facial features first, before the entire face. However, for the former, the opposite is the case. One major challenge with using these methods is that the face detection process here is susceptible to light and noise, which may negatively affect the method's ability to detect faces.

Next, template-matching face detection methods detect faces by comparing the current input image with parameterised or pre-defined face templates. These methods typically take into consideration the face contours and positioning of the facial features in order to function. Even though these methods are effortless to implement, they might not be so sufficient for more complex face detection tasks due to their vulnerabilities to variation in face pose, shape, and scale.

Lastly, appearance-based face detection methods detect faces using machine learning and statistical analysis. Generally, appearance-based methods train various classifiers to detect faces using sets of face images. In terms of performance, appearance-based face detection methods prevail among the four methods of face detection. Multiple different classifiers can be used in these methods, such as support vector machines (SVMs), neural networks, Adaboost and etcetera. However, these classifiers take time to train on the face detection task. Therefore, when new information is introduced to the face database, the time-consuming training process has to be repeated.

Table 2.1 summarizes the descriptions and disadvantages of the four face detection methods above.

Table 2.1 Face Detection Methods [4]

Method	Description	Disadvantage
Knowledge-based	Detects a face based on predetermined rules	Challenging to come up with well-defined rules
Feature-based	Detects a face based on a person's facial features	Easily affected by light and noise

Template-matching	Detects a face based on the correlation between the current face and the faces in a database	Easily affected by variations in face pose, shape, and scale
Appearance-based	Detects a face characteristic using machine learning and statistical analysis	Hard to update the face database as the learning process is time-consuming

2.1.2 Face Alignment

Face alignment, also known as face normalization, is defined as the process of how the geometric shapes of human faces are identified. Face alignment is the second step of the pre-processing stage. The face alignment process aims to normalize the input image to match the images in the database in terms of image resolution, magnification, brightness, and orientation. The face aligning process is crucial so that the consecutive image analysis tasks can be done on a common basis.

Figure 2.2 illustrates the process by which two faces are compared. In the figure, A and B show two different faces. Then, the two faces are overlapped, and the differences between the two faces are highlighted. The differences of the faces can be discovered effectively because A and B are of the same colour, orientation, and magnification. If the faces were not similar in these ways, the differences highlighted would be false. Therefore, face alignment is essential to ensure established correspondences between two different faces.

Figure 2.2 Importance of Face Alignment [4]

Furthermore, the face alignment process also utilizes image pre-processing techniques to prepare the aligned face for the feature extraction process later on. These techniques include image scaling, greyscale conversion, and image contrasting.

Firstly, the image scaling process manipulates the overall size of a face image. Generally, images are scaled down rather than up to minimise the pixels involved and, thus, the processing time needed. In 2015, Singh, Rastogi, and Sharma proposed the use of face images with the same length and width [7]. This is to ensure that the pixels of the image were not distorted when passed to the feature extraction module. In all face images, each tiny pixel houses important special information. Spatial information can be used as a measure for even the most minuscule detail in a face image. Hence, image scaling should avoid distortions of pixels as not to lose essential details in the image itself.

Next, face images might also require the use of greyscale conversions. In 2012, a paper proposed by Kanan and Cottrell stated that greyscale images use less computational time and are less sensitive to lighting conditions [8]. This is due to the size of the pixels in a greyscale image, which takes up less storage space – up to 8-bits, compared to the 24 bits in a coloured image's pixel [9]. In some face recognition algorithms where colour is not essential, it is treated as noise and is chosen to be removed to speed up the entire face recognition process.

Following the greyscale conversion, image-contrasting techniques are typically applied to reduce uneven lighting in the face images. There are several techniques of contrast improvement. In 2016, a paper proposed by Pratiksha demonstrated the use of a conventional histogram equalization method, which uniformly distributes light intensity values over the light intensity axes [10]. In 2013, Sasi and Jayasree proposed the use of Contrast Limited Adaptive Histogram Equalization, also known as CLAHE, for image contrast improvement [11]. Unlike Pratiksha's histogram equalization method, CLAHE only modifies the contrast of certain regions of the image using a tile-by-tile method and not throughout the entire image like the former. An advantage of CLAHE is that it prevents any forms of over-enhancement to the image. However, compared to the conventional histogram equalization method, CLAHE is more sensitive to noise.

2.1.3 Feature Extraction and Feature Matching

Feature extraction, the third step in the face recognition process, marks the beginning of the recognition stage. The goal of the feature extraction process is to select

critical facial features from the original image to decrease the total number of features in a given dataset. Generally, facial landmarks that are focused on in the feature extraction process include the centres of eyes, the tips of noses, and the corners of mouths. Selecting these features manually is a tedious process. Therefore, this process makes use of feature extraction algorithms to automate the process and produce high accuracy results.

The final step of the face recognition process is feature matching. In feature matching, various algorithms are used to compare face images with a predefined face database. A common approach to feature matching is to compare the points-of-interests as extracted from the previous feature extraction process. The performance of the whole face recognition process depends heavily on the choice of feature extraction and feature matching algorithm as different algorithms have different approaches to selecting and comparing the points-of-interests.

According to Parmar and Mehta [12], the recognition process – feature extraction and feature matching – can be segregated into three different methods – feature-based, holistic-based, and hybrid methods. The most general category is holistic-based methods, commonly known as appearance-based methods. This category of method utilizes the whole face image as raw input for the feature matching process later on. However, in feature-based methods, only the critical features such as eye, mouth, and nose are located and fed into the feature matching process. As for hybrid methods, both the concept of holistic-based methods and feature-based methods are combined. This combination of methods allows the system to take in even the tiniest details of a face, such as the curves of the facial landmarks, as well as mathematical details such as depth and axis values [13]. Examples of the mentioned categories are listed down as below:

- Holistic-based methods: LDA (Linear Dependent Analysis), PCA (Principal Component Analysis)
- Feature-based methods: LBP (Local Binary Pattern)
- Hybrid methods: A cohesive fusion of both methods

Principal Component Analysis, also known as a dimension-reducing algorithm, works by compressing whole face image matrixes into single-columned vectors. PCA aims to maximize the reduction of the dimensionality of a face dataset, all while

maintaining the face dataset's variation. Generally, the phrase "principal component" in the algorithm's name refers to the set of orthogonal new variables that are extracted from the original variables [14]. The variables extracted are usually considered of higher importance compared to the other variables. In face recognition, these extracted variables are known as the eigenvectors, or more specifically, Eigenfaces, after the transformation of the covariance matrix of the whole face image. The feature matching process of PCA makes use of Euclidean distances to compare the distance between the facial landmarks of the face in the dataset and the current face. Commonly known for its high computation speed and robustness, PCA is often selected for use in face recognition. Examples of the use of PCA for face recognition is highlighted in papers proposed by Singh and Kumar [15], Paul and Sumam [16], and Bhuvaneshwari et al. [17]. Additionally, a paper proposed by Nithya [18] also supported the use of PCA for a computer-based face recognition attendance system. Figure 2.3 illustrates an outline of the PCA algorithm.

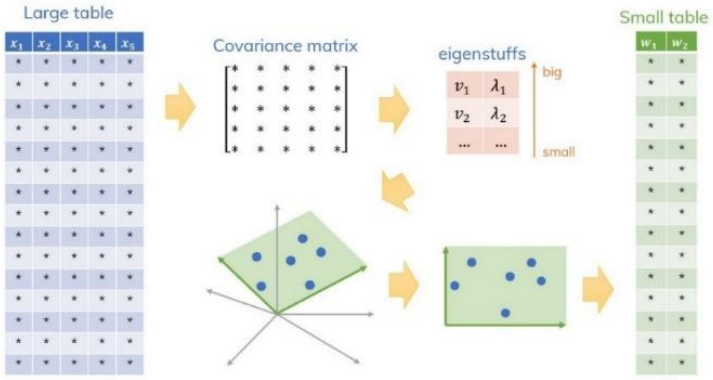


Figure 2.3 PCA Algorithm [19]

The next face recognition algorithm to be discussed is the Linear Discriminant Analysis (LDA) algorithm, more commonly known as the Fisher Face algorithm. Similarly, the features learned from LDA are known as Fisherfaces. Unlike the previous PCA face recognition method, LDA focuses on only the discriminant facial features when optimizing the image into a lower-dimensional one during the feature extraction process. Moreover, during the feature extraction method, LDA groups the images into different classes. Generally, a class should contain face images of a particular individual, regardless of background lighting or facial expressions. Because of this ability to categorise face images, LDA often outperforms PCA when dealing with variation in the face images, such as the two aforementioned factors. Thus, Bhattacharyya and

Rahul [20] proposed the use of LDA over PCA for their face recognition attendance system. In the feature matching process of LDA, Bayes' Theorem is used to estimate the probabilities that the input face image belongs to each class. The class with the highest probability will be returned as LDA's prediction. However, in order for LDA to outperform PCA in terms of recognition accuracy, LDA must be given many face samples of the same class and not have many different classes. Figure 2.4 shows a classification task before and after the application of the LDA algorithm.

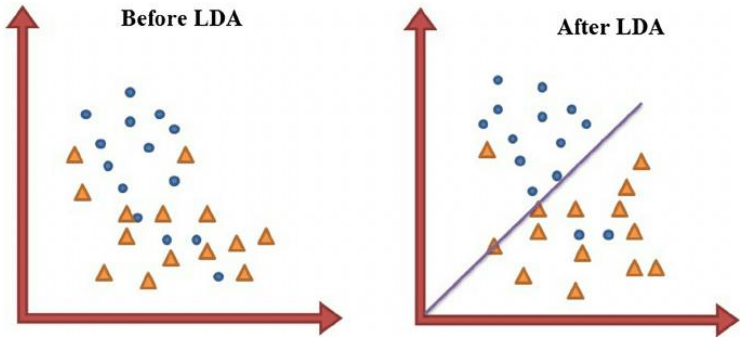


Figure 2.4 LDA Algorithm [21]

Next, Local Binary Pattern (LBP) is a type of feature-based recognition method proposed by Rahim et al. in 2013. Originally, the LBP concept was discovered by Ojala et al. in 2002, when they first discovered the LBP operator's high potential for the use in texture classification [22]. In the feature extraction process of LBP, the pixels of a face image are labelled by thresholding them against other neighbouring pixels. This step is generally done with a 3X3 pixel intensity matrix [23]. The results of this comparison are converted to the binary numbers 1 if the current pixel's intensity is more than the centred pixel's intensity and 0 if otherwise [24]. Then, the binary numbers are further converted to decimal values, and a concatenated histogram is generated to represent the pixel intensities. In the feature matching process of LBP, the histograms of the current image and the images in the face dataset are compared. Here, different approaches can be used, for example, Euclidean distance, absolute value, chi-square, and etcetera. In real-world applications, LBP is often preferred for complex, real-time recognition tasks due to its tolerance towards background light variations, ageing faces, and face rotations. Figure 2.5 shows an outline of the LBP algorithm.

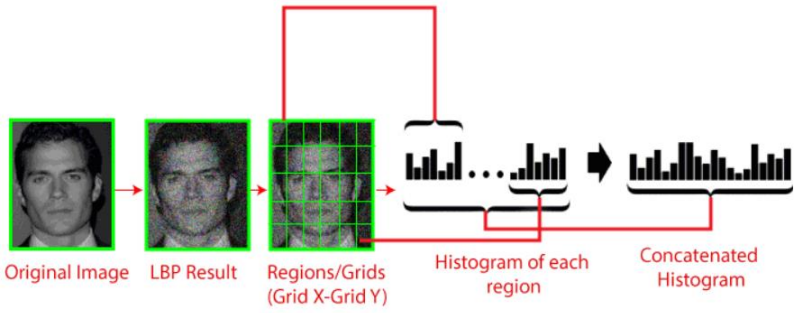


Figure 2.5 LBP Algorithm [25]

Furthermore, one of the more popular face recognition methods for noisy face datasets is the Artificial Neural Network or ANN for short. In earlier years, neural networks were only used for face detection, but not recognition. It was only in 2016 when Kasar et al. first proposed the use of ANN for face recognition [26]. An ANN model mimics a biological neural network through the use of a network of "nodes", which are artificial neurons. These nodes are all interconnected, and they have weights that determine the strengths of the connections. The node levels can be categorised into the input level, hidden levels, and output level. In ANN, an algorithm called backwards propagation of error, or backpropagation for short is used as the set of "learning" rules [27]. The backpropagation algorithm helps during the model training phase, where ANN starts with the output units and slowly traces its way back to the input units while adjusting the weights of the connections as it goes [28]. During the actual face recognition process, ANN is able to use binary formats to navigate through the neural network and output the results. The advantage of using ANN is that it is capable of learning complex, non-linear relationships between the input variable and output variable. This is especially useful in real-time scenarios such as face recognition attendance systems. However, to have high accuracy, ANN requires a massive database with possibly millions of face samples [26]. Hence, model training becomes a very time-consuming process.

Another type of neural network that is prevalent in the field of machine learning is the Convolutional Neural Network (CNN). Unlike ANN, the hidden layers in CNN can consist of multiple types of layers such as pooling layers, normalisation layers, convolutional layers, and fully-connected layers [29]. In the feature extraction process of CNN, kernels are used as "filters" to excerpt the input image's significant features hand-in-hand with the convolutional layer. The result of the feature extraction process

is known as a feature map. Next, the pooling layer's main task is to reduce the spatial size of the feature map. The purpose of this layer is to decrease the overall computational power needed for the model training process. The following layer is the fully-connected layer. The mentioned layer is responsible for the classification process. Similar to ANN, the fully-connected layer of CNN is able to learn the non-linear combination of the facial features represented by the feature map and produce the predictions. Generally, CNN provides high accuracy and is considered to be more powerful than ANN in terms of performance [30]. However, the disadvantage of CNN is that it also requires large amounts of training data, much like ANN.

Table 2.2 summarises the advantages and disadvantages of the face recognition methods described above as well as a sample accuracy as experimented by Kamila [31] and Islam et al. [30] in the year 2016 and 2017, respectively. These two papers performed their face recognition model testing using the AT&T public database of faces [32]. The AT&T face database consists of 400 92X112 pixels face images that are in greyscale.

Table 2.2 Face Recognition Algorithm Comparison [30] [31]

Algorithm	Pros	Cons	Accuracy (%)
Principal Component Analysis	Robust High computational speed	Accuracy depends solely on the training dataset	77.97
Linear Discriminant Analysis	Able to recognize images that vary in light intensity, face expression, and pose	Requires many images of the same individual for training purposes Accuracy depends on the dataset more than that in PCA	82.45
Local Binary Pattern	Tolerant towards background light variations, ageing faces, and face rotations	Longer model training compared to PCA and LDA	90.93
Artificial Neural Network	High accuracy when given a large training dataset	The model training process is time-consuming Need extremely large dataset for high accuracy	96.00
Convolutional Neural Network	High accuracy and has better performance than ANN		94.00

2.1.4 Factors Causing Face Recognition Difficulties

Much research has been done in the face recognition field to achieve 100% recognition accuracy. However, multiple environmental factors still hinder the overall accuracy of current face recognition systems. Based on Anwarul and Dahiya's paper in 2019 [33], the factors that cause difficulties in face recognition can be separated into two groups – extrinsic and intrinsic. Intrinsic factors are factors relating to the human face itself, such as ageing, facial expressions, and plastic surgery. On the other hand, extrinsic factors are background factors such as low resolution, occlusion, illumination, noise and pose variation. Table 2.3 tabulates the factors mentioned, as well as their descriptions.

Table 2.3 Factors Hindering Face Recognition [33]

Category (Intrinsic / Extrinsic)	**Factor**	**Description**
Extrinsic	Occlusion	Objects that cover parts of the face, such as sunglasses and scarves.
	Low Resolution	Low-resolution image inputs from cheap surveillance cameras are difficult to compare to high-resolution images in the face dataset
	Noise	Digital noise, also known as virtual distortion, can come in many forms. Generally, digital noise originates from the image capturing process
	Illumination	Illumination comes in the form of background lighting, brightness, shadow, contrast, and any other light-related factor.
	Pose Variation	Any non-frontal face is considered to have pose variation. The majority of face datasets are trained on frontal face images, so pose variation may pose a challenge for many face recognition models
Intrinsic	Facial Expression	Even a tiny variation in faces can create ambiguity for face recognition models due to the slight change in the positioning of eyebrows, lips, cheeks, and etcetera.
	Ageing	Over time, a person's face may undergo physical maturing, such as the formation of wrinkles or the change in face

		shape. In face recognition datasets, the stagnant face images do not age, so the model might have problems recognising aged faces.
	Plastic Surgery	After plastic surgery, a face's features may be totally different. Thus the face recognition model might treat it as a foreign face altogether.

2.2 Current Attendance Taking Systems in Classrooms

Ever since the technology development surge at the turn of the 21st century, people have been coming up with innovative methods to automize the attendance-taking process in schools. The vast range of attending-taking systems vary from cheap and simple pen-and-paper attendance to sophisticated and computerized biometrics attendance, RFID (Radio Frequency Identification) based attendance, GPS (Global Positioning System) based attendance and etcetera. Table 2.4 summarizes the main pros and cons of current attendance-taking systems from a paper written by Katara et al. [34].

Table 2.4 Types of Attendance System [34]

Type of Attendance System	**Advantages**	**Disadvantages**
Manual (Pen and Paper, Name Calling)	No technology expenses involved	Possible human errors Time-consuming
RFID Card	Simple to implement	The system is prone to manipulation
Fingerprint Recognition (Biometric)	Harder to forge; not transferable (increased security)	Exclusion problems with older people or people with skin problems
Iris Recognition (Biometric)	Mathematical patterns of iris stay constant throughout a lifetime, Increased reliability	Obstacles such as contact lenses, artificial lenses, and reflections might cause mistakes
Voice Recognition	-	Not accurate in noisy classrooms
Bluetooth & GPS Based	Low power consumption	Easily forged using GPS spoofing
QR Code	Quick and Efficient	Easily forged
Face Recognition	Fast and accurate No human-sensor contact needed	May be hindered by environmental factors

As seen in the comparison table, all attendance-taking methods have their own pros and cons. However, face recognition possesses an adequate compromise between computation time, cost, and accuracy as its main advantage. This main advantage of

face recognition, as stated by Katara et al. [34], makes it the most suitable biometric recognition technique to be included in the proposed mobile application for class attendance. As for the main disadvantage of face recognition, as mentioned above, potential mitigation techniques will be explored further in the development of this project.

2.3 Review of Existing Face Recognition Applications for Class Attendance

In this section, the existing face recognition applications for class attendance are reviewed in terms of their implementation methods, strengths, and weaknesses. The face recognition applications to be reviewed in this section varies from computer-based applications to mobile-based applications. Moreover, some of the applications discussed below utilise still images for the face recognition process. In contrast, others make use of video sequences and are even capable of handling real-time, video-based face recognition. The reason behind selecting a variety of face recognition applications is to explore the previous research and development projects done in the application of face recognition in taking class attendance, regardless of user interface platforms. In the following sections, the face recognition applications to be reviewed are categorised into computer-based and mobile-based. These two categories are then further split into still image-based and video-based.

2.3.1 Computer-Based Face Recognition Applications

Computer-based face recognition applications refer to desktop applications or web-based applications that utilize face recognition technologies. Generally, the advantage of computer-based face recognition applications is that there are more available face recognition libraries and resources available for use on computer-based applications rather than mobile-based applications.

Moreover, computers have higher processing powers when compared to the average, everyday mobile device. At times, the limited computational speed of mobile devices hinders their abilities to handle sophisticated tasks, such as face recognition, when the dataset involved is huge. Therefore, many face recognition tasks have been carried out through computer-based applications or systems. In computer-based face

recognition applications, the face recognition process can be based on still images or real-time videos, both of which are reviewed in the following sections.

2.3.1.1 Image-Based Face Recognition

Most computer-based face recognition systems utilize still images to capture the attendance of students in a class. For example, in 2012, Balcoh et al. proposed an external wired camera setup that was used to feed in images to a computer, which housed the modules needed for the face recognition process [35]. The modules in the computer, in their respective orders, were the image enhancement module, a face detection module, a face recognition module, and finally, the attendance marking module. Figure 2.6 below illustrated the setup and summarized flow of Balcoh et al.'s system.

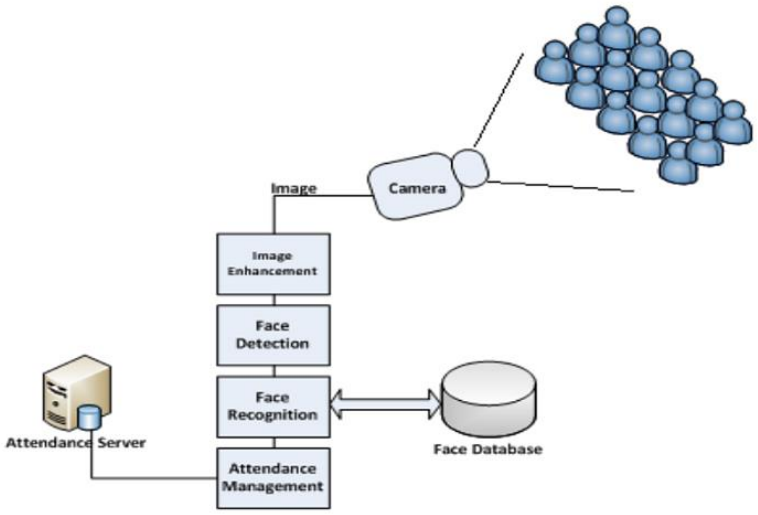


Figure 2.6 Setup of Balcoh et al.'s Attendance System [35]

The attendance system proposed by Balcoh et al. utilised the Viola-Jones method in the initial face detection process, and the Eigenfaces face recognition method later on. In the face detection process, Balcoh et al. used skin classification techniques to increase the accuracy of the Viola-Jones face detection model. Then, the cropped images were feed into the face recognition module. Next, the face recognition process in the system identifies the students based on a database of faces. Then, the attendances of the students were recorded in another database, where both the students and the

parents were allowed to view them. The recognition accuracy of this model was 85% for unveiled students with no beards.

Eight years later, in 2020, Khan et al. [36] proposed an attendance-taking system using a similar setup with a different implementation method. Khan et al.'s web-based face attendance application's GUI was done through Tkinter, with the integration of SQLite 3 for face databases' managements. The modules of this system included face detection, model training, student counting, and face recognition. Figure 2.7 below shows the block diagram of the attendance system developed by Khan et al.

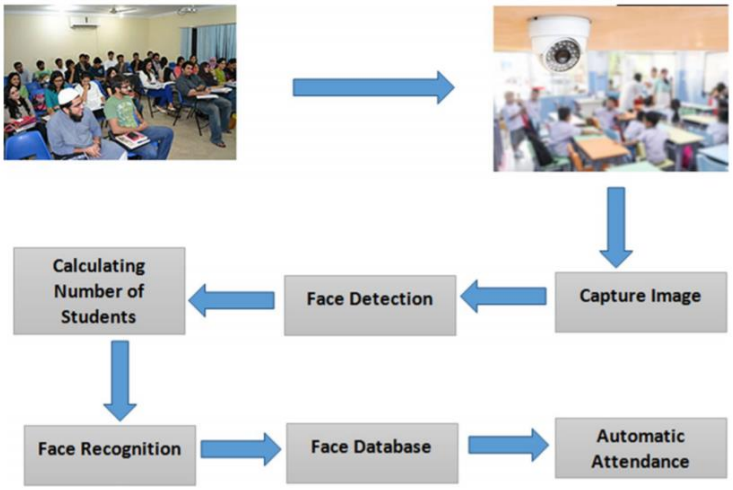


Figure 2.7 Block Diagram of Khan et al.'s system [36]

Both the image-based face recognition attendance systems shared similar advantages. The strength of the two systems was that they were able to record the attendance of all the students using only one picture. This method was time-consuming as teachers did not have to take each student's picture manually each time a class was conducted. In addition to this, Khan et al.'s attendance system was able to effectively count the number of attendees, absentees, and unknown faces in a class image.

On the flip side of the coin, the two attendance systems also exhibited disadvantages linked to the use of still images. Both the attendance systems required all students to be present in the same place at the same time in order to record the attendance. Moreover, the students had to line up for their pictures to be taken by the camera. Hence, these attendance systems were not suitable for large lecture classes with over 100 students as it would be hard for lecturers to arrange a large number of students to fit into a single frame at the same time.

2.3.1.2 Video-Based Face Recognition

To mitigate the inconvenience of using still images for the face recognition process, some researchers came up with the idea of using video-based face recognition in computer-based attendance systems. For example, in 2017, Raghuwanshi and Swami came up with an automated attendance system using video inputs and excel sheet outputs [37]. Similar to previously reviewed systems, the setup of Raghuwanshi and Swami's attendance system included a high-definition camera and a PC with MATLAB installed. The attendance system consisted of 5 consecutive modules – face database creation, video recording, face detection, face recognition, and attendance registration. Figure 2.8 below illustrates the block diagram of the video-based attendance system.

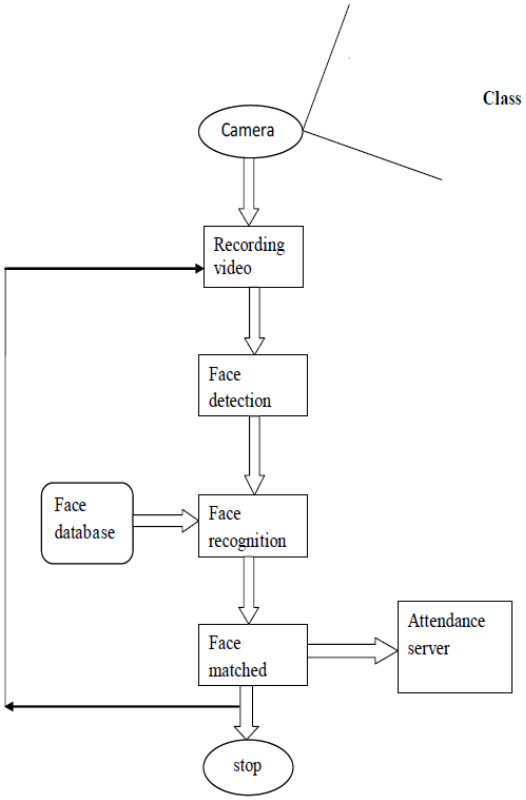


Figure 2.8 Block Diagram of Raghuwanshi and Swami's Attendance System [37]

Raghuwanshi and Swami's attendance system collected five images of each student to train the recognition system. The input of the attendance system was the recording video from the external camera. Once the video was received, the system read the frames one by one and send each frame for face detection, which utilized the Viola-

Jones algorithm. After the face detection process, the cropped faces were extracted and saved to the face dataset, which was saved on the PC itself. For the face recognition process, PCA and LDA algorithms were used. The feature extraction was performed using Eigenfaces and Fisherfaces subspace projection. Furthermore, the face matching process was performed using the Euclidean distance classifier. The last step of the attendance process was to mark the attendance into an exportable excel workbook. The recognition of the LDA algorithm was seen to be 83.57%, which was higher than the PCA algorithm, which stood at 66.07%. However, the paper concluded that the PCA algorithm was faster than the LDA algorithm by 81.94% [37]

Recently, in 2020, Smitha et al. proposed another computer-based face recognition system that is able to take class attendance using live-streaming videos [38]. The set-up for this face recognition system consisted of a computer and a webcam. The GUI of the face recognition system allowed the user to perform three different operations – student registration, faculty registration, and attendance marking. Figure 2.9 below shows the system architecture for Smitha et al.'s attendance system.

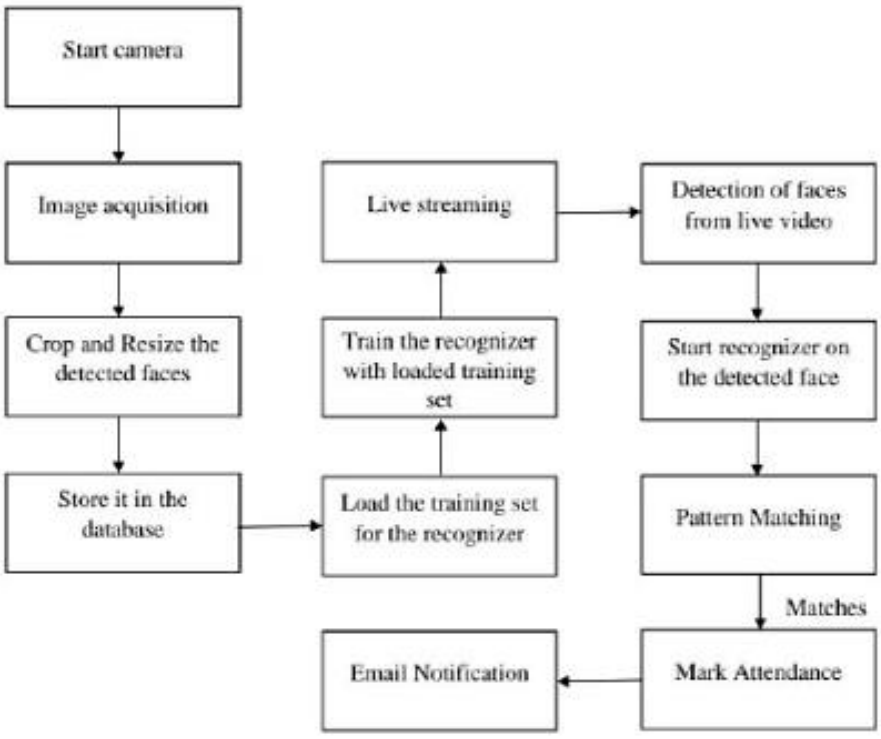


Figure 2.9 System Architecture of Smitha et al.'s Attendance System [38]

The student registration option triggered one of the modules of the system – dataset creation. In this process, 60 face images of the student were captured automatically and stored in the local database along with the student's details. As for the faculty registration, the faculty details, course information, class information, and lecturers' emails were collected and kept in the system's database. Next, the attendance marking option triggered the main modules of the system – face detection, face recognition, and attendance updating. The algorithm choices for the face detection and face recognition process were the Haar-Cascade classifier and the LBP Histogram, respectively. After identifying the students, the attendance was updated in real-time via a Microsoft Excel spreadsheet. The accuracy of this face recognition model was 82%.

Both the video-based face recognition attendance systems had similar advantages. First of all, the use of a video-based input meant that the students did not have to squeeze into a single frame, as seen in attendance systems with image-based inputs. In terms of real-life usages, these two attendance systems were more plausible as students tended to enter the class at different times. With video-based attendance systems, the attendance of students who arrived early and later could be recorded separately in two streams of video. Additionally, the newer attendance system created by Smitha et al. was able to automatically notify the course's lecturers of the list of absentees via email. This made it easier for lecturers to contact students who were constantly absent from class.

However, the two video-based attendance systems reviewed shared a disadvantage between them that revolved around the use of a webcam. In image-based attendance systems, the students are typically positioned far from the webcam during the attendance-taking session, with the teacher in charge of the webcam at all times. On the contrary, with video-based attendance systems, the students generally had to walk past the webcam at a close distance in a single file. This made the webcam vulnerable to damage that may be caused by younger students without adult supervision. In the long run, the use of web cams was costly and not feasible for schools on a smaller budget.

2.3.2 Mobile-Based Face Recognition Applications

With the ever-increasing popularity of mobile phones, it is more surprising to find a person without a mobile phone than with one. According to the report by the Department of Statistics Malaysia ,in April 2020, around 98.2% of Malaysians own mobile phones, whereas only 71.3% of Malaysians own computers [39]. With this thought in mind, many developers have been coming up with ways to integrate complex technologies which were once only runnable on powerful GPUs into the tiny gadgets right at a user's fingertips. Face detection and recognition are some of the most commonly used technologies in mobile applications, mainly for security and entertainment purposes. Pertaining to this paper, there are already several face recognition mobile applications being used for taking class attendance. Three of the more recent applications will be discussed and reviewed in the following section in terms of their implementation methods, advantages and limitations.

2.3.2.1 Image-Based Face Recognition

In 2017, Samet and Tanriverdi proposed a trio of mobile applications that worked cohesively to automate the attendance-taking process in schools [40]. The three different mobile applications with different built-in modules were to be downloaded by students, parents, and teachers, respectively. Through the teacher's module, teachers were able to take class photos of all the students together for attendance taking. In the student's module, students were able to sign in to their respective courses and upload pictures or 3-second videos for their attendance to be taken. As for the parent's module, the students' immediate family members were able to keep track of their child's attendance records. The result of the attendance-taking process was viewable in all three of the mentioned modules. Next, all three of these modules communicated with an external server via RESTful web services. Each client-server request was transmitted in JSON format via the POST method. The external server was in charge of handling the face detection and recognition process, as well as storing large databases. Figure 2.10 below illustrates the system architecture of Samet and Tanriverdi's attendance mobile application.

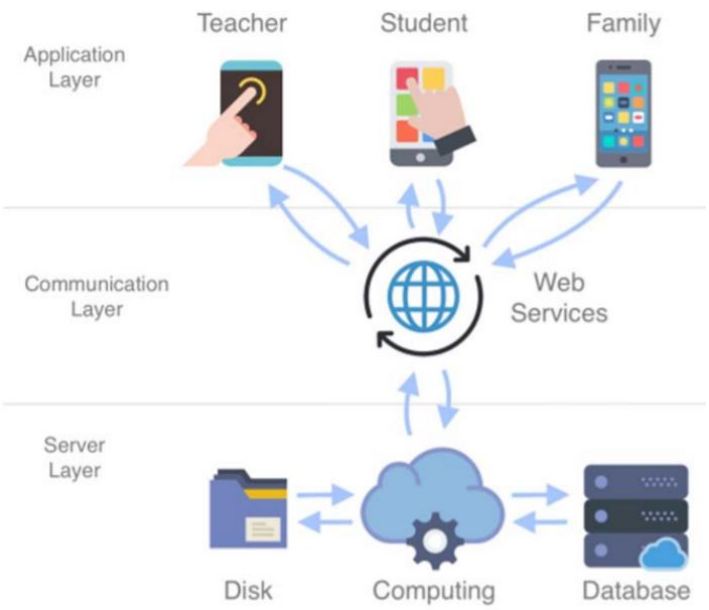


Figure 2.10 System Architecture of Samet and Tanriverdi's Attendance Application [40]

In this paper, several algorithms were tested for the face recognition attendance-taking process. However, the final algorithms selected for the face detection and face recognition process were the AdaBoost-assisted Viola-Jones algorithm and the Local Binary Pattern algorithm. The accuracy of the final face recognition model was 84.81%.

Furthermore, in 2019, Sunaryono et al. proposed the development of an Android application that was able to take attendance using still image-based face recognition and QR code technologies [41]. The hardware required in the attendance-taking process were an Android mobile phone, a Raspberry Pi screen, and a desktop-based server. Sunaryono et al.'s proposed mobile application had two separate modules for both students and lecturers. In both modules, the mobile application had to connect to an external computer-based server that was in charge of processing the face recognition process, providing course information, and generating the QR code. Before the attendance could be taken, students had to capture ten frontal face images using the mobile application for model training purposes. During the attendance-taking process, students had to scan the given QR code generated by the lecturers' module before they were prompted for the face recognition process. Both students and lecturers could track the results of the attendance via the mobile application itself. Figure 2.11 below shows the system architecture of Sunaryono et al.'s attendance application.

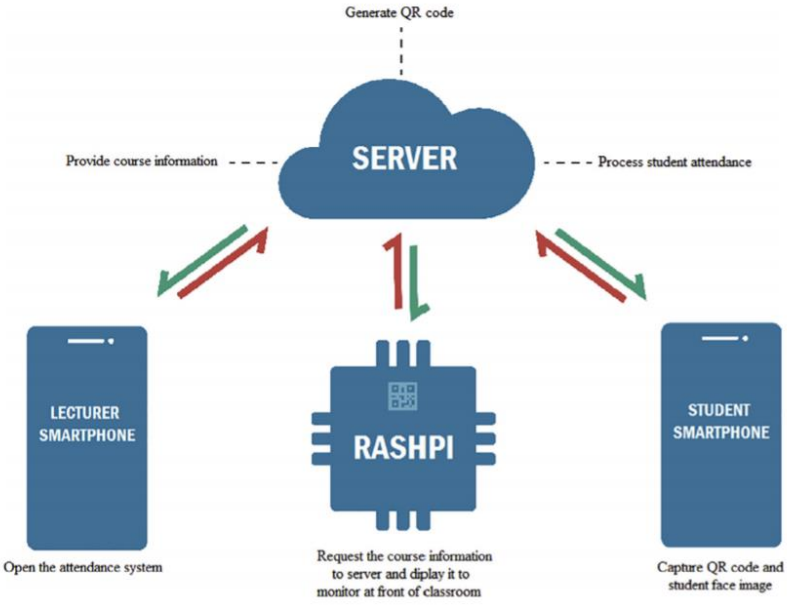


Figure 2.11 System Architecture of Sunaryono et al.'s Attendance Application [41]

The purpose of the Raspberry Pi monitor was to communicate with the server, retrieve the QR code and project it for students to view. The implementation of the server's web client application was performed through the use of Volley (an HTTP library) developed in PHP language. The server also integrated MySQL for database managing purposes. The algorithms selected for the face detection and face recognition process were Viola-Jones and Linear Discriminant Analysis, respectively. The accuracy of the face recognition mobile application stood at 97.29%, with a 0.000096s time needed to communicate with the external server.

One year later, in 2020, Isinkaye et al. proposed a face recognition mobile application using still images to mitigate attendance malpractice in schools [42]. The hardware needed for this mobile application was only an Android phone. The two key modules in this mobile application were the registration module and the attendance verification module. In Isinkaye et al.'s proposed attendance-taking procedure, only the teacher had to install the mobile application. Then, the teacher would be in charge of registering the students' details and taking the students' attendance. The attendance records were taken by taking a picture of the student's face and manually pressing the "proceed" button once the student's face has been detected. Other sub-modules of Isinkaye et al.'s proposed mobile application included the student registration module and the view attendance module. Figure 2.12 below illustrates the system architecture of the attendance application.

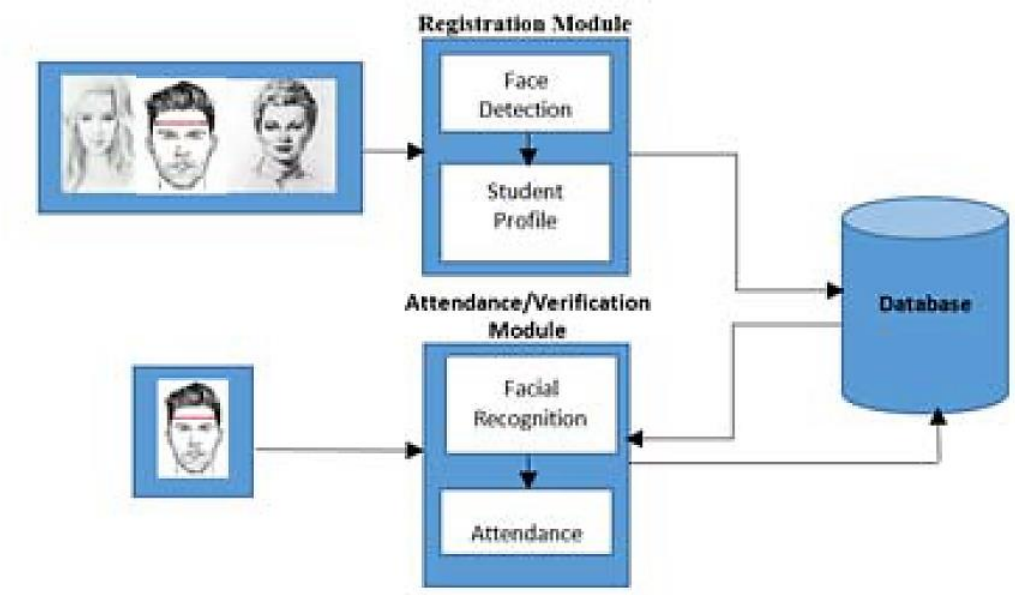


Figure 2.12 System Architecture of Isinkaye et al.'s Attendance Application [42]

The algorithms chosen for the face detection and face recognition process were the AdaBoost-assisted Viola-Jones algorithm and the Eigenface algorithm, respectively. The implementation of the face recognition attendance-taking system was done through the use of Android Studio, XML, and SQLite DatabaseS. Android Studio and XML were utilized cohesively to create and style the user interface of the mobile application. On the other hand, SQLite Database was embedded into the mobile application itself to manage the face databases needed for the face recognition process. The results of the attendance-taking process were displayed in the mobile application itself. The recognition accuracy of the face recognition process was 95%, whereas the face detection accuracy was 78%.

All three of the image-based face recognition attendance applications had their own strengths in terms of convenience and security. The face recognition mobile application proposed by Sunaryono et al. [41] used QR codes to trigger the face recognition process for attendance in the students' phones. Moreover, the mobile application used only campus intranet communications among the devices to prevent off-campus students from accessing the QR codes. Similarly, the face recognition mobile application proposed by Samet and Tanriverdi [40] made use of the school's network by using unique web tokens during the attendance taking process. Moreover, for privacy purposes, only the teacher would be allowed to view the class photos and the individual student face photos. On the contrary, the face recognition mobile

application proposed by Isinkaye et al. [42] controlled the application's security by only allowing teachers to take attendance through a single device. Students were not required to bring their own devices to class.

However, all three of the image-based face recognition attendance applications shared similar disadvantages. Firstly, similar to computer-based attendance systems using still images, these three systems required for students to all be present at the same time for the attendance-taking process. Secondly, the mobile applications proposed by Sunaryono et al. [41], Samet and Tanriverdi [40] required students to bring their own mobile phones to class, which was not feasible for younger students below secondary school level. On the flip side of the coin, the mobile application proposed by Isinkaye et al. [42] allowed the attendance taking to be performed from a single device only. However, the lack of communication with an external server may lead to sluggish performances. Since the mobile device in this implementation method had to store all of the face images and student databases, the overall system's performance might deplete due to the intense use of storage capacities.

2.4 Summary of Literature Review

Table 2.5 summarizes the advantages, disadvantages, algorithms used, and recognition accuracies of the six face recognition applications reviewed in section 2.3:

Table 2.5 Types of Face Recognition Attendance Applications

Application Proposed by	Face Detection Algorithm	Face Recognition Algorithm	Advantages	Disadvantages	Recognition Accuracy
Computer-based; Still Images [35]	Viola-Jones	Eigenfaces	Able to capture the whole class's attendance in one frame Able to automatically notify teachers of absentees	Required all students to be captured in one frame Bulky set-up caused congestions in classrooms	85%
Web-Based; Still Images [36]	YOLO V3	Microsoft Azure Face API			100% (Max 12 students)
Computer-Based; Live-Video	Haar-Cascade Classifier	LBPH			82%

[38]					
Computer-Based; Recorded-Video [37]	Viola-Jones	PCA and LDA			83.57% (LDA), 66.07%. (PCA)
Mobile-Based; Still Images [41]	Viola-Jones	LDA	Higher security due to the use of both face recognition & QR code	- Students had to bring their own mobile devices for attendance taking - The teacher module required all students to be captured in one frame	97.29%
Mobile-Based; Still Images [42]	AdaBoost-assisted Viola-Jones	Eigenfaces	All modules could be accessed from one mobile device only		78%
Mobile-Based; Still Images [40]	AdaBoost-assisted Viola-Jones	Local Binary Pattern	Higher security due to the use of private POST requests for client-server communication		84.81%

As reviewed in section 2.3, computer-based face recognition applications required bulky and costly equipment, and the lack of portability made it inefficient as a replacement for manual attendance-taking methods. On the other hand, even though the use of mobile applications solved these issues, they still do not fully automate the attendance-taking process completely. In the mobile applications reviewed earlier, teachers still had to manually capture the images of students for the attendance to be taken. In order to fully automate the attendance-taking process and bring convenience to both students and teachers, this paper realizes the use of face recognition in a mobile application using live-streaming videos for real-time attendance taking. Inspired by the quick computational speed of previous face recognition mobile applications, the developed face recognition application also uses an external server for the management of the databases used. The methodologies used in this project's implementation are discussed further in the following chapter.

Chapter 3

System Methodology/Approach

In this chapter, the system methodologies and overview of each module in the project are introduced and discussed briefly before diving deeper into each module in the following chapters. Wrapping up this chapter are the hardware and software requirements for the developed application as well as the project setup.

3.1 System Methodology

The developed application is a real-time face recognition mobile application named FaceIt. There are five main modules in the developed application – the Login and Register module, Class module, Student module, Enrolment module and the Attendance module. In this section, the general flow of the entire application is explained with the help of a block diagram.

The developed application has five modules that can be presented as a flow of events. Figure 3.1 below shows the block diagram for the application.

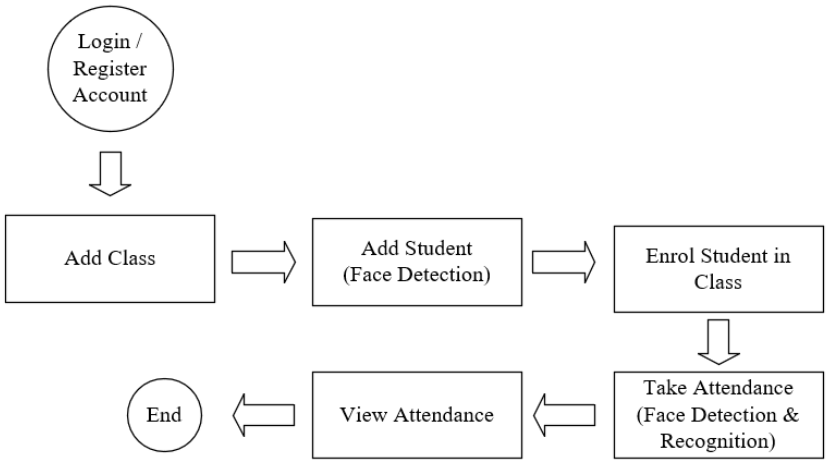


Figure 3.1 Overall Block Diagram of FaceIt

As evidenced by the figure above, the five modules of the developed application each appear as a part of the system's general flow. First, users have to either login into an existing

account or register a new account with FaceIt. After that, a first-time user would have to add a new class into the system through the Class module. Then, the user would have to add a new student into the system through the Student module. The Student module uses face detection technologies to obtain the face image of the new student. Subsequently, the new student would have to be assigned to a class. This is where the Enrolment module comes in. After the student is enrolled into an existing class, the Attendance module can be triggered during physical classes in order to record the students' attendance. In the block diagram above, the Attendance module is shown as two separate blocks – take attendance and view attendance. The attendance-taking functionality uses face detection and recognition technologies in order to capture the students' attendance records with ease.

In the following subsections, the general workflows of the five modules are described.

3.1.1 Login and Register Module

The Login and Registration module is the very first module that the users dawn upon when they launch the developed application, FaceIt. This module allows users to login into an existing account or registers a new account in FaceIt using Firebase Authentication. Listed below are the main functionalities that come under the Login and Registration module:

1. **Login** – Users can log into FaceIt using an existing account (Email/password or Gmail)
2. **Register** – Users can register a new account with FaceIt (Email/password or Gmail)
3. **Forgot Password** – Users can reset their passwords at the Login interface if they ever forget their passwords.
4. **Profile Management** – Once logged in, users can view and edit their personal details, including name, email, password and profile picture.
5. **School Management** – Once logged in, users that categorize under the "Admin" role are able to view the school authorization code and the list of accounts registered under the current school.

Figure 3.2 below shows the flowchart of the main functions in this module – the Login function and the Register function:

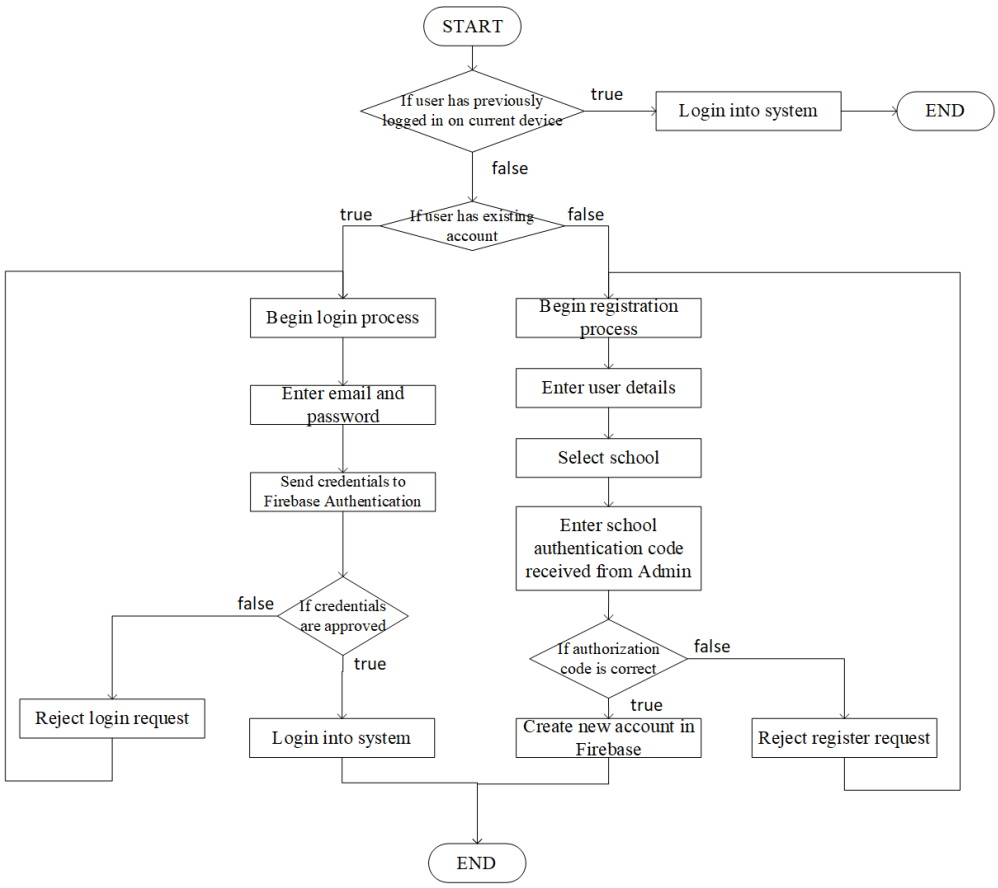


Figure 3.2 Login and Registration Flowchart

First, upon opening the application, FaceIt checks if the user had previously logged into the application on the current mobile device. If so, FaceIt redirects the user straight to the main dashboard interface. If this is the user's first-time logging into the application, then the user is presented with the Login interface. If the user has already registered an account with FaceIt, they can use their login credentials (email and password) to log into the application. If the user does not have a FaceIt account, they can register a new account in the Register interface. During the registration process, users are prompted for their personal particulars. Then, users have to choose the school that they wish to register an account under and provide the school authorization code which is unique to each school. In this project, it is assumed that the school's head of administrator will provide users with the school's authorization code. Once FaceIt authenticates the user's entered details, a new account will be created, and the user will be logged in immediately.

As listed in the functionality list earlier in this subsection, the two submodules that come under the Login and Registration module include School Management and Profile Management, each of which are described in detail in Chapter 4.

3.1.2 Class Module

The next module in the general workflow is the Class module. The Class module allows users to manage the classes used in the application. The Class module is crucial as it houses the class list selection page, which is used to initiate the Enrolment and Attendance modules later on. Without any existing classes added, users are not able to access the Enrolment and Attendance modules. Listed below are the functionalities that are included in the Class module:

1. **Add Class –** Users can add a new class into the application's database by entering the new class's code, name, semester, year, lecture/tutorial/practical (LTP) group, and description. Each newly added class can only be accessed, viewed, and updated by the user who created it. The application ensures that no duplicated classes are added by the same user.
2. **View Class –** Users can view a list of previously added classes and all the details. This functionality also has an option that redirects the user to the Enrolment module.
3. **Update Class –** Users can update the class code, class name, LTP Group, and description of a class
4. **Delete Class –** Users can delete an existing class. Upon deletion of a class, all the related enrolment records are also deleted.
5. **Search Class –** Users can search for a particular class in the list of classes using the class code or class name.

3.1.3 Student Module

After adding a class, users can proceed to the Student module, where they have to add new students into the application's database. In this module, Firebase ML Kit's Face Detection API is used to detect the faces of the student to-be-added. The functionalities of the Student module are listed below:

1. **Add Student –** Users can add a new student into the application's database. This functionality consists of two activities. The first activity prompts the user to enter the new student's details, and the second activity utilizes the user's mobile phone camera and Firebase's Face Detection API to capture the face image of the new student.
2. **View Student –** Users can view the list of students that were previously added. Users can only view students within the same school.
3. **Update Student –** Users can update the details of an existing student, including student ID, student name, phone number, email address, and face image.
4. **Delete Student –** Users can delete an existing student. Upon deletion, the enrolment records associated with the selected student are deleted as well.
5. **Search Student –** Users can search for an existing student in the student list using the student ID or student name.

The main functionality of the Student module is Add Student. Figure 3.3 below illustrates the general flow of the Add Student process:

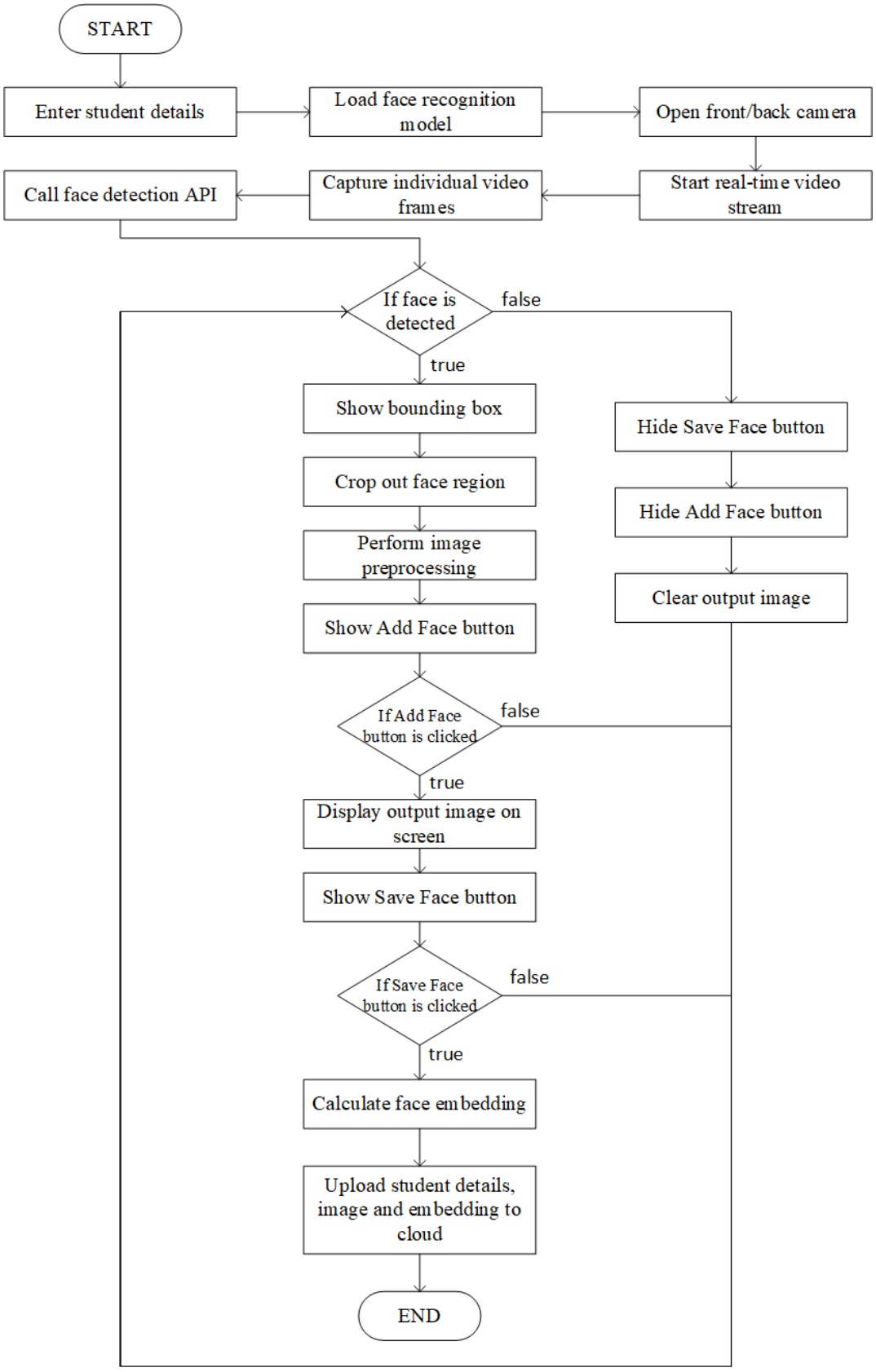


Figure 3.3 Add Student Flowchart

As evidenced in the flowchart above, the first step in the Add Student process is for users to enter student details such as the student's full name, gender, phone number, email and the student ID of an identical sibling. The last field mentioned is optional but plays an important role in the application's ability to detect the presence of identical siblings. The next step is for the new student to add a face image – When the student's face is detected, a bounding box will be shown. Then, the face image portion is cropped out, processed, and saved as a face embedding in the application's database.

3.1.4 Enrolment Module

Having added classes and students into the application, users can now proceed to enrolling students into the existing classes available through the Enrolment module. The functionalities of the Enrolment module are listed below:

1. **View Enrolled Students –** Users can view the list of enrolled students in a particular class
2. **Add Enrolment –** Users can enrol students from the current school's complete student list into a particular class.
3. **Delete Enrolment –** Users can unenroll a student from the enrolled student list of a particular class.
4. **Search Students –** Users can search for students by name or by student ID when trying to enrol or unenroll students.

3.1.5 Attendance Module

Having enrolled students into existing classes, user can now use the Attendance module to handle the attendance-taking process. The Attendance module is the final and most crucial module in FaceIt. In the attendance module, Firebase ML Kit's Face Detection API and a face recognition model, MobileFaceNet, are used to recognize the faces of enrolled students and take the attendance of the students automatically. Listed below are the functionalities of the Attendance module:

1. **Take Attendance** – Users can take the attendance of the students in a class using a real-time video stream. The attendance of the student is taken and stored in the application's cloud database, completed with the student's ID, name, and the timestamp of the attendance.
2. **View Attendance** – Users can view the previously taken attendance records, categorized by class and by date.
3. **Add Attendance (Manual)** – Users have the option to add in attendance records manually in the event that a student arrives late and misses the attendance-taking session.
4. **Generate Attendance Report** – Users can generate an Excel sheet populated with the attendance records of a particular class on a particular day.

The main functionality of the Attendance module is Take Attendance. Figure 3.4 below shows the flowchart of the Take Attendance functionality:

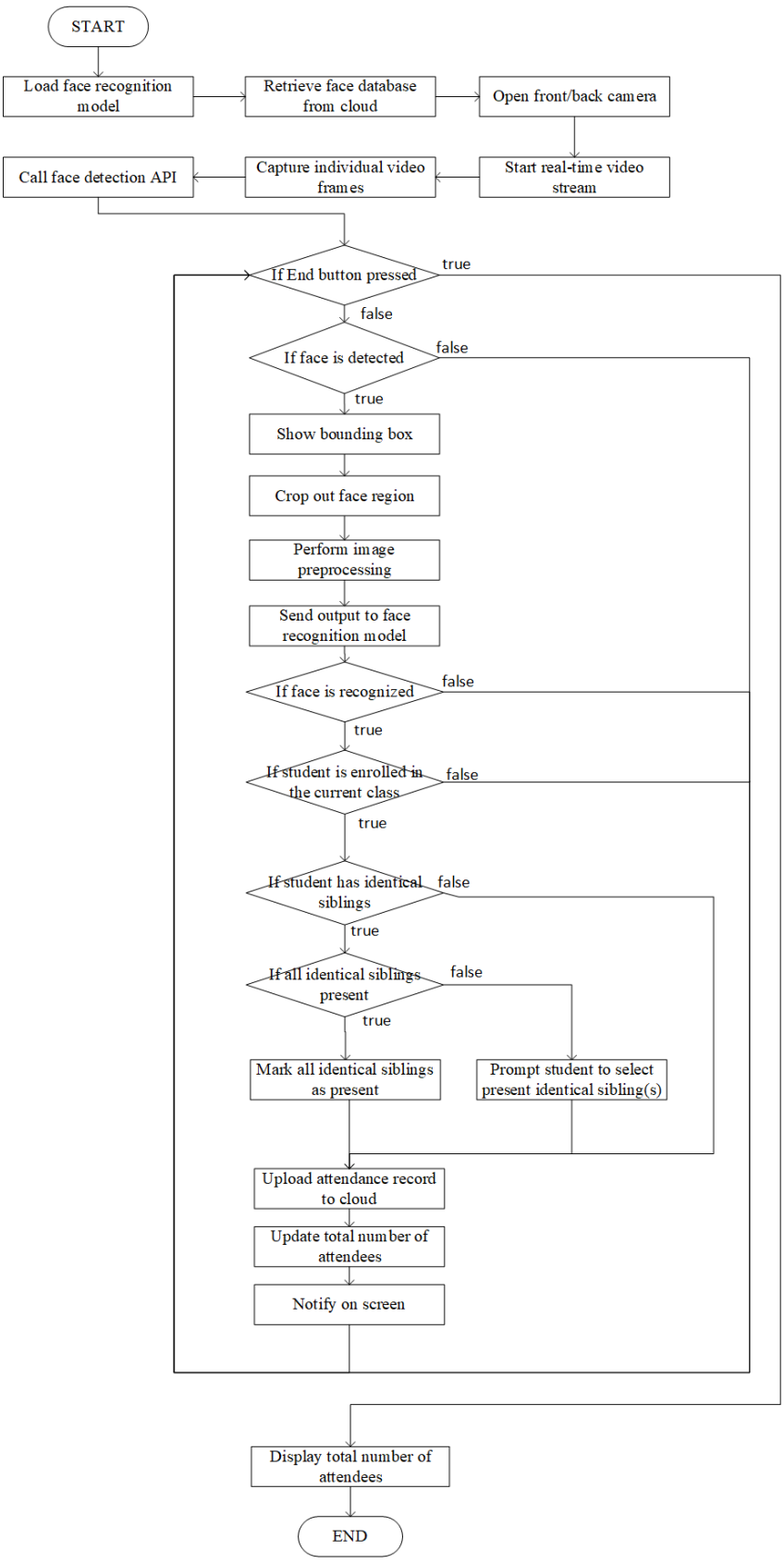


Figure 3.4 Take Attendance Flowchart

During the Take Attendance process, the first step is for the application to load the face recognition model. Then, the application fetches the face embeddings from the cloud database and starts up the user's mobile phone's camera. During the real-time video stream, the application captures individual video frames and detects if there are any students present in frame. Once a student's face has been detected, the face region is cropped out, processed, and recognize. Then, the application checks to see if the recognized student has any identical siblings. If so, the application prompts all the identical siblings to stand in the same frame for the attendance to be taken. If one or more identical siblings are absent, the application displays a checklist and prompts the student to tick the present identical sibling(s). After that, the attendance records of the present identical siblings are uploaded to the cloud database and the student is notified on screen.

On the other hand, if the student recognized on screen has no identical siblings, then the application takes the attendance for the student straight away, notifies the student and uploads the attendance record to the cloud database. This process repeats for every video frame until the teacher ends the attendance-taking process. Once the attendance-taking process has been terminated, the application displays the total number of present students and returns to the main menu.

Before proceeding to the detailed implementation elaborations for each module in the following chapters, the hardware and software requirements have to be first understood. The next section lists out the tools used, including hardware, software and external libraries called in the application.

3.2 Project Setup

In this section, the hardware and software requirements of the developed application in this project are discussed in terms of the tools used and the reasons for selection. Then, the last subsection explains the external libraries used in the project as well as some key settings and configurations of the development environment.

3.2.1 Hardware Setup

In this project, the hardware used in the development process are a laptop and an Android smartphone. However, for end users (teachers, lecturers, and school admins), only an Android smartphone is needed to operate the application. Table 3.1 below shows the minimum and recommended specifications for the hardware used in the development of this project:

Table 3.1 Hardware Specifications

Device	**Specification**	**Minimum Requirement**	**Recommended Requirement (Used in this Project)**
Laptop	Operating System	-	Windows 10 (64-bit)
	Memory	-	12GB
	Graphic Card	-	NVIDIA GeForce 940MX
	Storage	-	10GB available space
Smartphone	Android Version	Android Oreo	Android 11
	Memory	4GB	6GB
	Storage	1GB available space	3GB available space
	Camera	13MP	20MP

3.2.2 Software Setup

Next, the various software and technologies used in this project and the reason for selection are listed below.

1. Android Studio

Android Studio is Google's official IDE for the Android mobile operating system. In this project, Android Studio is utilized along with the Java programming language for mobile application development due to its IDE stability and easy-to-use interface.

2. Firebase

Firebase is a developer toolset created by Google that provides a wide variety of services such as analytics, authentication, real-time database, file storage, machine learning, and etcetera. The Firebase features utilized in this project are as listed below:

- **Authentication** – Used to allow users to log in or register into FaceIt using email/password or Gmail integration.
- **Firestore Database** – A NoSQL, real-time database used to store all the information used in the developed application except image files.
- **Firebase Storage** – A cloud file storage feature used to store image files such as the face embeddings used in the face recognition process and user profile images.
- **ML Kit** – ML Kit stands for Firebase Machine Learning Kit. Developed by Microsoft, Firebase ML Kit is a mobile software development kit that makes use of Google's machine learning technologies. In this project, the Face Detection API from Firebase ML Kit is called from the mobile application itself to trigger the continuous face detection process.

3. TensorFlow & TensorFlow Lite

TensorFlow is a publicly available artificial intelligence software library developed by Google used to build machine learning models. The face recognition model used in this project is constructed using TensorFlow libraries because of the seamless library management supported by Google and its capabilities to create large-scale, layered neural networks.

TensorFlow Lite is a subset of TensorFlow. The former optimizes models for on-device machine learning processes. The MobileFaceNet face recognition model used in this project was originally a TensorFlow model that was converted to a TensorFlow Lite model in order to operate on mobile phones with high performance, maximum model optimization, minimum model size and minimum latency.

3.2.3 Project Setting and Configuration

In Android Studio, the Java programming language was chosen to develop the mobile application. In order for the mobile application to support Android 11.0 APIs, the compileSdkVersion and targetSdkVersion were set to 30 in the app's build.gradle file. However, taking into consideration that some users might not have Android 11.0 smartphones, the minSdkVersion was set to 21 (supports Android 5.0) so that more than 94.1% of mobile devices are supported.

Furthermore, external libraries were implemented in the Android Studio project to add on features and functions to aid the overall development process. The essential libraries used in this project and their respective functions are tabulated below in Table 3.2:

External Library Name	Implementation Name	Description
Material Component	com.google.android.material:material:1.3.0	Used to provide various themes and designs for UI widgets such as buttons, textfields and etcetera.
Firebase Core	com.google.firebase:firebase-core:16.0.0	Used to provide access to Firebase configurations and to perform full initialization of Firebase in the application
Firebase Authentication	com.google.firebase:firebase-auth:20.0.4	Used to provide the email-and-password login functionality in the Login and Register module
Cloud Firestore	com.google.firebase:firebase-firestore:22.1.2	A NoSQL cloud database used to store the application's data in the form of collections and documents
Firebase Storage	com.google.firebase:firebase-storage:19.2.2	Used to store student face images and the user's profile pictures
Google Services Authentication	com.google.android.gms:play-services-auth:19.0.0	Used to provide the Gmail login functionality in the Login and Registration module

Picasso	com.squareup.picasso:picasso:2.71828	An image downloading and catching library used to load images straight from Firebase
Firebase ML Kit Face Detection	com.google.mlkit:face-detection:16.0.3	Used to provide the face detection capabilities of the application
GSON	com.google.code.gson:gson:2.8.6	Used to transform Java objects to JSON and vice versa
CameraX	androidx.camera:camera-core:1.0.0-rc01	Used to implement camera usages in the application
Tensorflow-Lite	org.tensorflow:tensorflow-lite:2.4.0	Used to support the MobileFaceNet face recognition model used in the application
RxPermissions	com.github.tbruyelle:rxpermissions:0.10.1	Used to manage camera and storage permissions
SDP (Scalable dp)	com.intuit.sdp:sdp-android:1.0.6	Used to ensure that the application is responsive (can work on mobile devices of different sizes)
SSP (Scalable sp)	com.intuit.ssp:ssp-android:1.0.6	Used to ensure that the application is responsive (can work on mobile devices of different sizes)

Table 3.2 List of External Libraries Used

Furthermore, the source code of the project was uploaded to GitHub in a private repository named FaceIt. The APK file of the project can be obtained here. For testing purposes, a new account should be registered under the school name UTAR-FICT with the school authorization code as 123. If further testing is required, please send an email to jctmq25@gmail.com requesting access to the source code.

In the following chapters, the implementations of the modules are described in detailed, along with comprehensive test results for each module according to chapter. In the very next chapter, the Login and Registration module is discussed first.

Chapter 4

Login and Register Module Implementation

In this chapter, the development, implementation and testing processes of the Login and Register module are discussed in detail. This chapter starts off with the database schema used for this module, then proceeds with the flow of the Login and Registration module, supported with screenshots of the application's interfaces. Last but not least, this chapter wraps up with the testing results for this module.

4.1 Database Design

The Login and Registration module stores data into Firebase's Cloud Firestore. As Cloud Firestore is a NoSQL database, the user details are stored under the *users* collection. Each of the documents in the *users* collection has its own autogenerated primary key. Each of the documents represents one user and houses three attributes as described below in Table 4.1.

Table 4.1 Database Design (Login and Registration Module)

Attribute Name	Attribute Description	Attribute Type	Attribute Example
Email	Email	String	jctmq25@1utar.my
SchoolName	School name	String	UTAR
fName	Full name of a user	String	Jacynth Tham
Role	Role of a user	String	admin

Even though each user has a unique autogenerated primary key, each user record is also validated to ensure that the email used is unique.

Additionally, each user is allowed to set a profile picture, which is stored in Firebase Storage. Firebase Storage is a cloud database used to store large files such as images and videos. Under the *users* folder, each user has their chosen profile picture stored in a folder named using their autogenerated primary key value. An example of a user's profile picture in Firebase Storage is shown below in Figure 4.1.

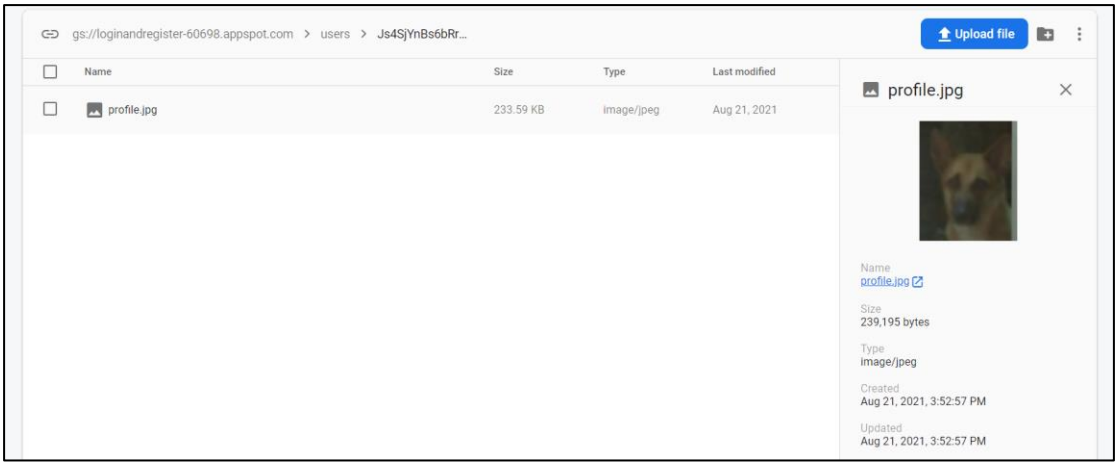


Figure 4.1 Firebase Storage Profile Picture Example

In the next section, the implementation of the Login and Register module is discussed in terms of the UI designs and functionalities.

4.2 Implementation of UI and Functionalities

The Login and Register module is the first module that users dawn upon when using the developed application, FaceIt. When users first launch the application, an animated splash screen is shown on-screen, as illustrated in Figure 4.2.

Figure 4.2 Application Splash Screen

The figure above shows the name and logo of the developed application. The name of the application is FaceIt, whose literal meaning is for the user to face the application's camera for their attendance to be taken. The application's name also explicitly hints at the use of face recognition technologies to the user. The logo of FaceIt is an AI face with attendance records for "brains". Inspired by humanoids, the logo was designed to represent the words "face" and "attendance". The catchphrase of the application – "Class Attendance with Ease", is also displayed in the splash screen to tell users the main gist of FaceIt, which is to capture class attendance. As a whole, the splash screen also hints at the colour theme of the developed application, which consists of blue and white. This colour scheme is adhered to by all the other activities' UIs in order to maintain a professional outlook.

Once the user has bypassed the splash screen, FaceIt checks to see if the user has any existing FirebaseUser object. If this object is not null, it means that the user has previously logged into the application and has yet to log out. Therefore, the application redirects the user straight to the main dashboard without the need for the user to log in again. This feature was developed with the intention to bring users convenience so that they do not have to log into the application multiple times in a single day.

However, if the user does not have a FirebaseUser object, then FaceIt redirects the user to the Login page. The UI of the Login page is illustrated below in Figure 4.3.

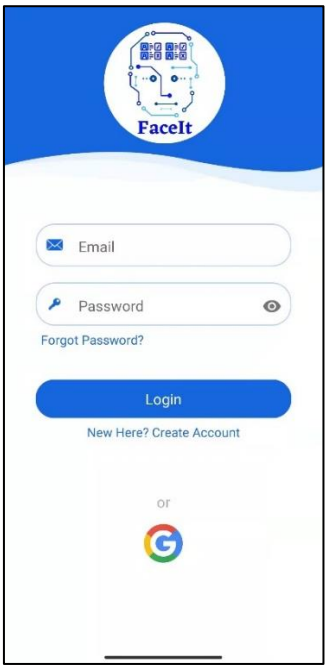


Figure 4.3 Login Page

In the Login page, the user has two login options – email-and-password and Gmail. Both authentication methods are handled by Firebase Authentication. For the email-and-password login method, users have to input a registered email and password into the input fields provided. Validations have been implemented at this point to ensure that the email and password inputs are valid and not empty. As for the password field, an additional "eye" icon is included to allow users to toggle the visibility of the typed-in password between visible and hidden. Once the user clicks on the Login button, the email-password authentication is performed by Firebase Authentication, which checks the inputted email-password pair against the existing ones in the database and decides whether to approve or reject the user's login request. If the user's login request has been approved, the user will be redirect to the main dashboard page. Otherwise, the user will be prompted to enter the login details again.

As for the second option – Gmail login, pressing the Gmail icon near the bottom of the screen will bring up a dialog box prompting the user to select the Gmail account to be associated with FaceIt. Thereafter, in future logins, FaceIt will select the same Gmail account upon clicking the Gmail login button. Figure 4.4 below shows the Gmail login dialogue box.

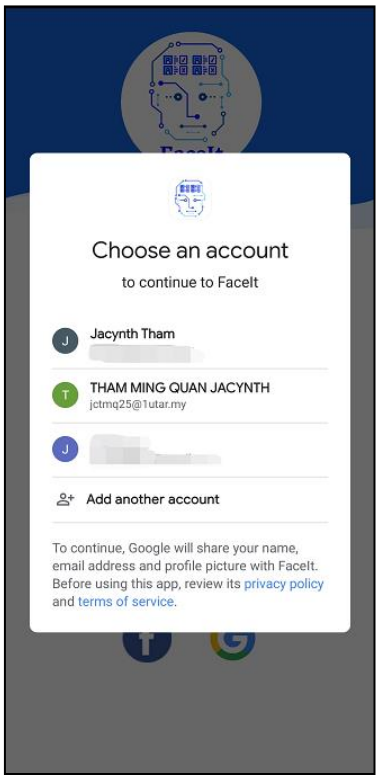


Figure 4.4 Gmail Login Option

During the login process, if the user ever forgets their login password, the user can click on the “Forgot Password?” text, which displays a pop-up dialogue box that prompts the user for an existing email previously registered with the application. Once the user has submitted a valid email, the application triggers Firebase Authentication to send an email to the user’s inbox with the password resetting link. Figure 4.5 below shows the reset password dialogue box and the email sent to the user’s inbox.

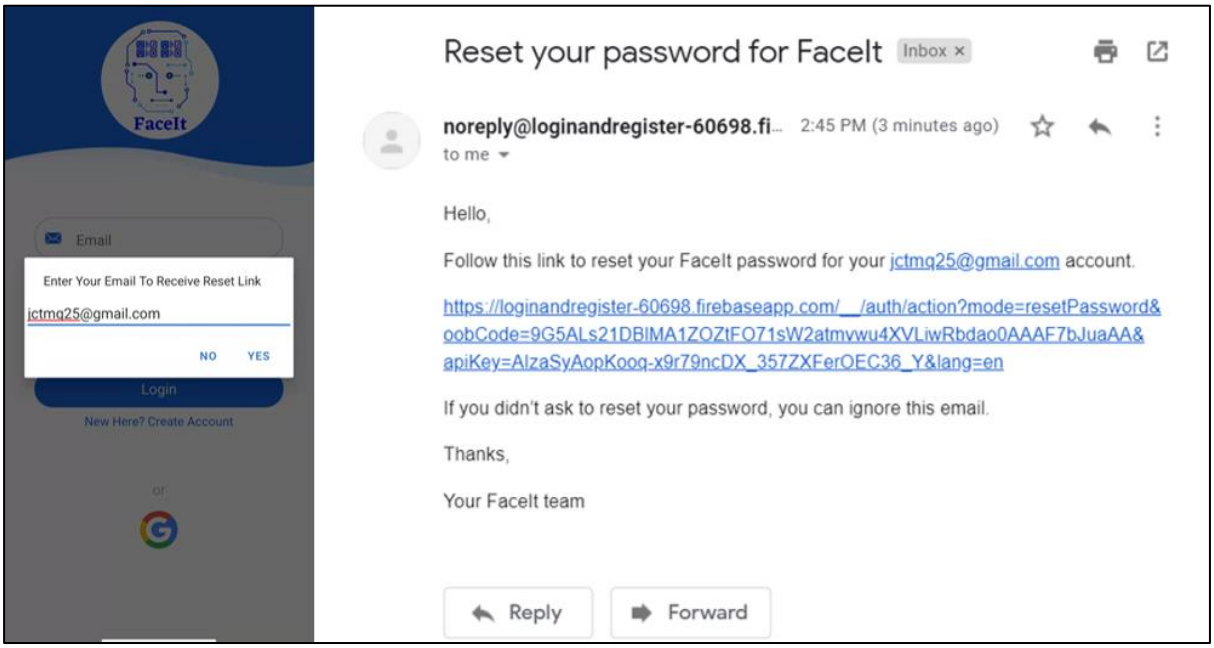


Figure 4.5 Reset Password Through Email

Once the link in the email is clicked, the webpage below in Figure 4.6 is shown to the user to reset the password.

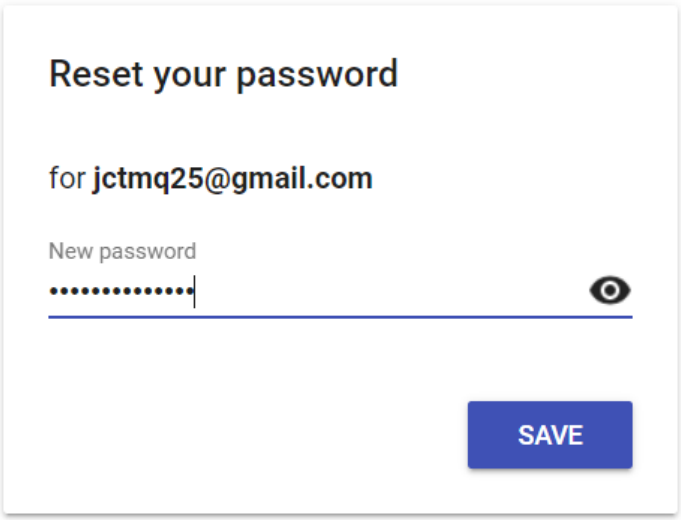


Figure 4.6 Reset Password Webpage by Firebase

Once the user clicks the Save button, the webpage shows a success message to the user and lets the user know that they can log into FaceIt using the new password, as evidenced below in Figure 4.7.

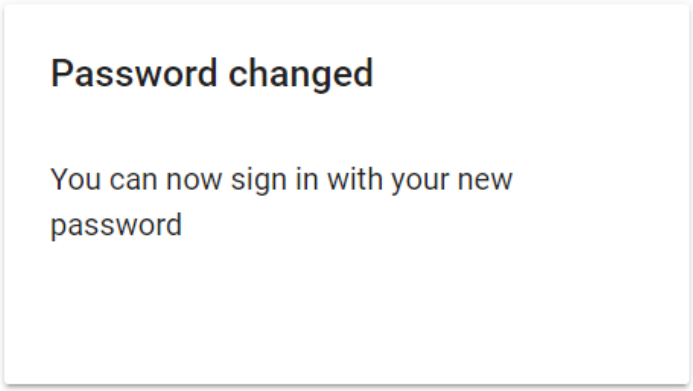


Figure 4.7 Reset Password Success Message by Firebase

Next, from the Login page, users can also navigate to the Register page to create a new account by clicking on the "New Here? Create Account" text beneath the Login button on the Login page. The Register page is shown below in Figure 4.8.

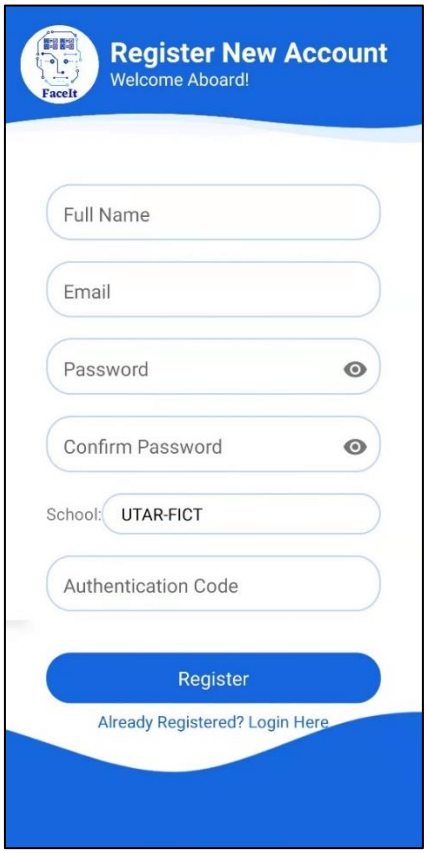


Figure 4.8 Register Page

In the Register page, users can create new accounts with FaceIt by providing their full name, email, password (required to type in twice for confirmation), school name and school authentication code. Validations are provided for all fields so that no field is allowed to be empty during the submission process. Additional validations are performed on the email input field to ensure that the input is a valid email. Similarly, validations are performed on the password input field to make sure that the chosen password is at least 6 characters long. As for the school selection, users can choose the school they wish to register under from a list of existing schools. For security reasons, users are not allowed to add in new schools. During the development of FaceIt, it is assumed that when new schools wish to use the application, the head administrator has to contact the developer of this project to add the school into the list of schools for users to register under, as well as to obtain a unique school authentication code. Then, when teachers within a particular school wish to register a new account, they have to contact the school administrator for the school authentication code to be filled in during the registration process. The main purpose of this feature is to ensure that the student and class details are not leaked to outsiders who do not belong in a particular school.
Once users have filled in all the details required, they can click on the Register button to register a new account and be directed straight to the main dashboard once the registration has been completed.

If users wish to register with Gmail, they can click on the Gmail button from the Login page shown previously in Figure 4.3. If the application detects that the user has yet to register an account, the application will redirect the user to the Register page as shown in Figure 4.8 above. However, with the Gmail registration process, the first name and email fields will be prefilled with the information obtained from the user's Gmail account. Moreover, no password will be required to be filled in by the user. The interface of the Register page when users select Gmail as their registration option is shown below in Figure 4.9:

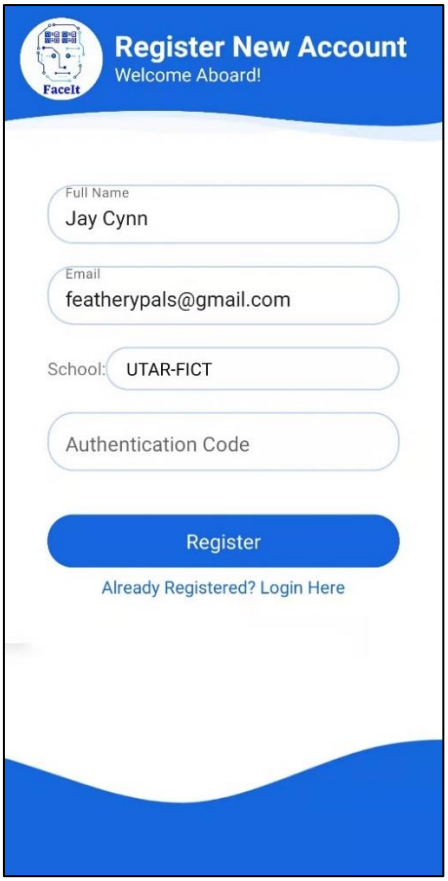


Figure 4.9 Register Page (Gmail)

Once the user has successfully registered a new account, the user's credentials are stored in Cloud Firestore under the *users* collection. The document ID is set to be exactly the same as the User UID as determined by Firebase Authentication during the user's registration process. This is to ensure that the application can retrieve the user's credentials to be displayed in the main dashboard of the application as well as determine the classes and students that should remain visible to the user (determined by userID and school name respectively). An example document of the user's collection is illustrated below in Figure 4.10.

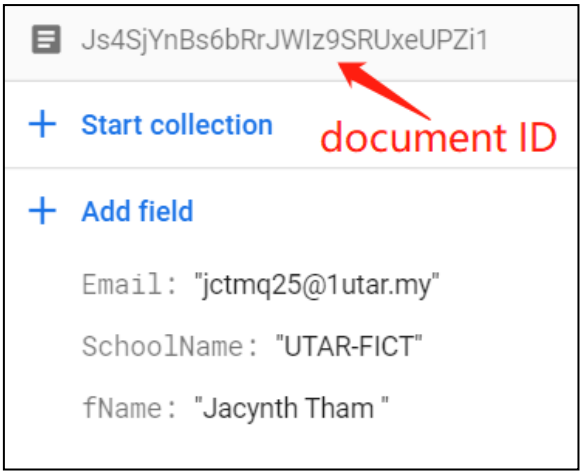


Figure 4.10 Example Document in *Users* Collection

After the user has logged into the application, the user will be redirected to the main dashboard. Figure 4.11 below shows the UI of the main dashboard of FaceIt.

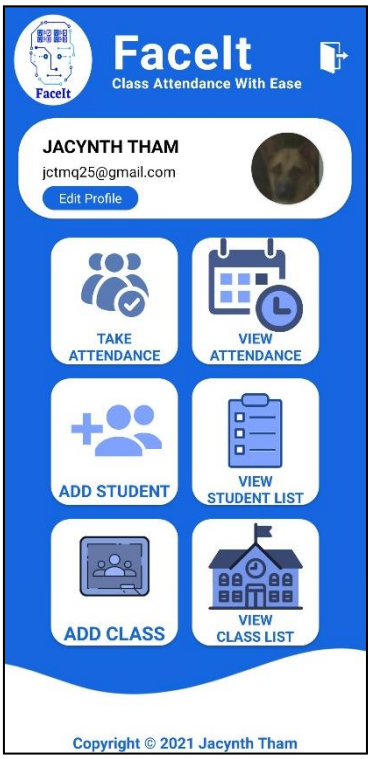


Figure 4.11 Main Dashboard

In the main dashboard, the key elements are the user profile details, the six navigation buttons and the logout button at the top right corner.

Firstly, users can view and modify their user credentials – full name and email – by clicking the Edit Profile button in the main dashboard. Upon clicking this button, users will be directed to the Edit Profile page, as shown in Figure 4.12 below.

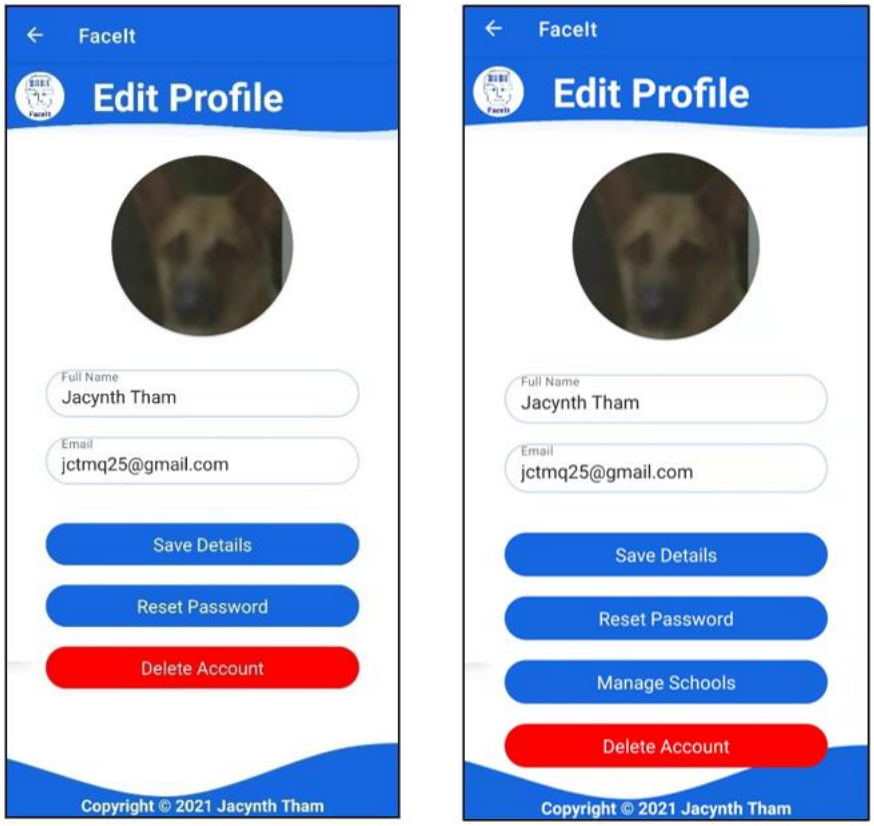


Figure 4.12 (From Left) Edit Profile User View, Edit Profile Admin View

As evidenced by Figure 4.12 above, there are two different profile views – one for basic users such as teachers and lecturers, and one for the head administrator of every school. In Firebase, there are two possible roles of accounts, which are user and admin. As mentioned in earlier paragraphs, when new schools wish to register with FaceIt, the head administrator of the school has to contact the developer of FaceIt to register the school into the list of schools and obtain an authentication code. During this process, an admin account is also created for the head administrator, whose Edit Profile interface looks like the image shown on the Right in Figure 4.12 above. The only difference between a user account and as admin account is that admins can view an additional button – Manage Schools. The functionalities of each button are explained in the following paragraphs.

In this Edit Page, both users and admins are allowed to change their credentials or profile picture. If users wish to modify the latter, they only have to click on the circular profile picture shown and they will be allowed to select an image from their phone's gallery. On the other hand, if users wish to modify their full name or email, users have to click on the Save Details button to lock in their changes.

Next, users are also allowed to reset their passwords in the Edit Profile page by clicking on the Reset Password button. Upon clicking this button, a pop-up dialogue box will be shown to the users, prompting the new password. Figure 4.13 illustrates the reset password dialogue box.

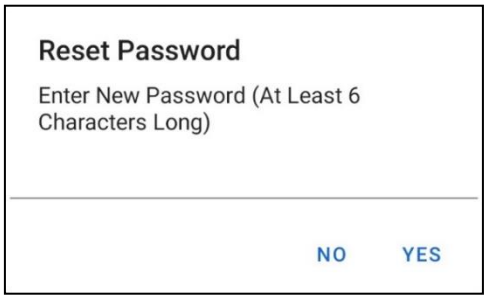


Figure 4.13 Reset Password Dialogue Box

The next button – Manage Schools – is only available for admin accounts. When clicked, the Manage School interface is shown as in Figure 4.14 below:

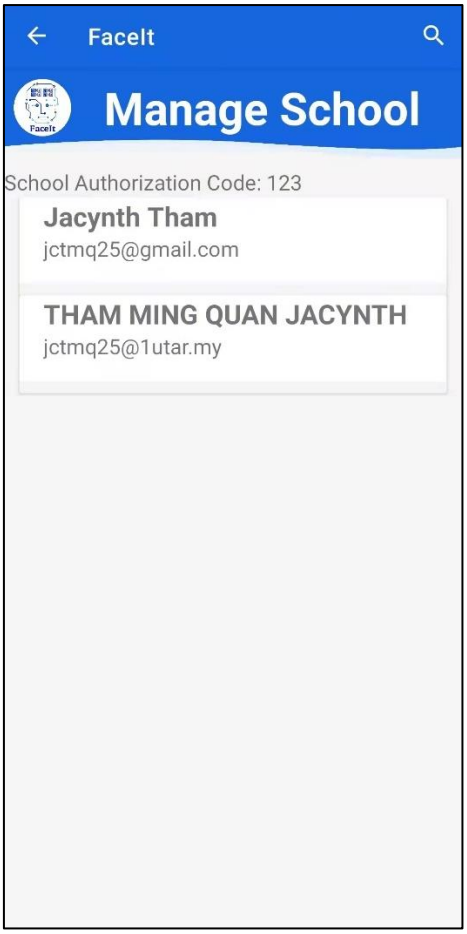


Figure 4.14 Manage School Interface

In the Manage School interface, the admin (users with the "admin" role) can view the school authorization code and the list of registered users under the current school. The school authorization code has to be given to any new teacher who wishes to register an account under the current school for authentication purposes. As for the list of users, the users' full names and emails are shown so that the admin can keep track of all the teachers registered under the current school.

The last option in the Edit Profile page, available to both user and admin accounts, is the Delete Account button. This function allows users to delete the user account and all related documents in the cloud database. This button was intentionally coloured red in the Edit Profile page so that users do not delete their accounts by accident. Upon clicking the Delete Account button, a pop-up dialogue box is also shown to the user to confirm that the user wants to delete the account. Figure 4.15 shows the Delete Account dialogue box.

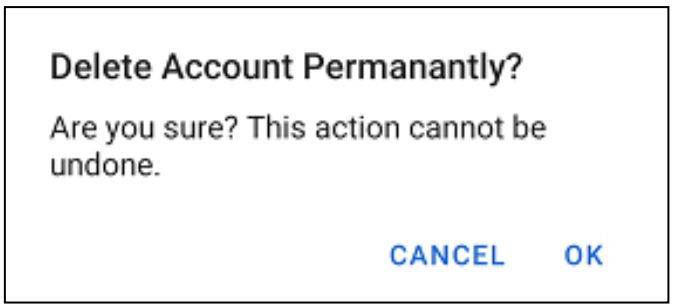


Figure 4.15 Delete Account Dialogue Box

Once the user confirms the account deletion, all classes added by the user and their respective attendance records are removed from the cloud database as well.

The last and most crucial part of the Main Dashboard interface is the grid in the centre of the page that houses six different navigation buttons, each leading to a specific action as described in the buttons' names. The six buttons are Take Attendance, View Attendance, Add Student, View Student List, Add Class and View Class List respectively. Each of the six buttons redirects the user to the corresponding activity in the Class module, Student module or Attendance module. The Enrolment module is an exception and can only be accessed from within the Class module.

Lastly, the Logout button in the main dashboard allows users to logout of the current account. Behind the scenes, Firebase invalidates the user's login request, so that the next time the user opens the application, the user will be prompted to log into a valid account. However, if the user exits the application without using the Logout button, the user will be directed to the main dashboard the next time the user opens the application.

4.3 Module Testing

In this subsection, the test results of the Login and Register module are documented as tables. Each table represents a single test case, complete with test case ID, name, description, expected output as well as input and output (supported with screenshots) as evidence to the test cases.

4.3.1 Login

Table 4.2 to 4.9 below show the test cases of the Login functionalities. The test case IDs in this subsection all start with TL, which stands for "Test Login"

Table 4.2 TL01: Open Application with Existing Login

Test Case ID	TL01
Test Case Name	Open Application with Existing Login
Test Case Description	User opens the application with an existing, authenticated login
Expected Output	Application redirects user to main dashboard straight away.
Input	Application closed without logging out (press back button on mobile device)

Results	`V/FA: onActivityCreated`
	`D/TAG: Current User is not null. UserID is Js4SjYnBs6bRrJWIz9SRUxeUPZi1`
	Console message shows that the userID is not null upon opening the application and redirects the user to the main dashboard.

Status (Pass/Fail)	Pass

Table 4.3 TL02: Login with Empty Fields

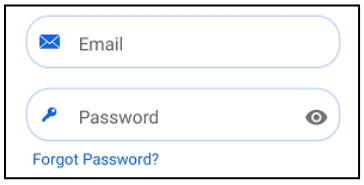
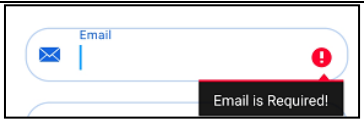
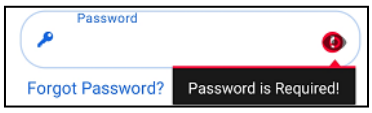
Test Case ID	TL02
Test Case Name	Login with Empty Fields
Test Case Description	User presses the Login button without keying in any fields
Expected Output	Error message shown
Input	
Results	 
Status (Pass/Fail)	Pass

Table 4.4 TL03: Toggle Password Button

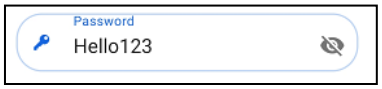
Test Case ID	TL03
Test Case Name	Toggle Password Button
Test Case Description	User presses the Toggle Password Visibility button.
Expected Output	Password field value toggled between visible and hidden
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.5 TL04: Email/Password Login with Invalid Credentials

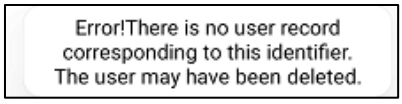
Test Case ID	TL04
Test Case Name	Login With Invalid Credentials
Test Case Description	User presses the Login button with an invalid username and password
Expected Output	Error message shown
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.6 TL05: Email/Password Login with Valid Credentials

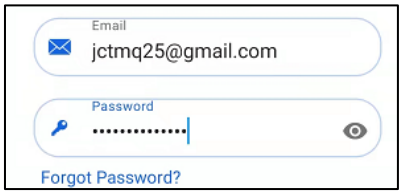
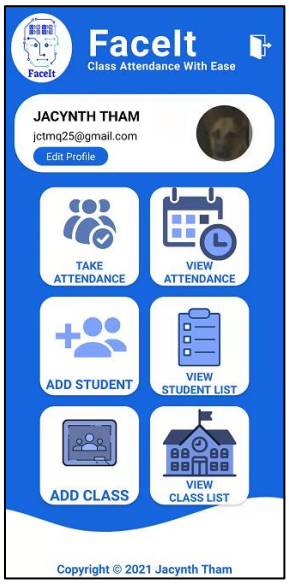
Test Case ID	TL05
Test Case Name	Login With Valid Credentials
Test Case Description	User presses the Login button with a valid username and password combo
Expected Output	Login and redirect user to the Main Dashboard interface
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.7 TL06: Gmail Login with Previously Registered email

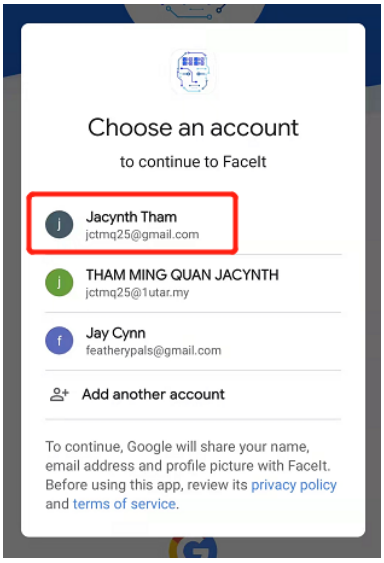
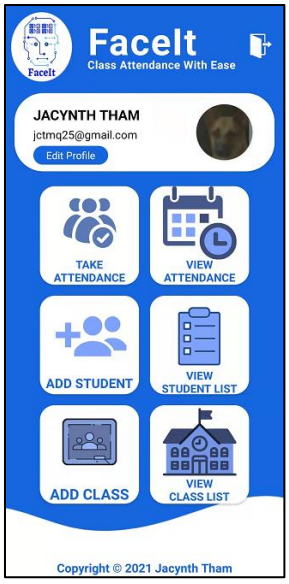
Test Case ID	TL06
Test Case Name	Login With Previously Registered Gmail
Test Case Description	User presses the Gmail button and selects an email that has previously been registered with FaceIt
Expected Output	Login and redirect user to the Main Dashboard interface
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.8 TL07: Gmail Login with Unregistered Email

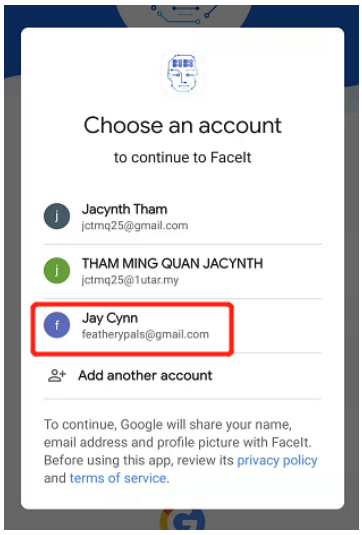
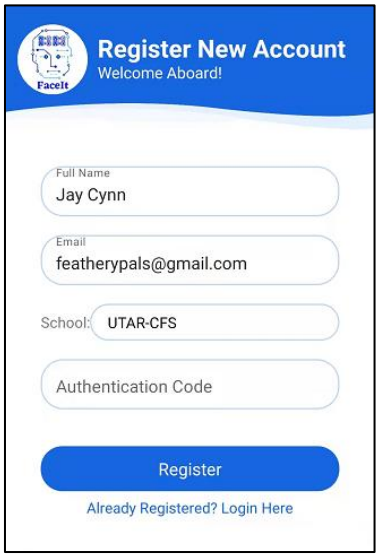
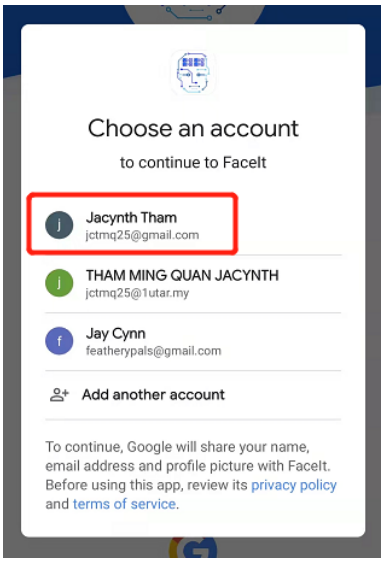
Test Case ID	TL07
Test Case Name	Login With Unregistered Gmail
Test Case Description	User presses the Gmail button and selects an email that has not previously been registered with FaceIt
Expected Output	Redirect user to Register page and prefill user's full name and email fields
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.9 TL08: Gmail Login Authentication Failure

Test Case ID	TL08
Test Case Name	Gmail Login Authentication Failure
Test Case Description	User presses the Gmail button to login, but cancels the authentication process halfway by clicking back or due to network loss
Expected Output	Display error message
Input	
Results	
Status (Pass/Fail)	Pass

4.3.2 Register

Table 4.10 to 4.16 below show the test cases of the Register functionalities. The test case IDs in this subsection all start with TR, which stands for "Test Register".

Table 4.10 TR01: Register with Empty Fields

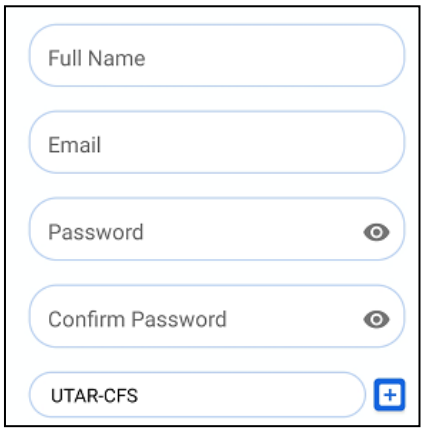
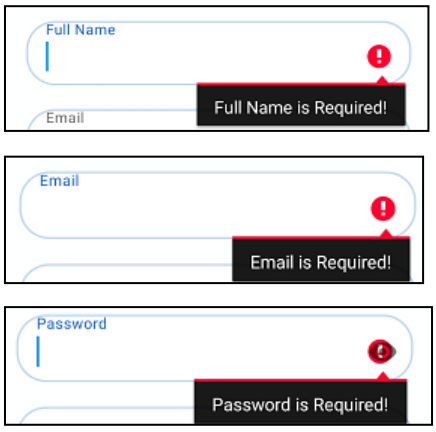
Test Case ID	TR01
Test Case Name	Register with Empty Fields
Test Case Description	User presses the Register button with one or more empty fields
Expected Output	Error message shown
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.11 TR02: Register with Invalid Email Format

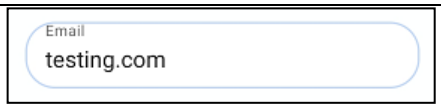
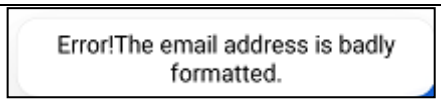
Test Case ID	TR02
Test Case Name	Register with Invalid Email Format
Test Case Description	User enters email address without the @ and . symbols
Expected Output	Error message shown
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.12 TR03: Register with Invalid Password

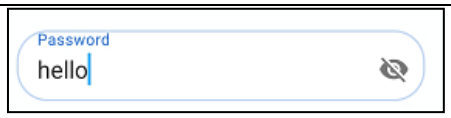
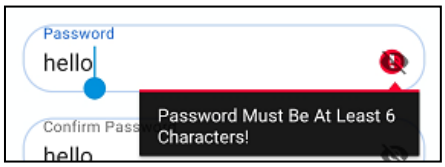
Test Case ID	TR03
Test Case Name	Register with Invalid Password
Test Case Description	User enters password less than six characters long
Expected Output	Error message shown
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.13 TR04: Register with Non-Matching Passwords

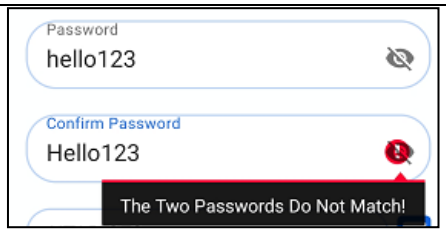
Test Case ID	TR04
Test Case Name	Register with Non-Matching Passwords
Test Case Description	User enters passwords that do not match
Expected Output	Error message shown
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.14 TR05: Gmail Registration

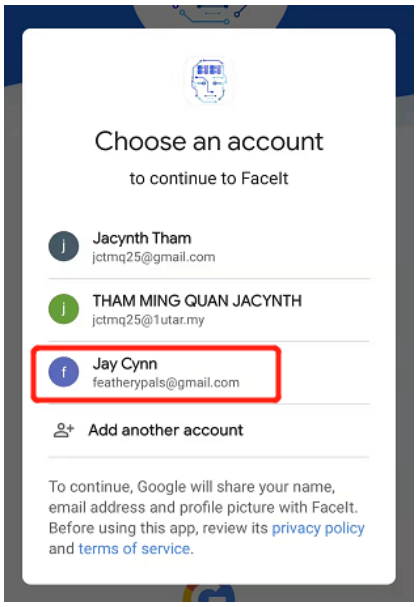
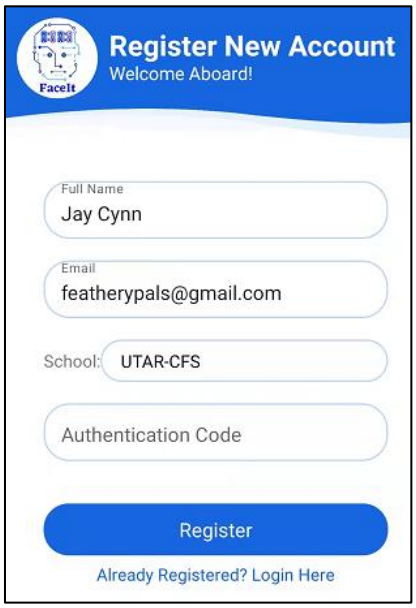
Test Case ID	TR05
Test Case Name	Gmail Registration
Test Case Description	User registers a new account with Gmail integration
Expected Output	User's full name and email are prefilled. The password and confirm password fields are hidden
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.15 TR06: Registration with Invalid School Authorization Code

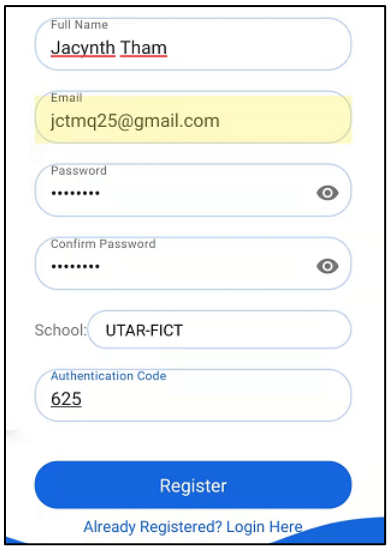
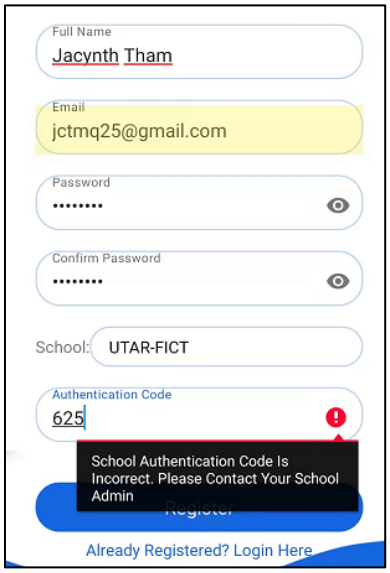
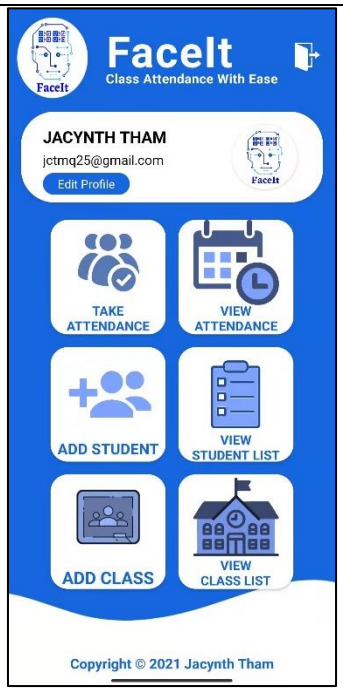
Test Case ID	TR06
Test Case Name	Registration with Invalid School Authorization Code
Test Case Description	User enters an invalid school authorization code for the chosen school
Expected Output	Display error message
Input	School authorization code as seen in Firebase Console: SchoolAuthCode: "123" SchoolName: "UTAR-FICT" User input in application: 
Results	
Status (Pass/Fail)	Pass

Table 4.16 TR07: Registration with Valid Fields

Test Case ID	TR07
Test Case Name	Registration with Valid Fields
Test Case Description	User presses the Register button with all input fields as valid
Expected Output	Application registers a new account and redirects user to Main Dashboard page
Input	
Results	
Status (Pass/Fail)	Pass

4.3.3 Main Dashboard

Table 4.17 to 4.24 below show the test cases of the functionalities in the Main Dashboard page, mainly related to application navigation. The test case IDs in this subsection all start with TD, which stands for “Test Dashboard”.

Table 4.17 TD01: Edit Profile Button

Test Case ID	TD01
Test Case Name	Edit Profile Button
Test Case Description	User presses the Edit Profile button in the Main Dashboard page
Expected Output	User is redirected to the Edit Profile page
Input	
Results	 *User shown is an admin, therefore, has access to Manage Schools button
Status (Pass/Fail)	Pass

Table 4.18 TD02: Take Attendance Button

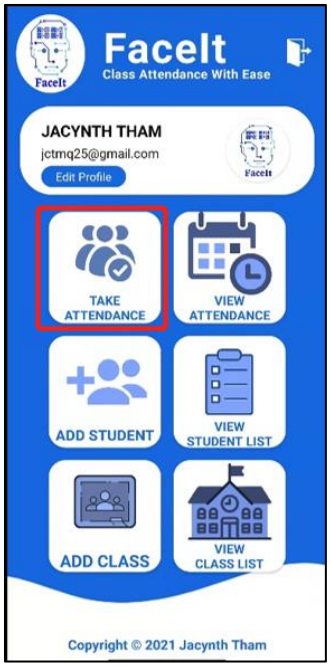
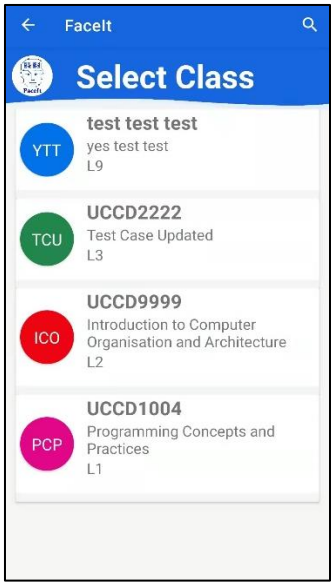
Test Case ID	TD02
Test Case Name	Take Attendance Button
Test Case Description	User presses the Take Attendance button in the Main Dashboard page
Expected Output	User is redirected to the Take Attendance page (Select Class activity)
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.19 TD03: View Attendance Button

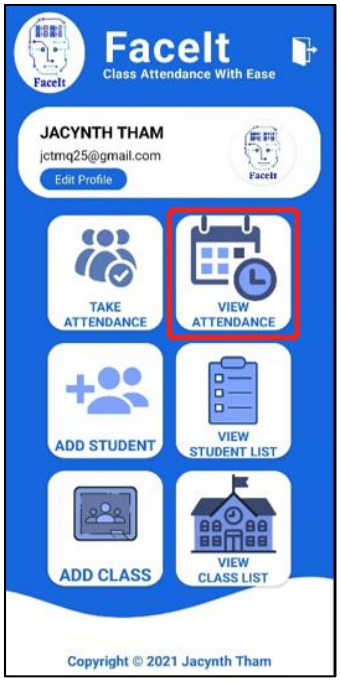
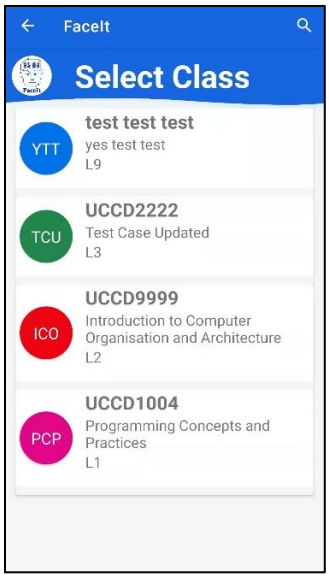
Test Case ID	TD03
Test Case Name	View Attendance Button
Test Case Description	User presses the View Attendance button in the Main Dashboard page
Expected Output	User is redirected to the View Attendance page (Select Class activity)
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.20 TD04: Add Student Button

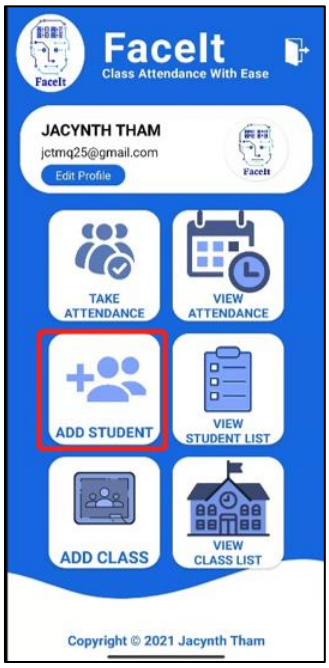
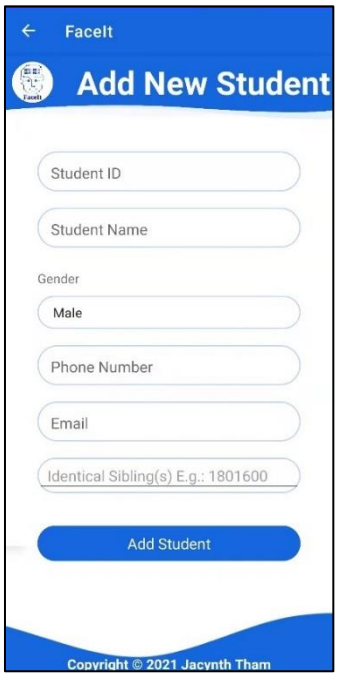
Test Case ID	TD04
Test Case Name	Add Student Button
Test Case Description	User presses the Add Student button in the Main Dashboard page
Expected Output	User is redirected to the Add Student page
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.21 TD05: View Student List Button

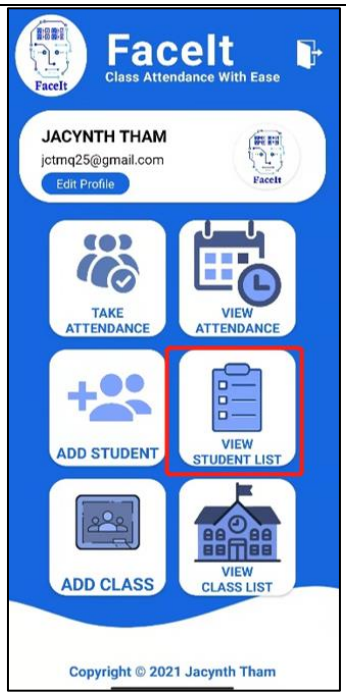
Test Case ID	TD05
Test Case Name	View Student List Button
Test Case Description	User presses the View Student List button in the Main Dashboard page
Expected Output	User is redirected to the View Student List page
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.22 TD06: Add Class Button

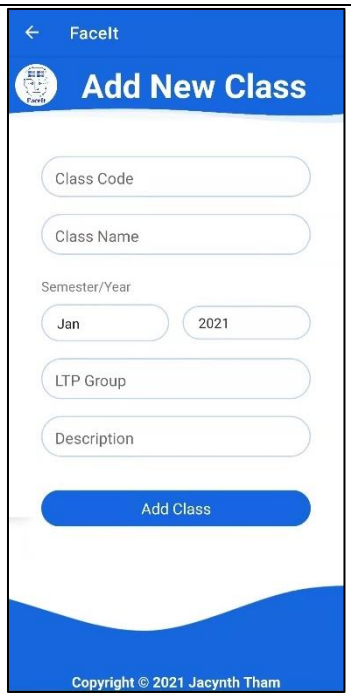
Test Case ID	TD06
Test Case Name	Add Class Button
Test Case Description	User presses the Add Class button in the Main Dashboard page
Expected Output	User is redirected to the Add Class page
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.23 TD07: View Class List Button

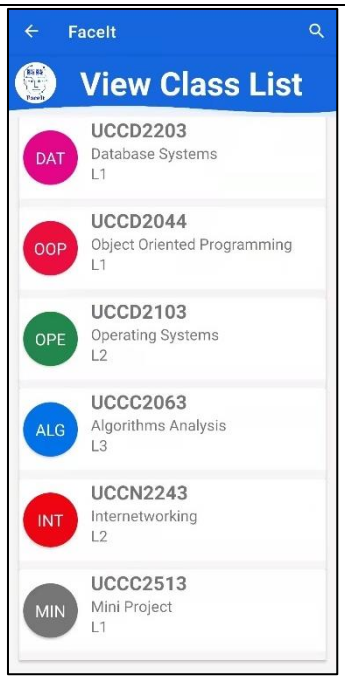
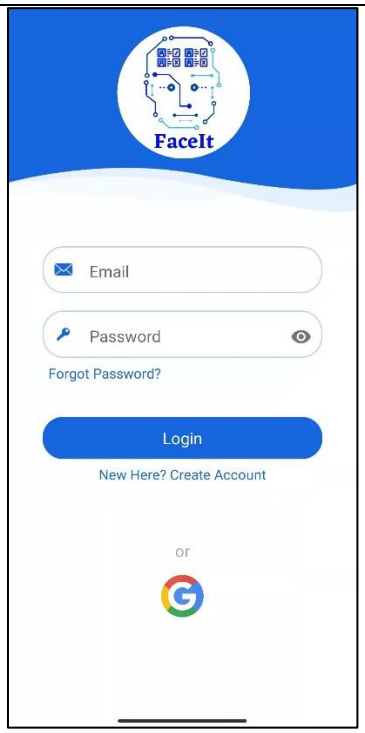
Test Case ID	TD07
Test Case Name	View Class List Button
Test Case Description	User presses the View Class List button in the Main Dashboard page
Expected Output	User is redirected to the View Class List page
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.24 TD08: Logout Button

Test Case ID	TD08
Test Case Name	Logout Button
Test Case Description	User presses the Logout button in the Main Dashboard page
Expected Output	Application logs the user out and redirects the user to the Login page
Input	
Results	
Status (Pass/Fail)	Pass

4.3.4 Edit Profile

Table 4.25 to 4.34 below show the test cases of the functionalities in the Edit Profile page. The test case IDs in this subsection all start with TP, which stands for "Test Profile".

Table 4.25 TP01: Display Edit Profile Page

Test Case ID	TP01
Test Case Name	Display Edit Profile Page
Test Case Description	User presses the Edit Profile button in the Main Dashboard page
Expected Output	Application displays the Edit Profile page with the user's full name and email prefilled with existing values
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.26 TP02: Change Profile Picture

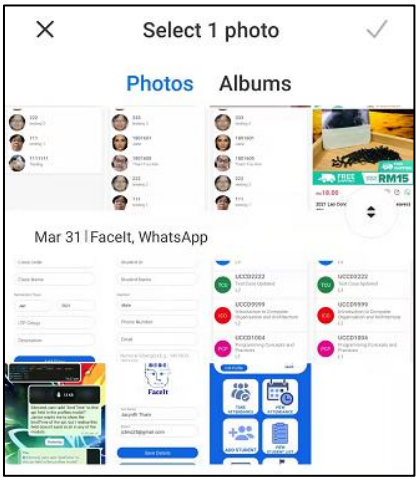
Test Case ID	TP02
Test Case Name	Change Profile Picture
Test Case Description	User clicks on the rounded profile picture at the top of the Edit Profile page
Expected Output	Application launches the user's phone gallery and allows the user to select a new profile picture. Then, the profile picture on the Edit Profile page updates itself.
Input	
Results	 
Status (Pass/Fail)	Pass

Table 4.27 TP03: Edit User Profile Details with Valid Fields

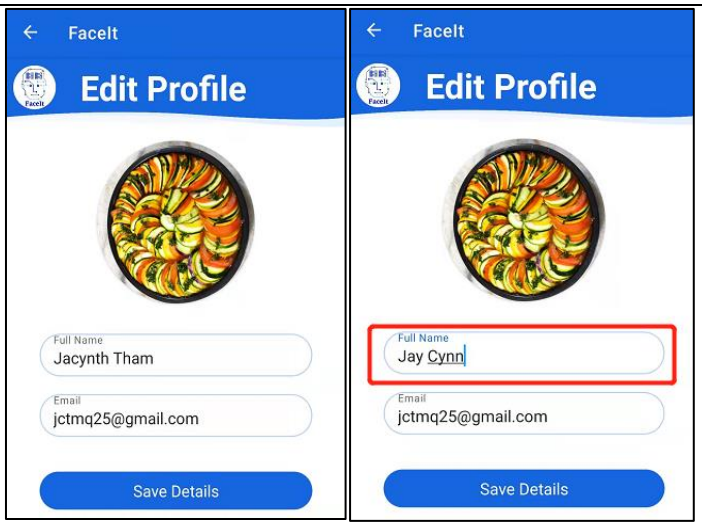
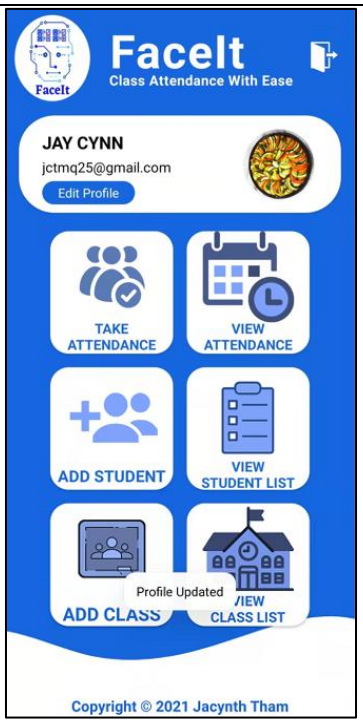
Test Case ID	TP03
Test Case Name	Edit User Profile Details with Valid Fields
Test Case Description	User modifies the values of Full Name and Email (both valid values) and clicks on the Save Details button
Expected Output	Application updates user's profile details in database and displays success message
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.28 TP04: Edit User Profile Details with Invalid Email Format

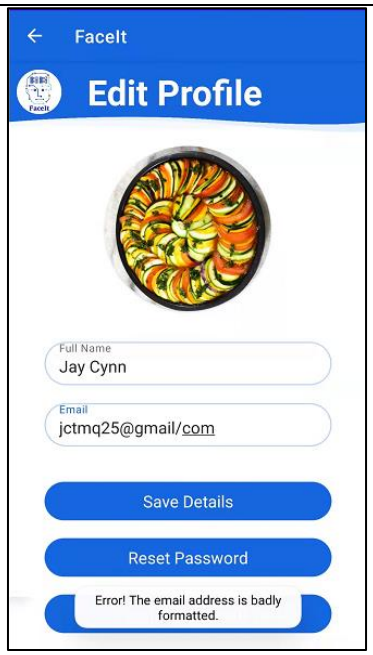
Test Case ID	TP04
Test Case Name	Edit User Profile Details with Invalid Email Format
Test Case Description	User clicks on the Save Details button with an invalid email entered
Expected Output	Display error message
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.29 TP05: Edit User Profile Details with Empty Fields

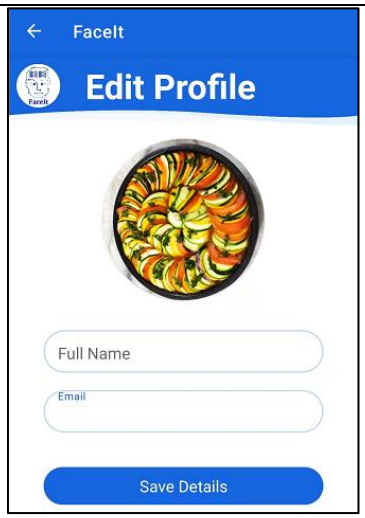
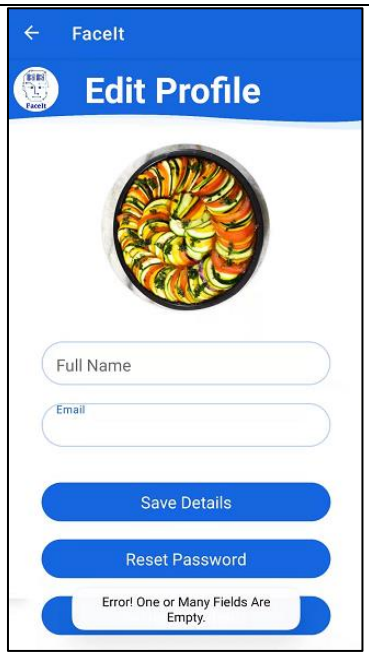
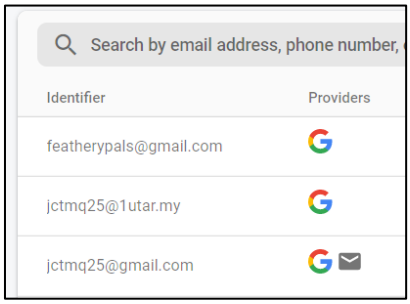
Test Case ID	TP05
Test Case Name	Edit User Profile Details with Empty Fields
Test Case Description	User clicks on the Save Details button with empty fields (full name or email)
Expected Output	Display error message
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.30 TP06: Edit User Profile Details with Non-Unique Email

Test Case ID	TP06
Test Case Name	Edit User Profile Details with Non-Unique Email
Test Case Description	User clicks on the Save Details button with an entered email that belongs t another account
Expected Output	Display error message
Input	Existing emails regstered in Firebase:  Email in input field: 

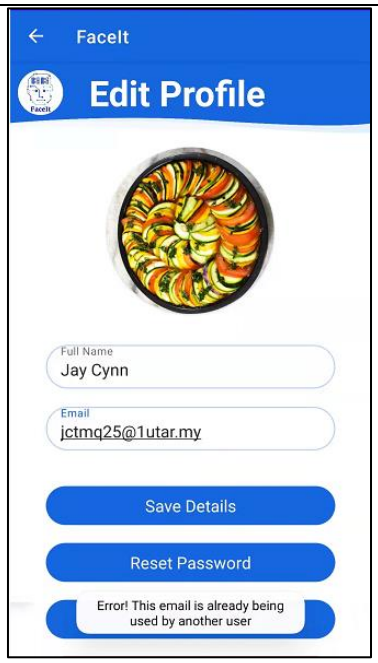
Results	
Status (Pass/Fail)	Pass

Table 4.31 TP07: Reset Password Button

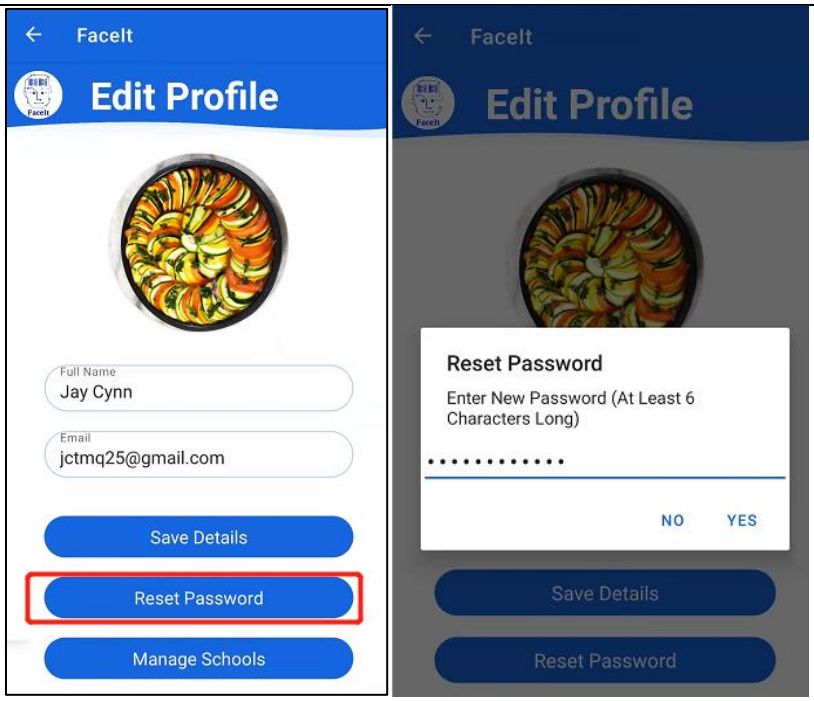
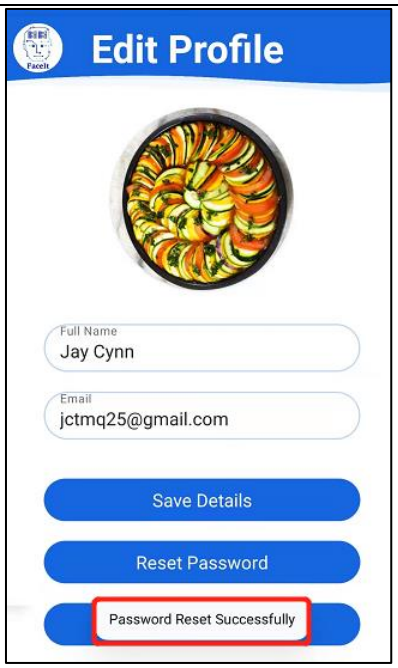
Test Case ID	TP07
Test Case Name	Reset Password Button
Test Case Description	User clicks on the Reset Password Button
Expected Output	Application displays a pop-up modal prompting users to enter a new password
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.32 TP08: Reset Password – New Password Is Empty

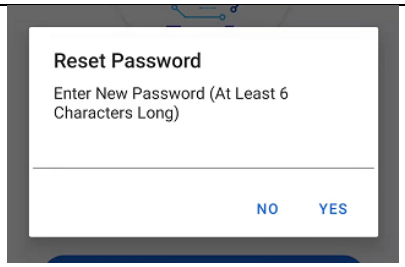
Test Case ID	TP08
Test Case Name	Reset Password – New Password Is Empty
Test Case Description	User enters an empty new password in the Reset Password modal
Expected Output	Application displays an error message
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.32 TP08: Show Manage School Button only for Users with Admin Role

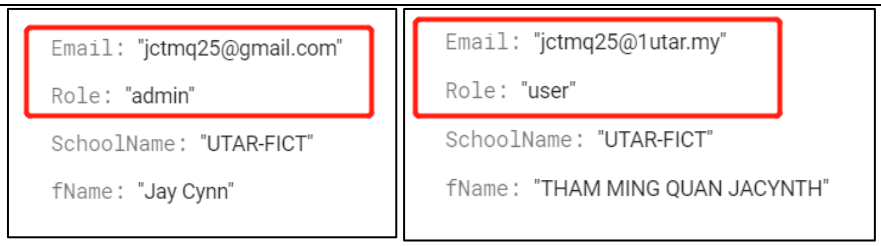
Test Case ID	TP08
Test Case Name	Show Manage School Button only for Users with Admin Role
Test Case Description	Application will only display the Manage School button if the current account has the “admin” role. Otherwise, the button will be hidden.
Expected Output	Show the Manage School button for users with the “admin” role and hide the button for users with the “user” role.
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.33 TP09: Manage School Button

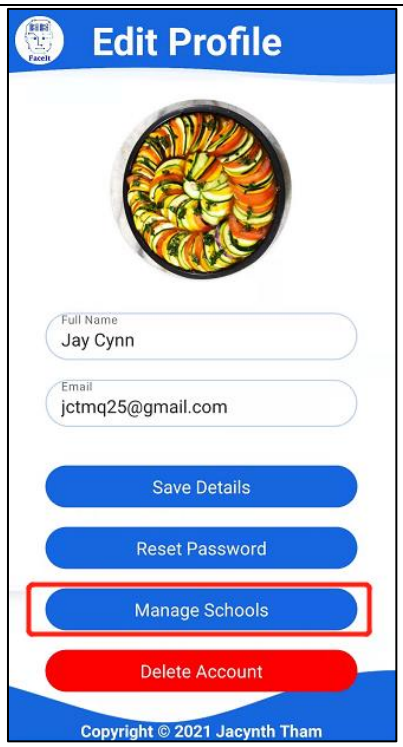
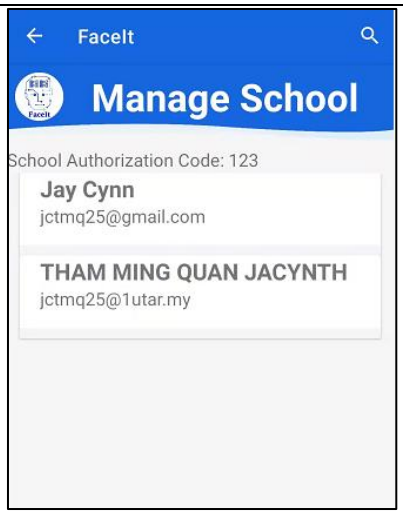
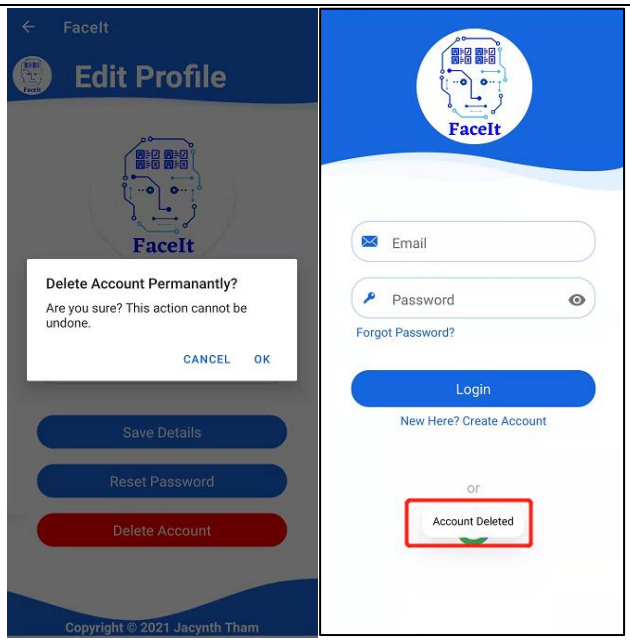
Test Case ID	TP09
Test Case Name	Manage School Button
Test Case Description	Admin clicks on the Manage School button
Expected Output	Redirects admin to the Manage School Interface and show list of registered users in the current school
Input	
Results	
Status (Pass/Fail)	Pass

Table 4.34 TP10: Delete Account Button

Test Case ID	TP10
Test Case Name	Delete Account Button
Test Case Description	User clicks on the Delete Account button in the Edit Profile page
Expected Output	Prompts confirm modal. If confirmed, the application deletes the user's account and all classes, enrolments, attendances, and profile pictures associated with the user's account. Redirect the user to the Login page
Input	
Results	
Status (Pass/Fail)	Pass

Having performed a comprehensive testing on the Login and Register module, it is evidenced that the functionalities of this module are sufficient for users to access the developed application as well as manage their own profile details. Not only are users able to register new accounts with FaceIt, but they can also login via email/password or Gmail integration. In the Main Dashboard page, users can navigate to the other modules in this project such as the Student and Class module, and they can also update their personal information in the Edit Profile page.

In the next chapter, the functionalities of the Class module are explained and tested.

Chapter 5

Class Module Implementation

In this chapter, the development, implementation, and testing processes of the Class module are discussed in detail. This chapter starts off with the database schema used for this module, then proceeds with the flow of the Class module, supported with screenshots of the application's interfaces. Last but not least, this chapter wraps up with the testing results for this module.

5.1 Database Design

The Class module stores data into Firebase's Cloud Firestore. The class details are stored under the *classes* collection. Each of the documents in the *classes* collection has its own autogenerated primary key, and each of the documents represents one class. The attributes of each class are as described below in Table 5.1.

Table 5.1 Database Design (Class Module)

Attribute Name	Attribute Description	Attribute Type	Attribute Example
ClassCode	Class code	String	UCCN2243
ClassDesc	Class description	String	300 Students
ClassLTP	Class lecture (L), tutorial (T), or practical (P) group	String	L2
ClassName	Class name	String	Internetworking Principles
ClassSem	Semester in which the class commences	String	May
ClassYear	Year in which the class commences	String	2021
UserID	The autogenerated primary key of the user who added the class	String	Js4SjYnBs6bRrJWIz9SRUxeUPZi1

LogoRandomNum	A random number between 1 and 8 that determines the colour of the logo as displayed in FaceIt. For example, 1 corresponds to dark green, 2 corresponds to purple and so on.	Number	1

Even though each class record has its own autogenerated primary key, a unique class is defined as having a unique combination of a class code, class name, class sem, class year, and LTP group.

5.2 Implementation of UI and Functionalities

The Class module allows users to manage classes added to the developed application. The Class module consists of two activities – Add Class and View Class List. From the main dashboard, users can access these two activities through the corresponding buttons with the activities' names annotated on them, as shown in Figure 5.1 below.

Figure 5.1 Buttons Associated with the Class Module

Figure 5.2 below illustrates the UI of the Add Class activity.

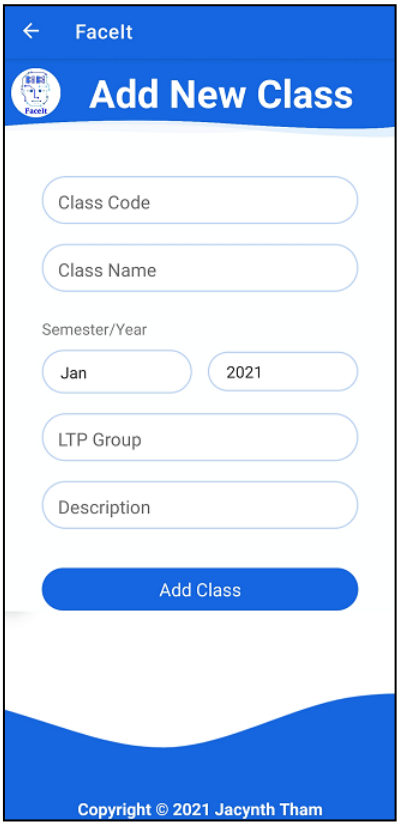


Figure 5.2 Add Class Page

In the Add Class activity, users can add in new classes by providing the new class code, class name, semester, year, LTP (Lecture, Tutorial, Practical) group and the class description. All the input fields should be manually filled in by the user except for the semester and year fields. These two fields allow users to select a single value from a dropdown list, as shown in Figure 5.3 below.

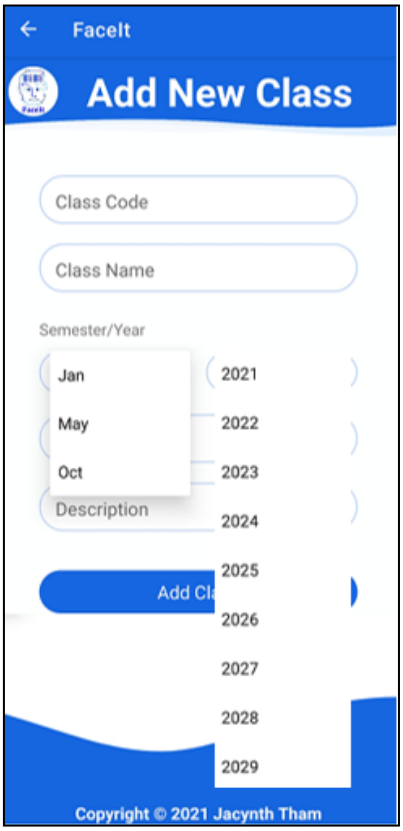


Figure 5.3 Drop-Down Lists for Semester and Year

Basic validations are implemented for all the input fields, including checking for empty fields or duplicated fields. Once the user has filled in all the mandatory fields, the new class will be added to the cloud database as a document in the *classes* collection. Figure 5.4 below shows an example of the document added into the *classes* collection.

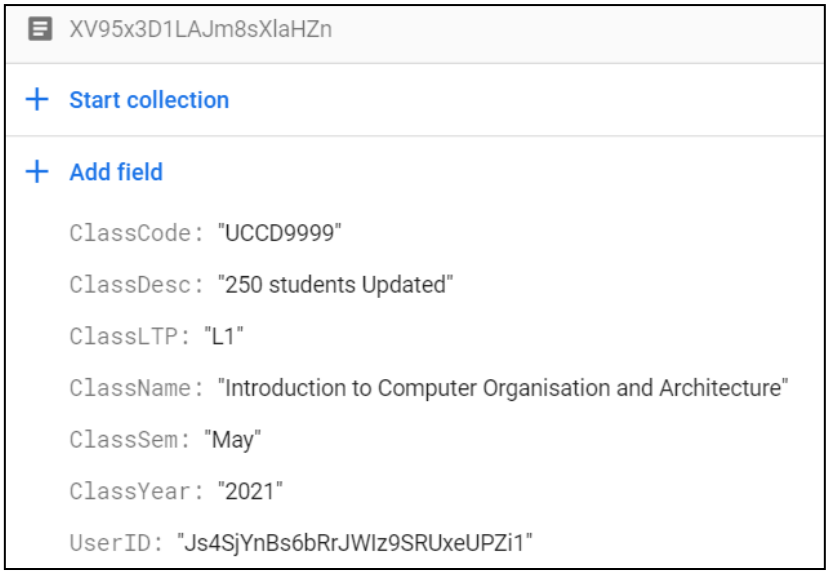


Figure 5.4 Example Document in *Classes* Collection

As seen in the figure above, an additional field, UserID, is also added to the cloud database upon the addition of a new class. This UserID is the same as the Firebase Authentication UserID and is used to ensure that users only manage classes that they have previously added. Other users are unable to view the classes of users other than themselves.

Next, users can view the list of existing classes by clicking on the View Class List button in the main dashboard. Figure 5.5 below shows the UI of the View Class activity.

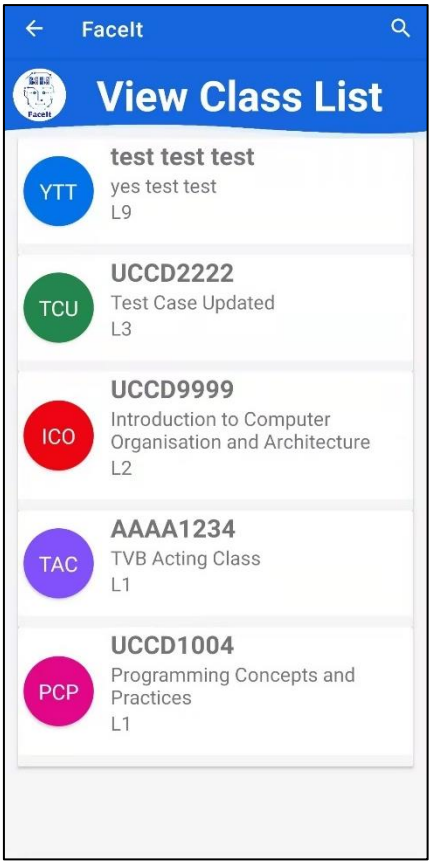


Figure 5.5 View Class List Page

For the convenience of users, each class in the class list is displayed along with a rounded icon with a particular colour and a three-lettered short-form. The colours of the class icons are determined by random during the Add Class process and stays the same for each particular class until they are deleted from the application. Furthermore, the short-form of the class name is determined at run time by taking the first letter of each word within the class name. Note that conjunctions such as “and”, “for”, “to” and so on are omitted when taking the first letter of each word in the class name. For example, the short-form of the class named “Programming Concepts and Practises” will be “PCP” because the word “and” is skipped. For cases where the class name consists of less than three words (excluding conjunctions), the first three letters of the first word of the class name will be used. For example, the class name “Internetworking” will yield “INT” as the short-form.

Next, if users wish to search for a particular class, they can click on the magnifying glass to trigger the search view. The search view displays classes according

to the keyword input and refreshes dynamically when a new character is entered. Figure 5.6 below illustrates the display of the search view.

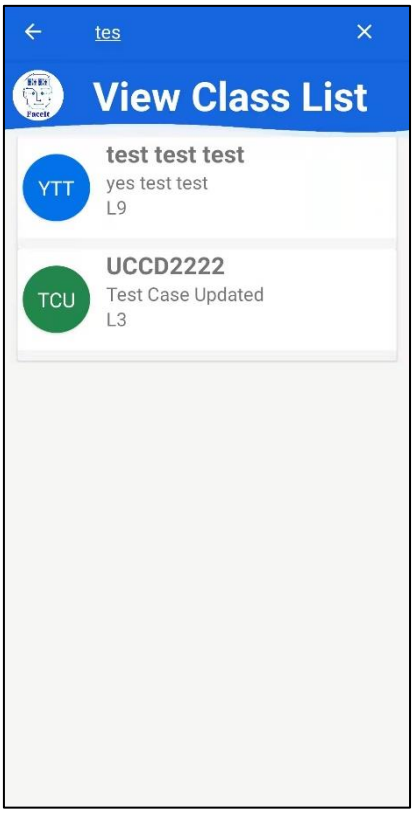


Figure 5.6 Class List Search View

Next, when users click on a particular class in the list, a dialogue box will be shown to the user. The dialogue box consists of the selected class's full details, as shown in Figure 5.7 below.

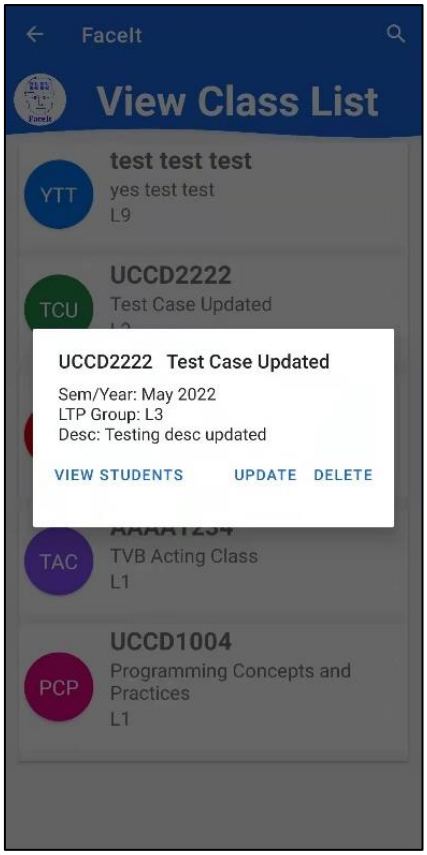


Figure 5.7 Class Details Dialogue Box

In the pop-up dialogue box shown in the figure above, users are presented with three options – view students, update and delete. The first option triggers the Enrolment module, which will list out the students who are currently enrolled in the selected class. This option will be further explained in Chapter 7. Furthermore, if users select the update option, another pop-up dialogue box will be displayed, prompting the user to update the fields of the selected class. Once users save the updated fields, the class list is refreshed, and the changes are reflected. The update class dialogue box is illustrated below in Figure 5.8.

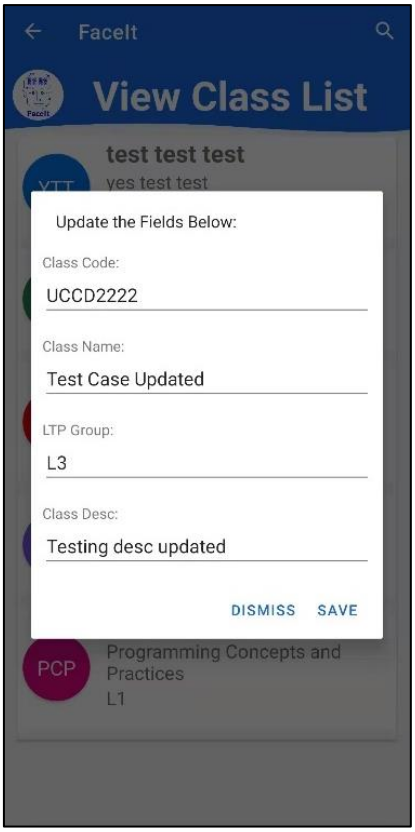


Figure 5.8 Update Class Dialogue Box

Furthermore, in the class details dialogue box, users can also opt to delete a class by clicking the Delete button. If a class is deleted, all the related enrolment records will be deleted too. Once a class has been deleted, the class list is refreshed automatically and a success message is shown to the user, as illustrated below in Figure 5.9.

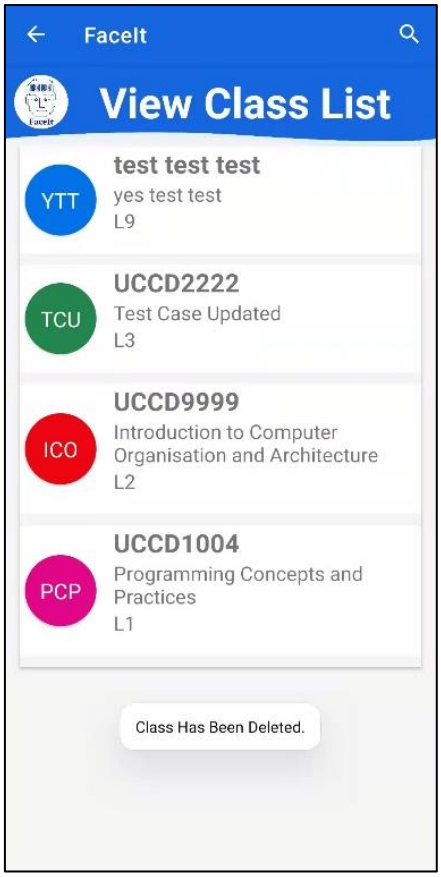


Figure 5.9 Delete Class Success Message

5.3 Module Testing

In this subsection, the test results of the Login and Register module are documented as tables. Each table represents a single test case, complete with test case ID, name, description, expected output as well as input and output (supported with screenshots) as evidence to the test cases.

5.3.1 Add Class

Table 5.2 to 5.4 below show the test cases of the Add Class functionalities. The test case IDs in this subsection all start with TAC, which stands for “Test Add Class”.

Table 5.2 TAC01: Add Class with Empty Fields

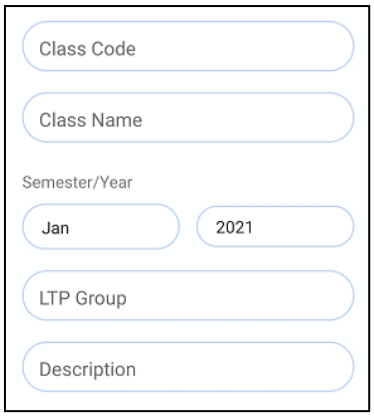
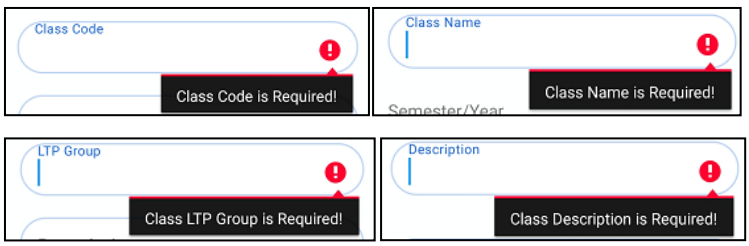
Test Case ID	TAC01
Test Case Name	Add Class with Empty Fields
Test Case Description	User presses the Add Class button with one or more empty fields
Expected Output	Display error message
Input	
Results	
Status (Pass/Fail)	Pass

Table 5.3 TAC02: Add Duplicated Class

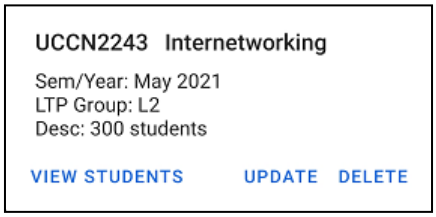
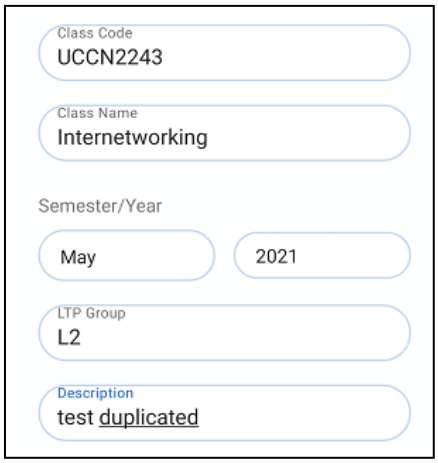
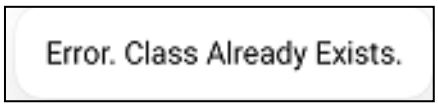
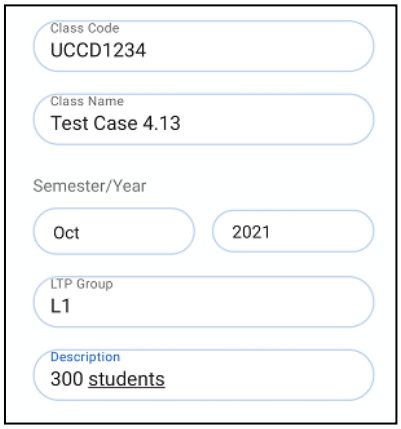
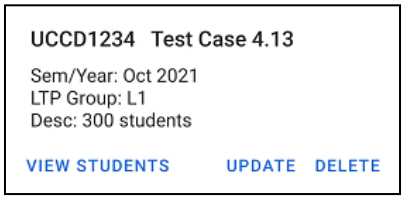
Test Case ID	TAC02
Test Case Name	Add Duplicated Class
Test Case Description	User enters a class that already exists
Expected Output	Error message shown
Input	 
Results	
Status (Pass/Fail)	Pass

Table 5.4 TAC03: Add Class with Valid Inputs

Test Case ID	TAC03
Test Case Name	Add Class with Valid Inputs
Test Case Description	User presses Add Class button with all the fields entered correctly
Expected Output	Add class to Firestore and show class details in class list
Input	
Results	
Status (Pass/Fail)	Pass

5.3.2 View Class List

Table 5.5 to 5.12 below show the test cases of the View Class List functionalities, including search, view, update and delete. The test case IDs in this subsection all start with TVC, which stands for "Test View Class".

Table 5.5 TVC01: View Class List Display

Test Case ID	TVC01
Test Case Name	View Class List Display
Test Case Description	User clicks on the View Class Listing button in the Main Dashboard page
Expected Output	Display the classes that **only** belong to the current user
Input	
Results	
Status (Pass/Fail)	Pass

Table 5.6 TVC02: View Class Details

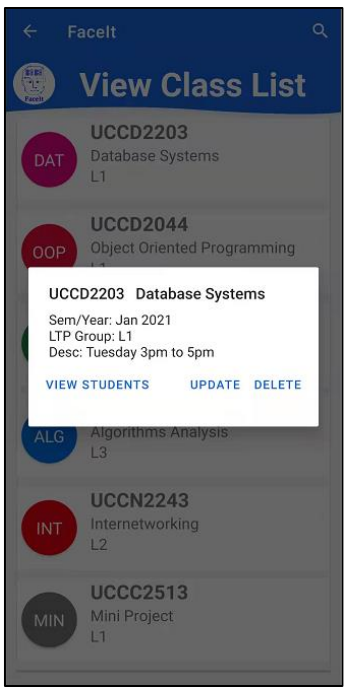
Test Case ID	TVC02
Test Case Name	View Class Details
Test Case Description	User clicks on a class record in the list of classes
Expected Output	Display a popup modal with the details of the clicked class record
Input	
Results	
Status (Pass/Fail)	Pass

Table 5.7 TVC03: Search Class

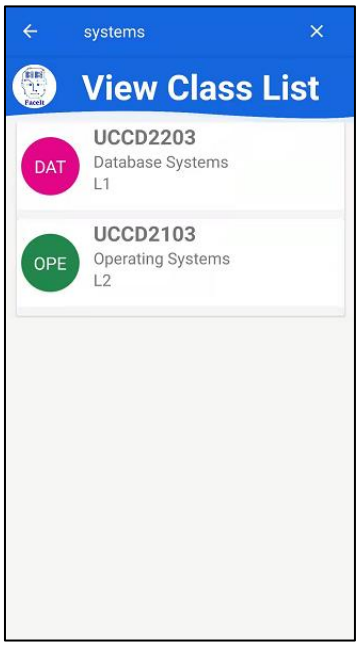
Test Case ID	TVC03
Test Case Name	Search Class
Test Case Description	User searches for a particular class using keyword-matching
Expected Output	Filter the list of classes in real-time as users type characters into the search box at the top of the page
Input	
Results	
Status (Pass/Fail)	Pass

Table 5.8 TVC04: Update Class Button

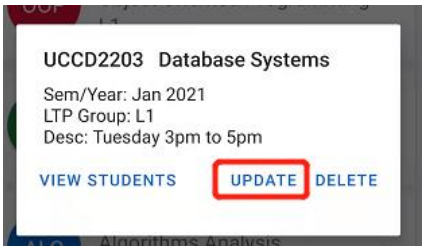
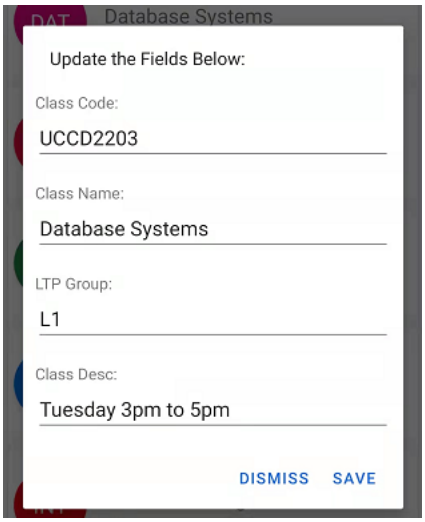
Test Case ID	TVC04
Test Case Name	Update Class Button
Test Case Description	User clicks the Update button in the View Class Details modal
Expected Output	Replace the current modal with a modal populated with editable textboxes.
Input	
Results	
Status (Pass/Fail)	Pass

Table 5.9 TVC05: Update Class with Empty Fields

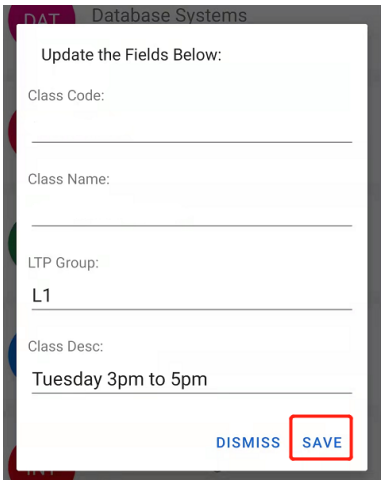
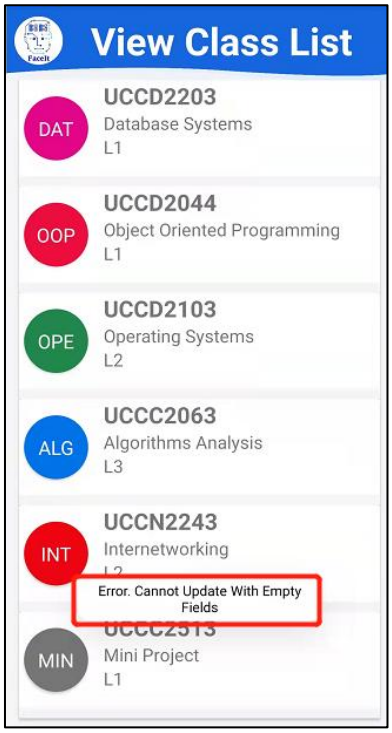
Test Case ID	TVC05
Test Case Name	Update Class with Empty Fields
Test Case Description	User clicks on the Update button with empty fields present
Expected Output	Display error message
Input	
Results	
Status (Pass/Fail)	Pass

Table 5.10 TVC06: Update Class with a Duplicated Class

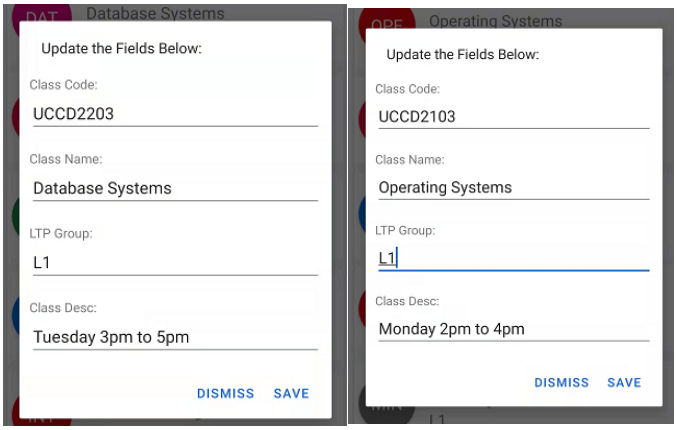
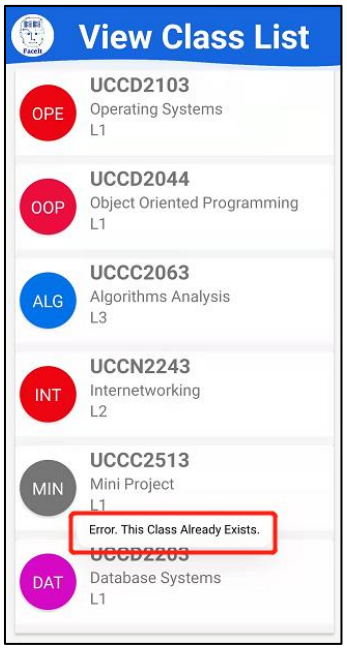
Test Case ID	TVC06
Test Case Name	Update Class with a Duplicated Class
Test Case Description	User clicks on the Update button with class details that clash with another class under the same user
Expected Output	Display error message
Input	The classes original details are shown on the left, and then the class details were updated to match that of an existing class in the class list, as shown on the right. 
Results	
Status (Pass/Fail)	Pass

Table 5.11 TVC07: Update Class with Valid Inputs

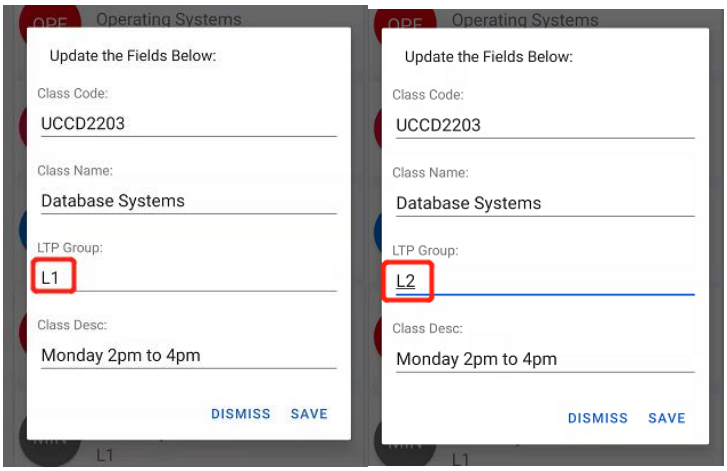
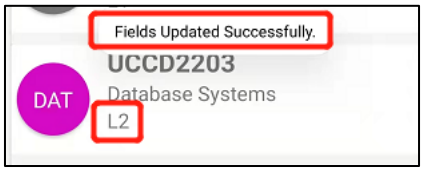
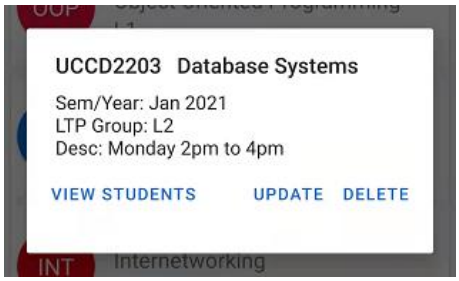
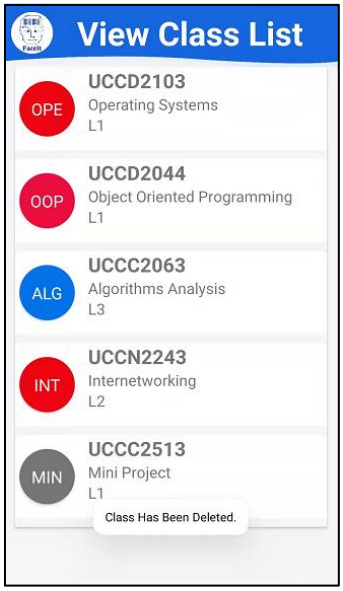
Test Case ID	TVC07
Test Case Name	Update Class with Valid Inputs
Test Case Description	User presses Update button with all fields entered correctly.
Expected Output	Display success message and redirect user back to the View Class List activity.
Input	
Results	
Status (Pass/Fail)	Pass

Table 5.12 TVC08: Delete Existing Class

Test Case ID	TVC08
Test Case Name	Delete Existing Class
Test Case Description	User presses the Delete button in the view class details dialogue box
Expected Output	Display success message and refresh class list
Input	
Results	
Status (Pass/Fail)	Pass

The test cases in this section evidence that the functionalities of the Class module work as expected. In the Class module, users can manage classes (add, view, update and delete) that are only visible to them, and not other users.

Having added classes to the developed application, the next step is to add in new student records. In the next chapter, the functionalities of the Student module are explained and tested.

Chapter 6

Student Module Implementation

In this chapter, the development, implementation, and testing processes of the Student module are discussed in detail. This chapter starts off with the database schema used for this module, then proceeds with the explanation of the MobileFaceNet face recognition model before demonstrating the flow of the Student module, supported with screenshots of the application's interfaces. Last but not least, this chapter wraps up with the testing results for this module.

6.1 Database Design

The Student module stores student details and student face images into Firebase's Cloud Firestore and Storage respectively. The latter is used mainly to store larger files such as image and video files. The student details are stored in two separated collections – students and embeddings. Each of the documents in the students and embeddings collections has its own autogenerated primary key. The attributes of each student are tabulated below in Table 6.1.

Table 6.1 Database Design (Student Module)

Collection Name	Attribute Name	Attribute Description	Attribute Type	Attribute Example
Students	StudentID	Student's ID	String	1801600
	StudentName	Student's Name	String	Jacynth Tham Ming Quan
	StudentPhone	Student's phone number	String	0122799255
	StudentEmail	Student's email address	String	jctmq25@gmail.com
	StudentGender	Student's gender	String	Female
	SchoolName	Student's school name (automatically set to follow current user)	String	UTAR

	IdentiSibs	Concatenated list of identical siblings' student IDs and names	String	0000001, Max; 1111111, Drew
Embeddings	map	JSON-formatted HashMap containing student ID, name, and face embedding float array	String	N.A.

The primary key of each student is autogenerated, but the developed application also checks to ensure that each student has a unique student ID.

6.2 MobileFaceNet Face Recognition Model

The Student module is one of the two modules that require the use of the MobileFaceNet face recognition model, the other being the Attendance module. Therefore, before the implementation of the UI and functionalities are discussed, the concept of MobileFaceNet is explained in this section.

MobileFaceNet is an efficient CNN model that aims to be able to recognize faces in real-time on mobile devices. With a file size of only 5.2 MB, MobileFaceNet can be swiftly integrated into mobile applications without the need for high progressing capabilities, unlike desktop-based face recognition models. Figure 6.1 below shows the details of the properties of the MobileFaceNet model, as observed using Lutz Roeder's Netron, a visualizer for deep learning and machine learning models [43].

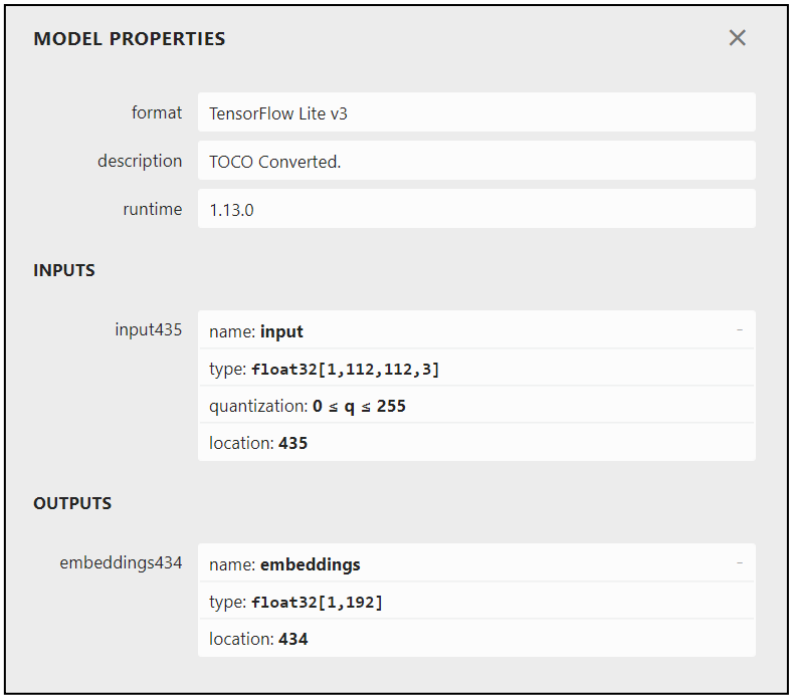


Figure 6.1 MobileFaceNet Model Properties

As evidenced by the figure above, MobileFaceNet is a TensorFlow Lite model, which means that it is compressed up to a point where even mobile applications with mediocre processing capabilities are able to use the model effectively. The input of the model is a 112 X 112 face image, which is obtained from the face detection and post-image processing pipeline. The output of the CNN model is a float array of [1, 192], which stores the embeddings of the face image. Face embeddings are numbers that represent the most important features of a face. Similar faces have similar embeddings; hence, these embeddings can be used in the attendance-taking process to distinguish between students. Figure 6.2 below illustrates the generation of the face embeddings.

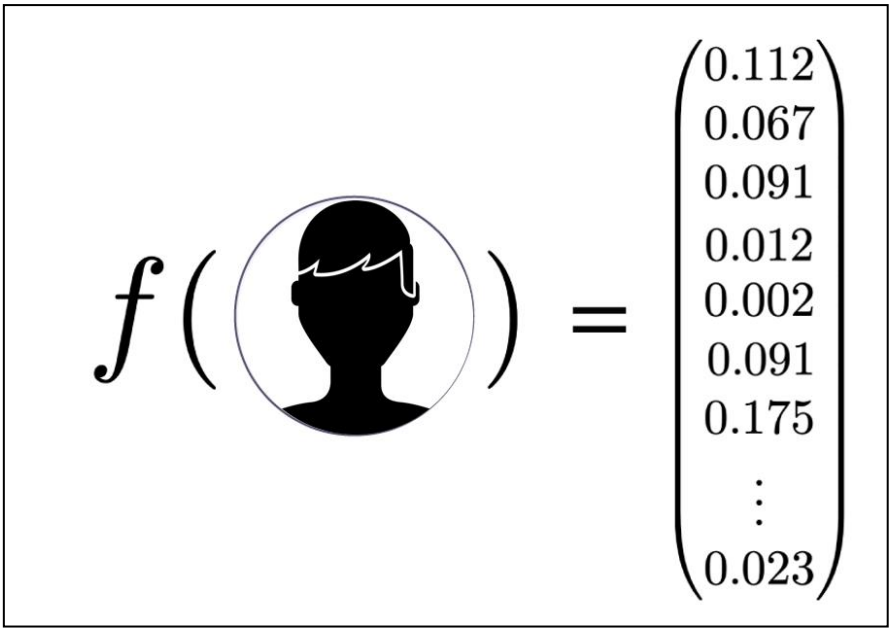


Figure 6.2 Face Embeddings Generation

The face embedding example above is similar to how the face embeddings of students are stored in the *embeddings* collection in Firebase. An example of a document in the *embeddings* collection is shown below in Figure 6.3.

{"111,testing 1,[222,testing 2;333,testing 3]":{"distance":-1.0,"extra":[[-0.0028201102,0.013234848,0.0039795553,-0.01667367,-0.02187877,-0.093243085,-0.05437645,-0.024657812,-0.14780812,-0.2527184,0.012683691,0.004508838,4.4323973E-4,0.0070509845,-6.6525943E-4,-0.08922681,-0.011699719,0.0026616661,0.0059563266,0.0029450255,0.26445475,-0.004070946,-0.083377786,0.008706882,0.082031995,0.0193902,-0.017313192,0.13727035,-0.09289302,0.0058812965,1.2294488E-4,0.048995737,0.14059955,-0.008658823,-0.017986758,-0.0022529406,-0.050243873,0.009502404,0.006510533,-0.0028245014,-0.009323587,-9.82665E-4,-0.008148588,0.0026238945,0.0058017094,0.013350498,-0.07466249,0.09885112,-0.012084255,0.008553415,-0.10609727,-0.002765175,-0.08647795,-0.0032244478,0.025742924,0.0059445114,0.097933024,-8.3716423E-4,-0.001465813,0.012159007,-0.029835023,0.02603151,0.04413137,0.09628356,0.0016891517,0.15644847,0.0012917799,0.013480399,0.005408481,0.0014257738,-3.7736082E-4,-0.25812316,-0.08446571,-0.004940844,-0.036391262,0.013126872,0.0037529657,0.0026393617,0.16557275,-0.03486491,0.005332898,0.094178766,-0.008073187,0.051214054,0.009263172,0.0016274261,-0.0011056421,-0.027751463,0.03140253,-0.07955401,0.10239355,-0.005673767,-0.006495044,-0.025549108,-0.059044342,0.07760128,-0.020821411,-0.011533014,-0.0086170705,0.029458217,0.002345433,0.0037939094,-0.0053456174,-0.0032606241,0.005806619,0.0076739043,-0.05441026,0.0086885905,0.016000224,0.015419929,-0.009580251,0.005011023,0.0051076133,-0.077611215,-0.005099198,0.08371476,0.002797931,0.009794326,0.11655743,-0.16616814,0.045716826,0.0077021746,-0.14346957,-5.4916356E-5,0.001401919,0.013527207,0.0072167967,-0.0092095025,0.008000619,-0.13460727,0.0013607936,0.011116416,-2.0238638E-4,0.011474847,-0.002105661,0.0029697106,-0.069775224,7.9376914E-4,-0.0075253197,0.004175214,0.0051130424,-0.0045095207,0.0046317247,-0.19329481,0.058117796,0.20285569,-0.010843955,0.004135185,-0.008983554,0.011702281,-0.011779148,-0.06267674,-0.07634656,-0.015545114,0.002531279,0.0035093199,0.006247374,0.0113611305,0.03935491,7.7573326E-4,-0.013650588,0.0028630244,-0.002787062,-0.0030678154,0.0061336067,0.00655688,-0.0034003416,0.33955556,-0.007993038,-0.0044819294,-0.10451092,-0.09213649,-6.075158E-4,-0.053434946,0.015659666,-0.00409299,-0.030128673,0.025076851,-0.0069567957,-5.1918207E-4,-0.15583158,-0.12903792,0.0033305504,-8.6323154E-4,-0.013559549,0.14441043,-0.04960943,0.05671279,0.3376471,-0.080660716,-0.010878728,-0.010685622]],"id":"0","title":""}}

Figure 6.3 Face Embeddings Example

The map field in the embeddings collection stores an Object that has been converted to a JSON string. This stored object only consists of one key-value pair. The key in the figure above is highlighted with a red border. In this project's implementation, the key contains a string value that is made up of the student name and student ID of the current student as well as the student names and IDs of all the identical siblings of the current student. The value to this key is another Object that contains the face embedding values such as in Figure 6.2 above.

In the following subsection, the functionalities of the Student module are explained, including details on how the face embeddings explained above are generated during the Add Student process.

6.3 Implementation of UI and Functionalities

The Student module consists of two activities – Add Student and View Student List. From the main dashboard, users can access these two activities through the corresponding buttons with the activities' names annotated on them, as shown in Figure 6.4 below:

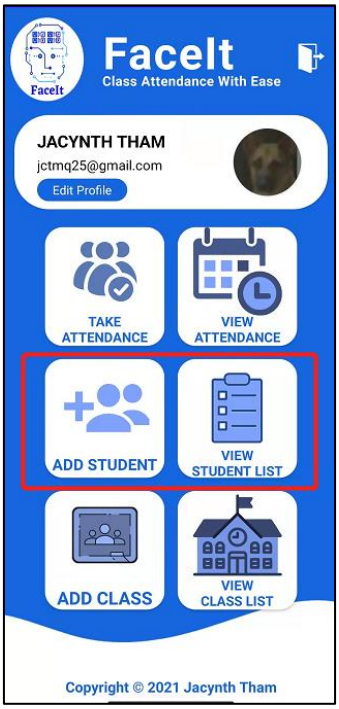


Figure 6.4 Buttons Associated with the Class Module

The Add Student activity can be further broken down into two consecutive sub-activities, which are the Add Student Detail and Add Student Face activities. When users click on the Add Student button in the Main Dashboard page, they will be presented with the Add Student Detail activity first. The UI for the Add Student Detail activity is shown below in Figure 6.5.

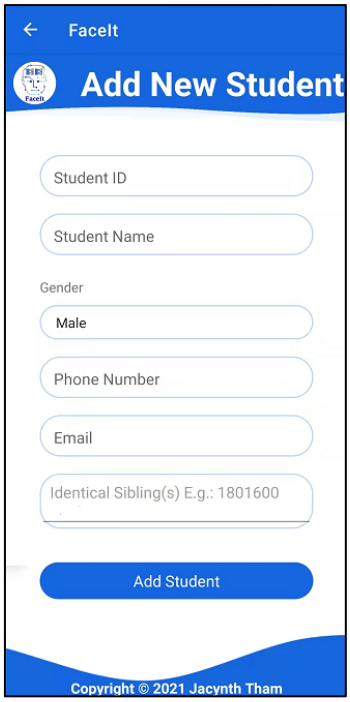


Figure 6.5 Add Student Detail Page

In the Add Student Detail activity shown above, users have to fill in the student ID, full name, gender, phone number, email and student ID of a single identical sibling of the student to-be-added. The last field is optional as not every student has an identical sibling. When the user presses the Add Student button, the application validates all the fields to ensure that the values entered are in the correct format. As for the identical siblings student ID, the application checks to see if the student ID exists in the database. if the student ID does not exist, the application displays an error message to the user. On the other hand, if the student ID of the identical sibling exists, the application searches for that particular student in the database and obtains the list of all identical siblings of the inputted student ID. Then, the application concatenates the student ID and name of the inputted identical sibling student ID and save it in the student record of the newly added student.

To illustrate, a set of triplets are used in the following example. First, Student A with student ID “1111” is added into the application, with the identical siblings field left empty because none of the other students exist in the application’s database. Next, Student B with student ID “2222” is added to the system. During this process, the identical siblings field of Student B shall have the input value “1111” to generate an identical sibling linkage between Student A and Student B. Next, Student C is added to

the application. At this point, the user only has to enter either Student A or Student B's student ID. Assuming that Student A's student ID has been entered by the user when adding Student C with student ID "3333", the application will detect that Student A already has a linkage to Student B, and therefore, will generate a linkage between Student A, B and C to signify that three of them are identical to one another. The final output of the IdentiSibs (identical siblings) field in the database for the three students is shown in Table 6.2 below:

Table 6.2 Identical Siblings Addition Example

Student Name	Student ID	IdentiSibs Field
Student A	1111	2222, Student B; 3333, Student C
Student B	2222	1111, Student A; 3333, Student C
Student C	3333	1111, Student A; 2222, Student B

The concept of the identical siblings as explained in the example above applies to students with multiple identical siblings and is highly scalable. In order to bring convenience to users, only the student ID of one identical sibling needs to be filled in, and the application generates the identical siblings linkage based on the identical siblings of the student ID inputted.

Once the user has submitted the Add Student Detail activity, the details of the new student are sent to the cloud database and stored in a document under the *students* collection. Figure 6.6 below shows the example of a document in the *students* collection.

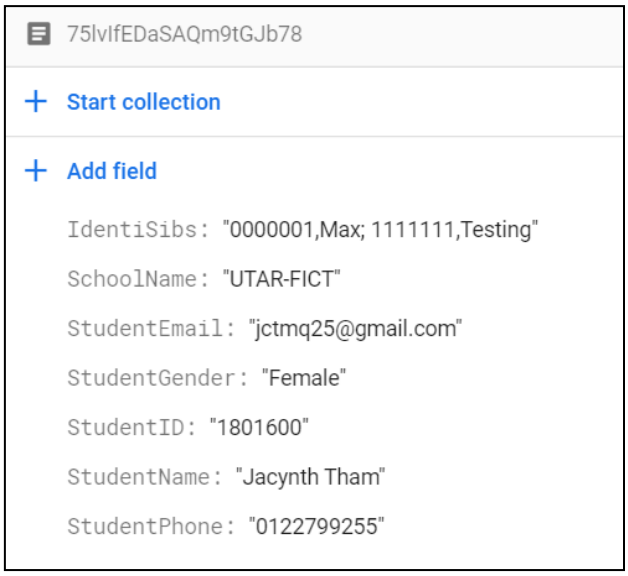


Figure 6.6 Example Document in the Students Collection

The figure above evidenced that an additional field – SchoolName – is added for every newly added student. The SchoolName value of the new student is the same as the school's name of the user who had added the student and is automatically added into the student record by the application. This is to ensure that only users within the same school may view the information of the students in that particular school. Moreover, the documents of the students selected as the identical siblings of the new student are also updated so that they contain the student ID and name of the new student in their IdentiSubs fields.

After the user has completed the Add Student Detail activity, the user will be directed to the next sub activity, which is the Add Student Face activity. This activity requires students to be present in order to add their face data into the application's database. This activity is responsible for capturing, processing and storing the students' faces to the cloud database. In this activity, the MobileFaceNet face recognition model is used to generate the face embeddings of the student to-be-added. When users first launch this sub activity, users are shown the UI display as seen below in Figure 6.7.

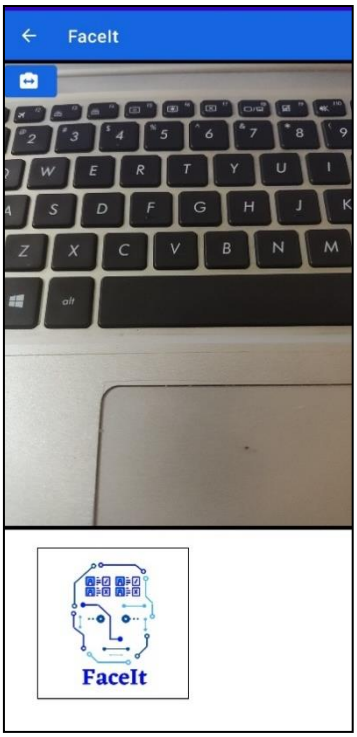


Figure 6.7 Add Student Face Activity (No Face Detected)

When no faces are detected, the image in the box at the bottom corner of the page displays the FaceIt logo. At this point, the camera feeds the real-time video stream

frames into Firebase's Face Detection API one by one. For each frame, the Face Detection API checks if any faces are present. If one or more faces are detected, then all the detected faces are highlighted with red bounding boxes and an Add Face button appears on screen. The bounding boxes are able to follow the detected face as the face move around within the frame. Figure 6.8 below illustrates this display.

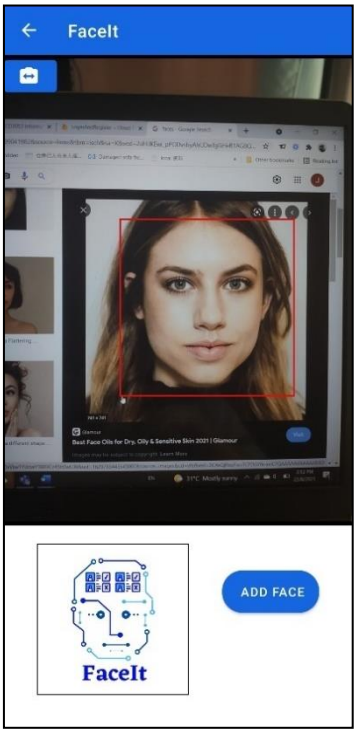


Figure 6.8 Add Student Face Activity (Face Detected)

When the Add Face button is clicked on, the most prominent face of all the bounded faces (more than one face can be bounded) is captured and displayed in the bottom-left box. This serves as a preview to users, and they are allowed to click the Add Face button as many times as they want in order to get the perfect shot. While a preview image is being shown in the bottom left box, a Save Face button will be shown on screen below the Add Face button. Figure 6.9 below shows the UI for this view.

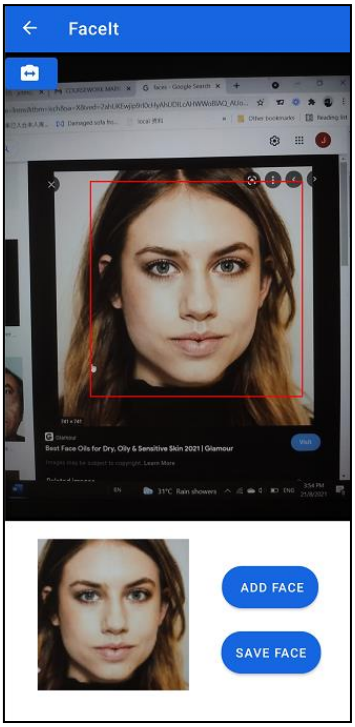


Figure 6.9 Add Student Face Activity (Preview Face)

When users click the Add Face button over and over, the preview image in the bottom left corner changes accordingly. If the detected face moves out of frame during this process, then the preview image is replaced with the default FaceIt logo and the display returns to the one shown in Figure 6.7 above.

Once the user is completely satisfied with the image displayed, the user may click on the Save Face button to store the newly captured face into the cloud database. When the Save Face button is clicked, the face image is cropped out, rotated, scaled, and fed into the MobileFaceNet model to be converted to a face embedding, which is represented by a float array. This float array is then concatenated with the new student's ID and name as a HashMap and converted to JSON format to be stored in the cloud database as a document in the *embeddings* collection. Figure 6.10 below shows an example of the face embeddings stored in the cloud database.

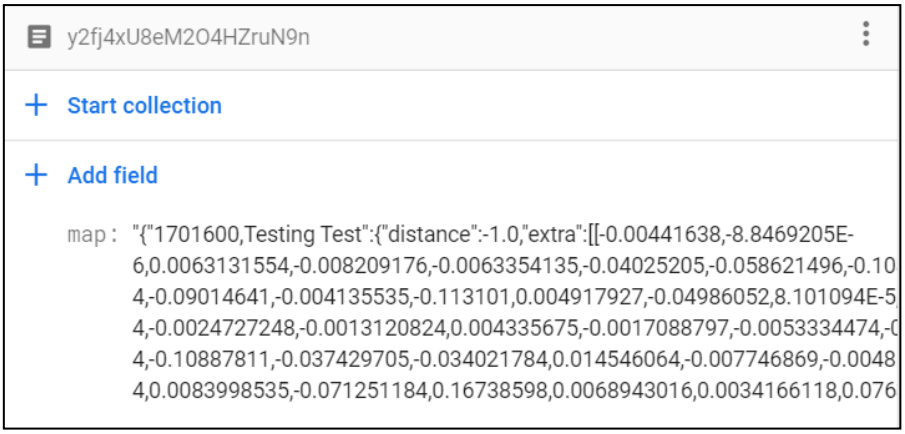


Figure 6.10 Example Document in the *Embeddings* Collection

Furthermore, the face image of the new student is also stored in Firebase Storage in the studentPic folder. The name of the face image is set to the documentID of the newly added student so that the application can retrieve the face image of each unique student later on. Figure 6.11 depicts an example of a face image in Firebase Storage.

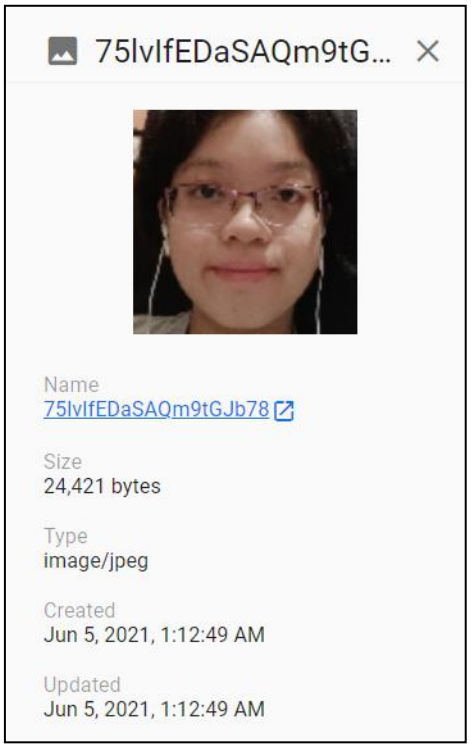


Figure 6.11 Example Face Image in Firebase Storage

Next, users can observe the full list of students by clicking the View Student List in the main dashboard. Figure 6.12 below shows the UI for the View Student List activity.

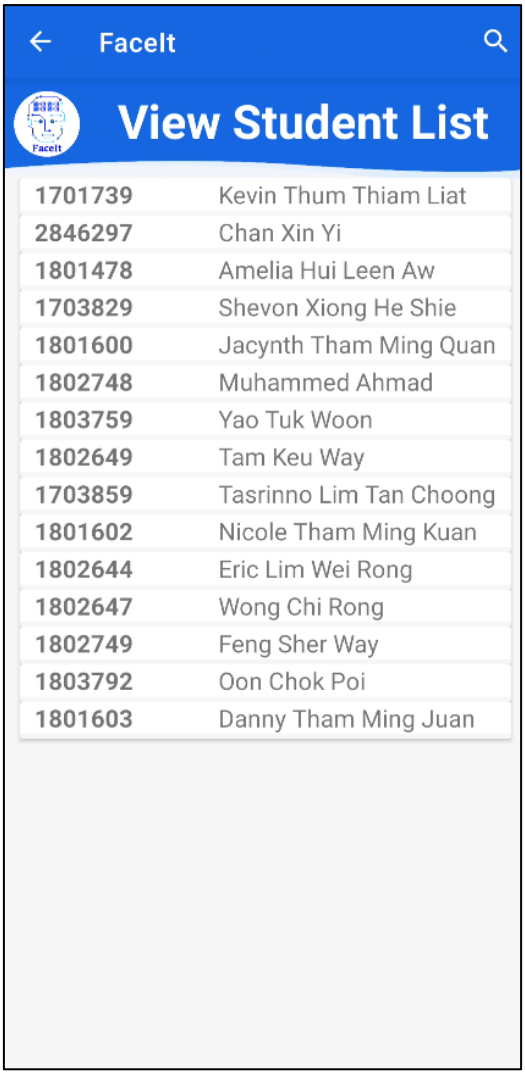


Figure 6.12 View Student List Page

In the View Student List activity, users can view the student's face icons, student ID and full name. If users wish to look for a particular student, users may click on the magnifying glass at the top right corner to initiate the search view. Similar to the search view of the View Class List activity, the searched student list is updated dynamically with every character entered. Figure 6.13 below shows the search view in the View Student List activity.

Figure 6.13 View Student List Search View

If the users click on a particular student, a pop-up dialogue box displaying the student's full details is shown to the user. Figure 6.14 below shows an example of the dialogue box shown.

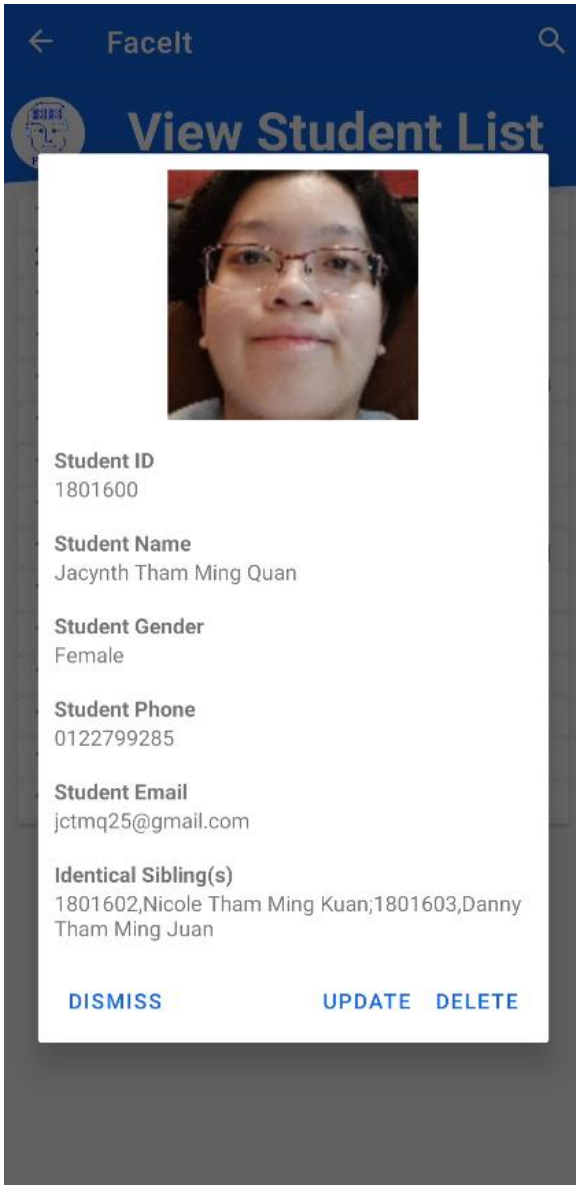


Figure 6.14 View Student Detail Dialogue Box

Besides displaying the selected student's full details, the View Student Detail dialogue box also shows the options to update or delete the selected student. If the Update button was clicked on, users are presented with another pop-up dialogue box with editable text fields so that they may update the required fields. Figure 6.15 below shows the Update Student dialogue box.

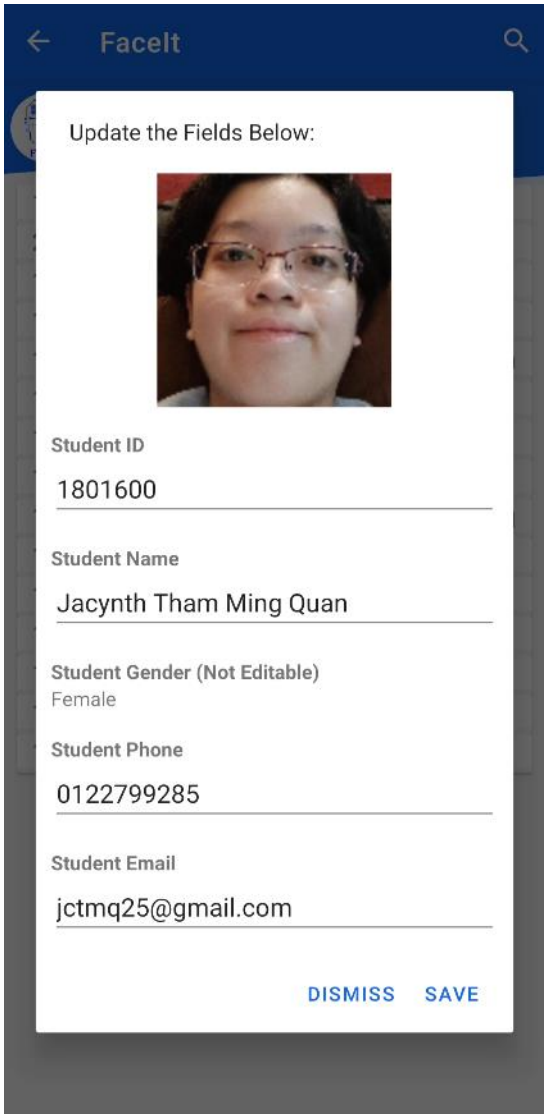


Figure 6.15 Update Student Dialogue Box

As evidenced by the figure above, the updated fields below include the student's face image (users have to click on the profile picture displayed at the top), student ID, student name, phone number and email address. The gender field is not editable as students are not expected to have their genders changed at any point thereafter. Once the user has filled in the fields to-be-updated, they have to click on the Save button to update the changes in the cloud database.

Next, users can also choose to delete a particular student by clicking on the Delete button in the View Student Details dialogue box. Once a particular student has been deleted, all the related enrolment records are deleted as well. However, all previous attendance records will be left as is. Furthermore, if the student to-be-deleted has identical siblings, then the IdentiSibs fields of all the identical siblings will be

updated to omit the student ID and name of the student to-be-deleted. After the process had been completed, the student list is refreshed automatically.

6.4 Module Testing

In this subsection, the test results of the Student module are documented as tables. Each table represents a single test case, complete with test case ID, name, description, expected output as well as input and output (supported with screenshots) as evidence to the test cases.

6.4.1 Add Student

Table 6.3 to 6.16 below show the test cases of the Add Student functionalities. The test case IDs in this subsection all start with TAS, which stands for “Test Add Student”.

Table 6.3 TAS01: Add Student Detail with Empty Fields

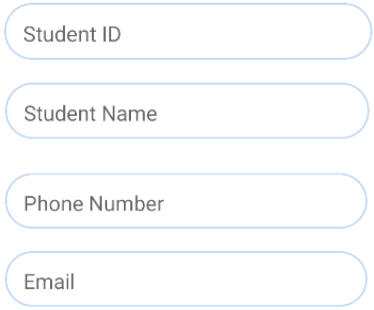
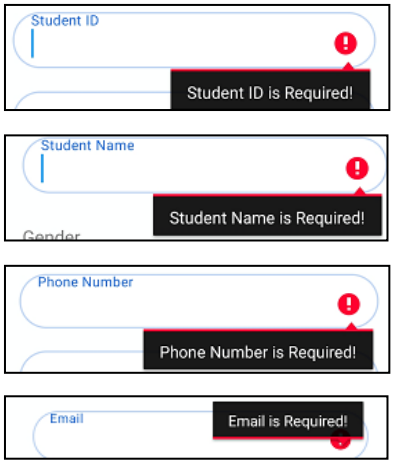
Test Case ID	TAS01
Test Case Name	Add Student Detail with Empty Fields
Test Case Description	User presses Next button with one or more empty fields (except identical sibling field, which is optional)
Expected Output	Display error message
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.4 TAS02: Add Student Detail with Invalid Email Format

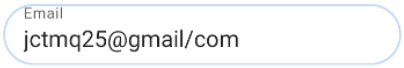
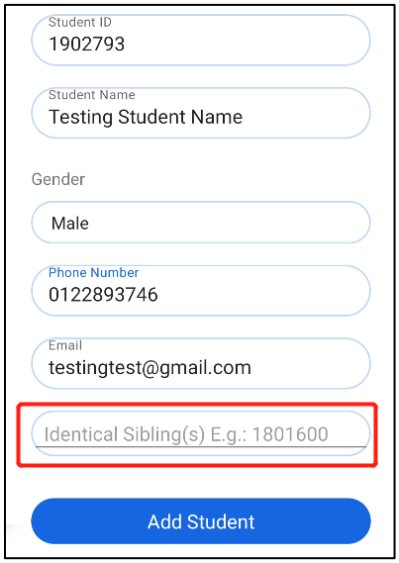
Test Case ID	TAS02
Test Case Name	Add Student Detail with Invalid Email Format
Test Case Description	User presses Next button with an invalid email entered
Expected Output	Display error message
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.5 TAS03: Add Student Detail with No Identical Siblings

Test Case ID	TAS03
Test Case Name	Add Student Detail with No Identical Siblings
Test Case Description	User presses Next button with no identical sibling student ID entered
Expected Output	Save the IdentiSibs field in the database as "None" and proceed to Add Student Face activity
Input	
Results	Firestore student record:

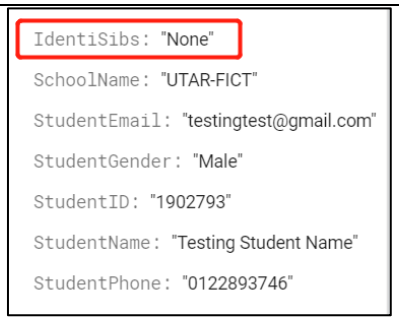
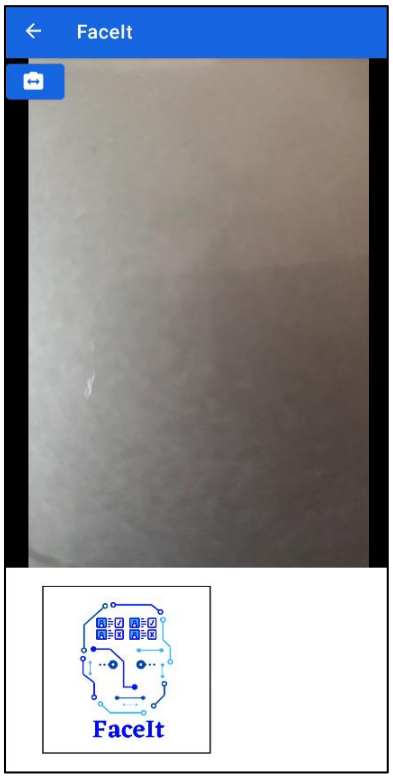
	 Redirected to Add Student Face activity: 
Status (Pass/Fail)	Pass

Table 6.6 TAS04: Add Student Detail with Unknown Identical Sibling Student ID

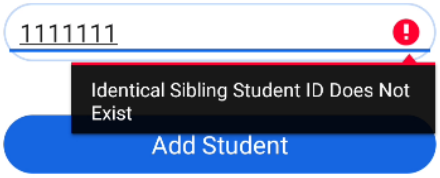
Test Case ID	TAS04
Test Case Name	Add Student Detail with Unknown Identical Sibling Student ID
Test Case Description	User presses Next button with an unknown identical sibling student ID entered
Expected Output	Display error message
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.7 TAS05: Add Student Detail with Duplicated Student ID

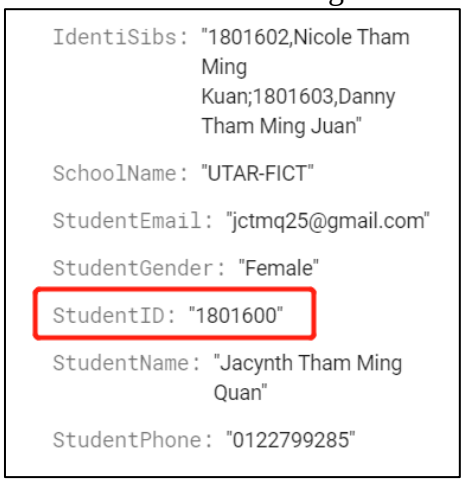
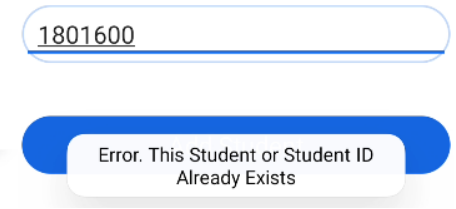
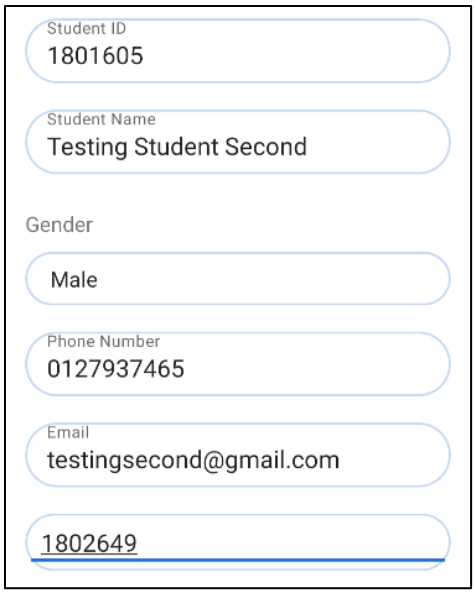
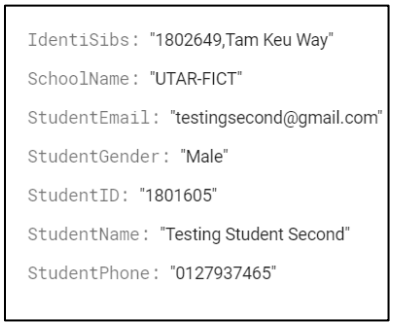
Test Case ID	TAS05
Test Case Name	Add Student Detail with Duplicated Student ID
Test Case Description	User presses Next button with a duplicated student ID entered
Expected Output	Display error message
Input	Firebase record of existing student:  Input in FaceIt: 
Results	
Status (Pass/Fail)	Pass

Table 6.8 TAS06: Add Student Detail with Valid Fields

Test Case ID	TAS06
Test Case Name	Add Student with Valid Fields
Test Case Description	User presses Next button with all fields entered correctly
Expected Output	Student is added successfully to Cloud Firestore and user is directed to the Add Student Face activity
Input	
Results	Firestore record:  Redirect to Add Student Face activity:

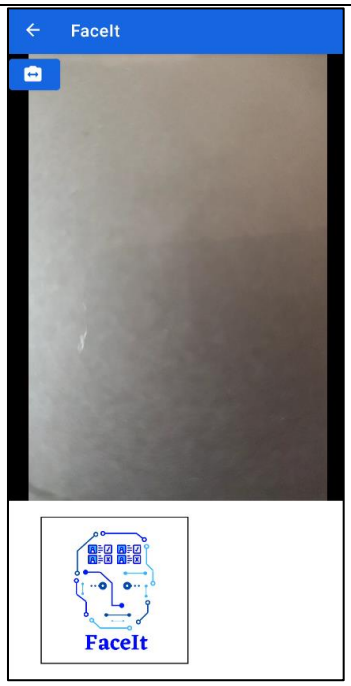
	
Status (Pass/Fail)	Pass

Table 6.9 TAS07: Add Student Face – No Face Detected

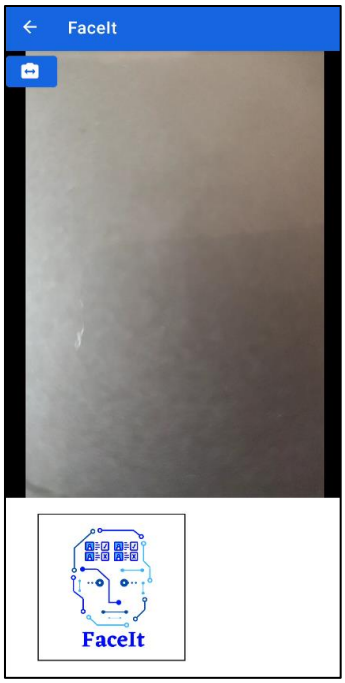
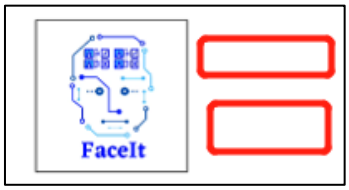
Test Case ID	TAS07
Test Case Name	Add Student Face – No Face Detected
Test Case Description	No faces detected in the Add Student Face activity
Expected Output	No bounded faces shown Hide Add Face and Save Face button
Input	
Results	No Add/Save button shown: 
Status (Pass/Fail)	Pass

Table 6.10 TAS08: Add Student Face – Single Face Detected

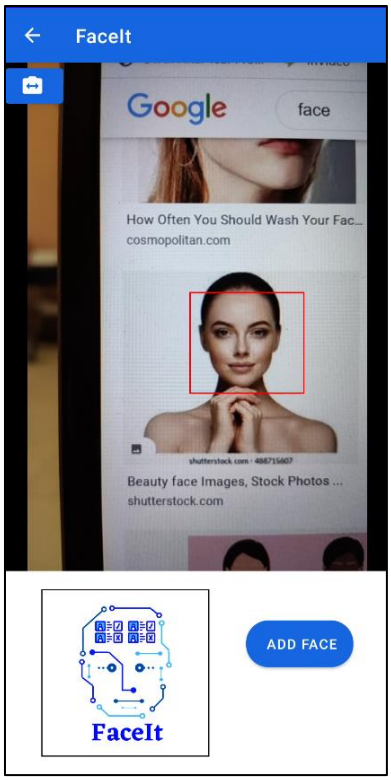
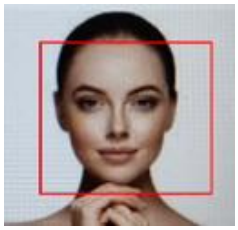
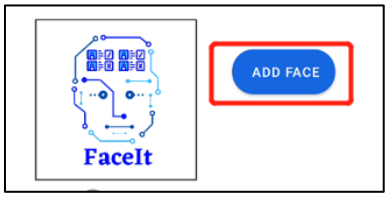
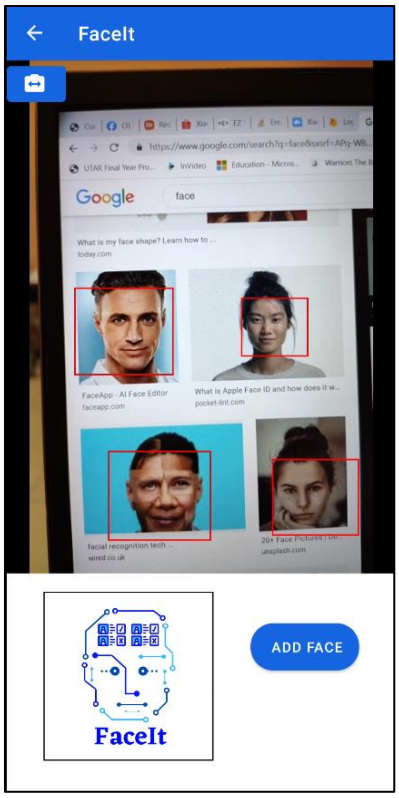
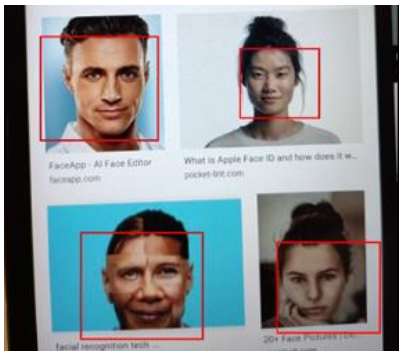
Test Case ID	TAS08
Test Case Name	Add Student Face – Single Face Detected
Test Case Description	One face detected in frame
Expected Output	Surround detected face with a bounding box Show Add Face button
Input	
Results	Bounding box shown:  Add Face button shown: 
Status (Pass/Fail)	Pass

Table 6.11 TAS09: Add Student Face – Multiple Faces Detected

Test Case ID	TAS09
Test Case Name	Add Student Face – Multiple Faces Detected
Test Case Description	More than one faces detected in frame
Expected Output	Surround detected faces with bounding boxes Show Add Face button
Input	
Results	Bounding box surrounding all detected faces:  Add Face button shown:

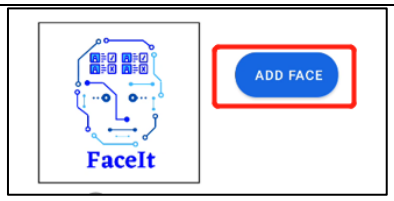
	
Status (Pass/Fail)	Pass

Table 6.12 TAS10: Add Face Button – Single Face Detected

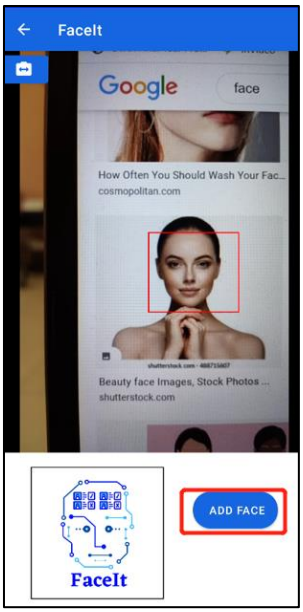
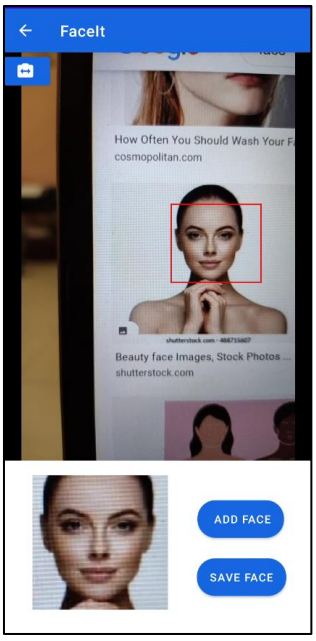
Test Case ID	TAS10
Test Case Name	Add Face Button – Single Face Detected
Test Case Description	User clicks on Add Face button with one face detected in frame
Expected Output	Bottom left frame is filled with the cropped image of the detected face
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.13 TAS11: Add Face Button – Multiple Faces Detected

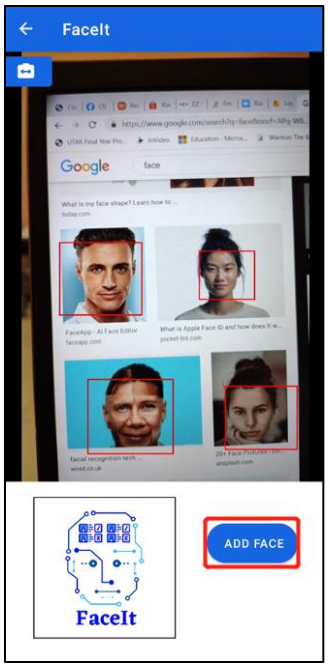
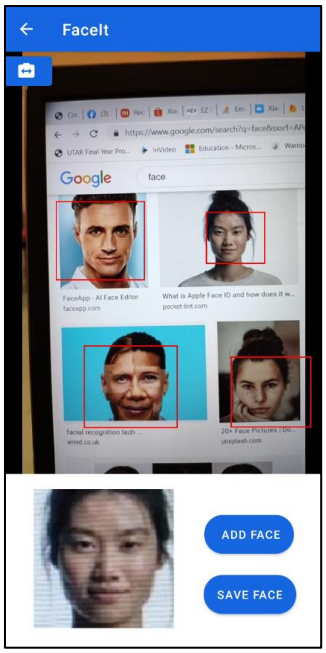
Test Case ID	TAS11
Test Case Name	Add Face Button – Multiple Faces Detected
Test Case Description	User clicks on Add Face button with more than one faces detected in frame
Expected Output	Bottom left frame is filled with the cropped image of the most prominent detected face
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.14 TAS12: Add Student Face – Detected Face Out of Frame

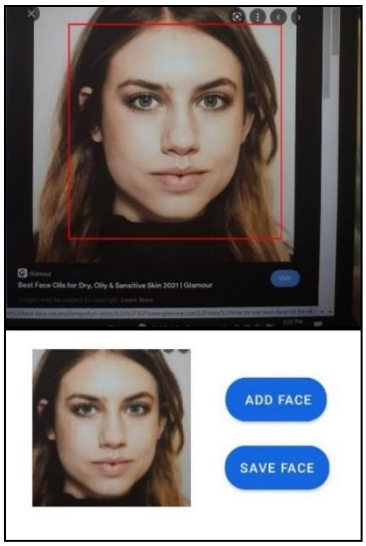
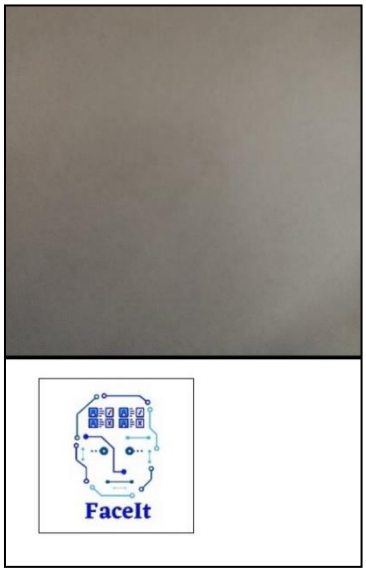
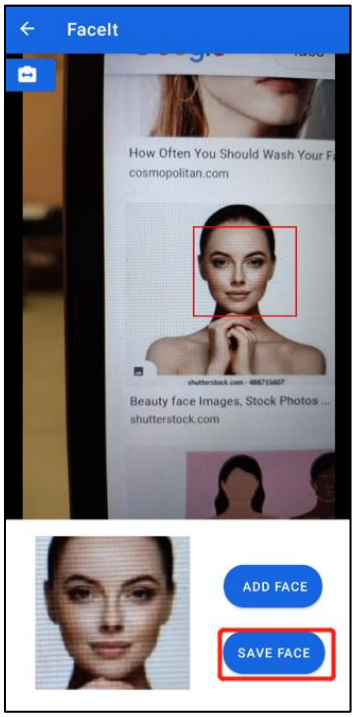
Test Case ID	TAS12
Test Case Name	Add Student Face – Detected Face Out of Frame
Test Case Description	Detected face moves out of frame
Expected Output	Bottom left frame changes to the default logo, the two buttons (Add Face and Save Face) turn invisible
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.15 TAS13: Save Face Button

Test Case ID	TAS13
Test Case Name	Save Face Button
Test Case Description	User clicks on the Save Face button
Expected Output	Convert preview face image to face embeddings. Store face embeddings in Firebase Firestore and preview image in Firebase Storage.
Input	
Results	In Firestore's embeddings collection: map: "{"1801605,Testing Student Second,[1802649,Tam Keu Way]":{"distance":-1.0,"ex 4,0.0020902622,-0.108434044,0.05824276,0.09733905,0.008002118,-0.05849 4,-0.069383934,0.015638173,0.0670718,-0.084808595,0.11098907,-0.0148037 4,-0.0047187344,-0.011105087,-0.21815541,-0.13937111,-0.13164115,-0.0829 4,0.0023332313,0.0035392733,0.0045275404,-0.007552585,0.006103754,-0.03 4,-8.391017E-4,0.0066038244,0.013257082,-0.0027815888,0.12000829,7.8052

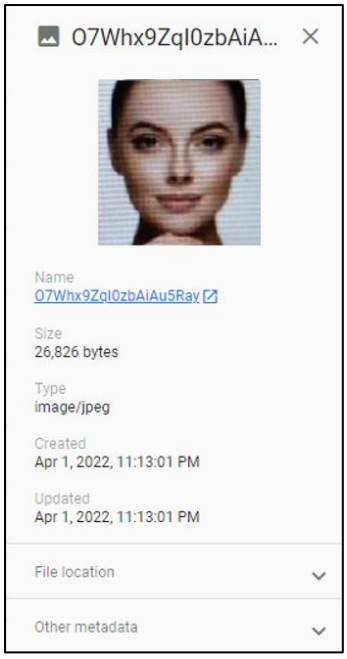
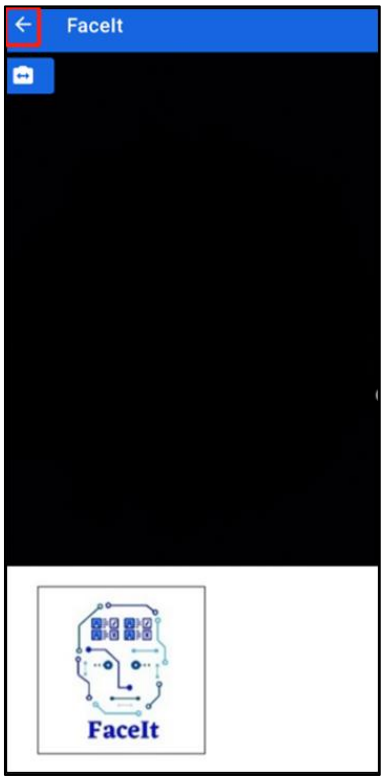
	In Firebase Storage: 
Status (Pass/Fail)	Pass

Table 6.16 TAS14: Add Student Face – Back Button

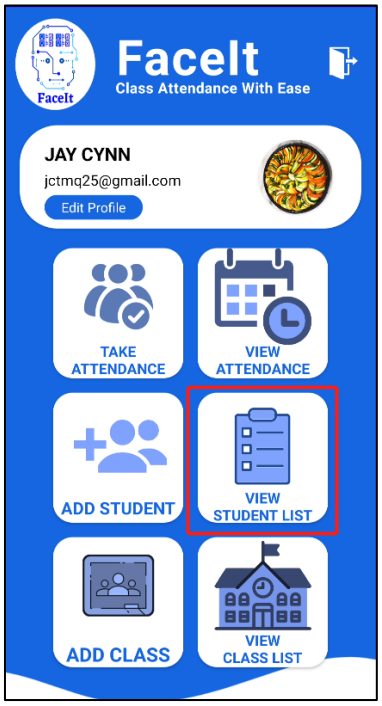
Test Case ID	TAS14
Test Case Name	Add Student Face – Back Button
Test Case Description	User clicks on the Back button while the Add Student Face activity has not been completed
Expected Output	Display error message
Input	

Results	
Status (Pass/Fail)	Pass

6.4.2 View Student List

Table 6.17 to 6.25 below show the test cases of the View Student List functionalities, including view details, search, update and delete. The test case IDs in this subsection all start with TVS, which stands for "Test View Student".

Table 6.17 TVS01: View Student List – Display List

Test Case ID	TVS01
Test Case Name	View Student List – Display List
Test Case Description	User clicks on the View Student List button in the Main Dashboard
Expected Output	Display students in the same school as the user only
Input	

Results	
Status (Pass/Fail)	Pass

Table 6.18 TVS02: View Student List – View Student Details

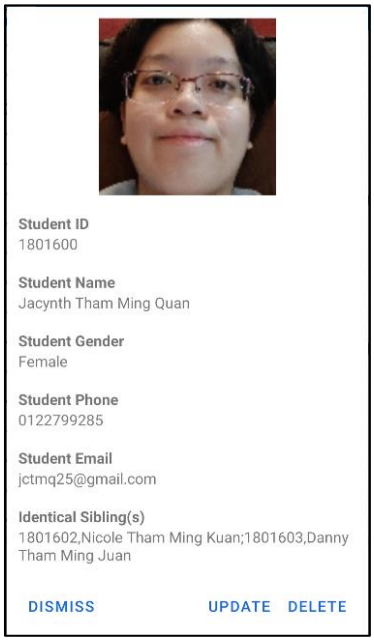
Test Case ID	TVS02
Test Case Name	View Student List – View Student Details
Test Case Description	User clicks on a student record in the list of students
Expected Output	Display a popup modal with all the student's details
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.19 TVS03: View Student List – Update Button

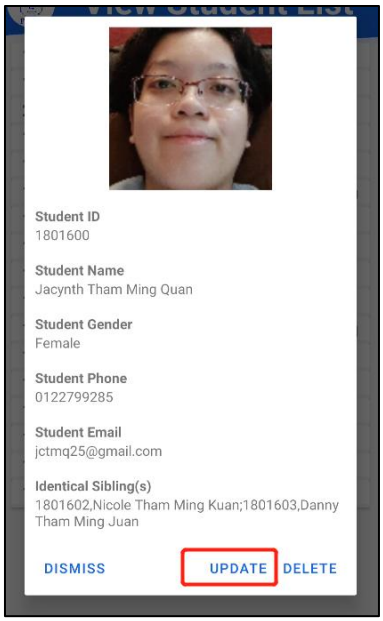
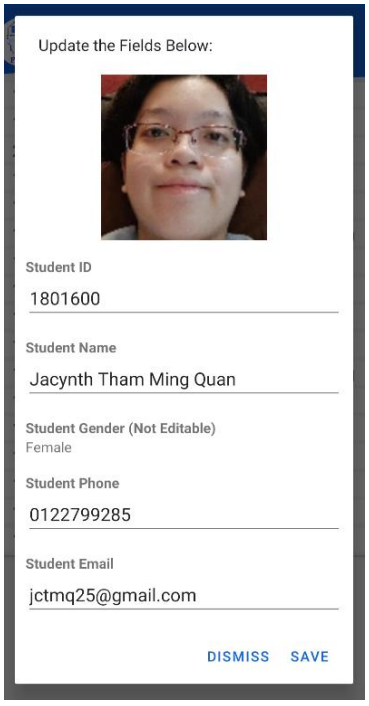
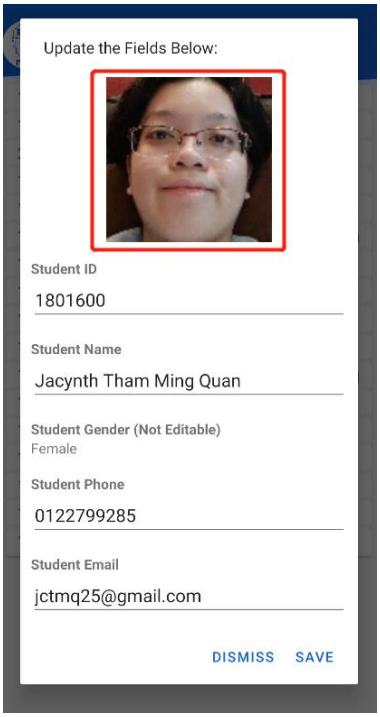
Test Case ID	TVS03
Test Case Name	View Student List – Update Button
Test Case Description	User clicks on the Update button in the View Student Details modal
Expected Output	Replace the current modal with a modal consisting of editable fields.
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.20 TVS04: Edit Student Image

Test Case ID	TVS04
Test Case Name	Edit Student Image
Test Case Description	User clicks on the student face image at the top of the Edit Student modal
Expected Output	Launch the Add Student Face activity to replace the student face image with a new one
Input	

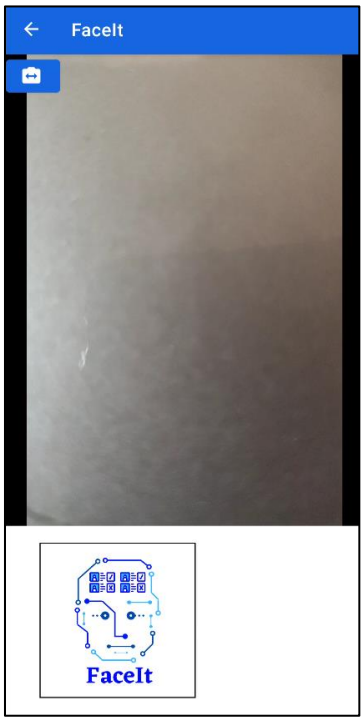
Results	
Status (Pass/Fail)	Pass

Table 6.21 TVS05: Edit Student Details (Empty Fields)

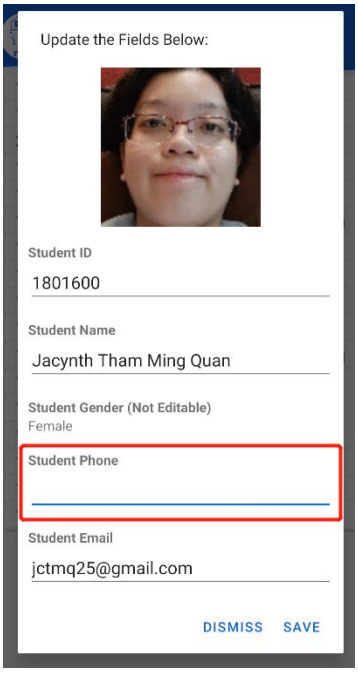
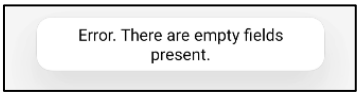
Test Case ID	TVS05
Test Case Name	Edit Student Details (Empty Fields)
Test Case Description	User tries to update the student with empty fields in the Edit Student modal
Expected Output	Display error message
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.22 TVS06: Edit Student Details (Invalid Email Format)

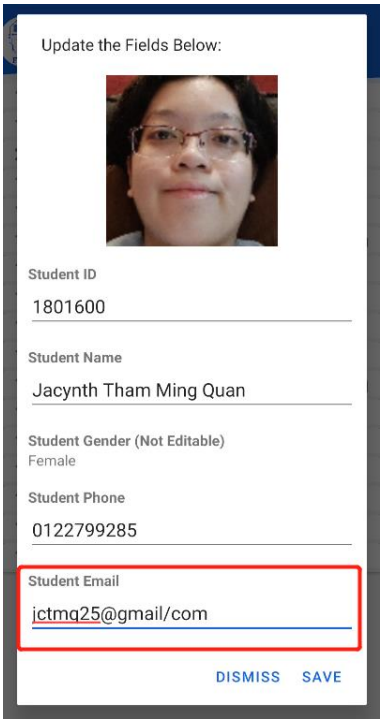
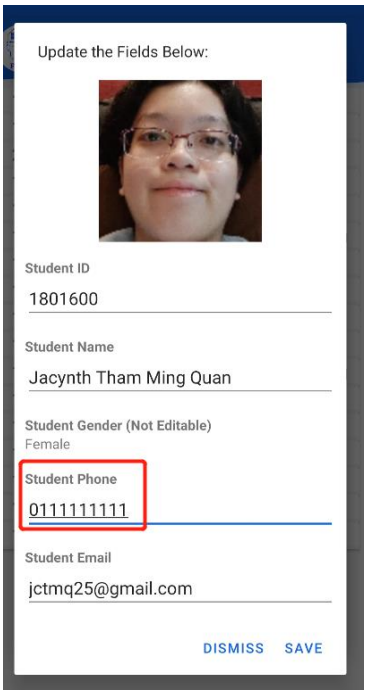
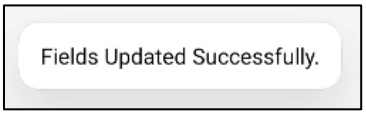
Test Case ID	TVS06
Test Case Name	Edit Student Details (Invalid Email Format)
Test Case Description	User tries to update the student with an invalid email address
Expected Output	Display error message
Input	
Results	
Status (Pass/Fail)	Pass

Table 6.23 TVS07: Edit Student Details (Valid Fields)

Test Case ID	TVS07
Test Case Name	Edit Student Details (Valid Fields)
Test Case Description	User updates the student record with all valid fields
Expected Output	Display success message and update student details in cloud database
Input	
Results	

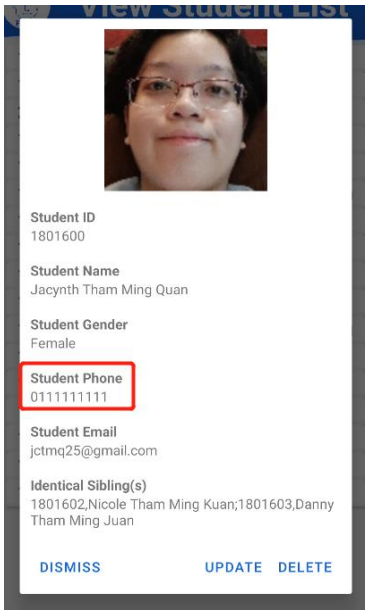
	
Status (Pass/Fail)	Pass

Table 6.24 TVS08: Delete Student

Test Case ID	TVS08
Test Case Name	Delete Student
Test Case Description	User clicks on the Delete button in the View Student Details modal
Expected Output	Display success message and update student list view
Input	Initial List:  

Results	
Status (Pass/Fail)	Pass

Table 6.25 TVS09: Search Student

Test Case ID	TVS09
Test Case Name	Search Student
Test Case Description	User searches for a student by keying in the corresponding student ID or name
Expected Output	Update the student list in real-time according to the user's keystrokes
Input	
Results	
Status (Pass/Fail)	Pass

The test cases in this section evidence that the functionalities of the Student module work as expected. In the Student module, users can manage students (add, view, update and delete) that are only visible to other users within the same school.

At this point, it is assumed that users have already added Classes and Students into the application. The next step is for users to enrol existing students into existing classes. The next chapter describes the Enrolment module, which is responsible for this.

Chapter 7

Enrolment Module Implementation

In this chapter, the development, implementation, and testing processes of the Enrolment module are discussed in detail. This chapter starts off with the database schema used for this module, then proceeds with the flow of the Enrolment module, supported with screenshots of the application's interfaces. Last but not least, this chapter wraps up with the testing results for this module.

7.1 Database Design

The Enrolment module stores enrolment records into Firebase's Cloud Firestore. The enrolment details are stored in the enrolments collection. Each document in the enrolments collection represents an enrolment record. The attributes are tabulated below in Table 7.1.

Table 7.1 Database Design (Enrolment Module)

Attribute Name	Attribute Description	Attribute Type	Attribute Example
EClassID	Autogenerated primary key from *classes* collection	String	jFf0CZkBQChWvSplWWXg
EStudentID	Autogenerated primary key from *students* collection	String	CgTNykYw1iSHyGj9vagO

In order for an enrolment record to be unique, both the EClassID and EStudentID fields have to be unique in the collection. In the following subsection, the functionalities of the Enrolment module are explained.

7.2 Implementation of UI and Functionalities

The Enrolment module is responsible for allowing users to enrol or unenroll existing students into existing classes. The Enrolment module can only be accessed if there are existing classes. Unlike the other modules of the application, the Enrolment module cannot be accessed directly from the Main Dashboard page. Instead, the user has to select a class from the class list in the class module and click the "View Students" button, as shown in Figure 7.1 below:

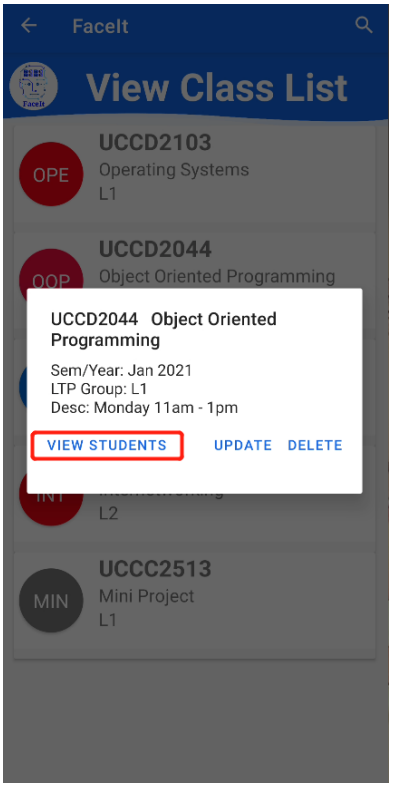


Figure 7.1 View Students Button in View Class Details Modal

Once users click on the button in the figure above, they are redirect to the View Enrolments page, as shown in Figure 7.2 below:

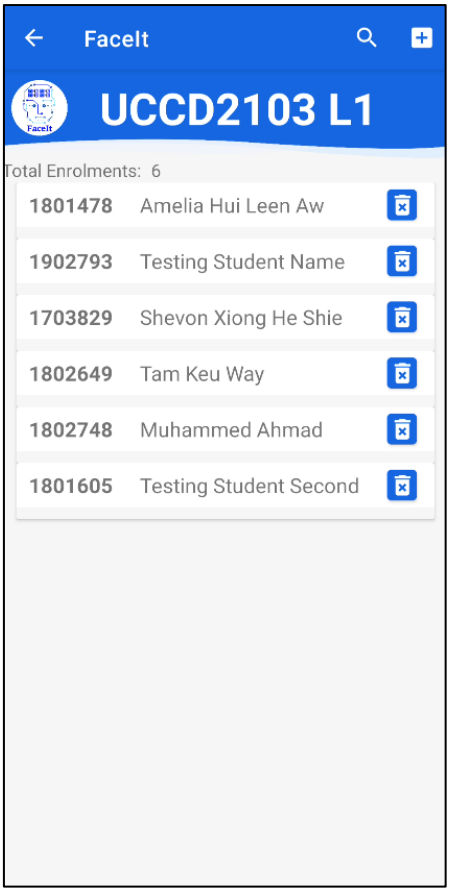


Figure 7.2 View Enrolments Page

In the View Enrolments page, the class code of the selected class is shown in the top banner. For the user's convenience, the total number of enrolled students is also displayed right under the class code. Then, the list of enrolled students is displayed, along with the student ID and student name. On the right-hand side of each student record card is a Delete Enrolment button with a trashcan symbol on it. If users click this button, the selected student will be unenrolled from the current class and the enrolment list will be updated in real-time to omit the unenrolled student. If users wish to enrol students into the current class, users have to click on the Plus button at the top right corner of the page, which will bring them to the interface as shown below in Figure 7.3.

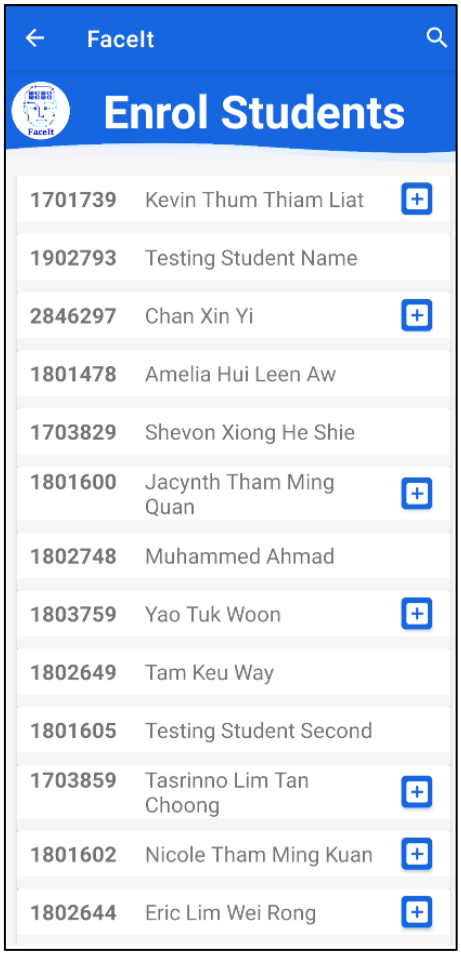


Figure 7.3 Add Enrolment Page

In the Add Enrolment page shown in the figure above, users can enrol students into the current class. This page shows the full list of students in the user's school, with a Plus button beside each student record shown if the student is not already enrolled in the current class. If the student is already enrolled, the Plus button will be hidden so that the user does not accidently enrol an already enrolled student into the current class. Moreover, both the View Enrolment and Add Enrolment pages have a small magnifying glass symbol at the top right corner of the page, which signifies the Search function. The Search function allows users to search for students by student ID or name when enrolling or unenrolling students. This makes it easier for users to look for certain students, rather than having to scroll through the entire student list, which may consist of hundreds of students.

Upon clicking the Add Enrolment or Delete Enrolment buttons, the corresponding Enrolment records will be added or deleted from the *enrolments* collections in the cloud database. Every document in the mentioned collection holds

the documentIDs of the selected class and student. An example is shown below in Figure 7.4.

Figure 7.4 Example Document in Enrolments Collection

The functionalities of the Enrolment module may be simple, but they are crucial to support the Attendance module discussed in the next chapter. In the next section, the functionalities of the Enrolment module are tested.

7.3 Module Testing

In this subsection, the test results of the Enrolment module are documented as tables. Each table represents a single test case, complete with test case ID, name, description, expected output as well as input and output (supported with screenshots) as evidence to the test cases.

Tables 7.2 to 7.8 below show the test results of the Enrolment module. The test cases below have IDs starting with “TE”, which stands for “Test Enrolment”.

Table 7.2 TE01: View Enrolments

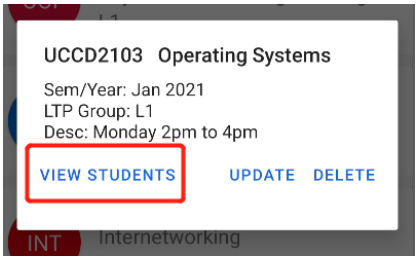
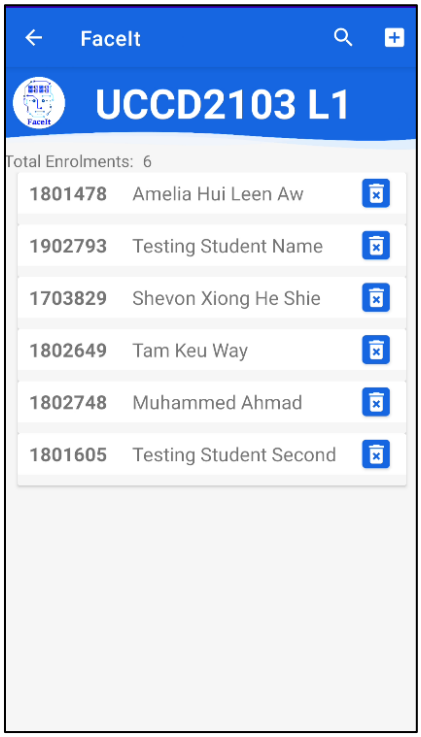
Test Case ID	TE01
Test Case Name	View Enrolments
Test Case Description	User clicks on the View Students button of a selected class's View Class Details modal
Expected Output	Display the list of enrolled students
Input	
Results	
Status (Pass/Fail)	Pass

Table 7.3 TE02: View Enrolments – No Students Enrolled

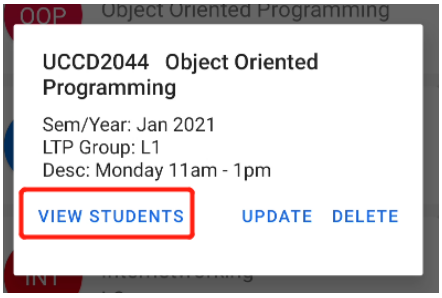
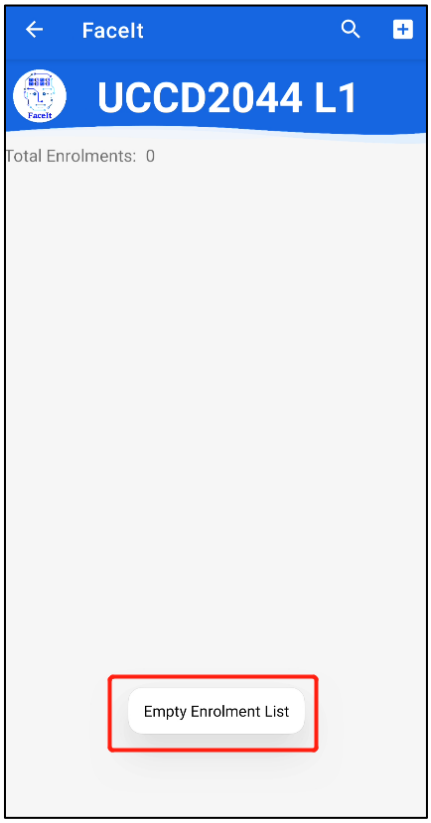
Test Case ID	TE02
Test Case Name	View Enrolments – No Students Enrolled
Test Case Description	User clicks on the View Students button of a selected class's View Class Details modal
Expected Output	Display an empty list with a message saying that no students have been enrolled into the current class
Input	This test input is a class with no enrolled students: 
Results	
Status (Pass/Fail)	Pass

Table 7.4 TE03: View Enrolments – Search Student

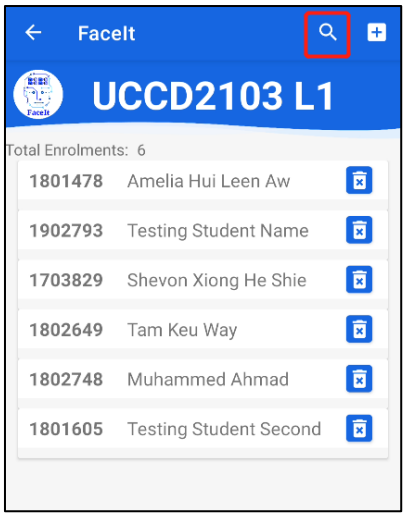
Test Case ID	TE03
Test Case Name	View Enrolments – Search Student
Test Case Description	User searches for a particular student by clicking on the magnifying glass icon on the top right corner of the screen in the View Enrolments page
Expected Output	Update enrolment list in real-time according to user's keystrokes
Input	
Results	
Status (Pass/Fail)	Pass

Table 7.5 TE04: View Enrolments – Unenroll Student

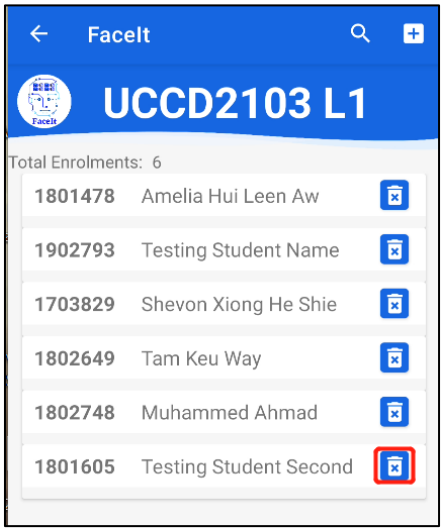
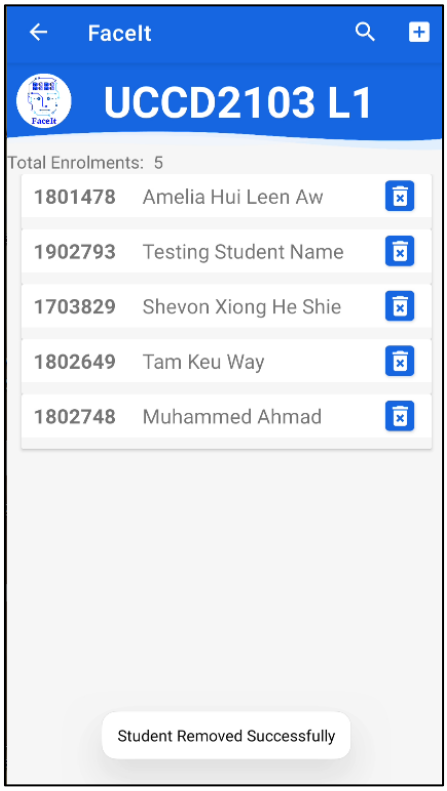
Test Case ID	TE04
Test Case Name	View Enrolments – Unenroll Student
Test Case Description	User clicks on the Delete Enrolment button (Trash can icon) at the right hand side of a particular student record in the list
Expected Output	Remove student from the enrolment list
Input	
Results	
Status (Pass/Fail)	Pass

Table 7.6 TE05: Add Enrolment

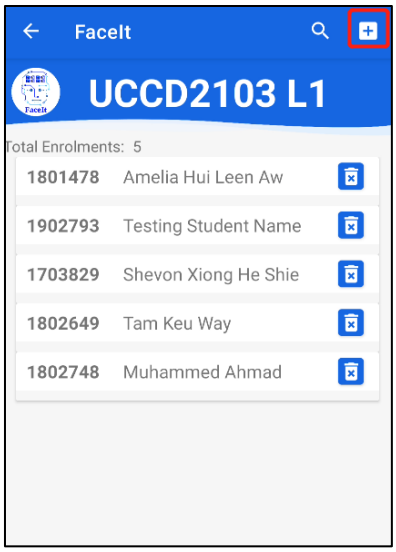
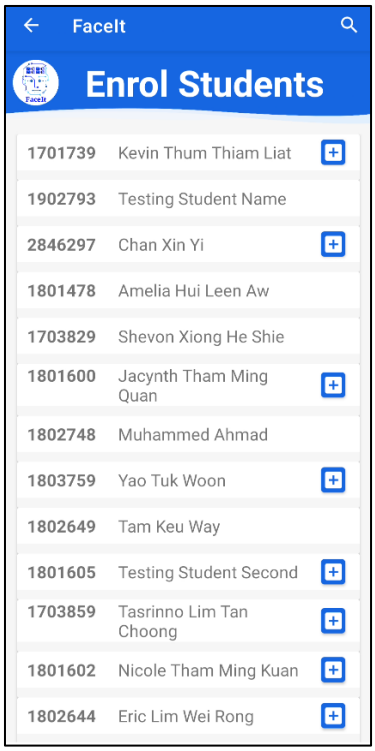
Test Case ID	TE05
Test Case Name	Add Enrolment
Test Case Description	User clicks on the Plus button at the top right corner of the screen in the View Enrolments page
Expected Output	Redirect user to the Add Enrolment page and display the student list of the user's school
Input	
Results	
Status (Pass/Fail)	Pass

Table 7.7 TE06: Add Enrolment – Search Student

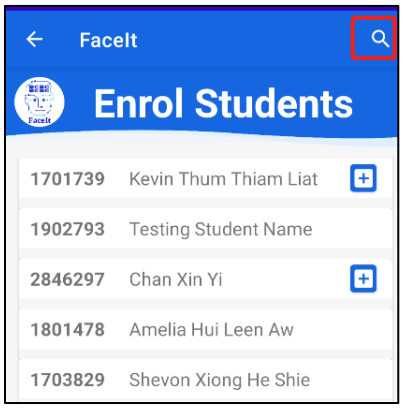
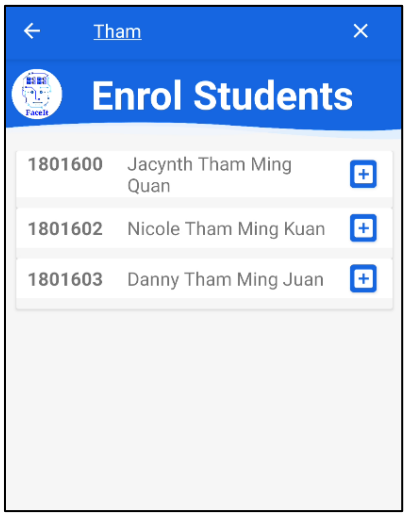
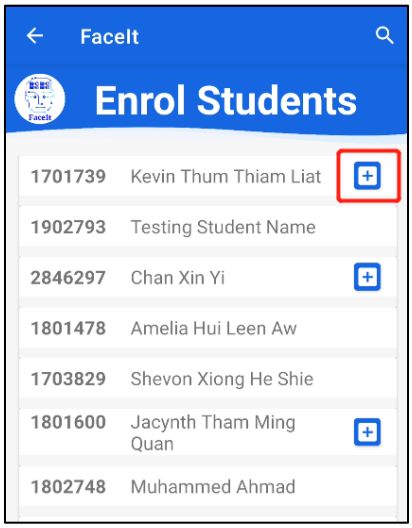
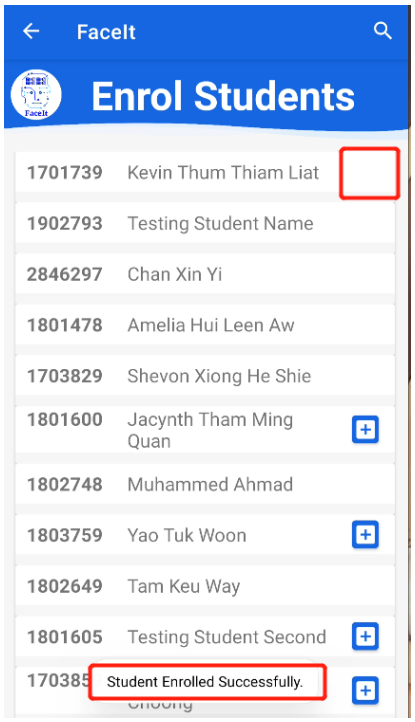
Test Case ID	TE06
Test Case Name	Add Enrolment – Search Student
Test Case Description	User searches for a particular student by clicking on the magnifying glass icon on the top right corner of the screen in the Add Enrolment page
Expected Output	Update add enrolment list in real-time according to user's keystrokes
Input	
Results	
Status (Pass/Fail)	Pass

Table 7.8 TE07: Add Enrolment – Enrol Student

Test Case ID	TE07
Test Case Name	Add Enrolment – Enrol Student
Test Case Description	User clicks on the Plus button on the right hand side of a particular student record in the add enrolment list
Expected Output	Enrol the student into the current class Hide the Plus button so users cannot enrol students that have already been enrolled into the current class
Input	
Results	
Status (Pass/Fail)	Pass

The test cases in this section evidence that the functionalities of the Enrolment module work as expected. In the Enrolment module, users can enrol or unenroll existing students into existing classes.

The next chapter describes the most important module in this application – the Attendance module, which is responsible for using face recognition technologies to automate the attending-taking process in classes.

Chapter 8

Attendance Module Implementation

In this chapter, the development, implementation, and testing processes of the Attendance module are discussed in detail. This chapter starts off with the database schema used for this module, then proceeds with the flow of the Attendance module, supported with screenshots of the application's interfaces. Last but not least, this chapter wraps up with the testing results for this module.

8.1 Database Design

The Attendance module stores data into Firebase's Cloud Firestore. The attendance record details are stored in the *attendance* collection. Each document in the *attendance* collection has an attribute named *dates*, which holds the concatenated string of attendance dates. Each of the dates reflects a subcollection in the document, which categorizes the attendance records by date. Each document in the *date* subcollection represents an attendance record for a single student. The attributes are tabulated below in Table 8.1.

Table 8.1 Database Design (Attendance Module)

Attribute Name	Attribute Description	Attribute Type	Attribute Example
ClassDocID	Autogenerated primary key of the class the attendance was taken in	String	RoZf03n1eGCRS86s56zq
StudentID	Student's ID	String	1801600
StudentName	Student's Name	String	Jacynth Tham
TimeStamp	The date and time of the attendance record	String	10-06-2021-14:40:21

Figure 8.1 below shows how the database structure looks like in the Firebase console.

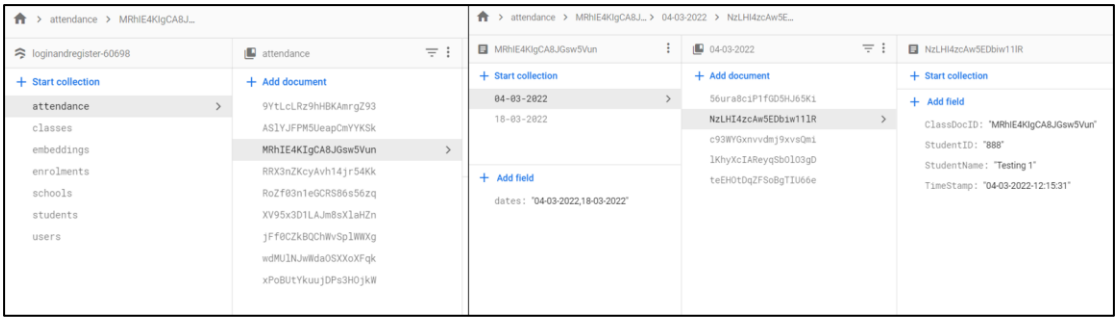


Figure 8.1 Attendance Module Database in Firebase Console

As seen in the figure above, the database schema for the Attendance module is somewhat different from the other modules because the Attendance database schema consists of nested collections and documents used to keep track of the date and class that the attendance records belong to. With the database schema above, it is assumed that one class only takes place once a day. Therefore, for the nested collection, only the date is stored, and not the entire timestamp.

From a logical standpoint, each main document represents a class. Thus, classes without any attendance records will not appear in the *attendance* collection. Each of the main documents has a field named "dates" and a subcollection. The documents in the subcollection correspond to the concatenated dates in the "dates" field. These documents store the attendance records for the students who were present on that date. In addition, the "dates" field is crucial for the developed application to loop through the subcollection to find the document with the selected date later on in the View Attendance activity.

In the next section, the implementation of the Attendance module's functionalities is discussed in detail.

8.2 Implementation of UI and Functionalities

The Attendance module consists of two activities – Take Attendance and View Attendance. From the Main Dashboard page, users can access these two activities through the corresponding buttons with the activities' names annotated on them, as shown below in Figure 8.2 below.

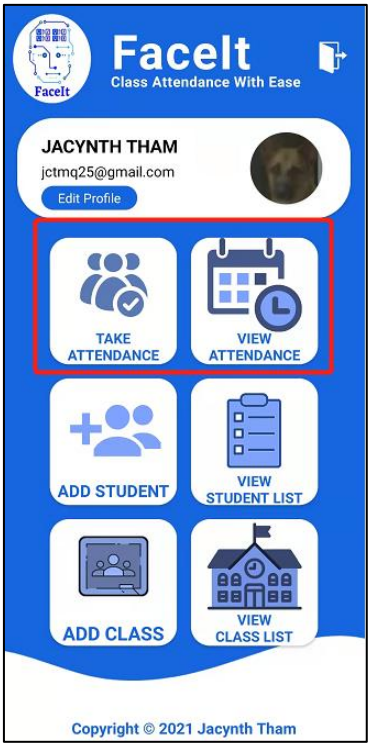


Figure 8.2 Take Attendance and View Attendance Buttons in Main Dashboard

Upon clicking the Take Attendance button, users are prompted to select the class that the attendance will be taken for. The Select Class activity is displayed below in Figure 8.3 below.

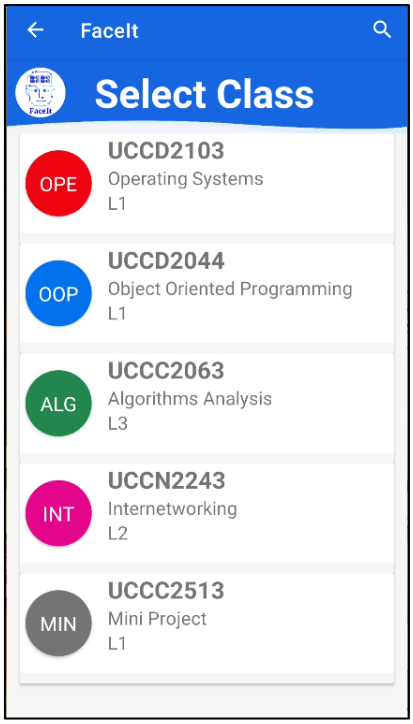


Figure 8.3 Take Attendance – Select Class Activity

Once the user has selected a class from the class list, the application triggers the user's mobile device's camera to capture the attendances of students. The UI of the Take Attendance view is illustrated below in Figure 8.4.

Figure 8.4 Take Attendance – Default Camera View with No Student in Frame

In this activity, users can take the attendance of the students using the MobileFaceNet face recognition model. The ideal setup is to have the mobile device supported by a tripod and have students line up and walk past the mobile device. The application extracts the camera frames one by one from the real-time video stream and processes them. First, the camera frame is fed into Firebase's Face Detection API to detect if any faces are present in frame. If so, then the face region is cropped out, mirrored, scaled and fed into the MobileFaceNet face recognition model for the face recognition part. At the very start of this activity's initiation, the entire *embeddings* collection is retrieved from Firebase and converted into a HashMap consisting of the students' information and face embedding arrays. For each detected face, the MobileFaceNet model transforms the cropped face region into a face embedding. Then, the face embedding is compared with the records stored in the HashMap retrieved earlier.

To ensure that the students were not wrongly recognized, the difference threshold of the face embedding comparison was set to 0.600f (f represents float). Note that a perfect match would give a difference of 0.00f while a face that does not match at all would yield a difference of more than 1.00f. While looping through the student embedding HashMap, the first face embedding with a difference of less than 0.600 would be used as the recognized student's label.

Once the student in frame has been recognized, the application checks to see if the recognized student is enrolled in the current class or not. If no, the application ignores the student. If so, the application checks again to see if the recognized student has any identical siblings. If not, then the application displays the student's ID, name and current timestamp in a success message on screen. Then, the application creates an attendance record in the corresponding path (following class ID and date). An example of attendance record is shown below in Figure 8.5.

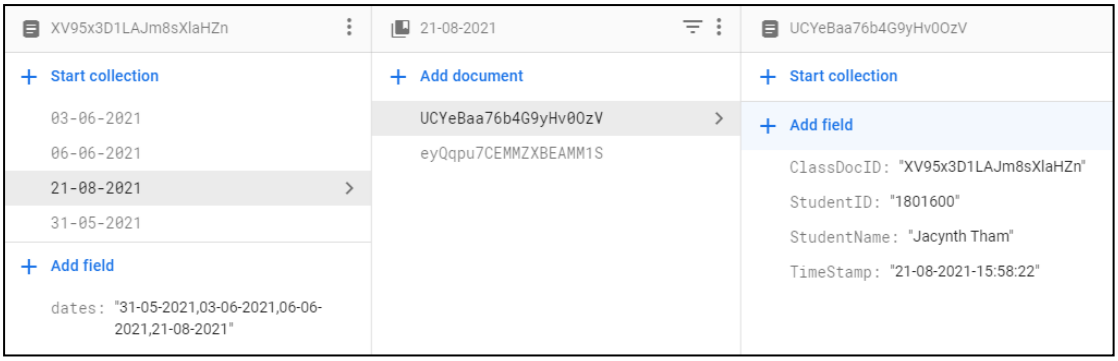


Figure 8.5 Example Attendance Record in Firebase Console

Figure 8.6 below shows the sample user feedback shown to the student whose attendance had just been taken.

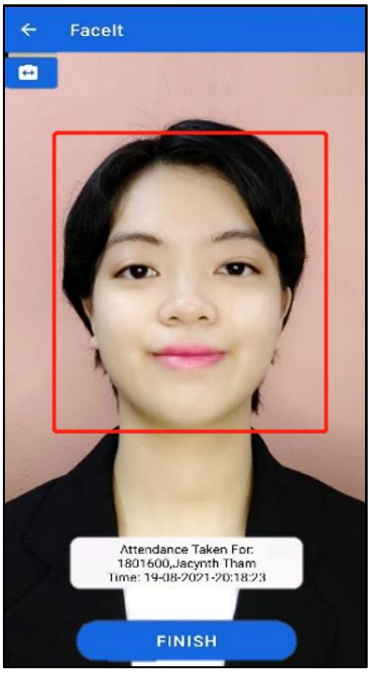


Figure 8.6 Take Attendance – Student Without Identical Siblings

On the other hand, if the identified student has identical siblings, then the application checks to see if all identical siblings are present in the same frame. This is done by comparing the number of faces detected against the number of identical siblings that a student has plus 1 (including the identified student themself). If the two numbers match, it is an indication that there is a possibility that all identical siblings are present. At this point, all the detected faces are cropped out, pre-processed and recognized to confirm the presence of all identical siblings. If all identical siblings have

been confirmed to be present, the application displays the success message as shown in Figure 8.7 below.

Figure 8.7 Take Attendance – Student with All Identical Siblings Present

In the example given above, the two images on the student ID cards were used during the Add Student process, which is why the application recognized them as students. In reality, students would have to use their own faces during the Add Student process, thus this situation where students are able to use their student ID cards to take attendance would not occur.

However, there are scenarios when not all identical siblings are present during a particular class. In cases like these, the application waits for 15 seconds. By the end of the time limit, if the application detects that not all identical siblings are present, the application displays a popup modal as shown in Figure 8.8 below.

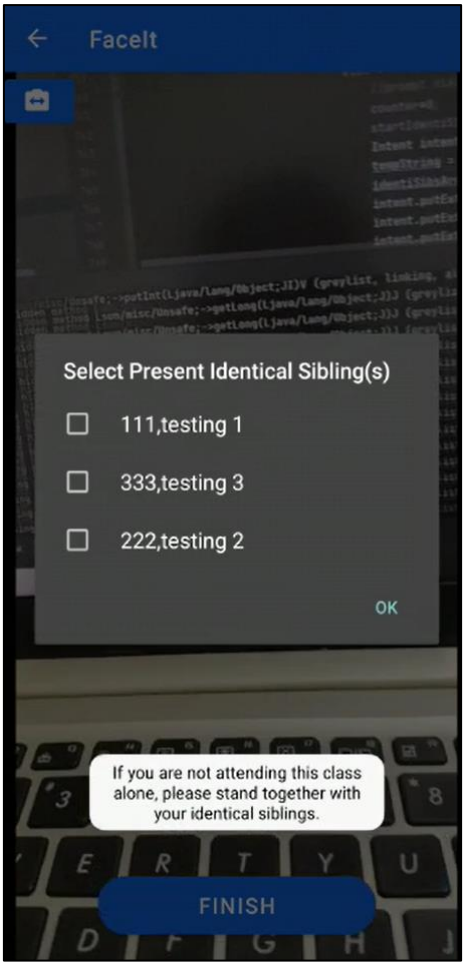


Figure 8.8 Take Attendance – Student with Absent Identical Siblings

As evidenced by the figure above, when there are one or more absent identical siblings, then the application shows a popup modal with checkboxes beside each identical sibling's details (student ID and name). At this point, the student has to tick the present identical sibling and press the OK button. When the OK button is pressed, the application takes the attendance for all the identical siblings marked as present, and takes note of the identical siblings marked as absent so that no duplicated attendance records are taken.

This process is repeated for all students who walk past the user's mobile device to have their attendance taken for a particular class. During this process, if a student whose attendance had already been taken comes into frame, the application will not take the attendance again. Instead, the application will simply ignore the student since the attendance record had already been taken previously. When the attendance of all students has been taken, users can click on the Finish button to end the Take Attendance

activity. At this point, the total number of attendance records taken is displayed on the screen, then the user is redirected to the main dashboard, as shown below in Figure 8.9.

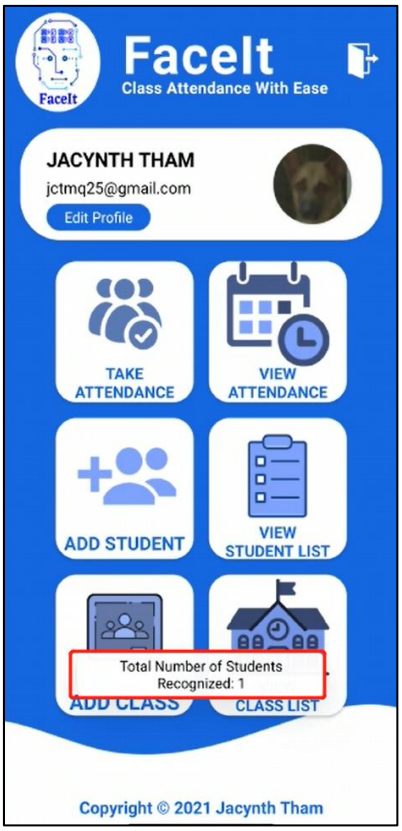


Figure 8.9 Take Attendance – Display Total Attendees

From the main dashboard, users can click on the View Attendance button to all the previously taken attendance records. Next, users have to select a class from the Select Class page, shown in Figure 8.10 below.

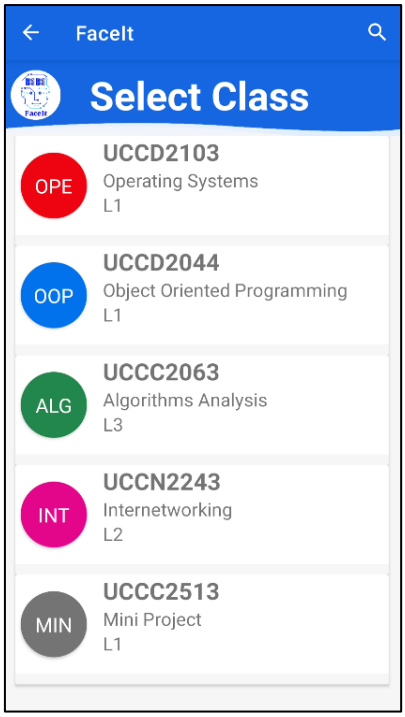


Figure 8.10 View Attendance – Select Class Activity

After that, users have to select the date of the attendance records to view. Figure 8.11 below illustrates the list of dates shown to the user for a particular class.

Figure 8.11 View Attendance – Select Date Activity

After users have selected a date, users are presented with a two-tabbed layout to show the list of present and absent students. Figure 8.12 and 8.13 below shows the UI for the View Attendance activity's present and absent tab respectively.

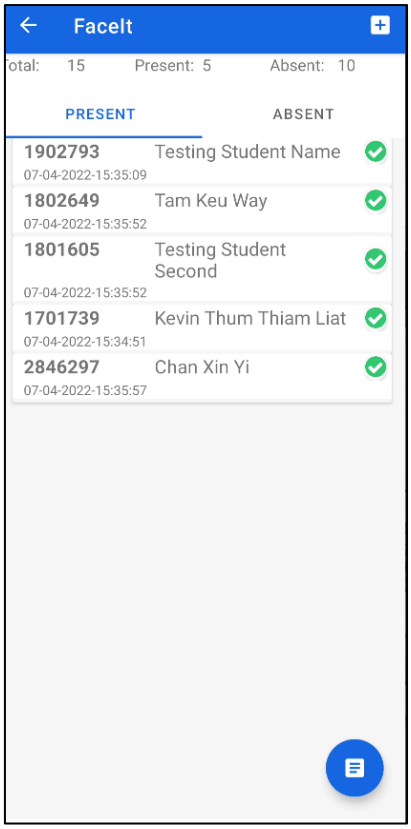


Figure 8.12 View Attendance – Present Tab

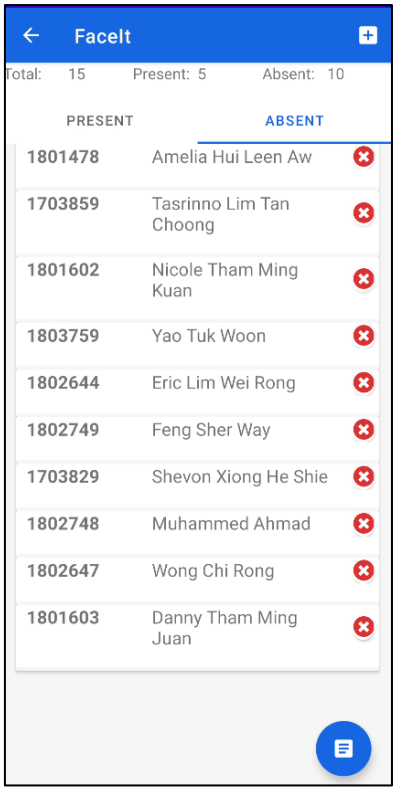


Figure 8.13 View Attendance – Absent Tab

In the two figures above, it is seen that the top rows of both the Present tab and the Absent tab show the total number of students, the number of present students and the number of absent students. All student records in the present tab have a green tick on the right-hand side of the record card and all student records in the absent tab have a red cross on the right-hand side of the record card. For all present students, the student's ID, full name and timestamp of attendance are shown in each student record. As for all absent students, only the student ID and name of each student are shown. However, it is noticed that the second student in Figure 8.12 does not have a timestamp shown, but rather, the phrase "Added By Lecturer". This is actually the result of the next functionality to be explained – the capability of the application to allow users to add attendance records manually. In certain scenarios, students might show up late or they might miss the attendance taking session. In another scenario, it is also possible for students to come in for replacement classes. However, as the application ignores recognized students who are not enrolled in the current class, an alternative way needs to be used to take the attendances of these students. In cases like these, these students can just approach the class lecturer and ask them for permission to have their attendance added manually. Users can perform this action by clicking on the Plus button at the top

right corner of the interface as shown in Figure 8.13. This button shows the popup modal as illustrated below in Figure 8.14.

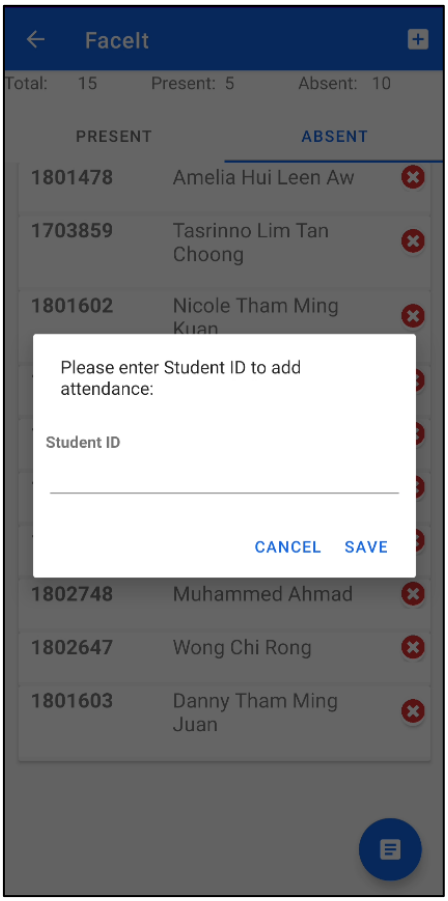


Figure 8.14 View Attendance – Add Attendance Record Manually

In the popup modal shown in the figure above, users have to key in the student ID of the student whose attendance is to be added manually into the application. Once the user clicks on the Save button, the application checks if the entered student ID exists in the *students* collection. If so, the application adds the student's attendance record into the *attendances* collection with the TimeStamp field as "Added By Lecturer" to signify that this particular attendance record was added manually by the user. If the student added does not belong to a particular class, the student's attendance record will be shown under the Present tab, but with a yellow question mark icon instead of a green tick. An example is shown below in Figure 8.15.

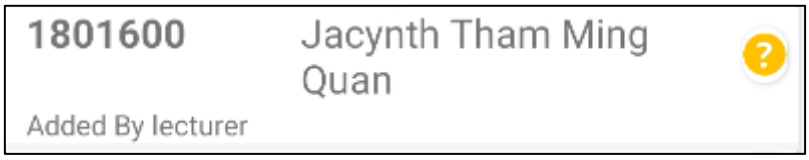


Figure 8.15 View Attendance – Unenrolled Student in Present Tab

The last functionality in the Attendance module is the ability to export the attendance records for a particular class and date into an Excel spreadsheet. This function can be accessed by clicking on the floating action button in the bottom right corner of the View Attendance interface, as circled out in Figure 8.16 below.

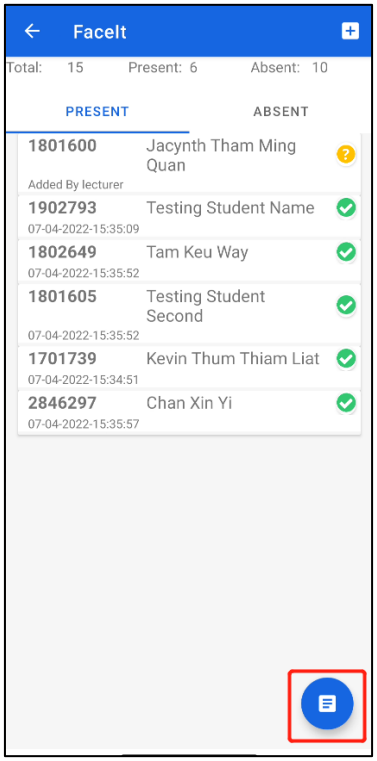


Figure 8.16 View Attendance – Generate Report Button

When users click on the button, a success message is shown to them, as shown below in Figure 8.17.

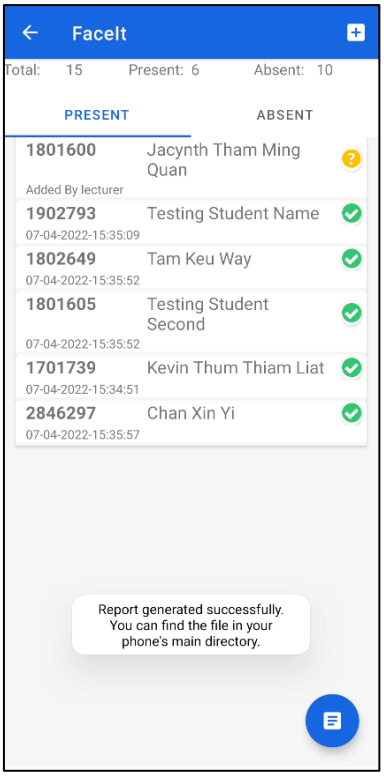


Figure 8.17 View Attendance – Generate Report Message

By default, the generated Excel file can be found in the user's device's main directory. The Excel file is named by concatenating the class name and date for the attendance records used to generate the report. An example of the Excel spreadsheet is shown below in Figure 8.18.

	A	B	C
1	Class Code:	UCCD2103	
2	Date:	07-04-2022	
3			
4	Student ID	Student Name	Present(1) / Absent(0)
5	1801478	Amelia Hui Leen Aw	0
6	1703859	Tasrinno Lim Tan Choong	0
7	1801602	Nicole Tham Ming Kuan	0
8	1803759	Yao Tuk Woon	0
9	1802644	Eric Lim Wei Rong	0
10	1802749	Feng Sher Way	0
11	1902793	Testing Student Name	1
12	1703829	Shevon Xiong He Shie	0
13	1801605	Testing Student Second	1
14	1802649	Tam Keu Way	1
15	2846297	Chan Xin Yi	1
16	1802748	Muhammed Ahmad	0
17	1802647	Wong Chi Rong	0
18	1801603	Danny Tham Ming Juan	0
19	1701739	Kevin Thum Thiam Liat	1

Figure 8.18 Attendance Report Excel Spreadsheet

In the Excel spreadsheet generated, the class code and date are shown at the top of the spreadsheet. Then, three different columns are populated for each attendance record – student ID, student name and the present or absent status. A “1” means the student was present during the class, and a “0” means the student was absent. This generate report function is crucial as some schooling institutions require teachers to submit Excel spreadsheets as proof of their students’ attendance in class.

In the following section, the functionalities of the Attendance module are put to the test and discussed in terms of the results obtained.

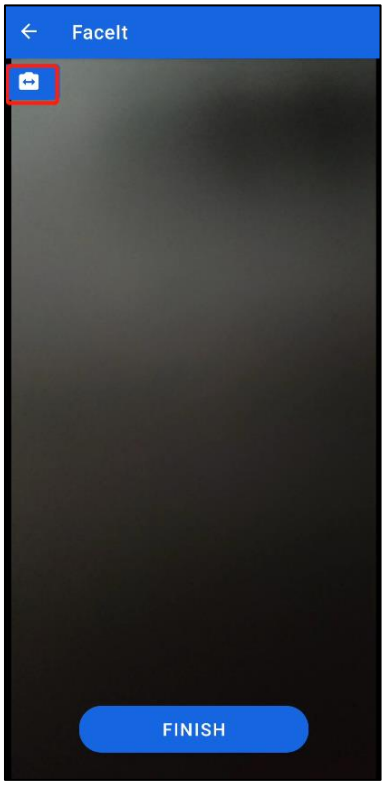
8.3 Module Testing

In this subsection, the test results of the Attendance module are documented as tables. Each table represents a single test case, complete with test case ID, name, description, expected output as well as input and output (supported with screenshots) as evidence to the test cases.

8.3.1 Take Attendance

Table 8.2 to 8.12 below show the test cases of the Take Attendance functionalities. The test case IDs in this subsection all start with TTA, which stands for "Test Take Attendance".

Table 8.2 TTA01: Switch Camera Button

Test Case ID	TTA01
Test Case Name	Switch Camera Button
Test Case Description	User clicks on the Switch Camera button
Expected Output	Switch the camera used from front to back or from back to front
Input	

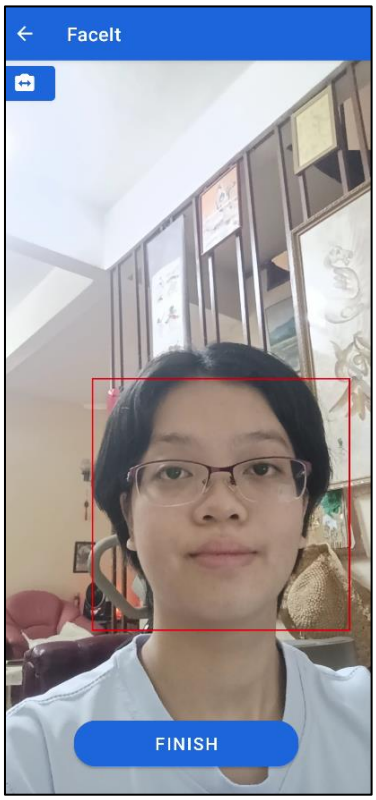
Results	
Status (Pass/Fail)	Pass

Table 8.3 TTA02: Unknown Student in Frame

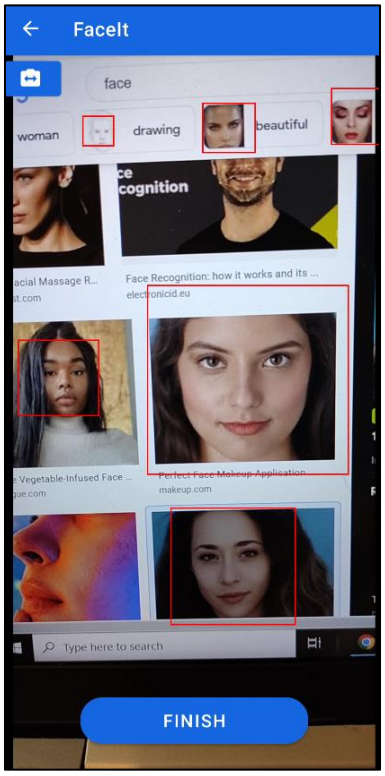
Test Case ID	TTA02
Test Case Name	Unknown Student in Frame
Test Case Description	Unregistered student comes into frame
Expected Output	Application ignores unregistered student
Input	
Results	Detected faces highlighted with bounding box, but no attendance taken because students are not registered
Status (Pass/Fail)	Pass

Table 8.4 TTA03: Unenrolled Student in Frame

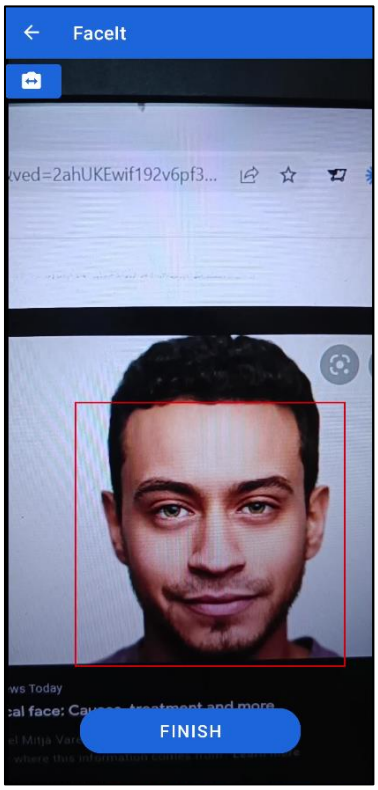
Test Case ID	TTA03
Test Case Name	Unenrolled Student in Frame
Test Case Description	Unenrolled student comes into frame
Expected Output	Application ignores unenrolled student
Input	
Results	Detected face highlighted with bounding box, but no attendance taken because student is not enrolled
Status (Pass/Fail)	Pass

Table 8.5 TTA04: Enrolled Student Without Identical Siblings in Frame (First Appearance)

Test Case ID	TTA04
Test Case Name	Enrolled Student Without Identical Siblings in Frame (First Appearance)
Test Case Description	Enrolled student with no identical siblings come into frame for the first time in the current attendance taking session
Expected Output	Application takes attendance and displays message to student
Input	

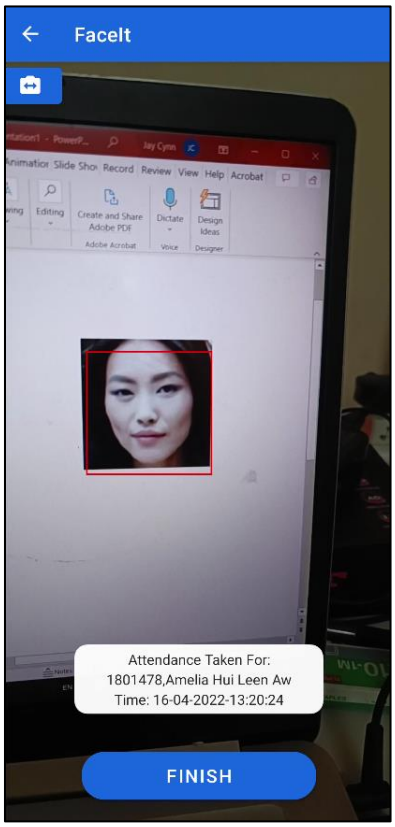
Results	
Status (Pass/Fail)	Pass

Table 8.6 TTA05: Enrolled Student Without Identical Siblings in Frame (Second Appearance and Beyond)

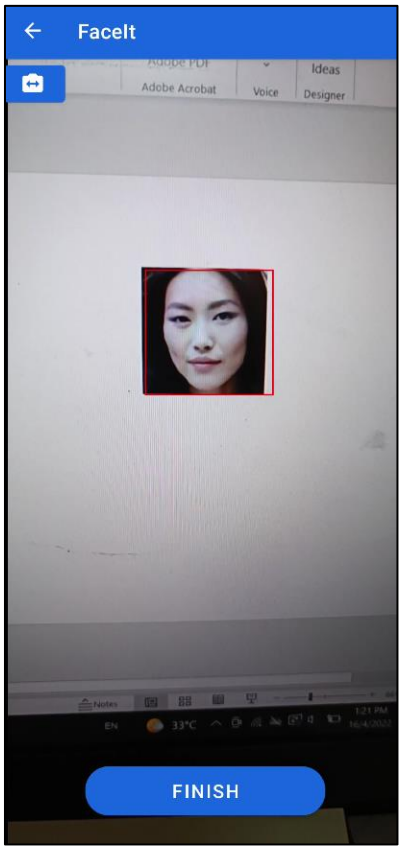
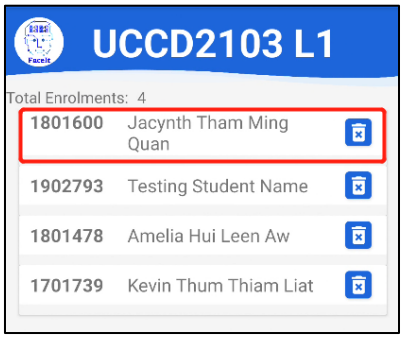
Test Case ID	TTA05
Test Case Name	Enrolled Student Without Identical Siblings in Frame (Second Appearance and Beyond)
Test Case Description	Enrolled student with no identical siblings come into frame again in the current attendance taking session
Expected Output	Application ignores student and does not take duplicated attendance records
Input	
Results	Detected face highlighted with bounding box, but no duplicated attendance is taken
Status (Pass/Fail)	Pass

Table 8.7 TTA06: Enrolled Student with Identical Siblings (No Identical Siblings Enrolled)

Test Case ID	TTA06
Test Case Name	Enrolled Student with Identical Siblings (No Identical Siblings Enrolled)
Test Case Description	In this scenario, the student in frame is enrolled and has 2 other identical siblings but for this particular class, none of the other identical siblings are enrolled.
Expected Output	Application takes attendance of student in frame without taking into consideration the presence of any of the identical siblings
Input	Screenshot below shows the enrolled student with two identical siblings, but none of which are enrolled in the current class. 

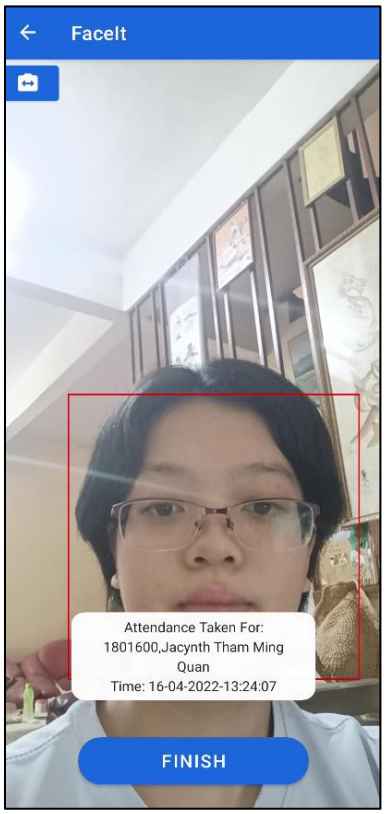
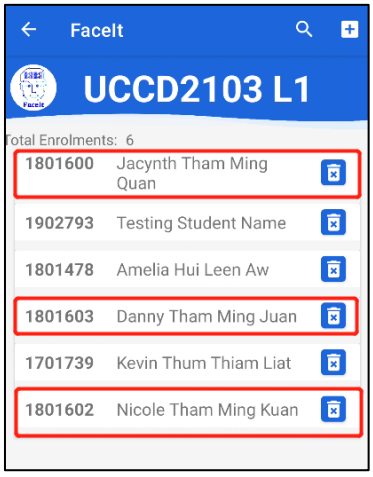
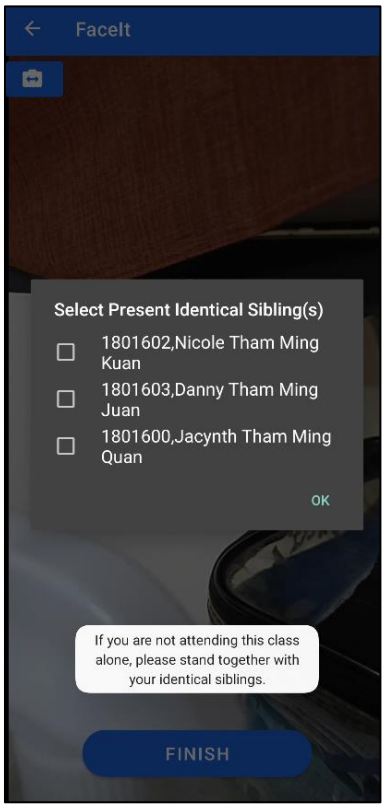
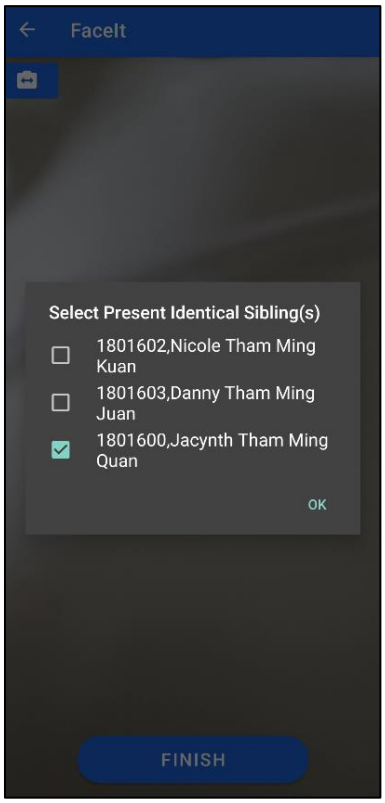
Results	
Status (Pass/Fail)	Pass

Table 8.8 TTA07: Enrolled Student with Identical Siblings (Not All Present)

Test Case ID	TTA07
Test Case Name	Enrolled Student with Identical Siblings (Not All Present)
Test Case Description	In this scenario, the student in frame is enrolled and has 2 other identical siblings enrolled in the same class but only 1 other identical sibling is present.
Expected Output	Application prompts student to select the present identical siblings through the use of a checkbox modal
Input	Screenshot below shows the three identical siblings enrolled in the same class. 

Results	 FaceIt Select Present Identical Sibling(s) 1801602,Nicole Tham Ming Kuan 1801603,Danny Tham Ming Juan 1801600,Jacynth Tham Ming Quan OK If you are not attending this class alone, please stand together with your identical siblings. FINISH  FaceIt Select Present Identical Sibling(s) 1801602,Nicole Tham Ming Kuan 1801603,Danny Tham Ming Juan 1801600,Jacynth Tham Ming Quan OK FINISH

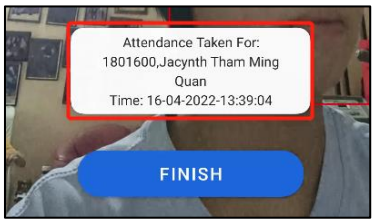
	
Status (Pass/Fail)	Pass

Table 8.9 TTA08: Enrolled Student with Identical Siblings (All Present)

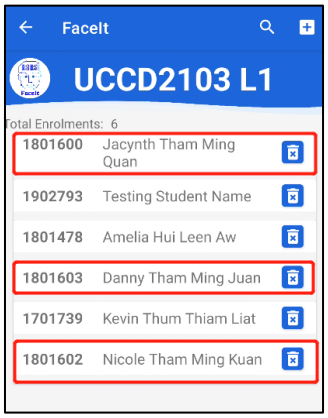
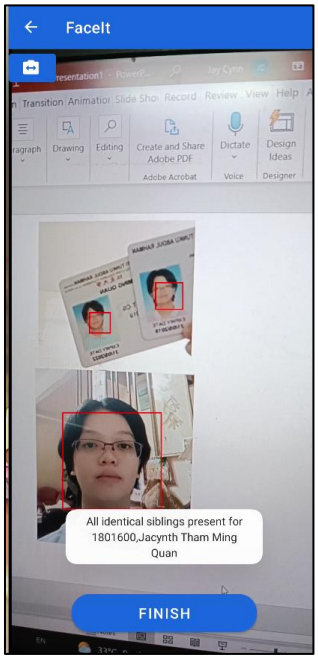
Test Case ID	TTA08
Test Case Name	Enrolled Student with Identical Siblings (All Present)
Test Case Description	In this scenario, the student in frame is enrolled and has 2 other identical siblings enrolled in the same class and all identical siblings are present
Expected Output	Application recognizes all identical siblings in the same frame and take attendance for all three students at once.
Input	
Results	To simulate the scenario where all identical siblings are present, the pictures of registered & enrolled “triplets” are used in the test case below. 
Status (Pass/Fail)	Pass

Table 8.10 TTA09: Enrolled Student with Identical Siblings (Second Appearance and Beyond)

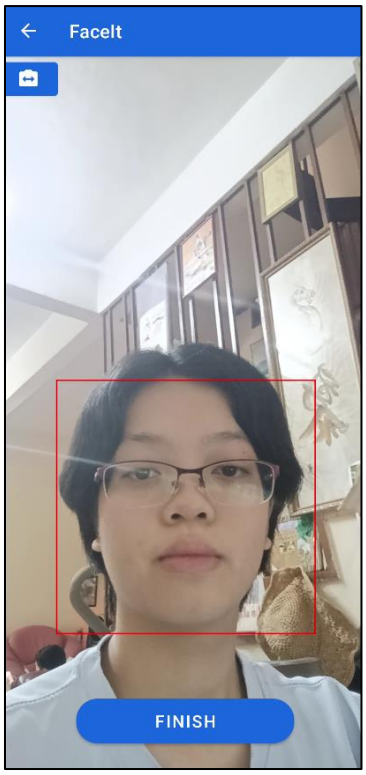
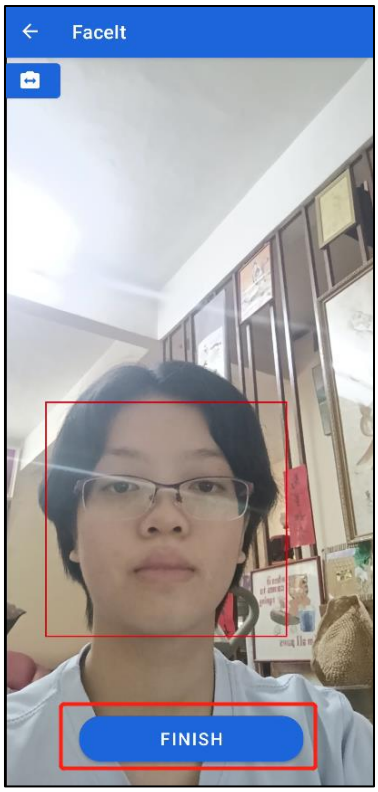
Test Case ID	TTA09
Test Case Name	Enrolled Student with Identical Siblings (Second Appearance and Beyond)
Test Case Description	This test case is a continuation from TTA07, where only 1 out of 3 identical siblings are present. In this scenario, the present identical sibling comes into frame again (attendance already taken previously)
Expected Output	Application ignores student and does not take duplicated attendance
Input	
Results	Detected face highlighted with bounding box, but no duplicated attendance is taken
Status (Pass/Fail)	Pass

Table 8.11 TTA10: End Take Attendance Activity (Finish Button)

Test Case ID	TTA10
Test Case Name	End Take Attendance Activity (Finish Button)
Test Case Description	User clicks on the Finish button
Expected Output	Redirect user to the Main Dashboard and display the total number of attendees.
Input	

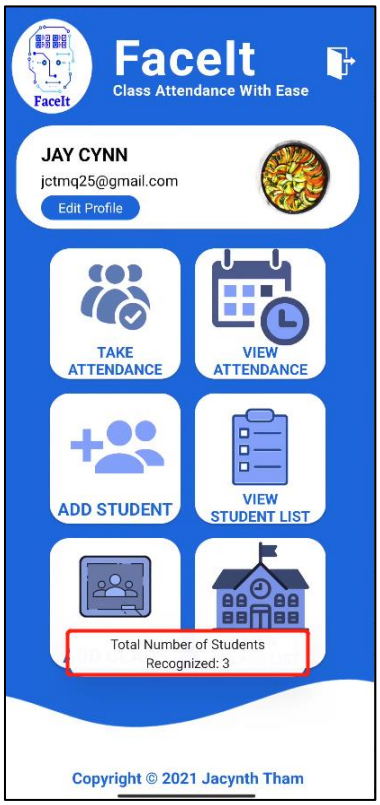
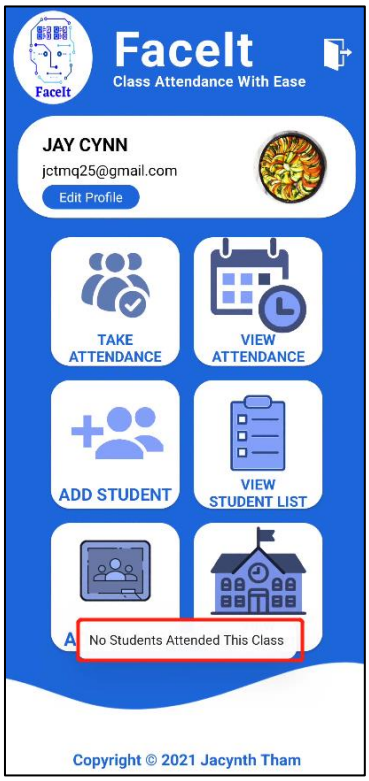
Results	
Status (Pass/Fail)	Pass

Table 8.12 TTA11: End Take Attendance Activity (Back Button)

Test Case ID	TTA11
Test Case Name	End Take Attendance Activity (Back Button)
Test Case Description	User clicks on the Back button on the top of the screen or on their mobile device
Expected Output	Redirect user to the Main Dashboard and display the total number of attendees. Previously taken attendance records are not cancelled
Input	

Results	
Status (Pass/Fail)	Pass

8.3.2 View Attendance

Table 8.13 to 8.21 below show the test cases of the View Attendance functionalities. The test case IDs in this subsection all start with TVA, which stands for "Test View Attendance".

Table 8.13 TVA01: Display Date Selection List (Dates Not Empty)

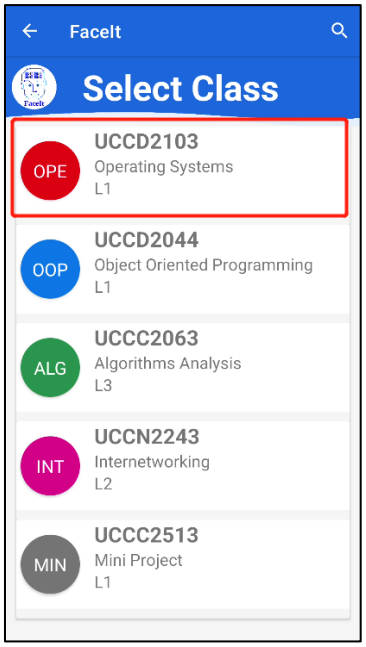
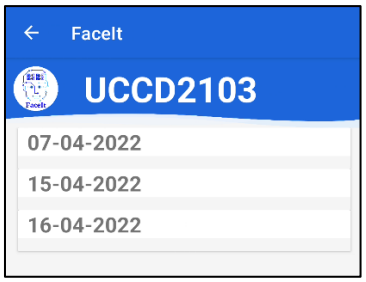
Test Case ID	TVA01
Test Case Name	Display Date Selection List (Dates Not Empty)
Test Case Description	User clicks on a particular class in the Select Class activity
Expected Output	Application displays a list of dates for the user to choose to display the corresponding attendance records.
Input	
Results	
Status (Pass/Fail)	Pass

Table 8.14 TVA02: Display Date Selection List (Dates Empty)

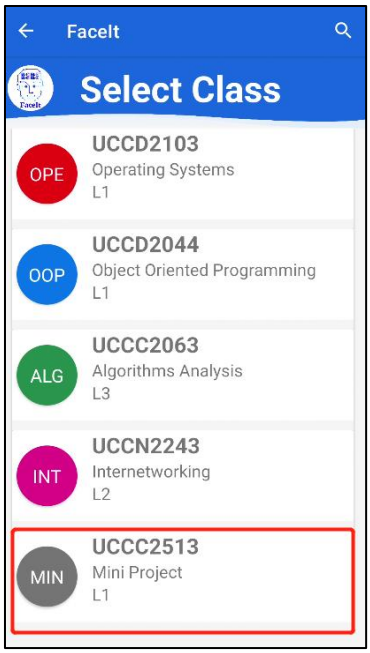
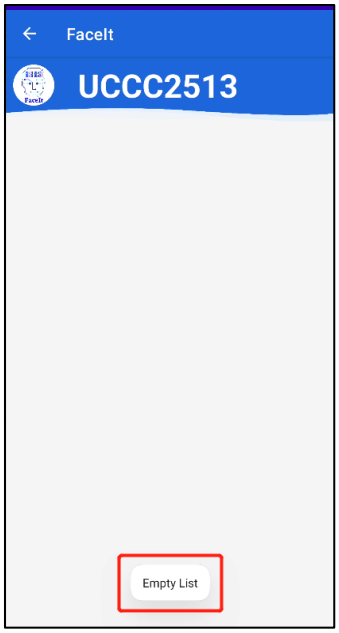
Test Case ID	TVA02
Test Case Name	Display Date Selection List (Dates Empty)
Test Case Description	User clicks on a particular class in the Select Class activity but the chosen class has not attendance records and thus, has an empty date list
Expected Output	Display error message and return to Select Class activity
Input	
Results	
Status (Pass/Fail)	Pass

Table 8.15 TVA03: Display Attendance List in Two Tabs (Present and Absent)

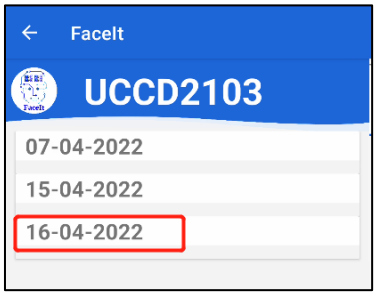
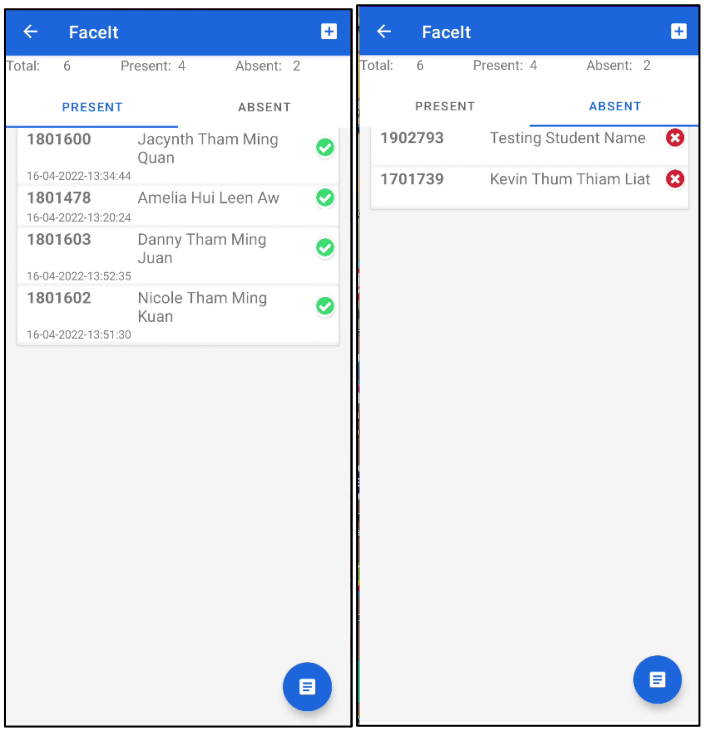
Test Case ID	TVA03
Test Case Name	Display Attendance List in Two Tabs (Present and Absent)
Test Case Description	User clicks on a particular date in the Select Date activity
Expected Output	Display attendance list separated into Present and Absent tab
Input	
Results	
Status (Pass/Fail)	Pass

Table 8.16 TVA04: Display Present Enrolled Students with a Green Tick Icon

Test Case ID	TVA04
Test Case Name	Display Present Enrolled Students with a Green Tick Icon
Test Case Description	-
Expected Output	Present students that have previously been enrolled in the particular class are displayed with a green tick icon on the right-hand side of each record in the list
Input	Enrolled student with attendance previously taken shown in Present Tab.
Results	
Status (Pass/Fail)	Pass

Table 8.17 TVA05: Display Present Unenrolled Students with a Yellow "?" Icon

Test Case ID	TVA05
Test Case Name	Display Present Unenrolled Students with a Yellow "?" Icon
Test Case Description	-
Expected Output	Present students that have not been previously enrolled in the particular class are displayed with a yellow question mark icon on the right hand side of each record in the list
Input	Unenrolled student with attendance previously taken shown in Present Tab. (Cases like this might occur when a student has taken his/her attendance previously but has been unenrolled from the class by the lecturer.
Results	
Status (Pass/Fail)	Pass

Table 8.18 TVA06: Display Absent Enrolled Students with a Red Cross Icon

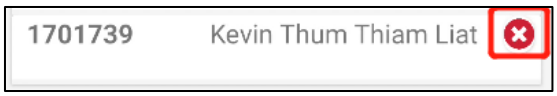
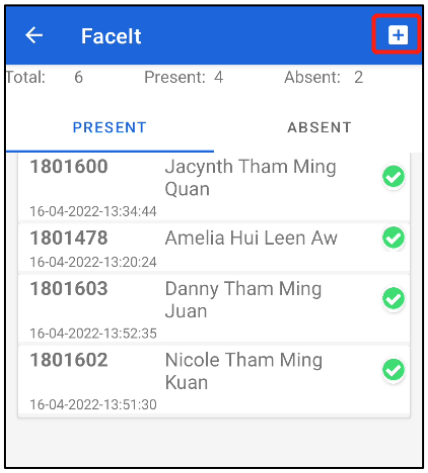
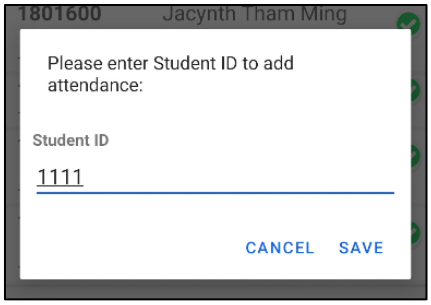
Test Case ID	TVA06
Test Case Name	Display Absent Enrolled Students with a Red Cross Icon
Test Case Description	-
Expected Output	Absent students are displayed with a red cross icon on the right hand side of each record in the list
Input	Enrolled student that was absent during class
Results	
Status (Pass/Fail)	Pass

Table 8.19 TVA07: Add Attendance Manually (Invalid Student ID)

Test Case ID	TVA07
Test Case Name	Add Attendance Manually (Invalid Student ID)
Test Case Description	Users click on the Plus button at the top right corner of the page and enters a non-existent student ID
Expected Output	Display error message
Input	 

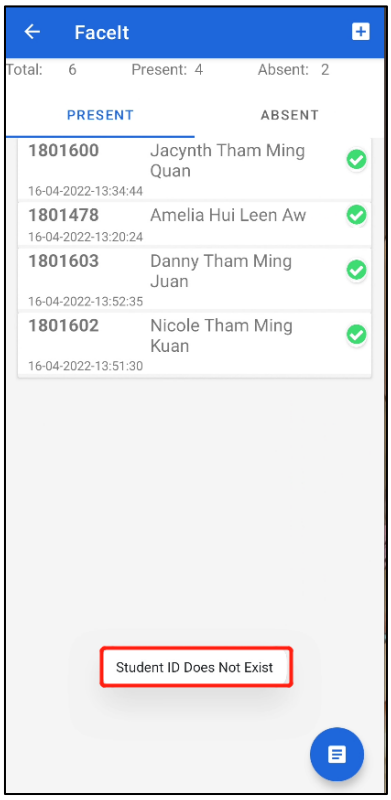
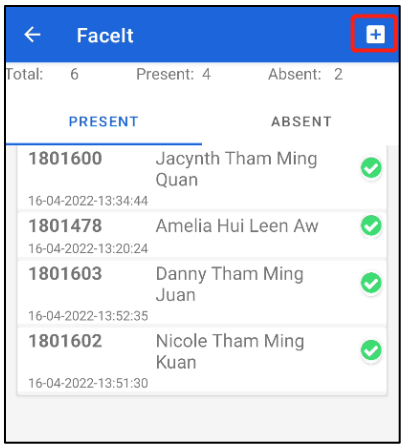
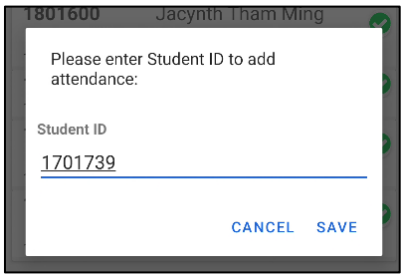
Results	
Status (Pass/Fail)	Pass

Table 8.20 TVA08: Add Attendance Manually (Valid Student ID)

Test Case ID	TVA08
Test Case Name	Add Attendance Manually (Valid Student ID)
Test Case Description	Users click on the Plus button at the top right corner of the page and enters a valid student ID
Expected Output	Add student to Present tab with the timestamp as "Added By Lecturer"
Input	 

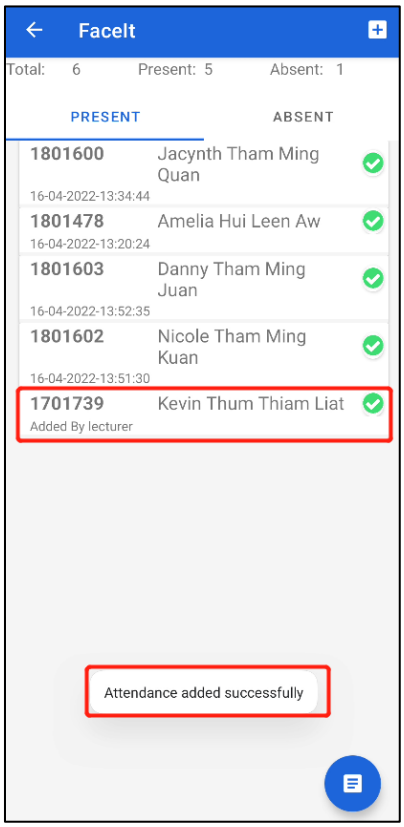
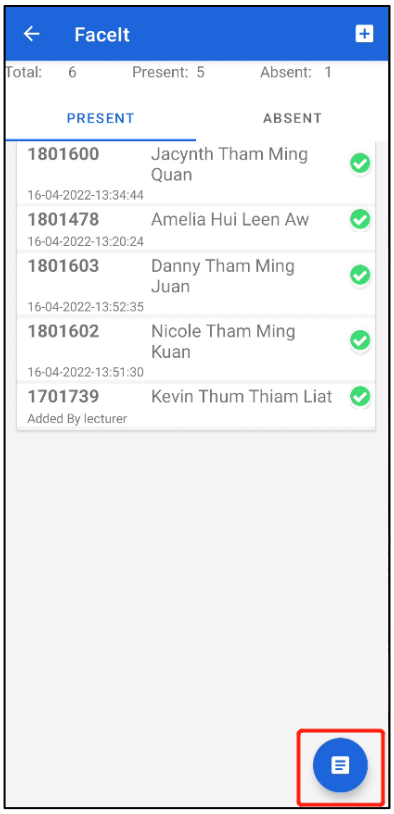
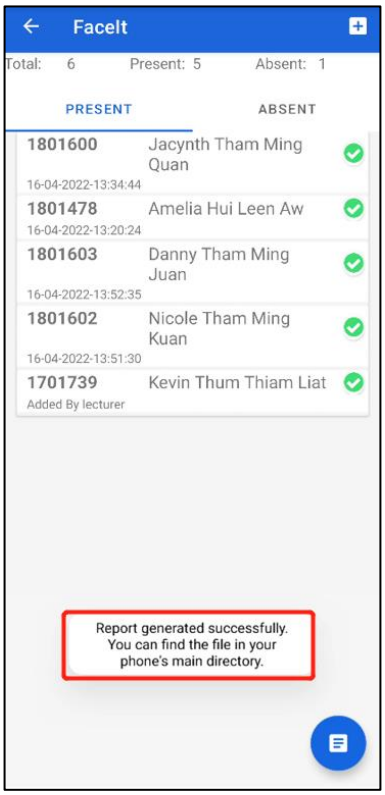
Results	
Status (Pass/Fail)	Pass

Table 8.21 TVA09: Generate Report Button

Test Case ID	TVA09
Test Case Name	Generate Report Button
Test Case Description	Users click on the Generate Report button at the bottom right corner of the page
Expected Output	Application exports attendance records into an Excel spreadsheet and displays a success message to the user.
Input	

<table>
<tr><td>Results</td><td>


<table>
<tr><th></th><th>A</th><th>B</th><th>C</th></tr>
<tr><td>1</td><td>Class Code:</td><td>UCCD2103</td><td></td></tr>
<tr><td>2</td><td>Date:</td><td>16/4/2022</td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>Student ID</td><td>Student Name</td><td>Present(1) / Absent(0)</td></tr>
<tr><td>5</td><td>1801600</td><td>Jacynth Tham Ming Quar</td><td>1</td></tr>
<tr><td>6</td><td>1801478</td><td>Amelia Hui Leen Aw</td><td>1</td></tr>
<tr><td>7</td><td>1801603</td><td>Danny Tham Ming Juan</td><td>1</td></tr>
<tr><td>8</td><td>1801602</td><td>Nicole Tham Ming Kuan</td><td>1</td></tr>
<tr><td>9</td><td>1902793</td><td>Testing Student Name</td><td>0</td></tr>
<tr><td>10</td><td>1701739</td><td>Kevin Thum Thiam Liat</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Status (Pass/Fail)</td><td>Pass</td></tr>
</table>

The test cases in this section evidence that the functionalities of the Attendance module work as expected. In the Attendance module, users can take attendance of students using face recognition technologies and manage the attendance records accordingly.

The next chapter wraps up the development process of this application with the system evaluation results and discussion.

Chapter 9

System Evaluation and Discussion

In this chapter, the system evaluation and discussion of the entire development process of FaceIt are reported. The chapter starts off with the testing and system performance metrics, then the project challenges and objectives evaluation. Lastly, this chapter closes up with a concluding remark regarding the entire project.

9.1 System Testing and Performance Metrics

Having performed and reported the test cases in Chapter 4 to 8 based on individual modules, it is evidenced that the functionalities of each module work as intended. On top of the correctness of the functionalities, extra validations were added to all input fields to ensure that the user input is of the correct format and that the value is valid. Once the individual modules have been tested thoroughly, the entire application was put to the test in a mock class environment with 25 students, representing the average size of a tutorial class in a university. However, due to the Covid-19 pandemic, a combination of close friends, family members, relatives and online face images were used to resemble students in the real-world.

Initially, all students had to register themselves by filling up their student particulars in the application and then adding their face image to the application. The application faced no issues in detecting the students' faces and converting them to face embeddings to be added to the cloud database. After that, a mock class was selected for the attendance taking process. All students were enrolled into the class to test whether the application is able to filter out unenrolled students.

During the first round of attendance taking, the application was able to identify majority of the students in less than 3 seconds. The average was 2 seconds per student and the maximum was 5 seconds (due to the presence of an identical sibling trio in the test subjects). However, the application had issues differentiating between two of the mock students, who were mother and daughter in reality that look alike due to similar facial features and hairstyles. Therefore, in order to resolve this issue, an identical sibling link was created in the application between the mother and daughter that the

application was having a hard time differentiating between. Then, a second round of attending taking was performed with the same set of 25 students.

During the second round of attending taking, the application was able to identify each student correctly, and was able to prompt the students with identical siblings to select only the present identical siblings. The total time taken for the second round of attendance taking was 2 minutes and 43 seconds, because more time were taken to take the attendance of students with identical siblings.

The testing scenario performed proved that the application is able to take the attendance of recognized students efficiently. Moreover, the identical siblings link in the application can not only be used for real-life identical siblings, but also for students who the application might have a hard time recognizing (e.g. students that look alike but are not blood-related). This is, however, entirely up to the user of the device whether to set up identical sibling linkages between students who are not actually identical siblings. Another fall-back alternative would be to add the student's attendance manually using the Add Attendance button in the View Attendance activity.

All in all, the application performed as intended during the short testing session. In the future, more comprehensive tests should be performed, such as testing the application in real classes of different education levels and class sizes.

The next section discusses about the project challenges faced during the development process of FaceIt.

9.2 Project Challenges

The difficulties in the implementation of the developed application revolved around the use of face recognition technologies. Even with advanced technology and machine learning techniques today, it is still impossible to achieve a recognition accuracy of 100%, especially when the face recognition model has to be implemented in a mobile device. In the context of attendance taking, accuracy is crucial as the lack of an accurate face recognition model could lead to commotions during the attendance taking process when students are unable to take their attendances. Moreover, the problem amplifies when identical siblings are brought into the picture. Ordinary people

already face problems in distinguishing identical siblings, let alone artificial intelligence.

In order to overcome the difficulties revolving around the use of face recognition, the developed application uses a novel attendance-taking process to make up for the lack of a completely accurate face recognition model. In the case of identical siblings, the application is able to recognize multiple identical siblings in the same image frame. With the absence of one or more siblings, the application is able to prompt human intervention to select the present identical siblings. Thus, the novelty of the application is to be able to provide a fully automated attendance-taking process, with minimal human intervention needed only to counteract the vulnerabilities of face recognition models when presented with identical siblings.

The following section discusses the objective evaluation of the project.

9.3 Objective Evaluation

The three main objectives of this project are listed below again for reference:

- To develop a face recognition mobile application that is able to automate the attendance-taking process in schools.
- To develop a user-friendly mobile application interface for teachers to manage students' attendance.
- To implement a face recognition algorithm with more than 85% accuracy and a processing time of fewer than 3 seconds per student.

With reference to the individual module testing results and the system testing section above, it is evidenced that the objectives of this project have been met. As for the first objective, the developed application, FaceIt, is able to automate the attendance-taking process in schools in most scenarios, with the exception that students with identical siblings have to interact with the application to pick out the present identical siblings in the event that not all enrolled identical siblings are present in a particular class on that day. However, scenarios like these are scarce, since most of the time, it is assumed that students would not be absent for classes. Therefore, the first objective has been achieved.

As for the second objective, the developed application ensures that the user interface is simplified to its max and convenient for users to use. Taking into consideration that some lecturers and teachers in the education sector might not be very well-versed with technology, buttons have been labelled with meaningful phrases and error or success messages are clear and distinct when giving user feedback about the application. As such, the second objective has been achieved in this project.

The last of the three objectives states that the application should have a recognition accuracy of at least 85% and a recognition time frame of 3 seconds at max. During the testing process, the application was able to recognize students in a class of 25 students (standard tutorial class size) accurately with an average recognition time of 2 seconds per student, with the exception of students with identical siblings having recognition times of 5 to 10 seconds. As for the recognition accuracy, there are not much concern as the developed application provides several fall-back alternatives in the event that the application fails to recognize a student accurately –the first alternative is for users to set identical sibling linkages between look-alike students that the application finds hard to differentiate and the second alternative is for users to add the attendance of any students that the application fails to recognize. In previous systems, this fall-back alternative was not included in the system, which was why the recognition accuracy played a humongous role in the overall application. However, with the fall-back methods included in the developed application in this project, FaceIt is able to compromise for face recognition inaccuracies. Therefore, the third objective stated in this project has been achieved.

9.4 Concluding Remark

In a nutshell, the functionalities of the five modules in the application – Login and Registration, Class, Student, Enrolment and Attendance – have been tested thoroughly to ensure that all of them work as intended. Moreover, validations for formatting and invalid flows have been implemented and tested to ensure that the application is able to handle various scenarios that may happen in the real-world.

When evaluating the developed application against the project's objectives, it is evidenced that all three project objectives have been achieved. In the next chapter, this report is summed up with a fruitful conclusion.

Chapter 10

Conclusion

In this chapter, the project's report is summarized in terms of the project's objectives, motivation, and implemented methods and how the novelty of the project contributes to society by solving problems faced by previous similar systems. This chapter ends with a summary of the application that has been developed and future work that can be done to further improve the developed application.

10.1 Project Review and Discussion

In a nutshell, the developed application, named FaceIt, aims to conquer the problems of manual attendance-taking methods using face recognition technologies integrated into a mobile application.

Attendance-taking has always been a hassle to teachers and lecturers. Not only are manual-attendance taking methods time-consuming and susceptible to impersonation and forgery, but also wastes pricy resources such as paper. Thus, with the motivation to ease the attendance-taking process for teachers, this project reports the development of a user-friendly face recognition mobile application that is able to automate the attendance-taking process with a recognition accuracy of at least 85% and a processing time of fewer than 3 seconds per student.

The application is realised using Android Studio to design the front-end and Firebase to handle the back-end functionalities such as authentication and data storage. The face detection and face recognition technologies used are Firebase ML Kit's Face Detection API and MobileFaceNet respectively. The application consists of five modules – the Login and Registration module, Class module, Student module, Enrolment module and Attendance module.

The functionalities of the five modules are dependent on one another, and all come together in a general flow for the convenience of users. First, the Login and Registration module allows users to register new accounts securely with second-factor authorisation codes, log into existing accountings and manage their own profiles. Next,

once the users have an authorized account, they can begin by adding new classes into the application through the Class module. Thereafter, these classes can be managed from the class list page. Moving on, users can proceed to add students into the application through the Student module. The Student module not only keeps the personal particulars of students, but also capture their face images and convert them into face embeddings for the face recognition process later on. Having added classes and students into the application, users can manage the enrolment of students in the Enrolment module, which is crucial to support the Attendance module. The Attendance module is the most crucial module in the application as it handles the actual attending taking process and attendance list management.

Cohesively, the five modules make up the developed application which is able to take attendance for students automatically. The novelty of this application is its ability to handle identical siblings and look-alike students by generating identical sibling linkages between students and prompting students to confirm the present identical sibling in the event where there are one or more absent identical siblings in class. Therefore, the application mostly omits the need for humans to take attendance, with the exception of minimal human intervention only when identical siblings are enrolled in the class and not all of them are present.

In the following section, the recommendations and future work for this project are stated and discussed.

10.2 Recommendations and Future Work

In the future, there are certain enhancements that can be added to this project to increase the overall effectiveness in automating the attendance process in classrooms. The more crucial enhancements all revolve around the face detection and face recognition models.

First, the accuracy of the face recognition model can be improved. In the current implementation, only one training image is converted to a face embedding to be used in the attendance-taking process. In the future, data augmentation techniques can be used to create more training images, and thus, more training face embeddings for the recognition process from just one face input image. Generally, data augmentation is a

technique used with neural networks, such as the MobileFaceNet model used in this project, to transform a single face image to multiple face images as if they were images taken from different angles and under different lightings. This way, the application would be able to recognize students even faster and more accurately, even if students change their appearance such as a change in hairstyle or facial accessories such as glasses.

Secondly, an option for image-based attendance taking should be included. In smaller classes such as in kindergartens and lower primary school classes, taking the image of all students in the class may yield faster results compared to real-time, video-based attendance taking. However, with the current application, only video-based attendance taking can be used. In the future, the application could come with an option to input an image, in which the application will run the face detection and face recognition model and take multiple attendance records from one image. By adding this feature in the future, the application is able to cater to classes of all sizes and yield faster attendance-taking durations in more concise classes.

Last but not least, the application could be linked to the attendance system of universities and schools. Many schools have their own attendance systems in which students, parents and teachers can view and mange by logging into a portal (e.g. UTAR portal). In order to ease the workload of teachers, the application could be directly linked to the attendance management module within the school portal. This way, the attendance records taken can be directly updated on the spot, without the need for users to manually export spreadsheets and keying in the attendance records into the school portal.

Ultimately, the aim of this application is to fully automate the attendance-taking process in all levels of the education sector in the future as to free up teachers' workloads so that they have more time to prepare quality lessons.

REFERENCES

[1] K. Sennaar, "Facial Recognition Applications - Security, Retail, and Beyond", Emerj Artificial Intelligence Research, 2020. [Online]. Available: https://emerj.com/ai-sector-overviews/facial-recognition-applications/. [Accessed: 17- Mar- 2022].

[2] "Tamil Nadu schools to launch Facial recognition app to replace attendance registers", India Today, 2018. [Online]. Available: https://www.indiatoday.in/education-today/news/story/tamil-nadu-schools-facial-recognition-app-attendance-registers-artificial-intelligence-divd-1406813-2018-12-11. [Accessed: 17- Mar- 2022].

[3] A. Martinez, “Face recognition, Overview (Encyclopaedia of Biometrics),” SpringerLink, 01-Jan-2009. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-73003-5_84. [Accessed: 17-Mar-2022].

[4] M. Rouse, "What is Facial Recognition?", SearchEnterpriseAI, 2019. [Online]. Available: https://searchenterpriseai.techtarget.com/definition/facial-recognition. [Accessed: 17- Mar- 2022].

[5] "A Brief History of Facial Recognition", NEC, 2020. [Online]. Available: https://www.nec.co.nz/market-leadership/publications-media/a-brief-history-of-facial-recognition/#:~:text=The%20dawn%20of%20Facial%20Recognition,to%20recognise%20the%20human%20face. [Accessed: 17- Mar- 2022].

[6] E. Jaha and L. Ghouti, "Color face recognition using quaternion PCA", 4th International Conference on Imaging for Crime Detection and Prevention 2011, p. 3, 2011. Available: https://www.researchgate.net/publication/260728292_Color_face_recognition_using_quaternion_PCA. [Accessed 17 March 2022].

[7] S. Singh, R. Rastogi and P. Sharma, "Automatic Lecture Attendance System Using Face Reorganization", 2015. Available:

http://maioj.org/data/documents/april2015/041509.pdf. [Accessed 17 March 2022].

[8] C. Kanan and G. Cottrell, "Color-to-Grayscale: Does the Method Matter in Image Recognition?", 2012. Available: https://doi.org/10.1371/journal.pone.0029740. [Accessed 17 March 2022].

[9] N. Koren, "Pixels, Images, And Files.", 2013. Available: http://www.normankoren.com/pixels_images.html#:~:text=Each%20pixel%20typically%20consists%20of,levels%20(0%2D255). [Accessed 17 March 2022].

[10] M. Pratiksha, "Contrast Enhancement of Images and videos using Histogram Equalization", International Journal on Recent and Innovation Trends in Computing and Communication, vol. 14, no. 11, 2016. [Accessed 17 March 2022].

[11] N. Sasi and V. Jayasree, "Contrast Limited Adaptive Histogram Equalization for Qualitative Enhancement of Myocardial Perfusion Images", Engineering, vol. 5, no. 10, pp. 326-331, 2013. [Accessed 17 March 2022].

[12] N. Parmar and B. Mehta, "Face Recognition Methods & Applications", International Journal of Computer Technology and Applications, vol. 4, pp. 84-86, 2013. [Accessed 17 March 2022].

[13] M. Viegas et al., "Different approaches of face recognition", International Journal of Engineering Research & Technology, vol. 8, no. 6, 2019. Available: https://www.ijert.org/different-approaches-of-face-recognition. [Accessed 17 March 2022].

[14] "Principal Component Analysis Tutorial", DeZyre, 2020. [Online]. Available: https://www.dezyre.com/data-science-in-python-tutorial/principal-component-analysis-tutorial#:~:text=The%20main%20idea%20of%20principal,up%20to%20the%20maximum%20extent.&text=As%20a%20layman%2C%20it%20is%20a%20method%20of%20summarizing%20data. [Accessed: 17- Mar- 2022].

[15] A. Singh and S. Kumar, "Face Recognition Using PCA and Eigen Face Approach", 2012. Available: http://ethesis.nitrkl.ac.in/3814/1/Thesis.pdf. [Accessed 17 March 2022].

[16] L. Paul and A. Suman, "Face recognition using principal component analysis method", IJARCET, vol. 1, no. 9, 2012. [Accessed 17 March 2022].

[17] K. Bhuvaneshwari, A. Abirami and N. Sripriya, "Face recognition using PCA", International Journal of Engineering And Computer Science, vol. 6, no. 4, 2017. [Accessed 17 March 2022].

[18] D. Nithya, "Automated class attendance system based on face recognition using PCA algorithm", International Journal of Engineering Research and Technology, vol. 4, no. 12, 2015. [Accessed 17 March 2022].

[19] A. Dinh, "Principal Component Analysis (PCA)", 2019. [Online]. Available: https://dinhanhthi.com/principal-component-analysis. [Accessed: 17- Mar- 2022].

[20] S. Bhattacharyya and K. Rahul, "Face recognition by linear discriminant analysis", International Journal of Communication Network Security, vol. 2, pp. 33-35, 2013. [Accessed 17 March 2022].

[21] M. Mohammadi et al., "Data mining EEG signals in depression for their diagnostic value", BMC Medical Informatics and Decision Making, vol. 15, no. 1, p. 108, 2015. [Accessed 17 March 2022].

[22] T. Ojala, M. Pietekainen and T. Maepaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 971-987, 2002. [Accessed 17 March 2022].

[23] K. Prado, "Face Recognition: Understanding LBPH Algorithm", Medium, 2017. [Online]. Available: https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b. [Accessed: 17- Mar- 2022].

[24] M. Pietikainen and G. Zhao, "Two decades of local binary patterns: a survey", Advances in Independent Component Analysis and Learning Machines, pp. 175-210, 2015. [Accessed 17 March 2022].

[25] M. Rahim, M. Hossain and M. Azam, "Face recognition using local binary patterns (LBP)", Global Journal of Computer Science and Technology Graphics & Vision, vol. 13, no. 4, 2013. [Accessed 17 March 2022].

[26] M. Kasar, D. Bhattacharyya and T. Kim, "Face recognition using neural network: a review", International Journal of Security and Its Applications, vol. 10, no. 3, pp. 81-100, 2016. [Accessed 17 March 2022].

[27] J. Mahanta, "Introduction to Neural Networks, Advantages and Applications", Medium, 2017. [Online]. Available: https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207. [Accessed: 17- Mar- 2022].

[28] T. Le, "Applying artificial neural network for face recognition’", Advances in Artificial Neural Systems, vol. 2011, pp. 1-16, 2011. [Accessed 17 March 2022].

[29] M. Deshpande, "Convolutional Neural Network", Medium, 2017. [Online]. Available: https://towardsdatascience.com/cnn-part-i-9ec412a14cb1. [Accessed: 17- Mar- 2022].

[30] K. Islam, R. Raj and A. Al-Murad, "Performance of SVM, CNN, and ANN with BoW, HOG, and image pixels in face recognition", 2017 2nd International Conference on Electrical & Electronic Engineering, pp. 27-29, 2017. [Accessed 17 March 2022].

[31] K. Kamila, "Advances in computational intelligence and robotics", Handbook of Research on Emerging Perspectives in Intelligent Pattern Recognition, Analysis, and Image Processing, 2016. [Accessed 17 March 2022].

[32] "AT&T Database of Faces", Kaggle, 2019. [Online]. Available: https://www.kaggle.com/kasikrit/att-database-of-faces. [Accessed: 17- Mar- 2022].

[33] S. Anwarul and S. Dahiya, "A Comprehensive Review on Face Recognition Methods and Factors Affecting Facial Recognition Accuracy", Lecture Notes in Electrical Engineering, pp. 495-514, 2019. [Accessed 17 March 2022].

[34] A. Katara et al., "Attendance system using face recognition and class monitoring system", Internal Journal on Recent and Innovation Trends in Computing and Communication, vol. 5, no. 2, 2017. [Accessed 17 March 2022].

[35] N. Balcoh, M. Yousaf and M. Baig, "Algorithm for efficient attendance management: face recognition-based approach", International Journal of Computer Science Issues, vol. 9, no. 1, 2012. [Accessed 17 March 2022].

[36] S. Khan, N. Usman and A. Akram, "Real time automatic attendance system for face recognition using face API and OpenCV", Wireless Personal Comunications, vol. 113, pp. 469-480, 2020. [Accessed 17 March 2022].

[37] A. Raghuwanshi and P. Swami, "An automated classroom attendance system using video based face recognition", 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2017. [Accessed 17 March 2022].

[38] Smitha, P. Hedge and Afshin, "Face recognition-based attendance management system", Face recognition-based attendance management system', vol. 9, no. 5, 2020. [Accessed 17 March 2022].

[39] Department of Statistics Malaysia Official Portal, 2020. [Online]. Available: https://www.dosm.gov.my/v1/index.php?r=column/cthemeByCat&cat=395&bul_id=SFRacTRUMEVRUFo1Ulc4Y1JlLzBqUT09&menu_id=amVoWU54UTl0a21NWmdhMjFMMWcyZz09. [Accessed: 17- Mar- 2022].

[40] R. Samet and M. Tanriverdi, "Face recognition-based mobile automatic classroom attendance management system", 2017 International Conference on Cyberworlds, 2017. [Accessed 17 March 2022].

[41] D. Sunaryono, J. Siswantoro and R. Anggoro, "An android based course attendance system using face recognition", Journal of King Saud University – Computer and Information Sciences, 2019. [Accessed 17 March 2022].

[42] F. Isinkaye, J. Soyemi and O. Arowosegbe, "Android-based face recognition system for class attendance and malpractise control", International Journal of Computer Science and Information Security, vol. 18, no. 1, 2020. [Accessed 17 March 2022].

[43] L. Roeder, "Netron", Netron.app, 2019. [Online]. Available: https://netron.app/. [Accessed: 22- Mar- 2022].

APPENDIX A

A.1 Weekly Report

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 1
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

This week was the first week of the academic semester. As such, the first week was mainly spent on reviewing the report and application codes from Project 1 in order to come up with the goals to be accomplished this semester for Project 2.

A checklist was created, consisting of all the enhancements and changes that had to be made to the application based on Mr Tou Jing Yi's comments from two semesters ago and the future work section from Project 1's report.

2. WORK TO BE DONE

- Reread Project 1 report
- Review Project 1 code and application design
- Start working on enhancements as per checklist written

3. PROBLEMS ENCOUNTERED

- Had some issues with Android Studio as the android emulator was not working. Suspected that the issue had something to do with the programs that had been installed in the current device during the internship period.

4. SELF EVALUATION OF THE PROGRESS

- Needed to resolve the Android Studio problem as soon as possible.
- Had to plan out timeline in order to accomplish all Project 2 goals on time

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 2
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

This week, Mr Tou Jing Yi did not call for our weekly Final Year Project meeting yet, so I spent my time rereading my Project 1 report and reviewing my Project 1 code and application design. I also looked up the FYP2 guidelines to see what needed to be changed in terms of report organization.

2. WORK TO BE DONE

Listed below are the work that needs to be done in the upcoming week:

- Reformat Project 2 report
- Refer to some UI designs online to see which part of my application design can be improved.
- Work on coding enhancements

3. PROBLEMS ENCOUNTERED

The Android emulator in my Android Studio still could not be fixed. Therefore, I had to sought out an alternative to run my code in Android Studio

4. SELF EVALUATION OF THE PROGRESS

- Need to work on enhancements as soon as possible
- Need to plan week-by-week timeline so that Project 2 progress stays on track.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 3
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

This week, I worked on code enhancements. I fixed some existing bugs that I found in Project 1, as listed below:

- Unable to delete identical siblings from Student List page
- Unable to enrol/unenroll several students at once
- Unable to search for students and enrol them accordingly in the Enrolment module
- Unable to prompt permissions modal when user declines the first permission modal (E.g. Asking for camera permissions, storage permissions, etcetera)

2. WORK TO BE DONE

Listed below are the work that needs to be done in the upcoming week:

- Work on coding enhancements
- Decide on which enhancements to keep and which enhancements to scrap due to the possible lack of time.

3. PROBLEMS ENCOUNTERED

- None

4. SELF EVALUATION OF THE PROGRESS

- So far progress is still relevant, but need to spend more time to finish coding all the enhancements in the coming week

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 4
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

This week, I spent some time working on the report. I started by highlighting the sections that needed to be amended, then I also changed all 43 references in the report from Harvard style to IEEE, as required by the FYP2 guidelines. As for the coding part, I modified the UI designs on the Class List page so that each class has a unique icon that is coloured and has three letters that make up the short form for each of the class's names. I had to amend the codes in other files accordingly because the Class List View is also used by the Take Attendance and View Attendance activities.

As for the Android emulator problem that I had been facing for the last couple of weeks, I decided to connect to my mobile device through the Internet and transfer my APK file to my mobile device in debugging mode.

2. WORK TO BE DONE

- Work on coding enhancements for other modules
- Start writing FYP2 report

3. PROBLEMS ENCOUNTERED

- None

4. SELF EVALUATION OF THE PROGRESS

- So far, time is still mildly sufficient to complete the enhancements that I need to perform for FYP2.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 5
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

This week, I mainly spent my time on coding enhancements, as listed below:

- Modified identical siblings field in Add Student page so users have to type the student IDs of the identical siblings instead of select the siblings from a checklist.
- Started working on handling identical siblings during the Take Attendance process.

2. WORK TO BE DONE

- Work on the identical siblings handling in the Take Attendance activity
- Work on FYP2 report

3. PROBLEMS ENCOUNTERED

- Faced a technical issue when trying to complete the identical siblings handling in the Take Attendance activity.

4. SELF EVALUATION OF THE PROGRESS

- Need to spend more time thinking of the identical siblings handling in the Take Attendance process because it is a crucial part of my application and is one of the enhancements that have to be added in Project 2.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 6
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

This week, I completed the handling of identical siblings in the Take Attendance activity. The application was able to detect if identical siblings are present in frae and if so, the application will detect and recognize all the identical siblings together. If the application detects that one or more identical siblings are absent, then the application prompts a modal for the user to select the present identical sibling.

2. WORK TO BE DONE

- Work on FYP2 report
- Work on other crucial code enhancements, such as the Generate Report feature

3. PROBLEMS ENCOUNTERED

- None

4. SELF EVALUATION OF THE PROGRESS

- Need to spend more time on FYP2 report since more emphasis was placed on coding this week

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 7
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

This week, I added the Generate Report feature. I also cleaned up the codes in the Student module and Login and Register module (Remove Log.d, repetition codes, etcetera)

2. WORK TO BE DONE

- Work on FYP2 report
- Work on other code enhancements such as UI enhancements
- See if there are any major bugs to fix

3. PROBLEMS ENCOUNTERED

- None

4. SELF EVALUATION OF THE PROGRESS

- Need to start writing report next week.
- In terms of coding, still making adequate progress

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 8
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

This week, I spent most of my time writing the FYP2 report. I rewrote some parts of Chapter 1, rearranged Chapter 3 and started working on Chapter 4,5 6 and 7, which represents one module each.

2. WORK TO BE DONE

- Work on UI enhancements, especially for the Student Listing activity
- Complete FYP2 report draft before Week 10

3. PROBLEMS ENCOUNTERED

- None

4. SELF EVALUATION OF THE PROGRESS

- Need to spend more time on completing the FYP2 report draft

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 9
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Wrote a draft for FYP2 report to be checked by Mr. Tou Jing Yi. Reworded certain parts in chapter 1 and 2 so that they look more like a report instead of a proposal (E.g. Reworded all sentences containing the word "proposed"). Rewrote chapter 3 and conclusion chapter (chapter 10). Added implementation and testing chapters according to module (chapter 4 to chapter 9).

Added "Add Attendance" button for users to add students' attendance records manually as a fallback method in the event that the application cannot recognize particular students or if some students arrive late and missed the attendance taking session.

2. WORK TO BE DONE

- Check if any more enhancements need to be completed in the application.
- Finish writing test cases for report

3. PROBLEMS ENCOUNTERED

- Had issues changing the interface of the View Student List activity (upon click of a button, should hide the student images as to show more students on screen at once.

4. SELF EVALUATION OF THE PROGRESS

Progress in the upcoming weeks might be slower due to heavy workloads from other subjects.

Will keep track of the checklist to ensure that no enhancements are missed out. If enhancements are taking too long, will try to think of an alternative way of implementing them.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 10
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Fixed bugs related to the display of the Student List and the Generate Report function.

Added screenshots of the test cases input and output results to FYP2 report.

2. WORK TO BE DONE

- Test application thoroughly to see if there are any more bugs to be fixed.
- Enhance UI to make the application more user-friendly.

3. PROBLEMS ENCOUNTERED

- None

4. SELF EVALUATION OF THE PROGRESS

On track of FYP2 progress. Might have to chip in more time into finalizing report.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 11
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Fixed bugs in the Enrolment module as well as enhanced the UI of the Take Attendance and View Attendance activities.

Modified Chapter 9 (System Evaluation) of the FYP2 Report

2. WORK TO BE DONE

- Rewrite Conclusion chapter of FYP2 Report
- Rework bounding box of Take Attendance and Add Student Face activities

3. PROBLEMS ENCOUNTERED

- None

4. SELF EVALUATION OF THE PROGRESS

Most parts of FYP2 completed in terms of code. Need to work more on report in the coming week.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 12
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Finalized UI for the entire application

Added screenshots to Chapter 4 to 8 in FYP2 report for the module testing subsections.

2. WORK TO BE DONE

- Finalize report for submission next week

3. PROBLEMS ENCOUNTERED

- None

4. SELF EVALUATION OF THE PROGRESS

On track with FYP2 progress. Need to finalize and format report as soon as possible

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3 Year 3 (Y3S3)	**Study week no.:** Week 13
Student Name & ID: Jacynth Tham Ming Quan, 18ACB01600	
Supervisor: Mr Tou Jing Yi	
Project Title: Real - Time Face Recognition Mobile Application for Class Attendance	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Completed report and code for FYP submission

2. WORK TO BE DONE

Submit report to Turnitin, supervisor and FYP 2 portal

3. PROBLEMS ENCOUNTERED

- None

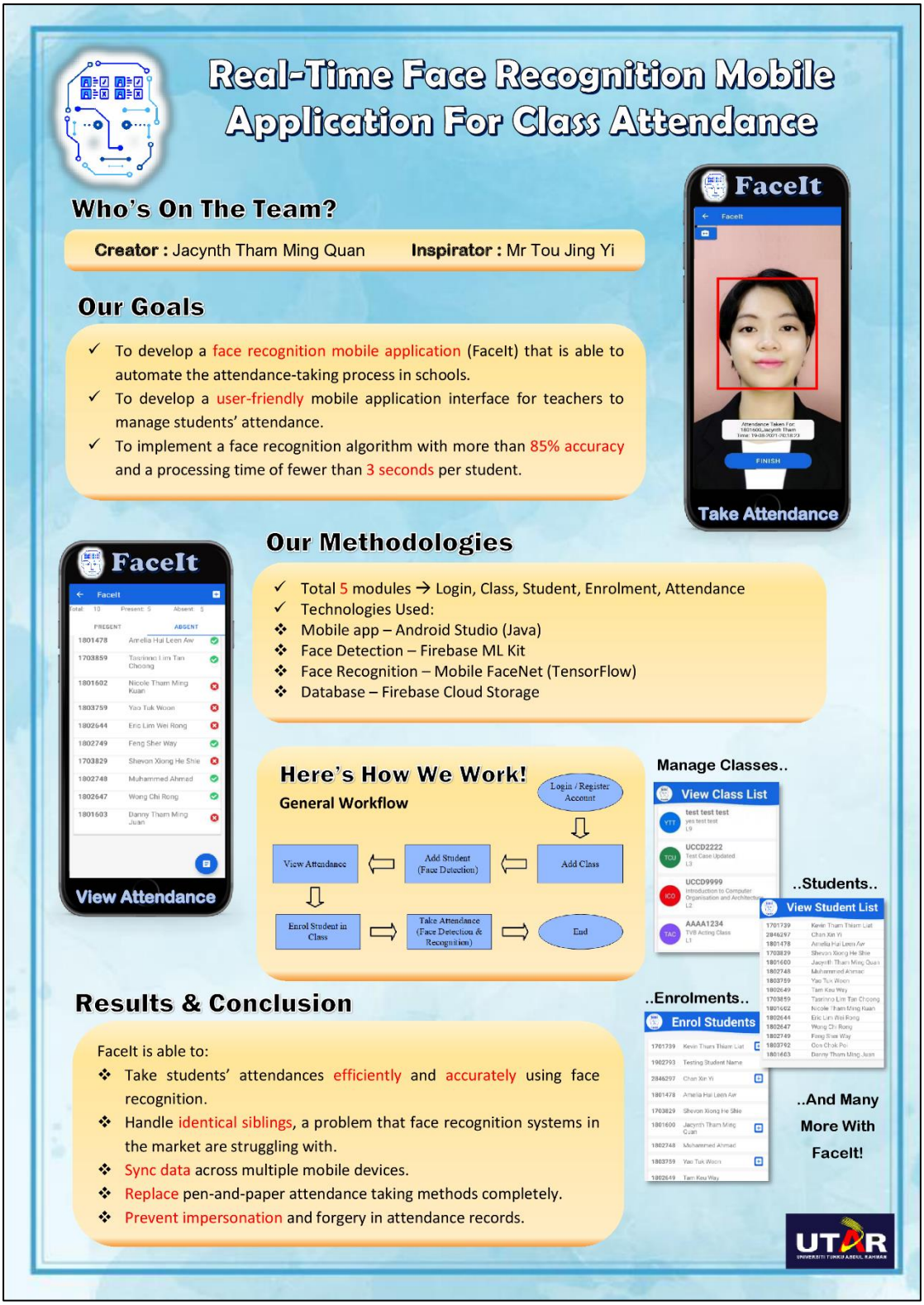
4. SELF EVALUATION OF THE PROGRESS

FYP2 has been completed, both report and codes.

Supervisor's signature

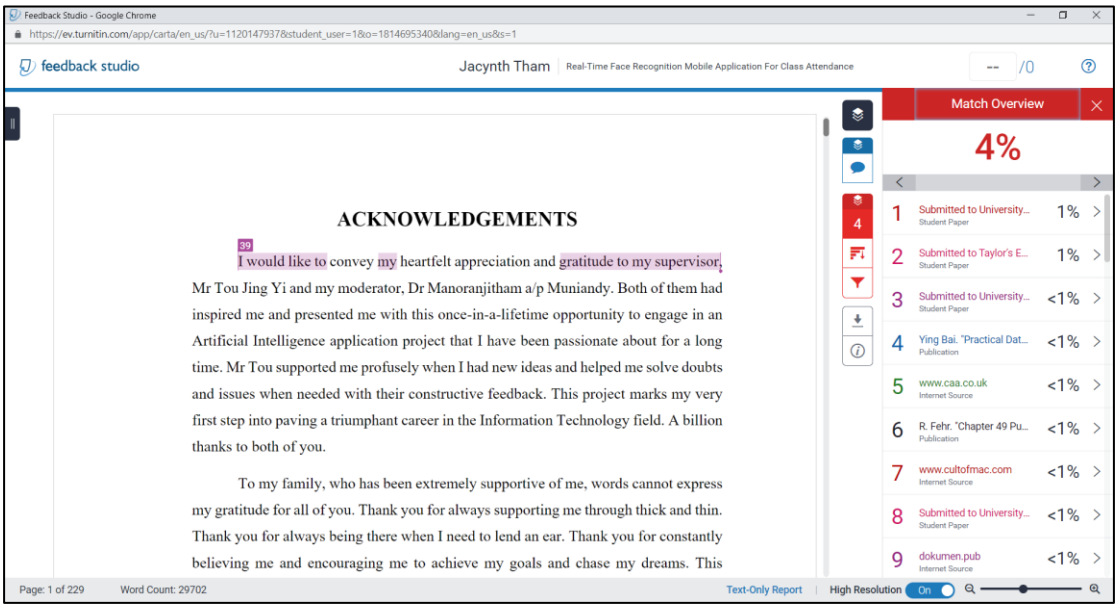
Student's signature

A.2 Poster

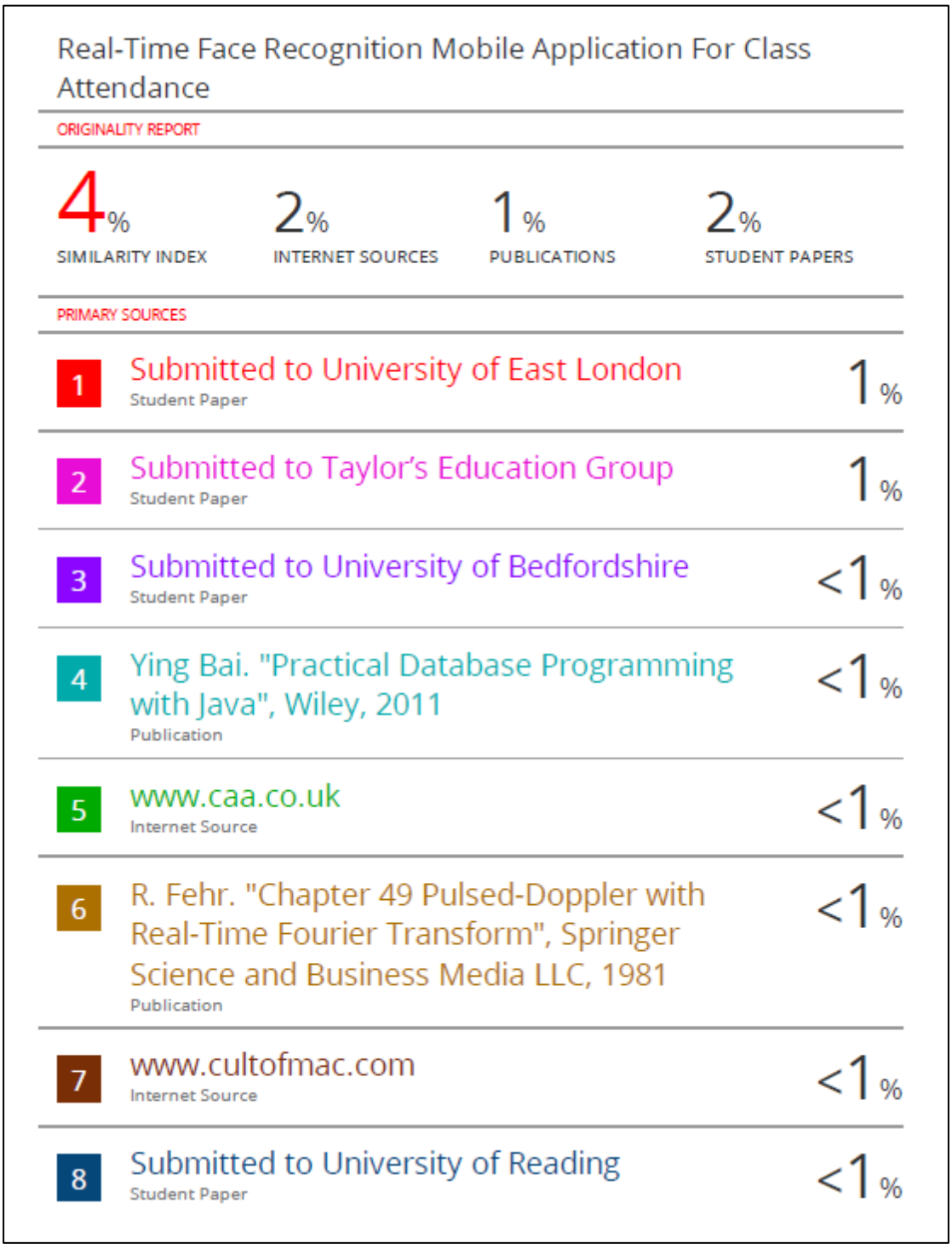

Real-Time Face Recognition Mobile Application For Class Attendance
Who's On The Team?
Creator : Jacynth Tham Ming Quan
Inspirator : Mr Tou Jing Yi
Our Goals
To develop a face recognition mobile application (FaceIt) that is able to automate the attendance-taking process in schools.
To develop a user-friendly mobile application interface for teachers to manage students' attendance.
To implement a face recognition algorithm with more than 85% accuracy and a processing time of fewer than 3 seconds per student.
FaceIt
Take Attendance
FaceIt
View Attendance
Our Methodologies
Total 5 modules → Login, Class, Student, Enrolment, Attendance
Technologies Used:
Mobile app – Android Studio (Java)
Face Detection – Firebase ML Kit
Face Recognition – Mobile FaceNet (TensorFlow)
Database – Firebase Cloud Storage
Here's How We Work!
General Workflow
Login / Register Account
Add Class
Add Student (Face Detection)
View Attendance
Enrol Student in Class
Take Attendance (Face Detection & Recognition)
End
Manage Classes..
View Class List
..Students..
View Student List
..Enrolments..
Enrol Students
..And Many More With FaceIt!
Results & Conclusion
FaceIt is able to:
Take students' attendances efficiently and accurately using face recognition.
Handle identical siblings, a problem that face recognition systems in the market are struggling with.
Sync data across multiple mobile devices.
Replace pen-and-paper attendance taking methods completely.
Prevent impersonation and forgery in attendance records.
UTAR

PLAGIARISM CHECK RESULT

Screenshot of Turnitin Report (First Page)



Turnitin Summary of Originality Report



Real-Time Face Recognition Mobile Application For Class Attendance

ORIGINALITY REPORT

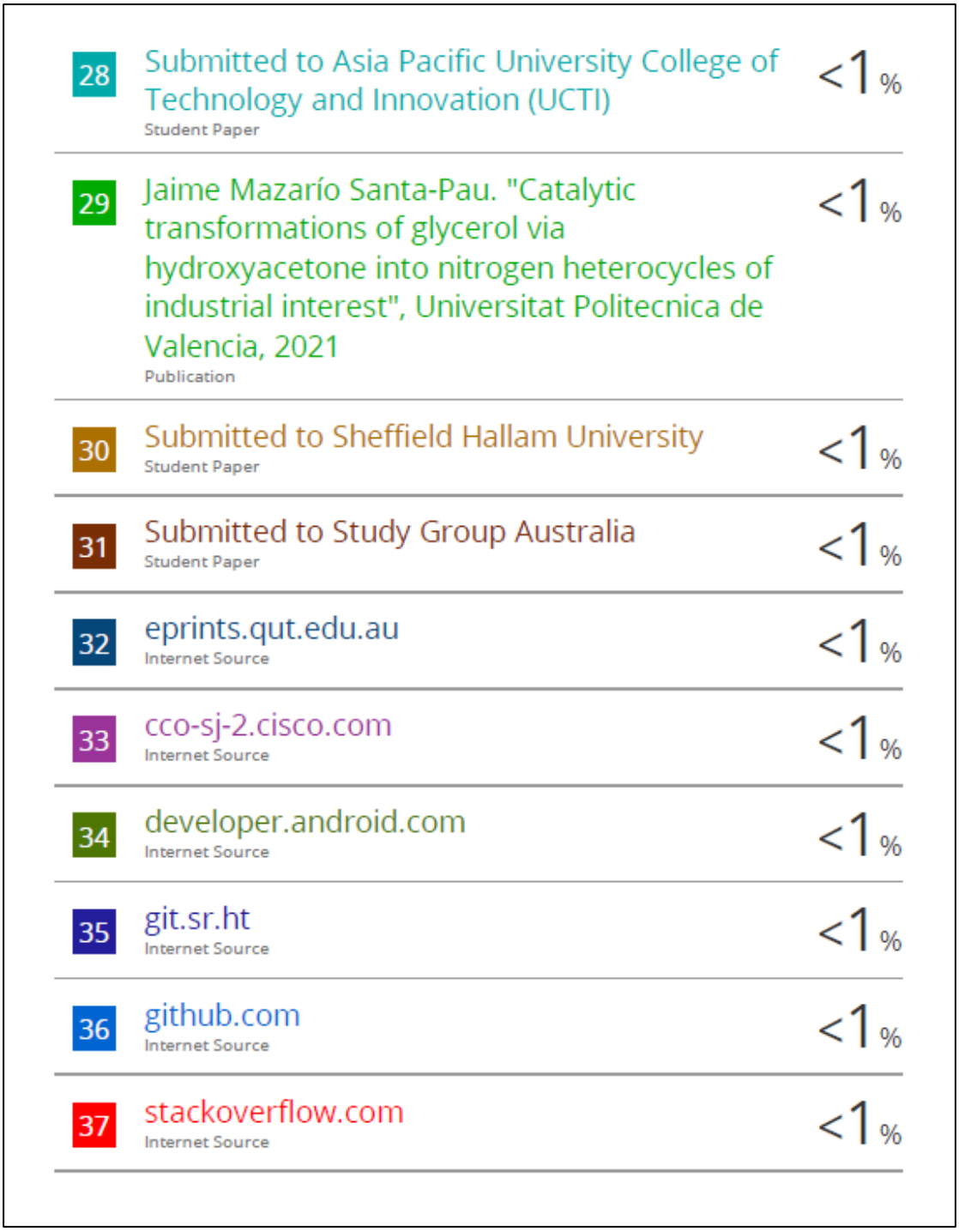
4%	2%	1%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

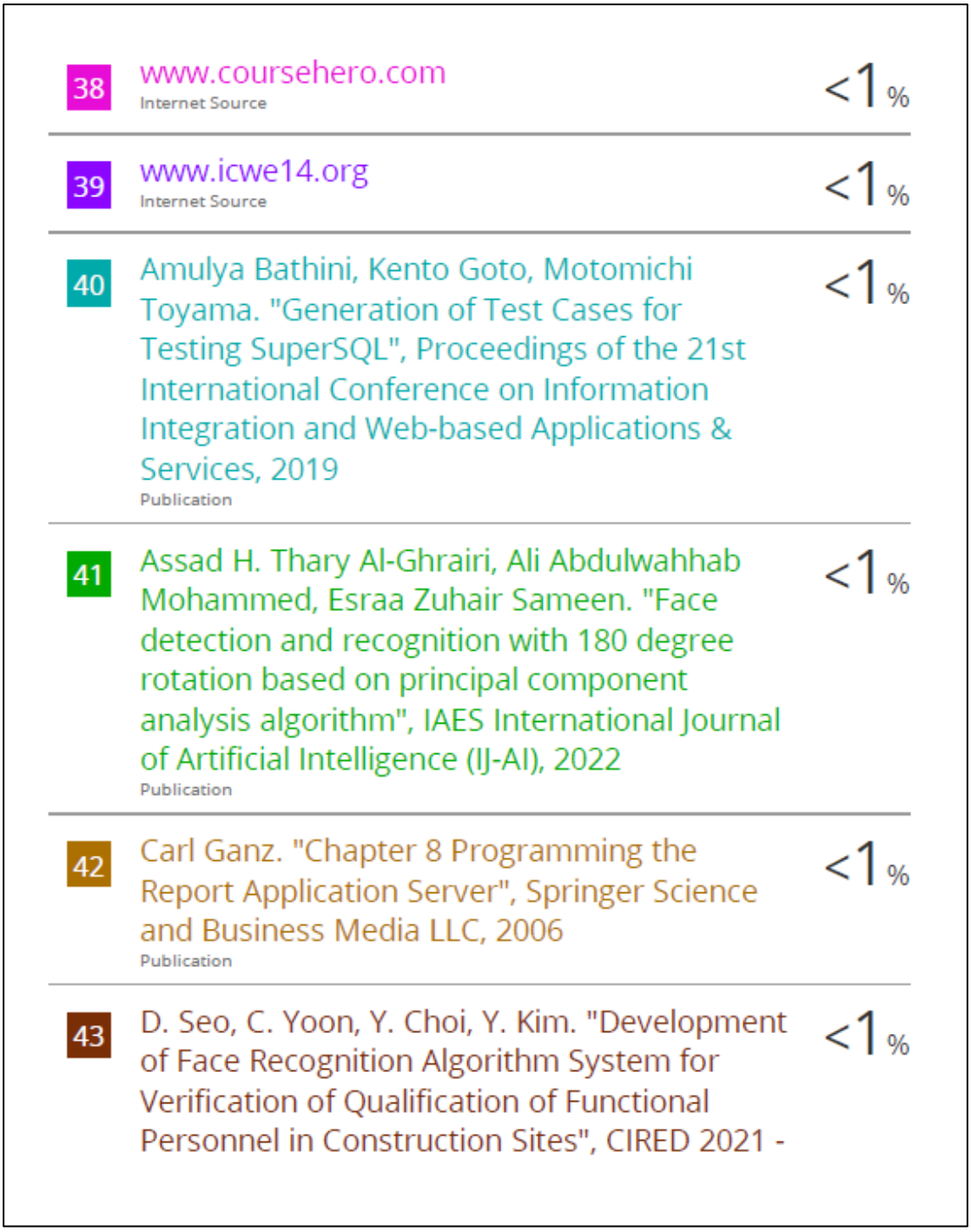
1	Submitted to University of East London Student Paper	1%
2	Submitted to Taylor's Education Group Student Paper	1%
3	Submitted to University of Bedfordshire Student Paper	<1%
4	Ying Bai. "Practical Database Programming with Java", Wiley, 2011 Publication	<1%
5	www.caa.co.uk Internet Source	<1%
6	R. Fehr. "Chapter 49 Pulsed-Doppler with Real-Time Fourier Transform", Springer Science and Business Media LLC, 1981 Publication	<1%
7	www.cultofmac.com Internet Source	<1%
8	Submitted to University of Reading Student Paper	<1%

9	dokumen.pub Internet Source	<1 %
10	docplayer.net Internet Source	<1 %
11	ir.unimas.my Internet Source	<1 %
12	firebase.google.com Internet Source	<1 %
13	Submitted to CSU, San Jose State University Student Paper	<1 %
14	Submitted to Western Governors University Student Paper	<1 %
15	repository.president.ac.id Internet Source	<1 %
16	Submitted to Institute of Research & Postgraduate Studies, Universiti Kuala Lumpur Student Paper	<1 %
17	ebin.pub Internet Source	<1 %
18	K. Meena, J. N. Swaminathan, T. Rajendiran, S. Sureshkumar, N. Mohamed Imtiaz. "Chapter 17 An Automated Attendance System Through Multiple Face Detection and	<1 %

	Recognition Methods", Springer Science and Business Media LLC, 2022 Publication	
19	www.codeproject.com Internet Source	<1 %
20	literature.rockwellautomation.com Internet Source	<1 %
21	Yiran Shen, Mingrui Yang, Bo Wei, Chun Tung Chou, Wen Hu. "Learn to Recognise: Exploring Priors of Sparse Face Recognition on Smartphones", IEEE Transactions on Mobile Computing, 2017 Publication	<1 %
22	Submitted to De Montfort University Student Paper	<1 %
23	pureapps2.hw.ac.uk Internet Source	<1 %
24	Submitted to Multimedia University Student Paper	<1 %
25	Submitted to Universiti Teknologi MARA Student Paper	<1 %
26	Submitted to University of London External System Student Paper	<1 %
27	Submitted to University of Teesside Student Paper	<1 %



28	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1 %
29	Jaime Mazarío Santa-Pau. "Catalytic transformations of glycerol via hydroxyacetone into nitrogen heterocycles of industrial interest", Universitat Politecnica de Valencia, 2021 Publication	<1 %
30	Submitted to Sheffield Hallam University Student Paper	<1 %
31	Submitted to Study Group Australia Student Paper	<1 %
32	eprints.qut.edu.au Internet Source	<1 %
33	cco-sj-2.cisco.com Internet Source	<1 %
34	developer.android.com Internet Source	<1 %
35	git.sr.ht Internet Source	<1 %
36	github.com Internet Source	<1 %
37	stackoverflow.com Internet Source	<1 %



38	www.coursehero.com Internet Source	<1%
39	www.icwe14.org Internet Source	<1%
40	Amulya Bathini, Kento Goto, Motomichi Toyama. "Generation of Test Cases for Testing SuperSQL", Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services, 2019 Publication	<1%
41	Assad H. Thary Al-Ghrairi, Ali Abdulwahhab Mohammed, Esraa Zuhair Sameen. "Face detection and recognition with 180 degree rotation based on principal component analysis algorithm", IAES International Journal of Artificial Intelligence (IJ-AI), 2022 Publication	<1%
42	Carl Ganz. "Chapter 8 Programming the Report Application Server", Springer Science and Business Media LLC, 2006 Publication	<1%
43	D. Seo, C. Yoon, Y. Choi, Y. Kim. "Development of Face Recognition Algorithm System for Verification of Qualification of Functional Personnel in Construction Sites", CIRED 2021 -	<1%

	The 26th International Conference and Exhibition on Electricity Distribution, 2021 Publication	
44	Luis M. Floría, Juan J. Mazo. "Dissipative dynamics of the Frenkel-Kontorova model", Advances in Physics, 1996 Publication	<1 %
45	Martin L. Shoemaker. "Chapter 8 Step 3: Assign Your Requirements to Components and Interfaces", Springer Science and Business Media LLC, 2004 Publication	<1 %
46	Nikolas Charlebois-Laprade, Evgueni Zabourdaev, Daniel Brunet, Bruce Wilson et al. "Expert Office 365", Springer Science and Business Media LLC, 2017 Publication	<1 %
47	archive.org Internet Source	<1 %
48	harshitabambure.medium.com Internet Source	<1 %
49	W. H. Cheung, A. J. Al-Khalili, W. Lynch. "Low power design of Motion Compensation module for MPEG-4 video transcoder in DCT domain", 2007 IEEE Northeast Workshop on Circuits and Systems, 2007 Publication	<1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Jacynth Tham Ming Quan
ID Number(s)	18ACB01600
Programme / Course	Computer Science
Title of Final Year Project	Real - Time Face Recognition Mobile Application for Class Attendance

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 4 % **Similarity by source** Internet Sources: 2 % Publications: 1 % Student Papers: 2 %	
Number of individual sources listed of more than 3% similarity: 0	
Parameters of originality required and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor

Name: Mr. Tou Jing Yi

Date: 21/04/2022

Signature of Co-Supervisor

Name: _____

Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	18ACB01600
Student Name	Jacynth Tham Ming Quan
Supervisor Name	Mr. Tou Jing Yi

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Plastic Cover (for hardcopy)
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)
Date: 20/04/2022