SMART MANAGEMENT SYSTEM FOR TUITION CENTRE OPERATIONS

BY

CHANG YUN QI

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)

JUNE 2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Tan Joi San, and my moderator, Ts. Dr. Mogana a/p Vadiveloo for their outstanding mentorship and guidance throughout the development of this project, "Smart Management System for Tuition Centre Operations". Their extensive knowledge and proficiency in the field provided me with critical insights. They consistently offered constructive feedback during each project phase and ensured that I stayed on track and met the project objectives.

Furthermore, I must also extend my deepest appreciation to my parents, who have given me support and encouragement constantly. They provided emotional support, empathy, motivation, and encouragement, especially during late-night coding sessions and moments of self-doubt. I am deeply grateful to my parents for providing a supportive atmosphere and confidence, which allowed me to concentrate on my task and accomplish the project's objectives.

ABSTRACT

This project, titled “Smart Management System for Tuition Centre Operations”, aims to develop a comprehensive solution to meet and fulfil the unique needs of small-sized tuition centres. The system addresses key challenges such as ineffective manual administrative and daily operational tasks, a lack of affordable management software, and poor communication between parents and tutors. This project prioritises rapid prototyping, iterative feedback from users, and continual refinement by utilising the Rapid Application Development (RAD) methodology. This ensures that the final product effectively satisfies user needs.

This system features a robust set of key modules, which include user authentication, student management, course management, class management with biometric facial recognition for attendance tracking, fee management, communication, and reporting and analytics. The system is created with Visual Studio Code (VS Code), Cloud Firestore Database, Firebase Functions, Firebase Authentication, Firebase Storage, Flutter, Dart, Node.js, JavaScript, and Python. The goals of this project are to increase communication between parents and tutors, streamline administrative and daily operational tasks, and boost the overall efficiency of tuition centres.

This project not only offers a practical tool for tuition centre management but also contributes valuable insights into the fields of educational technology and software development. The deployment of this system is expected to significantly reduce manual workload, reduce human errors, and foster better engagement between administrators, tutors, students, and parents.

Area of Study: Application Development, Software Development

Keywords: Tuition Centre Management System, Biometric Facial Recognition, Administrative Automation, Android Mobile Application, Flutter Mobile Application, Full-Stack Mobile Development

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SDLC	Software Development Life Cycle
RAD	Rapid Application Development
VS Code	Visual Studio Code
IDE	Integrated Development Environment
BaaS	Backend-as-a-Service
AI	Artificial Intelligence
VR	Virtual Reality
UI	User Interface
UML	Unified Modeling Language
FYP	Final Year Project

Chapter 1
Introduction

Chapter 1 consists of eight different sections, such as background information which highlights the vital role of tuition centres in enhancing student learning, problem statement and motivation which identify the current issues faced, project objectives that analyse the goals to be achieved, project scope which describe seven key modules that need to be developed in this project, proposed approach outlines the flowchart of the project, impact, significance, and contribution, report organisation that structures the report’s content flow, and a summary.

1.1 Background Information

In this fast-paced education-centred era, tuition plays an extremely vital role for primary school, secondary school, or university students. Tuition centres offer several extra courses outside of school hours for students to enhance their knowledge in different subjects [1]. However, the majority of tuition centre management still relies on traditional methods, which involve using paper and pen for a huge amount of paperwork every day. These manual processes are time-consuming, prone to mistakes, and inefficient.

With the continuous development of modern technology in the 21st century, digital solutions are becoming increasingly necessary as they can streamline daily operations, enhance communication, and simplify tasks. All the issues mentioned above can be resolved with the aid of a tuition centre management system. An effective tuition centre management system can help automate most of the administrative tasks and save valuable time. For example, the system will send related reminders and alert messages to all parents and students automatically, such as new announcements, fee reminders, or important updates [2].

This project aims to develop a “Smart Management System for Tuition Centre Operations”. This system is designed to streamline administrative workflows, manage all data centrally and transparently, foster the engagement between tutors and parents, and automate various academic functions. Hence, the tuition centres’ management team can operate their centre more effectively and simplify their daily tasks, tutors can focus on teaching, while students can focus on their academic progress. With the tuition centre management system, the entire process is super easy and efficient, and only takes a few seconds.

1.2 Problem Statement and Motivation

This project outlines three main problem statements which affect small-sized tuition centres. This includes the lack of an affordable tuition centre management system, inefficiency in attendance management and challenges in parental communication and engagement. These issues emphasise the need for an efficient and cost-effective tuition centre management solution.

i. Lack of an affordable tuition centre management system

Small-sized tuition centres often struggle to effectively manage their administrative and operational tasks due to limited financial resources. Nowadays, there are several types of tuition centre management systems or software development companies available in the market. However, all the systems are costly and unable to be afforded by the small-sized tuition centres. Rather than subscribing to or developing a costly system, they wish to manage their administrative tasks manually. As a result, this will reduce overall productivity and make it prone to human error.

ii. Inefficiency in attendance management

Tutors are required to take attendance manually, which is not only time-consuming but also prone to human error. For example, assuming there are 80 students in a class, the tutor needs to call their names one by one before each class commencement and record their attendance. If there are students who are late to class, the tutor will need to repeat the process at the end of the class to ensure that each student's attendance is recorded. This manual process detracts from the tutor's valuable teaching time and students' precious learning time.

iii. Challenges in parental communication and engagement

Parents face significant challenges in tracking their children's progress due to inadequate communication channels with tutors. Most of the small-sized tuition centres rely on the WhatsApp application to broadcast their latest updates or announcements. But there is a problem here, when the messages are sent out, there is no guarantee that all parents receive them on time, or some parents may overlook them. This results in inconsistencies and gaps in communication between parents and tutors. Moreover, they often remain uninformed about what their children are learning, hence leading to a disconnect between the family and the tuition centre.

This project, "Smart Management System for Tuition Centre Operations", is motivated and inspired by the previous teaching experience in a tuition centre. Due to the lack of implementation of the management system, the operational and administrative tasks can only be done manually, and the communication with parents can only be done via the WhatsApp application. For example, publish new announcements, check attendance before and after the class, update each student's learning status to their parents, etc. Resulting in time-consuming and adding additional tasks which could be automated by the management system. By developing a smart management system, this project not only improves the efficiency of the tuition centre operations but also enhances the educational experience for both students and tutors.

1.3 Project Objectives

The main goal of "Smart Management System for Tuition Centre Operations" is to develop an application to overcome the inefficiencies and inconveniences faced by small-sized tuition centres. This project identifies three key objectives such as to develop a user-friendly smart management system, to track biometric class attendance using facial recognition, and to enhance communication and transparency between parents and tutors. This system aims to enhance overall efficiency, streamline operations, and provide a cost-effective solution for tuition centre management.

i. To develop a user-friendly smart management system for tuition centre operations

The main objective of this project is to develop a user-friendly and comprehensive tuition centre management system that addresses the daily operational and administrative tasks of small-sized tuition centres that are handled manually. As the system develops, the traditional methods of administration, which use paper and pen, will be eliminated to encourage recycling, and all the data will be stored securely and transparently in the database. This system will be user-friendly to all the tuition centres in the market. The system will automate administrative tasks, improve communication between parents and tutors, improve educational experience for students and tutors, and enhance overall efficiency. Hence, tutors can focus on the teaching progress while students can focus on the learning process.

ii. To track biometric class attendance using facial recognition

This project aims to implement a robust attendance tracking system that utilises biometric face recognition technology. The biometric attendance tracking module addresses the problem of friend impersonation, resulting in inconsistent attendance performance. This system will also eliminate issues such as duplicate attendance taking and errors in manual recording. In the traditional method, tutors will take attendance one by one by calling their names before starting the class. If there are students who are late to class, the tutors will need to check the attendance once again before the end of the class. By automating attendance tracking, this module will save time, reduce paper usage, eliminate friend impersonation, and ensure that each student's presence is recorded reliably and efficiently.

iii. To enhance communication and transparency between parents and tutors

Another key objective is to create a platform of communication tools that will significantly improve communication and transparency between parents and tutors. Administrators will publish the latest announcements and real-time updates, ensuring that parents are always informed. Additionally, it will provide parents with tools to track their children's academic progress and directly communicate with tutors about their children's performance and behaviour in the class. This system will bridge the communication gap, increase engagement for parents, and ensure parents are fully informed about their children's learning activities and progress in the tuition centre.

1.4 Project Scope

The project scope encompasses the development of a “Smart Management System for Tuition Centre Operations”, which integrates multiple modules to streamline the administrative, academic, and learning tasks. Key modules include the user authentication module for role-based access control, the student management module for centralised student information, the course management module for courses organisation, the class management module for class scheduling and biometric attendance tracking, the fee management module will automate invoicing and payment reminders, the communication tools for enhancing the interaction between tutor and parents and the reporting and analytics module for providing valuable insights through performance and financial reports. These modules aim to create a comprehensive, user-friendly system that enhances the efficiency of operational and administrative tasks.

i. User authentication module

This system will be featured in a strong user authentication module with safe login, and it will have role-based access control. There are four different roles in this system, such as administrators, tutors, students, and parents. Each of the roles can only access relevant features and data to enhance security and ensure that sensitive information is protected. For example, the administrators can access and manage all the data in the system, tutors can only access and manage the students and courses that are assigned by the administrators, students can only access the courses that are registered, and parents can only manage their children’s profiles.

ii. Student management module

The project will include a comprehensive student management module that enables administrators and tutors to centrally manage all aspects of student information within a unified digital platform. This module allows for the effective management of students' personal information, such as contact information and emergency contacts. Besides, it will also ensure the security of enrolment records and class attendance histories and simplify the student registration process. By centralising the students' information, this module avoids duplicate data entry, decreases administrative errors, and allows for tailored learning experiences.

iii. Course management module

This module enables administrators to create new courses and manage existing courses by defining the learning objectives, duration, and price structures. Tutors are allowed to manage each course that is assigned by the administrator. For example, the tutor can create new class schedules. All the assessments will be managed within the system and provide a centralised platform for all academic activities. This dual-level management strategy ensures that administrators maintain supervision and strategic control, while tutors have the operational flexibility to provide personalised education.

iv. Class management module

The system enables tutors to create flexible class schedules by specifying dates, starting time, and ending time for their assigned courses. This module also allows for real-time synchronisation of class schedules and class information into administrators', students', parents', and tutors' calendars. Besides, this module is integrated with the advanced biometric facial recognition attendance tracking technology, which enables tutors and administrators to capture students' attendance easily throughout each class session. This biometric attendance tracking system replaces the manual roll call processes, reduces human error, prevents friends from impersonating, and provides real-time attendance data.

v. Fee management module

This module helps to automate and streamline the billing and payment process. Administrators are able to generate and manage the monthly student bills that include breakdowns of course fees, additional charges, and discounts. Besides, it also supports automated invoices and receipts generation, which makes invoices and receipts easily accessible by allowing users to download them in PDF format for record-keeping and offline reference. To improve communication efficiency, the module will send email notifications to parents immediately upon the bills being generated. This module also has the overdue payment reminder feature that automatically sends overdue reminders to parents seven days after the bill is generated, which helps in reducing manual follow-up efforts and improving fee collection rates. The module helps tuition centre manage their finances more effectively, reducing administrative tasks and ensuring transparency and timely payments.

vi. Communication module

The communication module eases communication between administrators, tutors, students, and parents. This module includes an integrated internal messaging system that allows for secure one-on-one contact between any users, which allows confidential discussions about specific academic concerns, administrative issues, or individualised feedback exchanges. Furthermore, the module also includes an advanced group messaging system that automatically creates course-specific communication groups by adding the administrators, respective assigned tutors, and all the respective registered students. This allows efficient broadcast communication, class-wide announcements, and collaborative discussions. This dual-messaging strategy, which integrates with email notification, is allowing for both private consultations and group coordination to take place on the same platforms, eliminating the need for external communication tools while keeping the conversations confidential and transparent.

vii. Reporting and analytics module

This project will include a comprehensive reporting and analytics module that provides valuable insights and performance data. This module gives administrators access to extensive financial data that breaks down revenue by grade level and specific course, which allows them to discover the most profitable academic offerings. Administrators can also view the tutors' and students' performance analytics, which include crucial indicators such as the number of courses each tutor teaches, workload distribution, and individual student attendance. Tutors are also granted access to their respective students' attendance performance analytics, parents have dedicated access to their children's attendance performance analytics, and students can check their own attendance performance analytics. These insights will enable transparent and data-driven decision-making, which enhances operational efficiency and prompts intervention for students with irregular attendance records.

1.5 Proposed Approach / Study

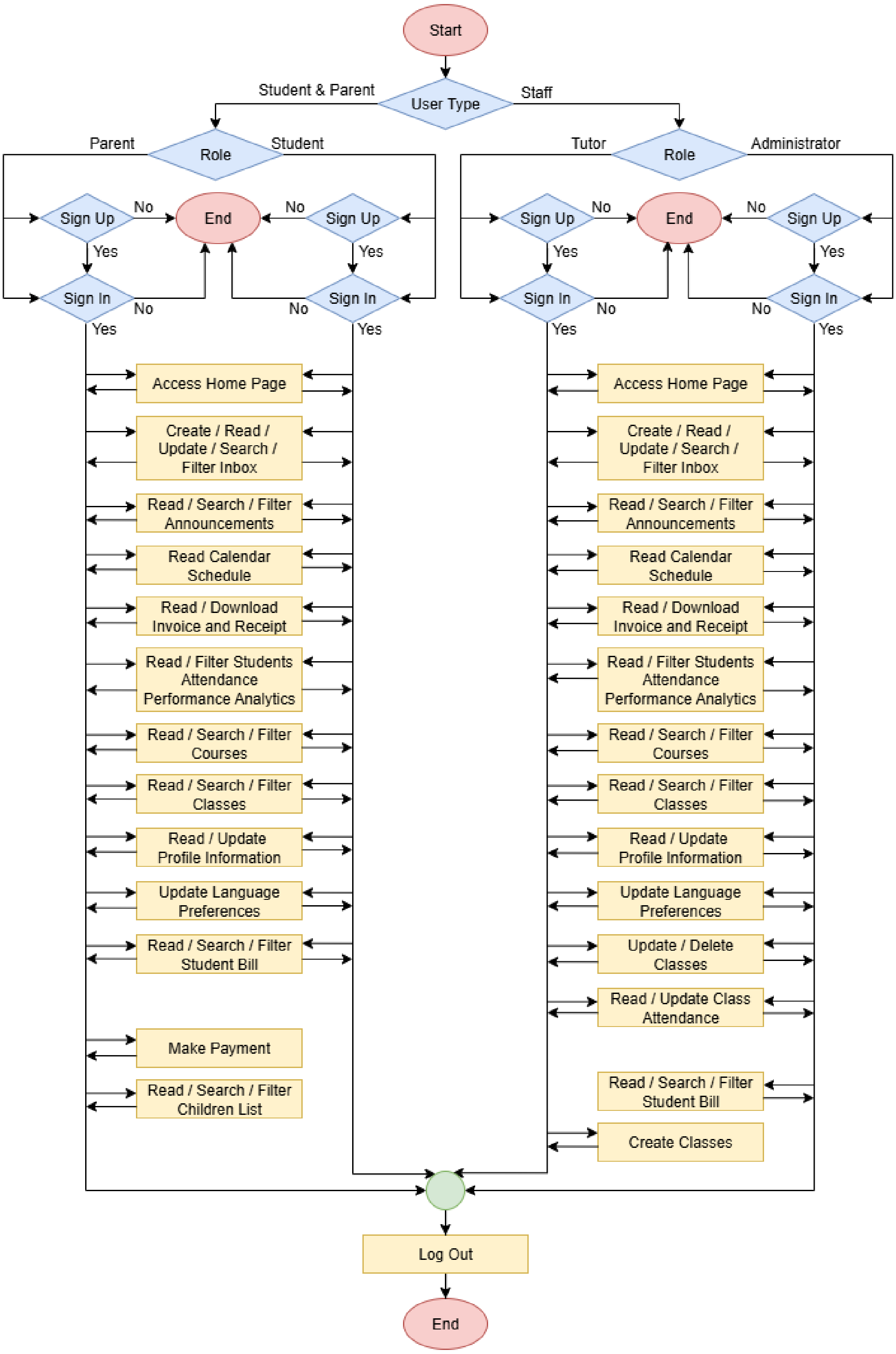

Start
User Type
Student & Parent
Staff
Role
Parent
Student
Role
Tutor
Administrator
Sign Up
No
Yes
End
Sign In
Yes
No
Access Home Page
Create / Read / Update / Search / Filter Inbox
Read / Search / Filter Announcements
Read Calendar Schedule
Read / Download Invoice and Receipt
Read / Filter Students Attendance Performance Analytics
Read / Search / Filter Courses
Read / Search / Filter Classes
Read / Update Profile Information
Update Language Preferences
Read / Search / Filter Student Bill
Make Payment
Read / Search / Filter Children List
Update / Delete Classes
Read / Update Class Attendance
Create Classes
Log Out
End

Figure 1.5.1 Flowchart of Smart Management System for Tuition Centre Operations

Figure 1.5.1 depicts the flowchart of the project “Smart Management System for Tuition Centre Operations”. The flowchart begins with a start point in the user authentication module, which branches into two main user categories, which are “Student & Parent” and “Staff”. Each category then splits further based on specific roles, including student, parent, tutor, and administrator. The authentication process is consistent across all roles. Users having the options of signing up or signing in, if the authentication attempts fail, then it will result in an “End” state. However, successful authentication allows users access to their distinctive home pages with role-specific features.

All roles have similar access privileges, which include the ability to access the home page, inbox communications, announcements, calendar schedules, download invoices and receipts, view attendance and performance analytics, search courses and classes, profile information, and language preference. In addition, tutors and administrators are allowed to manage classes and class attendance. Meanwhile, parents additionally have access to view their children's profile information lists and make payments to their children’s student bills, tutors can create new classes, and administrators can manage the student bills. Lastly, the system ends with a log-out function that ensures secure session termination for all roles.

1.6 Impact, Significance, and Contribution

The proposed “Smart Management System for Tuition Centre Operations” has a vital impact on small-sized tuition centres by addressing inefficiencies in conventional administrative and operational procedures. This project minimises human error and drastically cuts down on manual labour by integrating several key modules. These integrated modules allow tutors to allocate more time to teaching and allow students to concentrate on their academic development. Hence, this helps to improve and enhance the overall educational experience.

The significance of this project arises from its capacity to offer an extensive and cost-effective solution which tailored to the needs of small-sized tuition centres, which frequently lack access to reasonably priced management tools. By utilising this system, it bridges the gap between students, parents and tutors and fosters greater transparency and engagement. For example, parents can monitor their children’s progress conveniently through the system, get the latest updates and communicate with tutors directly. Tutors can focus on their teaching progress, students can prioritise their learning experiences, and administrators can handle their administrative tasks efficiently.

Besides, this project offers significant contributions to various stakeholders and the broader field of Information Technology. For researchers, it provides valuable insights and a reference point for similar studies in educational technology. Information Technology specialists can build upon the design and development strategies of this project to enhance functionalities in related systems. By introducing a cost-effective, comprehensive solution, this project could be developed into a new tool in the field of educational technology, which offers huge benefits to tuition centre owners by automating administrative and operational tasks and improving overall efficiency.

1.7 Report Organisation

The report is organised and divided into seven separate chapters. The first chapter is the introduction, which includes background information, problem statement, motivation, project objectives, project scope, proposed approach, impact, significance, contribution, report organisation, and summary. The second chapter is the literature review, which comprises discovering similar projects, reviewing existing systems, and a summary. The third chapter is the system methodology, which consists of methodology, system design, timeline, and a summary. The fourth chapter is the system design, which covers program development and a summary. The fifth chapter is the system implementation, which contains system requirements, settings and configurations, system operation, implementation issues and challenges, and a summary. The sixth chapter is the system evaluation and discussion, which outlines system testing, testing setup and results, objectives evaluation, and a summary. The last chapter is a conclusion and recommendations.

1.8 Summary

In conclusion, the "Smart Management System for Tuition Centre Operations" is designed to address the inefficiencies of small-sized tuition centres by automating administrative and operational tasks. In this fast-paced educational landscape, tuition centres play a crucial role in enhancing students' learning. However, many small-sized tuition centres still rely on conventional, outdated and error-prone manual processes like paper-based administrative tasks. This project highlighted three key problem statements and three primary project objectives by integrating seven essential modules. The proposed approach of this project is to employ a comprehensive role-based access control system, with each role having access to different features and functionalities while ensuring operational efficiency. The system's impact lies in reducing manual workload, minimising human error and improving efficiency. Its significance is in providing a cost-effective, tailored solution for small-sized tuition centres and fostering transparency and engagement. Lastly, this project also contributes to educational technology by providing insightful information for IT professionals and researchers.

Chapter 2

Literature Review

Chapter 2 consists of three different sections, which include similar projects, existing systems, and a summary. This chapter provides an in-depth analysis of the existing tuition centre management systems and similar projects on the Internet. The purpose of the analysis is to gain insightful information about the objectives, procedures, limitations, and recommendations for the future work of this project. Besides, this chapter aims to evaluate the strengths and weaknesses of the existing tuition centre management system.

2.1 Similar Projects

In this section, one similar project has been successfully analysed, which is "Automated Attendance System Based on Face Recognition Using OpenCV" [3]. This analysis offers valuable insights into leveraging biometric attendance-taking technology, which is facial recognition.

2.1.1 Automated Attendance System Based on Face Recognition Using OpenCV [3]

The proposed automated attendance system leverages image capturing, face recognising, pre-processing, database construction, and post-processing to streamline attendance tracking in educational settings. To ensure precise identification, the process starts with frontal images of the students, which are then followed by advanced face detection and pre-processing techniques. The system stores face models and attendance records in a MongoDB database. MongoDB makes it possible to retrieve data and generate reports quickly. The collected photos are refined by post-processing techniques to guarantee high facial recognition accuracy. The system also incorporates email and push notifications to enhance overall engagement and operational efficiency.

Figure 2.1.1.1 shows the attendance flow diagram using face recognition. The flow starts with student and teacher registration then followed by face registration. Once registered successfully, the face recognition algorithm starts. If the face is recognised by the algorithm, then the attendance will be marked and a report will be generated; else the user will be marked as absent, and it will report absentees.

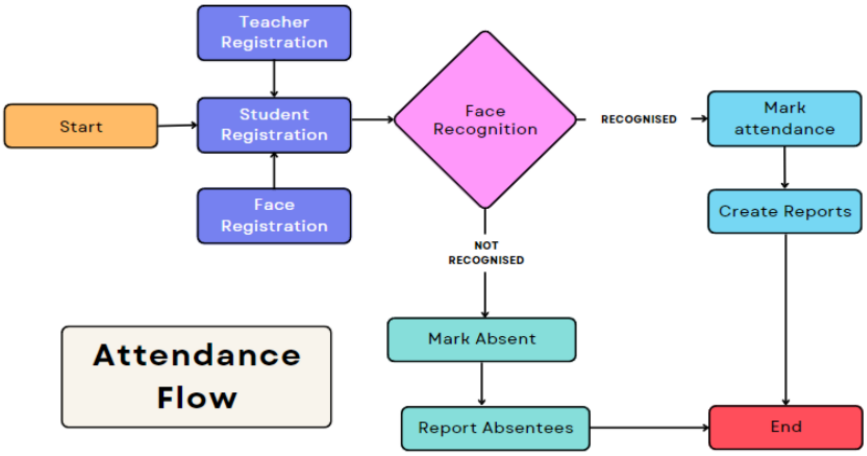


Figure 2.1.1.1 Screenshot of the attendance flow diagram [3]

i. Image capture

The automated attendance system begins by capturing the frontal images of students using a camera positioned directly in front of them. The camera is kept in focus, with adequate lighting to ensure distinct facial features. To have their whole face in the frame when taking the picture, students are encouraged to stand at least three feet away from the camera. After that, the photos are stored in the database for use in facial recognition and attendance tracking in the future.

ii. Face detection

For the face recognition system to be functional, a trustworthy face detection algorithm is essential. These algorithms could make use of facial geometry or adapt to changes in appearance due to ageing or other factors. Machine learning techniques can improve detection accuracy, as demonstrated in the provided attendance flow diagram, as mentioned in Figure 2.1.1.1. Face recognition is the first step in the process, after which either identified faces are marked as present or unidentified faces are marked as absent.

iii. Pre-processing

Pre-processing is done on images of faces to increase the accuracy of identification after they are recognised. Facial expression recognition follows a series of processes that include image enhancement, face matching, and comparison with stored data in the database. The pre-processing may break down the face into components like eyes, nose, and mouth. Each image is associated with a probability that indicates the likelihood of accurate detection based on the quality of pre-processed data.

iv. Database development

The system stores the face models and attendance information in a storage-level database known as MongoDB. The images are stored in JPEG format, which provides lossy compression to reduce the file size and ensure it can be retrieved quickly. Through data queries and one-to-one mapping between attendees and matching face models, the database also makes it possible for the application to display students' and teachers' profiles.

v. Post-processing

Post-processing techniques are employed to refine the captured images by eliminating extraneous elements like hats or sunglasses. Attendance reports are then produced by comparing detected faces with the database entries. These reports can be generated monthly or weekly and are shared with parents or guardians as needed. The system's accuracy improves with increased usage. The notifications are sent to users via email and push notifications to keep staff and students informed in real time.

2.2 Existing Systems

In this section, three existing systems have completed reviews, including MCPlus, iKEY, and Prime Tuition. Since the objective of this project is to build a mobile application, all the existing systems are reviewed for their mobile versions.

2.2.1 **MCPlus** [4]

MCPlus, formerly known as The Maths Clinic, is one of the biggest tuition centres in Klang Valley, Selangor, with the highest ranked [4]. MCPlus provides a web application and a mobile application for students. First, if the student did not have an account before, they are requested to sign up for a new account before using it. After logging in to the account, the student will be directed to the student profile, and the navigation bar will pop up on the screen's left side. In the navigation bar, the student can choose to perform any of the features. For example, home, enter class, timetable, feedback, payment history, notes bank, etc, as shown in Figure 2.2.1.1.

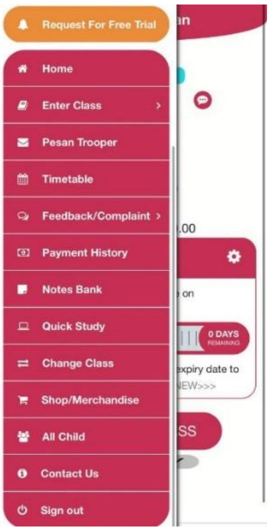
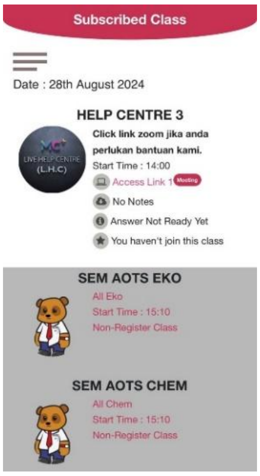


Figure 2.2.1.1 Screenshot of the navigation bar in the mobile application

When the student clicks on the button "Enter Class", there will be a drop-down list that contains the subscription class, replay video, and replay transaction. In the subscription class, the student can check all the classes that are subscribed to, as shown in Figure 2.2.1.2. The student can also click on the replay video, then they can check all the recorded videos that are categorised by date. A timetable that lists all available subjects in chronological order is also provided to allow the student user to check their schedule, and the student can download it for the full schedule, according to Figure 2.2.1.3. In Figure 2.2.1.4, the students can check their billing information, which includes payment and order history.



Subscribed Class

Date : 28th August 2024

HELP CENTRE 3

Click link zoom jika anda perlukan bantuan kami.
Start Time : 14:00
Access Link 1 Meeting
No Notes
Answer Not Ready Yet
You haven't join this class

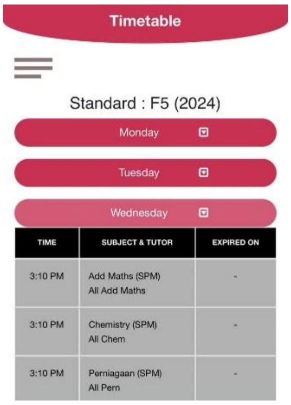
SEM AOTS EKO

All Eko
Start Time : 15:10
Non-Register Class

SEM AOTS CHEM

All Chem
Start Time : 15:10
Non-Register Class

Figure 2.2.1.2 Screenshot of the subscribed class

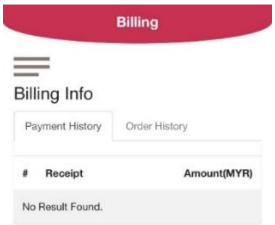


Timetable

Standard : F5 (2024)

TIME	SUBJECT & TUTOR	EXPIRED ON
3:10 PM	Add Maths (SPM) All Add Maths	-
3:10 PM	Chemistry (SPM) All Chem	-
3:10 PM	Perniagaan (SPM) All Pern	-

Figure 2.2.1.3 Screenshot of Timetable



Billing

Billing Info

#	Receipt	Amount(MYR)
No Result Found.		

Figure 2.2.1.4 Screenshot of Billing Info

Inside the notes bank, there are a vast number of notes provided according to each subject and tutor. Hence, the student can assess, view, and download the notes provided by the tutor in every class through Google Drive to ease revision, as shown in Figure 2.2.1.5 and Figure 2.2.1.6. Furthermore, the "Pesan Trooper" will publish all announcements posted by the tutor, such as the quote of the day. MCPlus allows the students to submit their feedback or complaints through their mobile application so that all the problems can be solved in a short time and satisfy all the students.

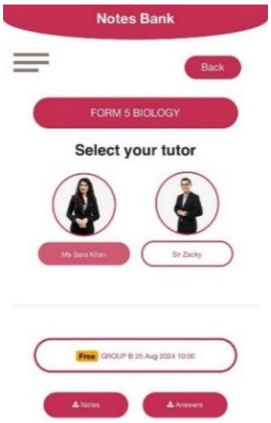


Figure 2.2.1.5 Screenshot of Notes Bank

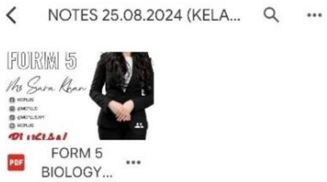
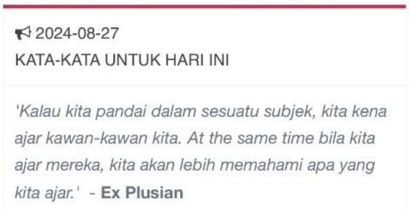


Figure 2.2.1.6 Screenshot of notes available in Google Drive

The pros of MCPlus are that it offers a well-designed visualisation and layout with a colourful and attractive interface for the student. This can help to enhance the student experience and be visually appealing. Besides, MCPlus allows the student to access and download all the notes in Google Drive for revision purposes and to streamline their learning experience. It also enables the students to track all their billing information and billing transactions with transparency. Additionally, a comprehensive timetable is also a highlight of MCPlus, which can help students plan and manage their study schedules efficiently and effectively.

However, there are a few drawbacks to the MCPlus mobile application, which is that the app suffers from a mixed language interface and affects the student who prefers consistency in language settings, according to Figure 2.2.1.7. Moreover, the mobile application does not include the attendance-taking feature, which allows the student to take their attendance, hence causing the tutor to fail to track student participation effectively. The mobile application may not fit well on certain devices, such as the iPhone 13 Pro, and will make it harder to operate. Lastly, the subscription for the application to use all the features may be deterred by the student, and the absence of the chat room feature makes it difficult for students to communicate with their tutor.



2024-08-27
KATA-KATA UNTUK HARI INI

'Kalau kita pandai dalam sesuatu subjek, kita kena ajar kawan-kawan kita. At the same time bila kita ajar mereka, kita akan lebih memahami apa yang kita ajar.' - **Ex Plusian**

Figure 2.2.1.7 Screenshot of mixed language interface

2.2.2 **iKEY** [5]

iKEY is a virtual assistant for an education mobile application that is available in the Google Play Store, Huawei App Gallery, and App Store. iKEY is a free mobile application designed for tutors, parents, and students to monitor educational progress. Tutors will use the educator mobile application, named “iKEY Educator”, which allows tutors to improve their teaching operation effectively and efficiently, while students and parents will use the application named “iKEY - Education & Family”, which will help students and parents to centralise all information and materials [5]. Upon logging in successfully to iKEY, the student and parents will be directed to the homepage. They are allowed to assess features such as albums, billing invoices, student attendance, announcements, applying for leave, etc, as shown in Figure 2.2.2.1. The tutor will be directed to the homepage, which contains features that are different from those of the students, such as announcements, leave requests, tutor attendance, etc, once they successfully log in to the iKEY Educator according to Figure 2.2.2.2.

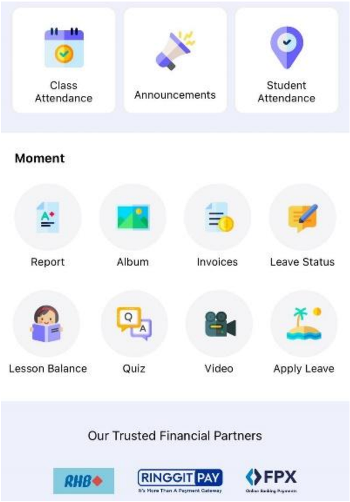


Figure 2.2.2.1 Screenshot of the homepage for iKEY - Education & Family

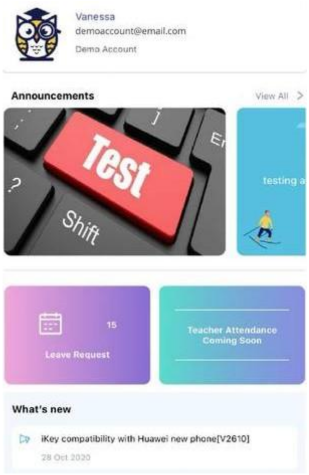
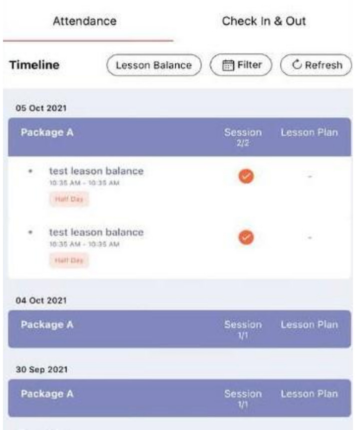


Figure 2.2.2.2 Screenshot of the homepage for iKEY Educator

In the iKEY - Education & Family applications, students and parents are authorised to check the attendance for each class, and they can filter it by lesson balance or date as depicted in Figure 2.2.2.3. iKEY enables parents to check their children's check-in and check-out time and their body temperature as illustrated in Figure 2.2.2.4. Moreover, parents can check the monthly report and the progress report card. The tutor will leave some comments, advice, or improvements for each student on the report card. Hence, parents can always keep track of their children's progress as represented in Figure 2.2.2.5. Payments can be made directly through the mobile application as it collaborates with trusted financial partners such as RHB Banking, RinggitPay payment gateway, and Financio Cloud Accounting. After making a payment, it will auto-generate an invoice and receipt for each transaction as shown in Figure 2.2.2.6.



Timeline

05 Oct 2021

Package A | Session 2/2 | Lesson Plan

- test leason balance 10:35 AM - 10:35 AM Half Day ✓ -
- test leason balance 10:35 AM - 10:35 AM Half Day ✓ -

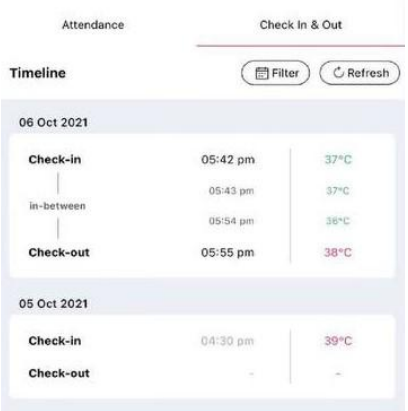
04 Oct 2021

Package A | Session 1/1 | Lesson Plan

30 Sep 2021

Package A | Session 1/1 | Lesson Plan

Figure 2.2.2.3 Screenshot of the attendance



Timeline

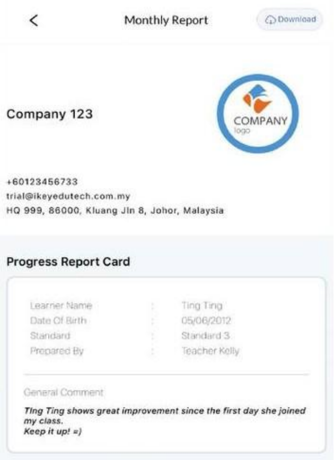
06 Oct 2021

Check-in	05:42 pm	37°C
in-between	05:43 pm	37°C
	05:54 pm	36°C
Check-out	05:55 pm	38°C

05 Oct 2021

Check-in	04:30 pm	39°C
Check-out	-	-

Figure 2.2.2.4 Screenshot of the check-in & check-out



Company 123

+60123456733
trial@ikeyedutech.com.my
HQ 999, 86000, Kluang Jln 8, Johor, Malaysia

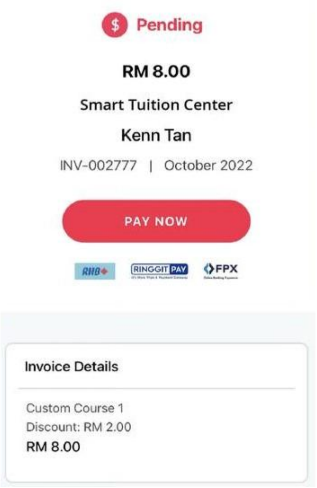
Progress Report Card

Learner Name	:	Ting Ting
Date Of Birth	:	05/06/2012
Standard	:	Standard 3
Prepared By	:	Teacher Kelly

General Comment

Ting Ting shows great improvement since the first day she joined my class.
Keep it up! =)

Figure 2.2.2.5 Screenshot of the monthly report



Pending

RM 8.00

Smart Tuition Center

Kenn Tan

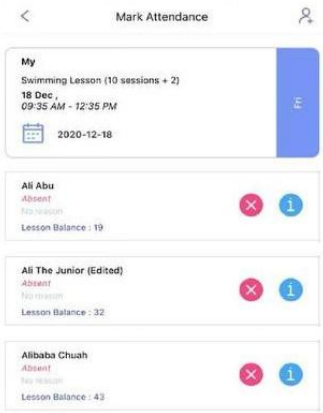
INV-002777 | October 2022

Invoice Details

Custom Course 1
Discount: RM 2.00
RM 8.00

Figure 2.2.2.6 Screenshot of the payment

On the other hand, for the iKEY Educator application, tutors need to mark attendance manually for each class as illustrated in Figure 2.2.2.7. After marking attendance, tutors can check all the attendance records in the attendance history. For example, there are a total of 4 students registered for the swimming lesson, and only 3 students are present for the lesson on 19 September, 09:35 am, as depicted in Figure 2.2.2.8. Besides, the tutor can check all the in-charge classes, and it is arranged chronologically. All details of the class are displayed, such as course name, time, number of students, and the latest date, as shown in Figure 2.2.2.9.



Mark Attendance

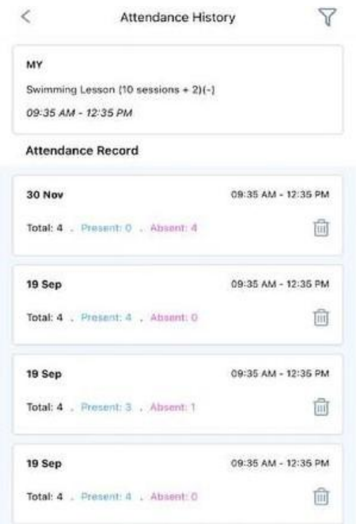
My
Swimming Lesson (10 sessions + 2)
18 Dec ,
09:35 AM - 12:35 PM
2020-12-18

Ali Abu
Absent
No reason
Lesson Balance : 19

Ali The Junior (Edited)
Absent
No reason
Lesson Balance : 32

Alibaba Chuah
Absent
No reason
Lesson Balance : 43

Figure 2.3.2.7 Screenshot of the attendance



Attendance History

MY

Swimming Lesson (10 sessions + 2)(-)

09:35 AM - 12:35 PM

Attendance Record

30 Nov 09:35 AM - 12:35 PM

Total: 4 . Present: 0 . Absent: 4

19 Sep 09:35 AM - 12:35 PM

Total: 4 . Present: 4 . Absent: 0

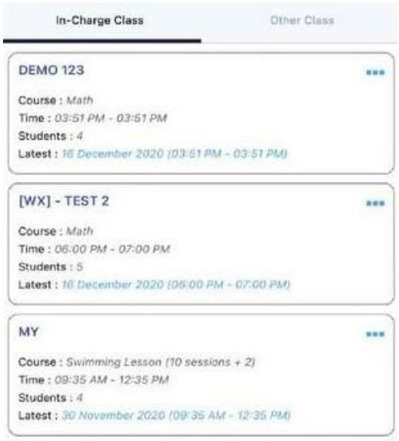
19 Sep 09:35 AM - 12:35 PM

Total: 4 . Present: 3 . Absent: 1

19 Sep 09:35 AM - 12:35 PM

Total: 4 . Present: 4 . Absent: 0

Figure 2.3.2.8 Screenshot of the attendance history



DEMO 123

Course : Math
Time : 03:51 PM - 03:51 PM
Students : 4
Latest : 16 December 2020 (03:51 PM - 03:51 PM)

[WX] - TEST 2

Course : Math
Time : 06:00 PM - 07:00 PM
Students : 5
Latest : 16 December 2020 (06:00 PM - 07:00 PM)

MY

Course : Swimming Lesson (10 sessions + 2)
Time : 09:35 AM - 12:35 PM
Students : 4
Latest : 30 November 2020 (09:35 AM - 12:35 PM)

Figure 2.3.2.9 Screenshot of the class in charge

The advantages of iKEY are free for all tutors, students, and parents; it provides an attractive interface with a lot of colorful elements and a neat layout for students and parents in iKEY - Education & Family. A gallery that can capture and upload the learning journey photos and send them to parents is one of the highlights of this application. Besides, iKEY collaborates with a few trusted financial partners so parents can directly make payments in the app securely, and all the transactions and receipts will be stored in the application.

The disadvantage of iKEY is that it is provided for two applications; one is for students and parents, while the second is for tutors. It will cause trouble for some tutors who also serve as parents at the same time, as they are forced to install two applications on their phones. The interface for iKEY Educator is null and boring due to a lack of elements and color. Lastly, it also needs the tutor to mark the attendance manually for every class, which results in inefficiency and a waste of time.

2.2.3 **Prime Tuition** [6]

Prime Tuition is an education management mobile application that allows parents to monitor their children's performance in the tuition centre [6]. Prime Tuition offers different types of features to ease parents keep track of their children's education progress. The application is designed with a navigation bar at the bottom of the screen that consists of home, receipt, fees, analytics, and profile, as demonstrated in Figure 2.2.3.1. Besides, it also has a sidebar such as a dashboard, my profile, view of children, pay fees, compensation, leave applications, etc, as portrayed in Figure 2.2.3.2.

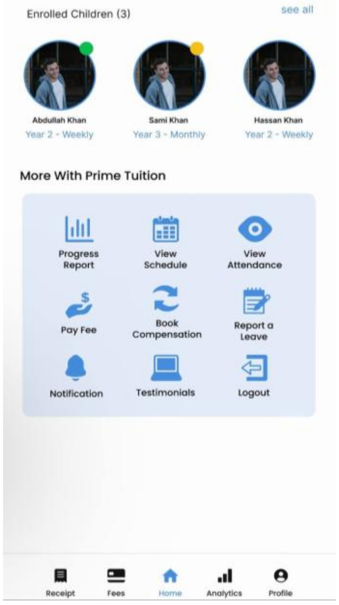


Figure 2.2.3.1 Screenshot of the homepage

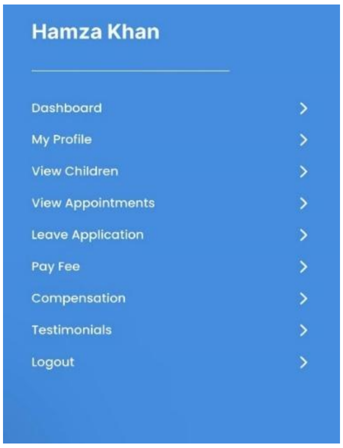
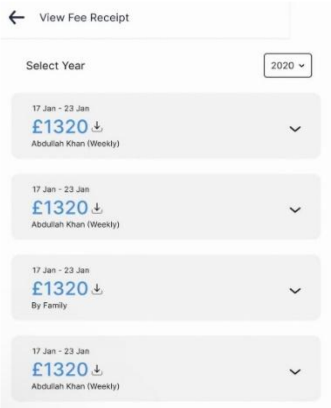


Figure 2.2.3.2 Screenshot of the sidebar

First and foremost, parents need to log in to the mobile application, and then they will be directed to the homepage. Parents can assess the progress report, view the schedule, view the attendance, pay fees, book compensation, report a leave, etc, via the homepage. Parents can make the payment directly and securely, and all the receipts will be stored and displayed in the application. Parents can download the fee receipt with just one click or review the previous fee receipts according to Figure 2.2.3.3. Parents can also apply for a leave application, such as medical leave or non-medical leave, on behalf of their children, as pictured in Figure 2.2.3.4.


View Fee Receipt

Select Year 2020

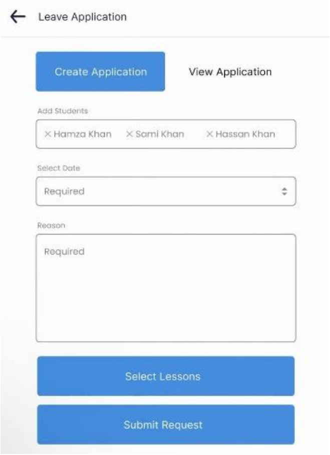
17 Jan - 23 Jan
£1320
Abdullah Khan (Weekly)

17 Jan - 23 Jan
£1320
Abdullah Khan (Weekly)

17 Jan - 23 Jan
£1320
By Family

17 Jan - 23 Jan
£1320
Abdullah Khan (Weekly)

Figure 2.2.3.3 Screenshot of the fee receipt



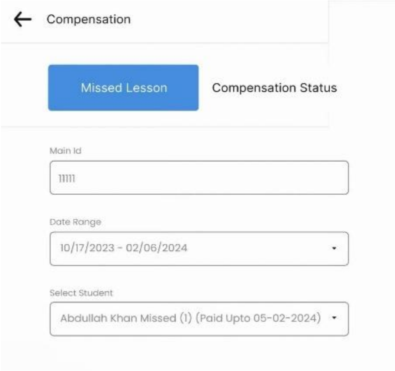
Add Students: × Hamza Khan × Sami Khan × Hassan Khan

Select Date: Required

Reason: Required

Figure 2.2.3.4 Screenshot of the leave application

Furthermore, parents can also claim compensation from the management if their children miss any lessons. They are requested to fill in the compensation form with the date range and student name, and then wait for the management to verify and handle it. Parents can also check for the compensation status to get the latest updates, as illustrated in Figure 2.2.3.5. Moreover, parents can check their children's attendance with the details of how many times they are absent, leave, or present for the class. Total fees and outstanding fees are also displayed under the analytics tab to notify parents of the unpaid fees and avoid overdue and extra charges, as mentioned in Figure 2.2.3.6.

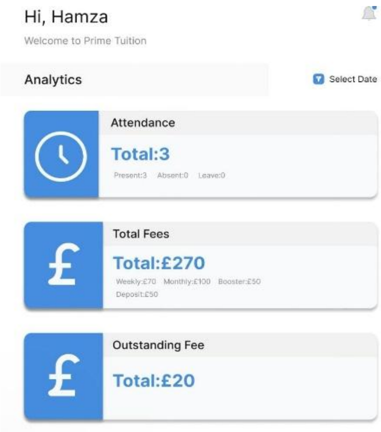


Main Id: 11111

Date Range: 10/17/2023 - 02/06/2024

Select Student: Abdullah Khan Missed (1) (Paid Upto 05-02-2024)

Figure 2.2.3.5 Screenshot of the compensation form



Hi, Hamza

Welcome to Prime Tuition

Attendance

Total:3

Present:3 Absent:0 Leave:0

Total Fees

Total:£270

Weekly:£70 Monthly:£100 Booster:£50 Deposit:£50

Outstanding Fee

Total:£20

Figure 2.2.3.6 Screenshot of the analytics

Prime tuition application offers several benefits for parents, such as parents can claim compensation for missed lessons if their children apply for leave. Hence, parents will not waste their money. Besides, it allows parents to register for more than one child in the application at the same time, making it easier for parents to check their children's performance. For example, if the parents have three children who also study in this tuition centre, then the parents can log in to their children's profiles simultaneously to check their attendance or make a one-time payment.

However, the drawback of this application is the dull interface, and it is not attractive to parents. This is due to this application only using blue and white colors, which fail to catch the users' eyes. Next, it also does not provide any chat room for parents if they or their children are facing any academic problems or confusion. Last but not least, the Prime tuition application does not authorize tutors to upload their latest teaching materials, notes, or recorded videos to the application. Therefore, students might encounter difficulties when reviewing.

2.3 Summary

This chapter provides an in-depth analysis of similar projects and existing applications. The reviewed paper, "Automated Attendance System Based on Face Recognition Using OpenCV [3]", proposed an automated attendance system that utilises advanced face recognition technology to streamline attendance tracking in educational environments. It involves several key steps, which include capturing frontal images of students, detecting faces using reliable algorithms, enhancing image accuracy through pre-processing techniques, utilizing a storage-level database for storing the face images, and refining the images by eliminating unwanted elements. The system stores face models and attendance records in a MongoDB database, which enables fast data retrieval and report generation. This approach significantly enhances the efficiency and accuracy of attendance management, providing a modern solution for educational institutions.

Besides, the existing tuition centre management applications that have been reviewed are MCPlus, iKEY, and Prime Tuition. The review focuses on mobile versions of these systems by examining their features, strengths, and limitations. The strengths and limitations are concluded and summarised in Table 2.3.1.

Table 2.3.1 Comparison of reviewed systems

Functions	**MCPlus**	**iKEY**	**Prime Tuition**
Mobile Application	Yes	Yes	Yes
Timetable Feature	Yes	No	Yes
Take Attendance Manually Feature	No	Yes	No
Attendance Record Feature	No	Yes	Yes
Make Payment Feature	No	Yes	Yes
Payment Receipt Feature	Yes	Yes	Yes
Teaching Materials Feature	Yes	No	No
Replay Recorded Videos Feature	Yes	No	No
Feedback Feature	Yes	No	No
Course List Feature	Yes	Yes	No
Albums Feature	No	Yes	No
Announcements Feature	Yes	Yes	Yes
Leave Application Feature	No	Yes	Yes
Monthly Report Feature	No	Yes	Yes
Separated into Tutor and Student Applications	No	Yes	No
Log in for more than one child at the same time	No	Yes	Yes

Chapter 3
System Methodology / Approach

This chapter consists of four different sections. The sections are system methodology, system design, timeline, and a summary. The system design includes a use case diagram, various use case descriptions, and an activity diagram for administrators, tutors, students, and parents. A Gantt Chart is also included in the timeline.

3.1 Methodology

Software development teams utilise a process named Software Development Life Cycle (SDLC) to design, develop, test, and develop new software [7]. It offers a methodical approach to software development, guaranteeing excellence and effectiveness all along the way. In this project, "Smart Management System for Tuition Centre Operations", Rapid Application Development (RAD) methodology is used to develop this system. RAD is an agile software development process that emphasises quick and iterative releases of prototypes, enabling rapid feedback and refinement of the system [8]. By using RAD, the development of the system can be accelerated with frequent iterations, ensuring that the final product meets the needs. According to Figure 3.1.1, RAD has four basic steps, which are defining the requirements, prototyping, construction, and deployment.

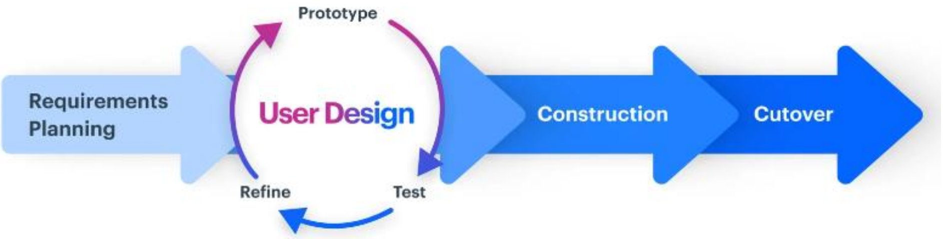


Figure 3.1.1 Screenshot of Rapid Application Development (RAD) [8]

i. Define the Requirements

Collecting and defining the system requirements is the first step in the RAD process. This stage focuses on understanding the specific requirements of tuition centres, such as user authentication, student management, course management, class management, fee management, communication, and reporting and analytics. This project gathered requirements through direct consultations with potential users, including tuition centre owners, administrators, tutors, students, and parents. The goal is to ensure that the system's features address the key problems identified in Chapter 1.

ii. Prototyping

In the prototype phase, the initial versions of the system are developed using the key modules and presented to stakeholders for feedback. Prototypes for key features such as user authentication, student management, course management, class management, fee management, communication, and reporting and analytics are built using Flutter with the Dart programming language, which emphasises user interface and functionality. These prototypes allow stakeholders to engage with the system early in the development phase, so they can offer insightful feedback that can be used to refine and optimize the system in subsequent iterations.

iii. Construction

Once the prototypes are validated, the construction phase involves building the entire system after the prototypes are approved and validated. In this phase, all modules will be integrated, and the back end will be developed with Python, Node.js, and Firebase. Additionally, the face_recognition library will be used to implement facial recognition for attendance tracking. Each module is created, tested, and integrated into the overall system. The construction phase focuses on ensuring that the system is reliable, scalable, and ready for deployment.

iv. Deployment

The final phase of RAD is deployment, where the system is made usable and available for use in real-world environments. The deployment process for the tuition management system involves setting up the system on a server, granting users access, and educating the users of the system. In this stage, the system is regularly reviewed to make sure it operates well and continues to meet the tuition centres' demands, and any final adjustments will be made based on user feedback.

3.2 System Design

The system design in this project consists of a use case diagram and several use case descriptions. In Unified Modeling Language (UML), a use case diagram is an illustration that shows the interactions between actors and the system. The actors include the administrator, tutor, student, and parent. Furthermore, a use case description offers a detailed textual explanation of each use case.

3.2.1 Use Case Diagram

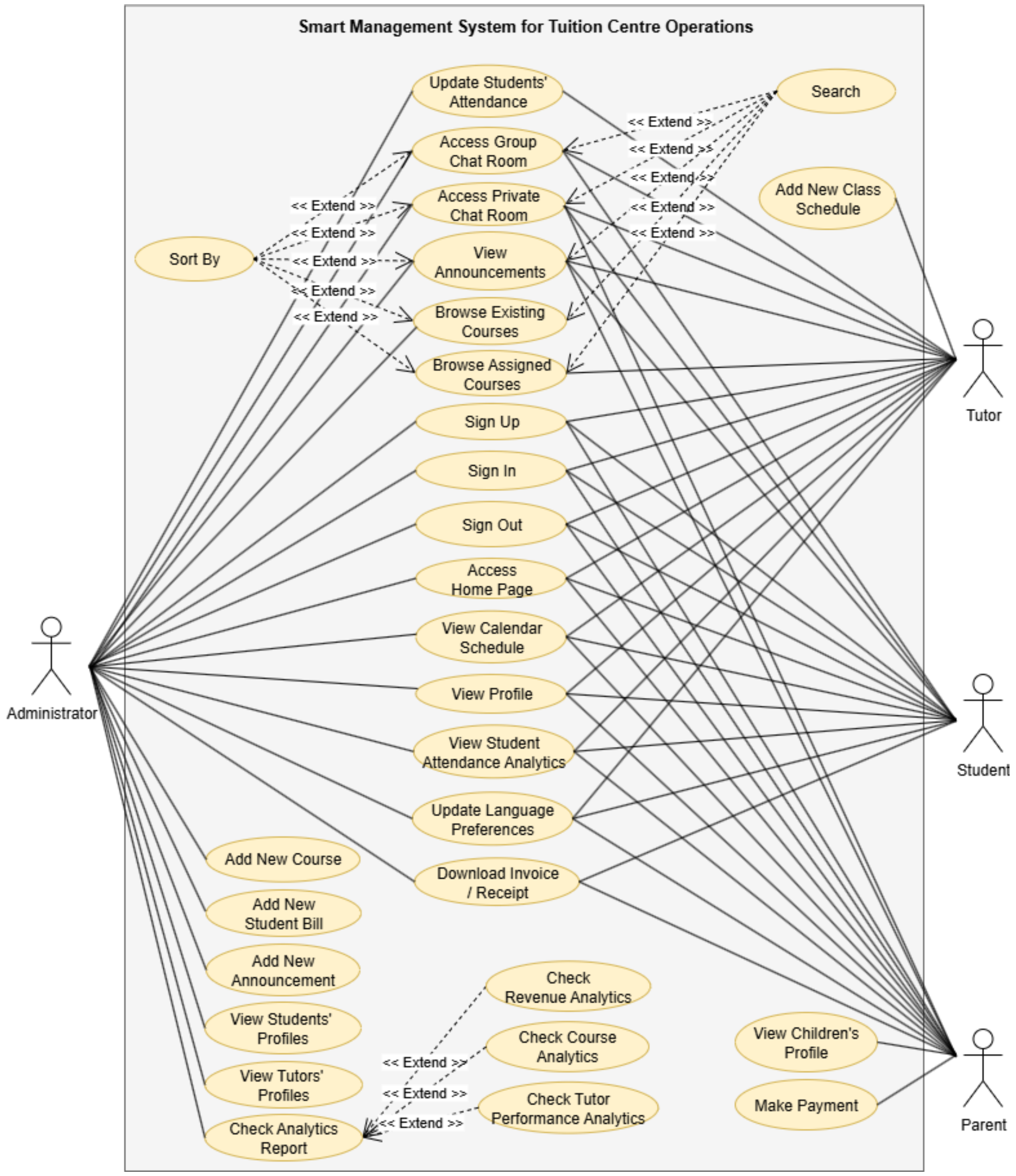


Figure 3.2.1 Use Case Diagram of Smart Management System for Tuition Centre Operations

3.2.2 Use Case Description

Table 3.2.2.1 Sign Up Use Case Description

<table>
<tr><td><b>Use Case Name:</b> Sign Up</td><td><b>ID:</b> 001</td><td><b>Importance Level:</b> High</td></tr>
<tr><td><b>Primary Actor:</b>
Administrator, Tutor, Student, Parent</td><td colspan="2"><b>Use Case Type:</b> Detail, Essential</td></tr>
<tr><td colspan="3"><b>Stakeholders and Interests:</b>
Administrator, Tutor, Student, Parent – wants to create a new account</td></tr>
<tr><td colspan="3"><b>Brief Description:</b> This use case describes how the administrator, tutor, student, or parent can create a new account in the system</td></tr>
<tr><td colspan="3"><b>Trigger:</b> Administrator, tutor, student, or parent wants to use the system but does not have an account

<b>Type:</b> External</td></tr>
<tr><td colspan="3"><b>Relationships:</b>
<b>Association:</b> Administrator, Tutor, Student, Parent
<b>Include:</b>
<b>Extend:</b>
<b>Generalization:</b></td></tr>
<tr><td colspan="3"><b>Normal Flow of Events:</b>
1. Administrator, tutor, student, or parent accesses the landing page.
2. Administrator, tutor, student, or parent selects their roles.
3. Administrator, tutor, student, or parent selects the “Sign Up” option.
4. Administrator, tutor, student, or parent enters their personal information, such as full name, gender, email address, password, and confirm password.
5. Administrator, tutor, student, or parent clicks the “Submit” button to verify their information.
6. Administrator, tutor, student, or parent successfully creates an account.</td></tr>
<tr><td colspan="3"><b>Sub Flows:</b>
1a. The system will display two different roles, which are “Student & Parent” and “Staff”.
2a. The system will display two options, which are “Sign In” and “Sign Up”.
5a. The system will verify their email address and password against their database.</td></tr>
</table>

6a. The system will display the message "Sign up successfully!".
Alternate / Exceptional Flows: 5a. If the administrator, tutor, student, or parent leaves any of the required fields empty, then the system will highlight the empty field and display the message "Please enter your information". 5b. If the entered password and the confirm password do not match, then the system will display the message "Passwords do not match". 5c. If the length of the password is less than 6 characters, then the system will display the message "Password must be at least 6 characters". 5d. If the email account already exists in the system's database, then the system will display the message "The email address is already in use by another account".

Table 3.2.2.2 Sign In Use Case Description

Use Case Name: Sign In	**ID:** 002	**Importance Level:** High
Primary Actor: Administrator, Tutor, Student, Parent	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator, Tutor, Student, Parent – wants to sign in to their respective account		
Brief Description: This use case describes how the administrator, tutor, student, or parent can sign in to their respective account in the system		
Trigger: Administrator, tutor, student, or parent wants to sign in to their respective account to use the system **Type:** External		
Relationships: **Association:** Administrator, Tutor, Student, Parent **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator, tutor, student, or parent accesses the landing page. 2. Administrator, tutor, student, or parent selects their roles. 3. Administrator, tutor, student, or parent selects the "Sign In" option.		

4. Administrator, tutor, student, or parent enters their personal information, such as email address and password. 5. Administrator, tutor, student, or parent clicks the "Submit" button to verify their information. 6. Administrator, tutor, student, or parent successfully signs in to their account.
Sub Flows: 1a. The system will display two different roles, which are "Student & Parent" and "Staff". 2a. The system will display two options, which are "Sign In" and "Sign Up". 5a. The system will verify their email address and password against their database. 6a. The system will display the message "Sign in successfully!".
Alternate / Exceptional Flows: 5a. If the administrator, tutor, student, or parent leaves any of the required fields empty, then the system will highlight the empty field and display the message "Please enter your information". 5b. If the email address does not exist in the database, then the system will display the message "The supplied auth credential is incorrect, malformed or has expired". 5c. If the password entered is incorrect, then the system will display the message "The supplied auth credential is incorrect, malformed or has expired". 5d. If the role selected is incorrect, then the system will display the message "Role mismatch. Please sign in using the correct tab".

Table 3.2.2.3 Sign Out Use Case Description

Use Case Name: Sign Out	**ID:** 003	**Importance Level:** High
Primary Actor: Administrator, Tutor, Student, Parent	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator, Tutor, Student, Parent – wants to sign out their respective account		
Brief Description: This use case describes how the administrator, tutor, student, or parent can sign out their respective account		
Trigger: Administrator, tutor, student, or parent wants to sign out their respective account **Type:** External		

Relationships: **Association:** Administrator, Tutor, Student, Parent **Include:** **Extend:** **Generalization:**
Normal Flow of Events: 1. Administrator, tutor, student, or parent accesses the profile page. 2. Administrator, tutor, student, or parent clicks the “gear” icon at the top right corner of the screen. 3. Administrator, tutor, student, or parent clicks the “Log Out” button. 4. Administrator, tutor, student, or parent successfully signs out of their account.
Sub Flows: Not applicable
Alternate / Exceptional Flows: Not applicable

Table 3.2.2.4 Access Home Page Use Case Description

Use Case Name: Access Home Page	**ID:** 004	**Importance Level:** High
Primary Actor: Administrator, Tutor, Student, Parent	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator, Tutor, Student, Parent – wants to view their home page		
Brief Description: This use case describes how the administrator, tutor, student, or parent can view their home page		
Trigger: Administrator, tutor, student, or parent wants to access their home page in the system **Type:** External		
Relationships: **Association:** Administrator, Tutor, Student, Parent **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator, tutor, student, or parent accesses the home page.		

2. Administrator, tutor, student, or parent can view the latest announcements at the top of the home page by scrolling left or right. 3. Administrator, tutor, student, or parent can view the calendar schedule at the bottom of the home page by clicking the date. 4. Administrator, tutor, student, or parent can click any of the announcements to view the details of the respective announcements. 5. Administrator, tutor, student, or parent can click any of the schedules to view the details of the respective schedules.
Sub Flows: 1a. The system will retrieve all the information from the database and display it here.
Alternate / Exceptional Flows: 2a. If there is no announcement created, then it will display the text “No announcements available”. 3a. If the data of the respective date in the calendar schedule is empty, then it will display “No schedules available for this date”.

Table 3.2.2.5 View Calendar Schedule Use Case Description

Use Case Name: View Calendar Schedule	**ID:** 0005	**Importance Level:** High
Primary Actor: Administrator, Tutor, Student, Parent	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator, Tutor, Student, Parent – wants to view the calendar schedule		
Brief Description: This use case describes how the administrator, tutor, student, or parent can view their calendar schedule		
Trigger: Administrator, tutor, student, or parent wants to access their calendar schedule in the system **Type:** External		
Relationships: **Association:** Administrator, Tutor, Student, Parent **Include:** **Extend:** **Generalization:**		

Normal Flow of Events: 1. Administrator, tutor, student, or parent accesses the calendar page. 2. Administrator, tutor, student, or parent can view their schedule for the current date. 3. Administrator, tutor, student, or parent can view the schedule for another date by choosing the new date in the pop-up calendar. 4. Administrator, tutor, student, or parent can click the respective schedule to view the course details of the schedule, such as course name, course description, day, time, etc.
Sub Flows: 1a. The system will retrieve all the information from the database and display it here.
Alternate / Exceptional Flows: Not applicable

Table 3.2.2.6 View Profile Use Case Description

Use Case Name: View Profile	**ID:** 006	**Importance Level:** High
Primary Actor: Administrator, Tutor, Student, Parent	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator, Tutor, Student, Parent – wants to view their profile page		
Brief Description: This use case describes how the administrator, tutor, student, or parent can view their profile page		
Trigger: Administrator, tutor, student, or parent wants to access their profile page to view their personal information **Type:** External		
Relationships: **Association:** Administrator, Tutor, Student, Parent **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator, tutor, student, or parent accesses the profile page. 2. Administrator, tutor, student, or parent can view all their personal information, such as profile photo, name, gender, age, date of birth, phone number, email address, etc.		

Sub Flows: 1a. The system will retrieve all the information from the database and display it here.
Alternate / Exceptional Flows: 2a. If the data in the corresponding field of the database is empty, then the system will display “None” in the corresponding field.

Table 3.2.2.7 View Student Attendance Analytics Use Case Description

Use Case Name: View Student Attendance Analytics	**ID:** 007	**Importance Level:** High
Primary Actor: Administrator, Tutor, Student, Parent	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator, Tutor, Student, Parent – wants to view the student attendance analytics		
Brief Description: This use case describes how the administrator, tutor, student, or parent can view the student attendance analytics		
Trigger: Administrator, tutor, student, or parent wants to access the student attendance analytics page to check their performance **Type:** External		
Relationships: **Association:** Administrator, Tutor, Student, Parent **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator, tutor, student, or parent accesses the student attendance analytics page. 2. Administrator, tutor, student, or parent can view all the attendance analytics records, which are displayed in the horizontal bar chart. 3. Administrator, tutor, student, or parent can filter the attendance analytics records by selecting the date. 4. Administrator, tutor, student, or parent can click the respective course bar in the bar chart to view the details of the respective course.		

Sub Flows: 1a. The system will retrieve all the information from the database and display it here.
Alternate / Exceptional Flows: 2a. If the data of the corresponding month in the database is empty, then the system will display the text “No courses found for the selected month”.

Table 3.2.2.8 Update Language Preferences Use Case Description

Use Case Name: Update Language Preferences	**ID:** 008	**Importance Level:** High
Primary Actor: Administrator, Tutor, Student, Parent	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator, Tutor, Student, Parent – wants to update the language preferences		
Brief Description: This use case describes how the administrator, tutor, student, or parent can update the language preferences		
Trigger: Administrator, tutor, student, or parent wants to update the language preferences of the system **Type:** External		
Relationships: **Association:** Administrator, Tutor, Student, Parent **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator, tutor, student, or parent accesses the profile page. 2. Administrator, tutor, student, or parent clicks the “gear” icon at the top right corner of the screen. 3. Administrator, tutor, student, or parent clicks the “Language Preferences” to display the language dropdown menu. 4. Administrator, tutor, student, or parent can select the new language preferences, such as English, Chinese, and Malay.		

<table>
<tr><td>5. Administrator, tutor, student, or parent successfully updates the language preferences of their system.</td></tr>
<tr><td><b>Sub Flows:</b>
5a. The system will display the message “Language preferences updated successfully”.</td></tr>
<tr><td><b>Alternate / Exceptional Flows:</b> Not applicable</td></tr>
</table>

Table 3.2.2.9 Download Invoice / Receipt Use Case Description

<table>
<tr><td><b>Use Case Name:</b> Download Invoice / Receipt</td><td><b>ID:</b> 009</td><td><b>Importance Level:</b> High</td></tr>
<tr><td><b>Primary Actor:</b>
Administrator, Student, Parent</td><td colspan="2"><b>Use Case Type:</b> Detail, Essential</td></tr>
<tr><td colspan="3"><b>Stakeholders and Interests:</b>
Administrator, Student, Parent – wants to download the invoice or receipt of the student bill</td></tr>
<tr><td colspan="3"><b>Brief Description:</b> This use case describes how the administrator, student, or parent can download the invoice or receipt of the student's bill</td></tr>
<tr><td colspan="3"><b>Trigger:</b> Administrator, student, or parent wants to download the invoice or receipt of the student’s bill to a local device

<b>Type:</b> External</td></tr>
<tr><td colspan="3"><b>Relationships:</b>
<b>Association:</b> Administrator, Student, Parent
<b>Include:</b>
<b>Extend:</b>
<b>Generalization:</b></td></tr>
<tr><td colspan="3"><b>Normal Flow of Events:</b>
1. Administrator, student, or parent accesses the student bill page.
2. Administrator, student, or parent can view all the details of the billing, such as billing month, registered courses, total amount, notes, etc.
3. Administrator, student, or parent can click the “Download Invoice” (if the payment status is “Unpaid” or “Overdue”) or “Download Receipt” (if the payment status is “Paid”) button at the bottom of the student bill page.</td></tr>
</table>

4. Administrator, student, or parent can check the download invoice or receipt on their local device.
Sub Flows: 1a. The system will retrieve all the information from the database and display it here. 3a. The system will display the message "PDF Generated Successfully. File downloaded to local storage successfully!".
Alternate / Exceptional Flows: 3a. If the local device is not granted the storage permission, then the system will display the text "Storage permission is required to save PDF".

Table 3.2.2.10 Update Students' Attendance Use Case Description

Use Case Name: Update Students' Attendance	**ID:** 010	**Importance Level:** High
Primary Actor: Administrator, Tutor	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator, Tutor – wants to update the students' attendance in the system		
Brief Description: This use case describes how the administrator or tutor can update the students' attendance in the system		
Trigger: Administrator or tutor wants to update the students' attendance in the system **Type:** External		
Relationships: **Association:** Administrator, Tutor **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator or tutor accesses the course history page. 2. Administrator or tutor selects any of the history records to view the student attendance list. 3. Administrator or tutor selects the student's name whose attendance needs to be updated.		

4. Administrator or tutor clicks the “Submit” button to verify the information. 5. Administrator or tutor successfully updates the attendance.
Sub Flows: 1a. The system will retrieve all the history records of the corresponding course and display them. 5a. The system will update the latest attendance information to the database and display the message “Attendance updated successfully”.
Alternate / Exceptional Flows: 2a. If there are no existing history records, then the system will display the message “No history available for this course”.

Table 3.2.2.11 Access Group Chat Room Use Case Description

Use Case Name: Access Group Chat Room	**ID:** 011	**Importance Level:** High
Primary Actor: Administrator, Tutor, Student	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator, Tutor, Student – wants to chat and ask some questions in the course group chat room		
Brief Description: This use case describes how the administrator, tutor, or student can access the course group chat room to chat in the course group		
Trigger: Administrator, tutor, or student wants to access the course group chat room to chat in the course group **Type:** External		
Relationships: **Association:** Administrator, Tutor, Student **Include:** **Extend:** Search, Sort By **Generalization:**		
Normal Flow of Events: 1. Administrator, tutor, or student accesses the inbox page. 2. Administrator, tutor, or student can select the target course group chat room from the list.		

3. Administrator, tutor, or student can start to communicate and ask some questions in the course group chat room. 4. The message will be sent to the target people, such as students who registered for the course, the tutor who handles the course, and the administrator, when the send button is clicked. 5. The system will notify the target people about the new message via Gmail.
Sub Flows: 1a. The system will retrieve all the information from the database and display it here.
Alternate / Exceptional Flows: 5a. An email notification will be sent to the target people to notify them of the new message.

Table 3.2.2.12 Access Private Chat Room Use Case Description

<table>
<tr><td><b>Use Case Name:</b> Access Private Chat Room</td><td><b>ID:</b> 012</td><td><b>Importance Level:</b> High</td></tr>
<tr><td><b>Primary Actor:</b>
Administrator, Tutor, Student, Parent</td><td colspan="2"><b>Use Case Type:</b> Detail, Essential</td></tr>
<tr><td colspan="3"><b>Stakeholders and Interests:</b>
Administrator, Tutor, Student, Parent – wants to chat and ask some questions</td></tr>
<tr><td colspan="3"><b>Brief Description:</b> This use case describes how the administrator, tutor, student, or parent can access the private chat room to chat with others</td></tr>
<tr><td colspan="3"><b>Trigger:</b> Administrator, tutor, student, or parent wants to access the private chat room to chat with others

<b>Type:</b> External</td></tr>
<tr><td colspan="3"><b>Relationships:</b>
<b>Association:</b> Administrator, Tutor, Student, Parent
<b>Include:</b>
<b>Extend:</b> Search, Sort By
<b>Generalization:</b></td></tr>
<tr><td colspan="3"><b>Normal Flow of Events:</b>
1. Administrator, tutor, student, or parent accesses the inbox page.
2. Administrator, tutor, student, or parent can select the target person’s chat room from the list.</td></tr>
</table>

3. Administrator, tutor, student, or parent can start to communicate and ask some questions in the chat room. 4. The message will be sent to the target person when the send button is clicked. 5. The system will notify the target person about the new message via Gmail.
Sub Flows: 1a. The system will retrieve all the information from the database and display it here. 2a. The system will retrieve and display all the names of the users categorized by their roles.
Alternate / Exceptional Flows: 4a. If the message is sent successfully to the target person, then the system will display the text "Message sent successfully". 5a. An email notification will be sent to the target person to notify them of the new message.

Table 3.2.2.13 View Announcements Use Case Description

Use Case Name: View Announcements	**ID:** 013	**Importance Level:** High
Primary Actor: Administrator, Tutor, Student, Parent	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator, Tutor, Student, Parent – wants to view the latest announcements		
Brief Description: This use case describes how the administrator, tutor, student, or parent can view the latest announcements		
Trigger: Administrator, tutor, student, or parent wants to view the latest announcements to get the most recent updates **Type:** External		
Relationships: **Association:** Administrator, Tutor, Student, Parent **Include:** **Extend:** Search, Sort By **Generalization:**		
Normal Flow of Events: 1. Administrator, tutor, student, or parent accesses the announcements page.		

2. Administrator, tutor, student, or parent can view all the announcements posted by the administrator. 3. Administrator, tutor, student, or parent can select and click any of the announcements to view the detailed information, such as announcement title, announcement description, date of publication, etc.
Sub Flows: 1a. The system will retrieve all the announcements from the database and display them here.
Alternate / Exceptional Flows: 1a. If there is no announcement in the database, then the system will display “No announcement recently”.

Table 3.2.2.14 Browse Existing Courses Use Case Description

<table>
<tr><td><b>Use Case Name:</b> Browse Existing Courses</td><td><b>ID:</b> 014</td><td><b>Importance Level:</b> High</td></tr>
<tr><td><b>Primary Actor:</b> Administrator</td><td colspan="2"><b>Use Case Type:</b> Detail, Essential</td></tr>
<tr><td colspan="3"><b>Stakeholders and Interests:</b>
Administrator – wants to browse all the existing courses in the system</td></tr>
<tr><td colspan="3"><b>Brief Description:</b> This use case describes how the administrator can browse all the existing courses in the system</td></tr>
<tr><td colspan="3"><b>Trigger:</b> Administrator wants to browse and check all the existing courses in the system
<b>Type:</b> External</td></tr>
<tr><td colspan="3"><b>Relationships:</b>
<b>Association:</b> Administrator
<b>Include:</b>
<b>Extend:</b> Search, Sort By
<b>Generalization:</b></td></tr>
<tr><td colspan="3"><b>Normal Flow of Events:</b>
1. Administrator accesses the course list page.
2. Administrator can view all the existing registered courses, which are arranged alphabetically based on the course name.
3. The details of the courses will be displayed, such as the course name, course description, day, time, fee, grade, tutor name, etc.</td></tr>
</table>

Sub Flows: 1a. The system will retrieve all the information from the database and display it here.
Alternate / Exceptional Flows: 1a. If there is no existing course in the database, then the system will display "No existing courses recently".

Table 3.2.2.15 Browse Assigned Courses Use Case Description

Use Case Name: Browse Assigned Courses	**ID:** 015	**Importance Level:** High
Primary Actor: Tutor	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Tutor – wants to browse the assigned courses in the system		
Brief Description: This use case describes how the tutor can browse the assigned courses in the system		
Trigger: Tutor wants to browse the assigned courses in the system **Type:** External		
Relationships: **Association:** Tutor **Include:** **Extend:** Search, Sort By **Generalization:**		
Normal Flow of Events: 1. Tutor accesses the course list page. 2. Tutor can scroll up and down to view all the assigned courses, which are arranged alphabetically based on the course name. 3. The details of the courses will be displayed, such as the course name, day, time, grade, etc.		
Sub Flows: 1a. The system will retrieve all the information from the database and display it here.		
Alternate / Exceptional Flows: 1a. If there is no existing course in the database, then the system will display "No existing courses recently".		

Table 3.2.2.16 Add New Course Use Case Description

Use Case Name: Add New Course	**ID:** 016	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator - wants to add a new course in the system		
Brief Description: This use case describes how the administrator can add a new course in the system		
Trigger: Administrator wants to add a new course in the system **Type:** External		
Relationships: **Association:** Administrator **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator accesses the course list page. 2. Administrator clicks the “add” icon at the top right corner of the page. 3. Administrator enters the course information, such as course name, course description, course fee, course grade, tutor name, schedule day, start time, and end time. 4. Administrator selects the students who want to participate in the course. 5. Administrator clicks the “Submit” button to verify the information entered. 6. Administrator successfully added the new course to the system.		
Sub Flows: 3a. The system will retrieve the names of all students once the grade is selected and display them in the “Student List”. 3b. The system will retrieve the names of all tutors and display them in the dropdown list of “Tutor Name”. 6a. The system will update the information of the new course to the database and display the message “Course added successfully”.		
Alternate / Exceptional Flows:		

5a. If the administrator leaves any of the required fields empty, then the system will highlight the empty field and display the message "Please enter the information". 5b. If the end time is before the start time, then the system will display the message "End time must be after start time".

Table 3.2.2.17 Add New Student Bill Use Case Description

Use Case Name: Add New Student Bill	**ID:** 017	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator – wants to add new student bills in the system		
Brief Description: This use case describes how the administrator can add new student bills to the system		
Trigger: Administrator wants to add new student bills to the system **Type:** External		
Relationships: **Association:** Administrator **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator accesses the payment list page. 2. Administrator clicks the "add" icon at the top right corner of the page. 3. Administrator can select the billing month and the grade to filter the student list. 4. Administrator selects or unselects the students' names by ticking or unticking the checkbox. 5. Administrator clicks the "Submit" button to verify the information entered. 6. Administrator successfully added the new student bill to the system. 7. The system will send an email notification to notify the students' parents that a new student bill has been generated.		
Sub Flows: 3a. The system will retrieve the names of all students and display them.		

6a. The system will update the information of the new student bill to the database and display the message “New bill added successfully”.
Alternate / Exceptional Flows: Not applicable

Table 3.2.2.18 Add New Announcement Use Case Description

Use Case Name: Add New Announcement	**ID:** 018	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator – wants to add a new announcement		
Brief Description: This use case describes how the administrator can add a new announcement		
Trigger: Administrator wants to add a new announcement to inform others about the latest updates **Type:** External		
Relationships: **Association:** Administrator **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator accesses the announcement page. 2. Administrator clicks the “add” icon at the top right of the screen. 3. Administrator uploads and enters the announcement information, such as photo, title, and description. 4. Administrator clicks the “Submit” button to verify the information. 5. Administrator successfully added a new announcement. 6. The system will send an email notification to notify all users that a new announcement has been generated.		
Sub Flows: 6a. The system will update the information of the new announcement to the database and display the message “Announcement added successfully”.		
Alternate / Exceptional Flows:		

5a. If the administrator leaves any of the required fields empty, then the system will highlight the empty field and display the message “Please enter the information”.

Table 3.2.2.19 View Students’ Profiles Use Case Description

Use Case Name: View Students’ Profiles	**ID:** 019	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator – wants to view information about all the students		
Brief Description: This use case describes how the administrator can view information about all the students		
Trigger: Administrator wants to view and access the students' information **Type:** External		
Relationships: **Association:** Administrator **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator accesses the student list page. 2. Administrator selects and clicks any of the existing students. 3. Administrator can view the corresponding student’s information, such as name, gender, age, date of birth, grade, phone number, email address, profile photo, courses registered, etc.		
Sub Flows: 1a. The system will retrieve all the existing student records from the database and display them here.		
Alternate / Exceptional Flows: 3a. If there are no existing student records, then the system will display the message “No student available recently”.		

Table 3.2.2.20 View Tutors’ Profiles Use Case Description

Use Case Name: View Tutors’ Profiles	**ID:** 020	**Importance Level:** High

Primary Actor: Administrator	**Use Case Type:** Detail, Essential
Stakeholders and Interests: Administrator – wants to view information about all the tutors	
Brief Description: This use case describes how the administrator can view information about all the tutors	
Trigger: Administrator wants to view and access the tutors' information **Type:** External	
Relationships: **Association:** Administrator **Include:** **Extend:** **Generalization:**	
Normal Flow of Events: 1. Administrator accesses the tutor list page. 2. Administrator selects and clicks any of the existing tutors. 3. Administrator can view the corresponding tutor's information, such as name, gender, age, date of birth, phone number, email address, profile photo, courses assigned, etc.	
Sub Flows: 1a. The system will retrieve all the existing tutor records from the database and display them here.	
Alternate / Exceptional Flows: 3a. If there are no existing tutor records, then the system will display the message "No tutor available recently".	

Table 3.2.2.21 Check Analytics Report Use Case Description

Use Case Name: Check Analytics Report	**ID:** 021	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator – wants to check the performance and analytics report		
Brief Description: This use case describes how the administrator can check the performance and analytics report		
Trigger: Administrator wants to check and view the performance and analytics report		

Type: External
Relationships: **Association:** Administrator **Include:** **Extend:** Check Revenue Analytics, Check Course Analytics, Check Tutor Performance Analytics **Generalization:**
Normal Flow of Events: 1. Administrator accesses the report and analytics page. 2. Administrator can view the analytics report by selecting the options, such as the revenue analytics, course analytics, student analytics, and tutor analytics. 3. After selecting the options, the system will display the respective analytics report.
Sub Flows: Not applicable
Alternate / Exceptional Flows: Not applicable

Table 3.2.2.22 Check Revenue Analytics Use Case Description

Use Case Name: Check Revenue Analytics	**ID:** 022	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator – wants to check the revenue analytics in the system		
Brief Description: This use case describes how the administrator can check the revenue analytics in the system		
Trigger: Administrator wants to check and view the revenue analytics in the system **Type:** External		
Relationships: **Association:** **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator accesses the revenue analytics page.		

<table>
<tr><td>2. Administrator can view all the revenue analytics records, which are displayed in the pie chart.
3. Administrator selects the desired month from the pop-up calendar to filter the data.
4. System will display the pie chart with the details.</td></tr>
<tr><td><b>Sub Flows:</b>
1a. The system will retrieve all the information from the database and display it here.</td></tr>
<tr><td><b>Alternate / Exceptional Flows:</b> Not applicable</td></tr>
</table>

Table 3.2.2.23 Check Course Analytics Use Case Description

<table>
<tr><td><b>Use Case Name:</b> Check Course Analytics</td><td><b>ID:</b> 023</td><td><b>Importance Level:</b> High</td></tr>
<tr><td><b>Primary Actor:</b> Administrator</td><td colspan="2"><b>Use Case Type:</b> Detail, Essential</td></tr>
<tr><td colspan="3"><b>Stakeholders and Interests:</b>
Administrator – wants to check the course analytics in the system</td></tr>
<tr><td colspan="3"><b>Brief Description:</b> This use case describes how the administrator can check the course analytics in the system</td></tr>
<tr><td colspan="3"><b>Trigger:</b> Administrator wants to check and view the course analytics in the system

<b>Type:</b> External</td></tr>
<tr><td colspan="3"><b>Relationships:</b>
<b>Association:</b>
<b>Include:</b>
<b>Extend:</b>
<b>Generalization:</b></td></tr>
<tr><td colspan="3"><b>Normal Flow of Events:</b>
1. Administrator accesses the course analytics page.
2. Administrator can view all the course analytics records, which are displayed in the pie chart.
3. Administrator selects the desired month from the pop-up calendar and the grade to filter the data.
4. System will display the pie chart with the details.</td></tr>
<tr><td colspan="3"><b>Sub Flows:</b>
1a. The system will retrieve all the information from the database and display it here.</td></tr>
<tr><td colspan="3"><b>Alternate / Exceptional Flows:</b></td></tr>
</table>

1a. If the course analytics data in the database is empty, then the system will display the text “No courses found for the selected grade”.

Table 3.2.2.24 Check Tutor Performance Analytics Use Case Description

Use Case Name: Check Tutor Performance Analytics	**ID:** 024	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator – wants to check the tutor performance analytics in the system		
Brief Description: This use case describes how the administrator can check the tutor performance analytics in the system		
Trigger: Administrator wants to check and view the tutor performance analytics in the system **Type:** External		
Relationships: **Association:** **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator accesses the tutor performance analytics page. 2. Administrator can view all the tutor performance analytics records, which are displayed in the horizontal bar chart. 3. Administrator selects the desired month from the pop-up calendar to filter the tutor performance analytics records. 4. Administrator can click the respective tutor bar in the bar chart to view the details of the respective tutor.		
Sub Flows: 1a. The system will retrieve all the information from the database and display it here.		
Alternate / Exceptional Flows: 1a. If the tutor performance analytics data in the database is empty, then the system will display the text “No analysis for this month”.		

Table 3.2.2.25 Add New Class Schedule Use Case Description

Use Case Name: Add New Class Schedule	**ID:** 025	**Importance Level:** High
Primary Actor: Tutor	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Tutor – wants to add a new class schedule in the system		
Brief Description: This use case describes how the tutor can add a new class schedule in the system		
Trigger: Tutor wants to add a new class schedule in the system **Type:** External		
Relationships: **Association:** Tutor **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Tutor accesses the class list page. 2. Tutor clicks the "add" icon at the top right corner of the corresponding class. 3. Tutor enters the class information, such as date, start time, and end time. 4. Tutor clicks the "Submit" button to verify the information entered. 5. Tutor successfully added the new class schedule to the system.		
Sub Flows: 5a. The system will update the information of the new class schedule to the database and display the message "Class schedule added successfully".		
Alternate / Exceptional Flows: 4a. If the tutor leaves any of the required fields empty, then the system will highlight the empty field and display the message "Please enter the information". 4b. If the end time is before the start time, then the system will display the message "End time must be after start time".		

Table 3.2.2.26 View Children’s Profile Use Case Description

Use Case Name: View Children’s Profile	**ID:** 026	**Importance Level:** High
Primary Actor: Parent	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Parent – wants to view their children's profiles and information in the system		
Brief Description: This use case describes how the parent can view their children's profiles and information in the system		
Trigger: Parent wants to view and check their children's profiles and information in the system **Type:** External		
Relationships: **Association:** Parent **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Parent accesses the children page. 2. Parent can view the corresponding children’s information, such as name, gender, age, grade, courses enrolled, attendance status, etc. 3. Parent selects and clicks any of their children to view all the details of the registered course. 4. Parent can view all the details of the registered course, such as the course name, course description, course fee, day, time, tutor name, etc.		
Sub Flows: 1a. The system will retrieve all the information from the database and display it here.		
Alternate / Exceptional Flows: 1a. If there are no children's records from the database, then the system will display the message “No children found”.		

Table 3.2.2.27 Make Payment Use Case Description

Use Case Name: Make Payment	**ID:** 027	**Importance Level:** High
Primary Actor: Parent	**Use Case Type:** Detail, Essential	

Stakeholders and Interests:

Parent – wants to make a payment for their child’s student bill

Brief Description: This use case describes how the parent can make a payment for their child’s student bill

Trigger: Parent wants to make a payment for their child’s student bill

Type: External

Relationships:

Association: Parent

Include:

Extend:

Generalization:

Normal Flow of Events:

1. Parent accesses the payment page.
2. Parent views and selects any of the child’s bill records.
3. Parent make payment via credit card by clicking the button “Make Payment”, if the payment status of the student's bill is “Overdue” or “Unpaid”.
4. Parent enters the relevant details of the credit card to make payment, such as card number, cardholder name, expiry date, and CVV.
5. Parent pays the student bill successfully, and the information will be updated to the database.

Sub Flows:

1a. The system will retrieve all the bill records from the database and display them here.

5a. The system will display the text “Payment Successful!”

Alternate / Exceptional Flows:

4a. If the credit card is declined or invalid, then the system will display the text “Payment Failed” and prompt the parent to try again to make payment.

4b. If the credit card has insufficient funds, then the system will display the text “Payment Failed” and prompt the parent to try again to make payment.

4c. If the parent leaves any of the required fields empty, then the system will highlight the empty field and display the message “Information Required”.

Table 3.2.2.28 Search Use Case Description

<table>
<tr><td><b>Use Case Name:</b> Search</td><td><b>ID:</b> 028</td><td><b>Importance Level:</b> High</td></tr>
<tr><td><b>Primary Actor:</b>
Administrator, Tutor, Student, Parent</td><td colspan="2"><b>Use Case Type:</b> Detail, Essential</td></tr>
<tr><td colspan="3"><b>Stakeholders and Interests:</b>
Administrator, Tutor, Student, Parent – wants to search for information by using keywords</td></tr>
<tr><td colspan="3"><b>Brief Description:</b> This use case describes how the administrator, tutor, student, or parent can search for information by using keywords</td></tr>
<tr><td colspan="3"><b>Trigger:</b> Administrator, tutor, student, or parent wants to search for information by using keywords

<b>Type:</b> External</td></tr>
<tr><td colspan="3"><b>Relationships:</b>
<b>Association:</b>
<b>Include:</b>
<b>Extend:</b>
<b>Generalization:</b></td></tr>
<tr><td colspan="3"><b>Normal Flow of Events:</b>
1. Administrator, tutor, student, or parent accesses the relevant page.
2. Administrator, tutor, student, or parent clicks on the search bar, which is located at the top of the page.
3. Administrator, tutor, student, or parent enters some keywords in the search bar and clicks the “search” icon.
4. System will search for all the existing records based on the entered keywords and display all the relevant records.</td></tr>
<tr><td colspan="3"><b>Sub Flows:</b>
1a. The system will retrieve all the information from the database and display it here.</td></tr>
<tr><td colspan="3"><b>Alternate / Exceptional Flows:</b>
3a. If the administrator or tutor wants to delete all the keywords in the search bar, then they can click the “cross” icon, which is located on the right side of the search bar.
4a. If there are no existing records based on the entered keywords, then the system will display the message “No records available”.</td></tr>
</table>

Table 3.2.2.29 Sort By Use Case Description

Use Case Name: Sort By	**ID:** 029	**Importance Level:** High
Primary Actor: Administrator, Tutor, Student, Parent	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator, Tutor, Student, Parent – wants to sort the existing records		
Brief Description: This use case describes how the administrator, tutor, student, or parent can sort the existing records		
Trigger: Administrator, tutor, student, or parent wants to sort the arrangement of the existing records **Type:** External		
Relationships: **Association:** **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator, tutor, student, or parent accesses the relevant page. 2. Administrator, tutor, student, or parent clicks the "filter" icon, which is located at the top right of the screen. 3. System will pop up a dialog box and ask for the sorting options. 4. Administrator, tutor, student, or parent needs to choose one of the sorting options. 5. After choosing the sorting options, the system will display all the existing records, which are sorted based on the options selected.		
Sub Flows: 1a. The system will retrieve all the information from the database and display it here.		
Alternate / Exceptional Flows: 5a. If there are no existing records, then the system will display the message "No records available".		

3.2.3 Activity Diagram

i. Administrator Activity Diagram

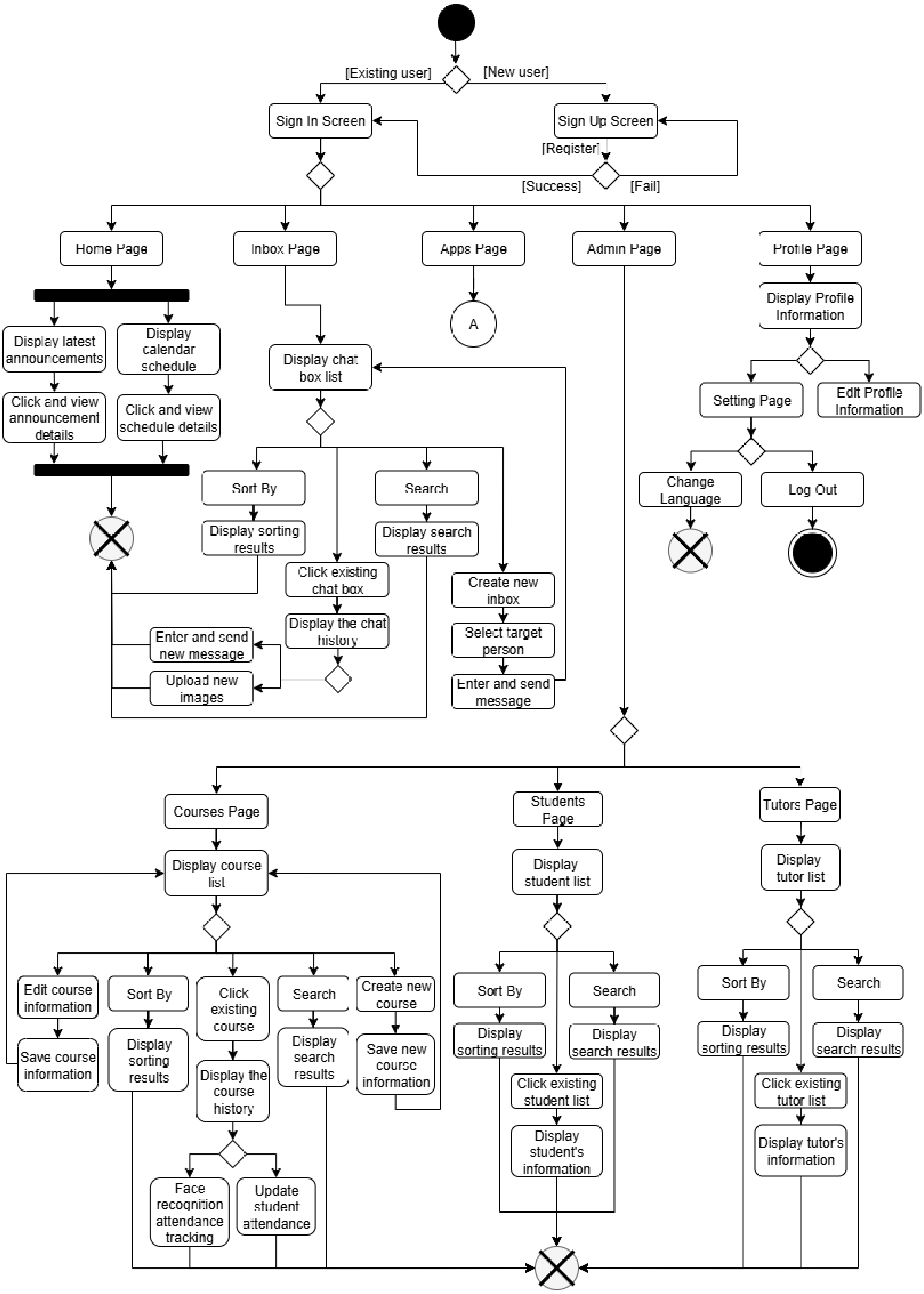


Figure 3.2.3.1 Administrator Activity Diagram (Part 1)

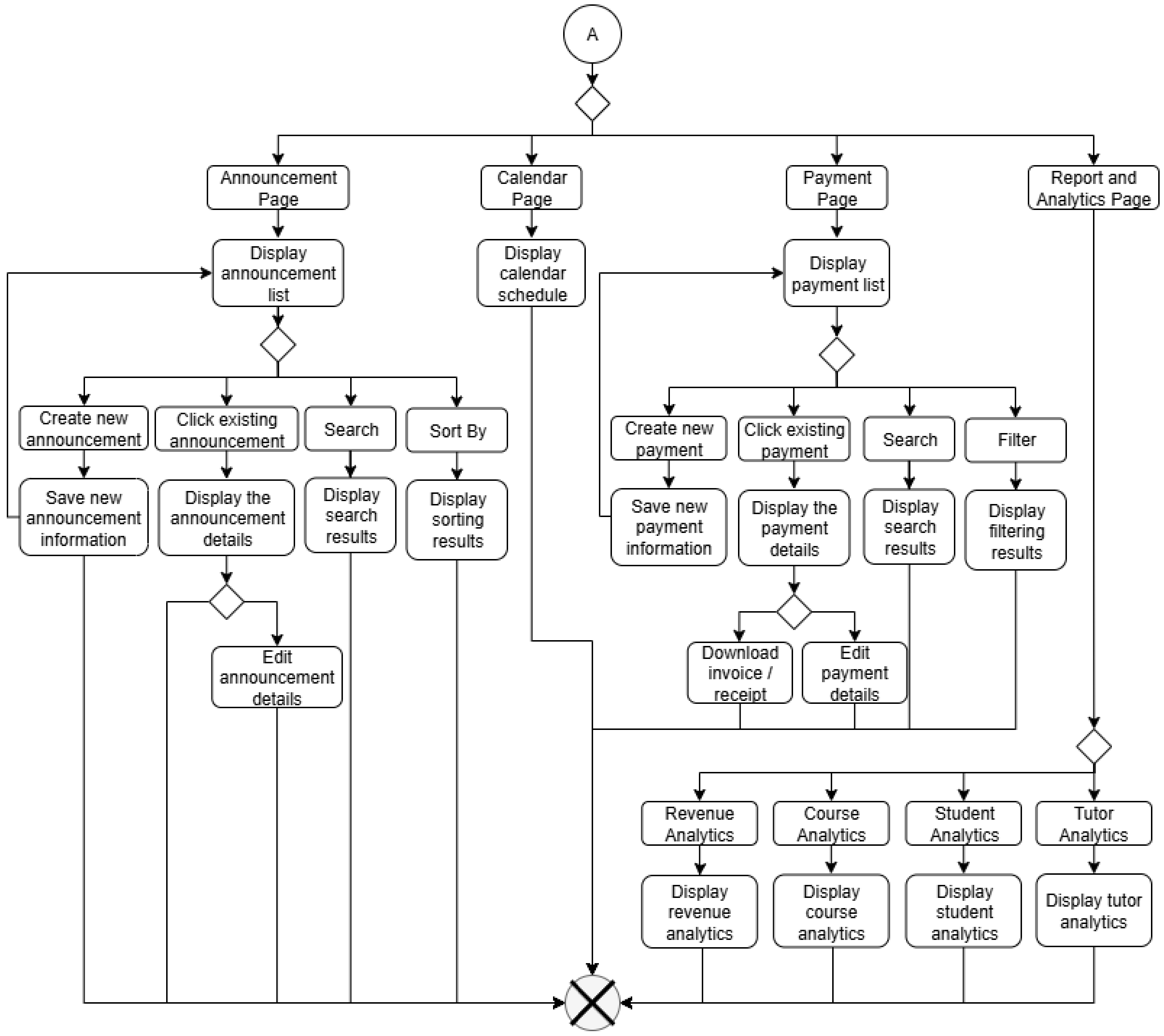


Figure 3.2.3.2 Administrator Activity Diagram (Part 2)

ii. Tutor Activity Diagram

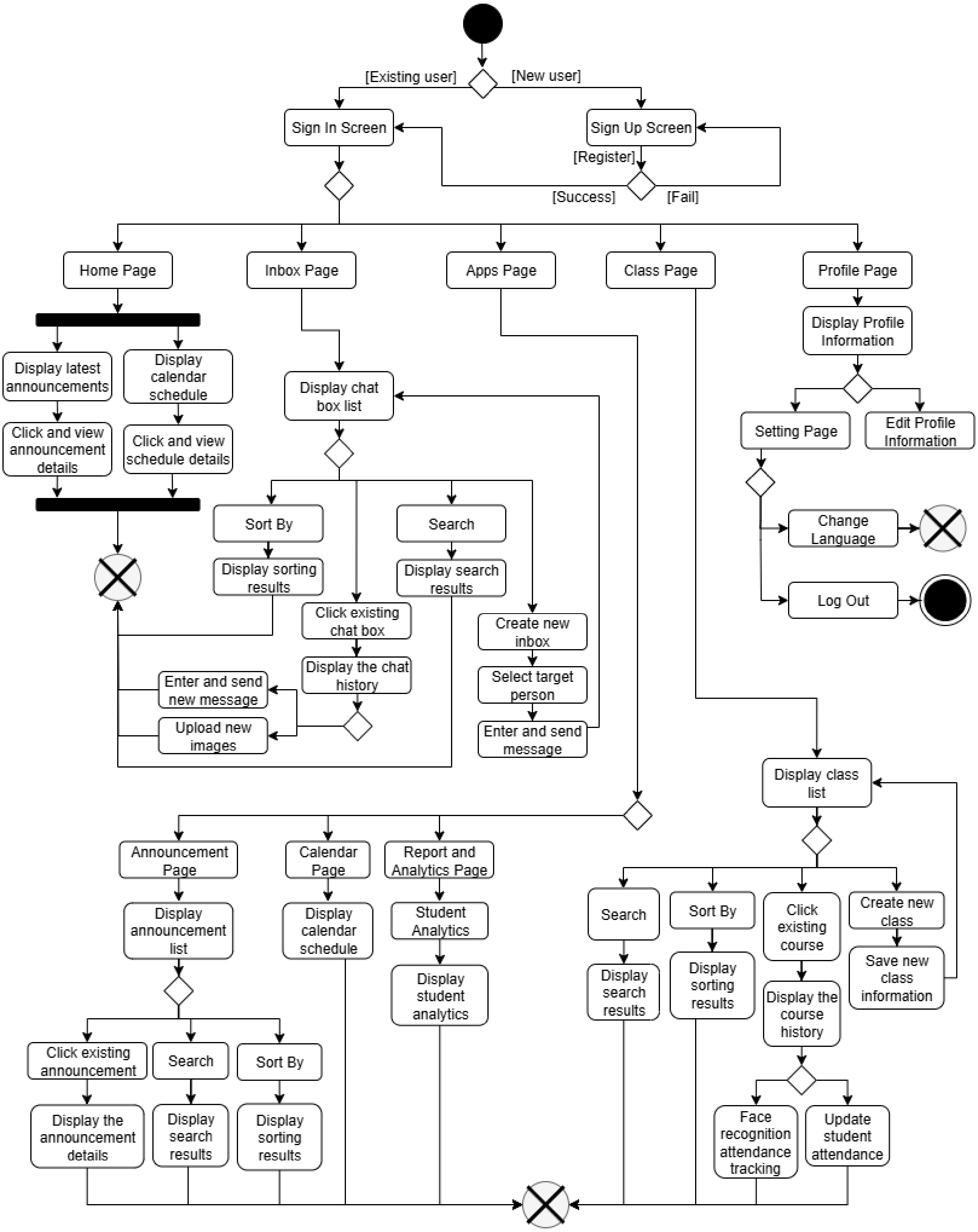


Figure 3.2.3.3 Tutor Activity Diagram

iii. Student Activity Diagram

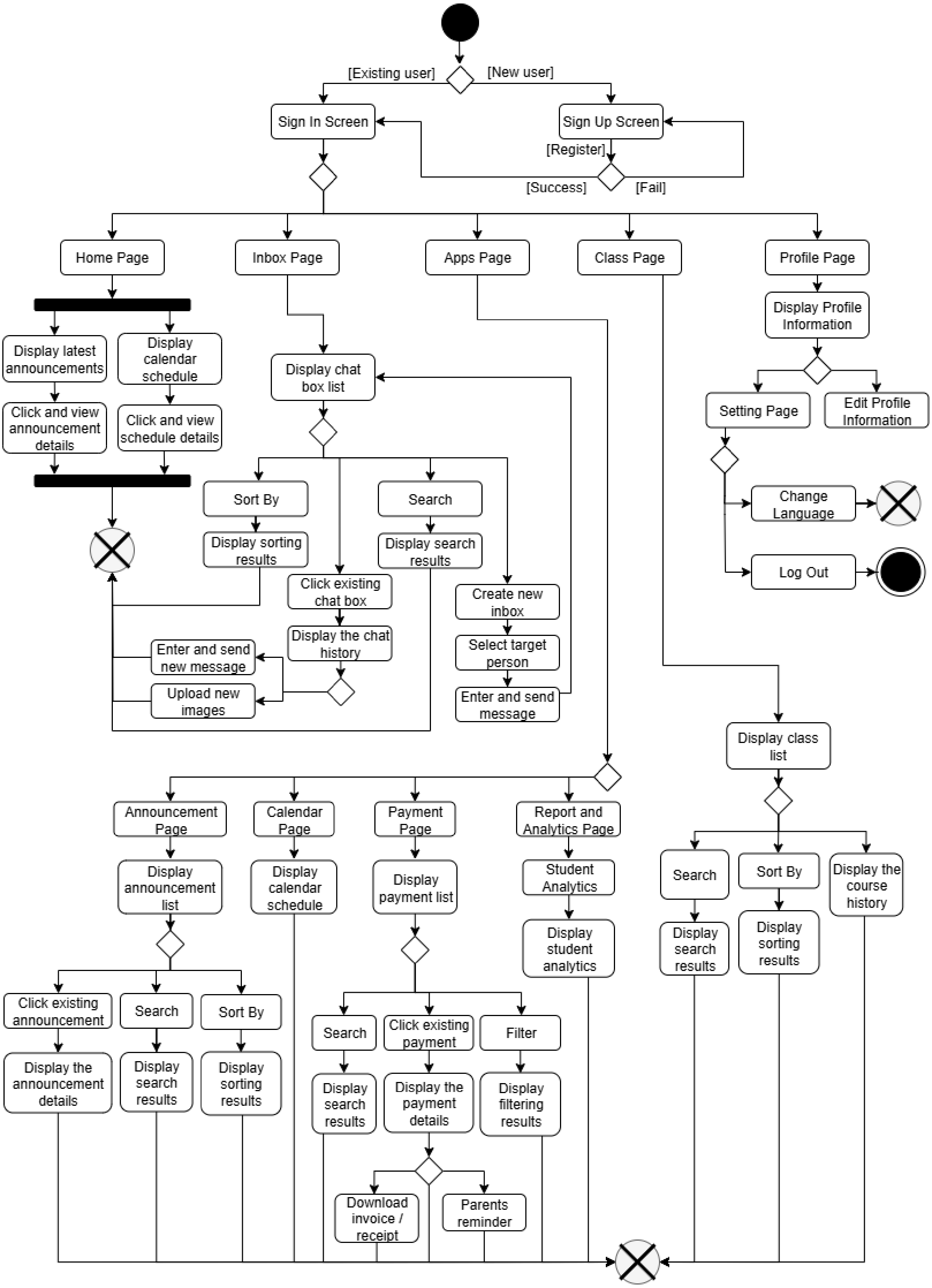


Figure 3.2.3.4 Student Activity Diagram

iv. Parent Activity Diagram

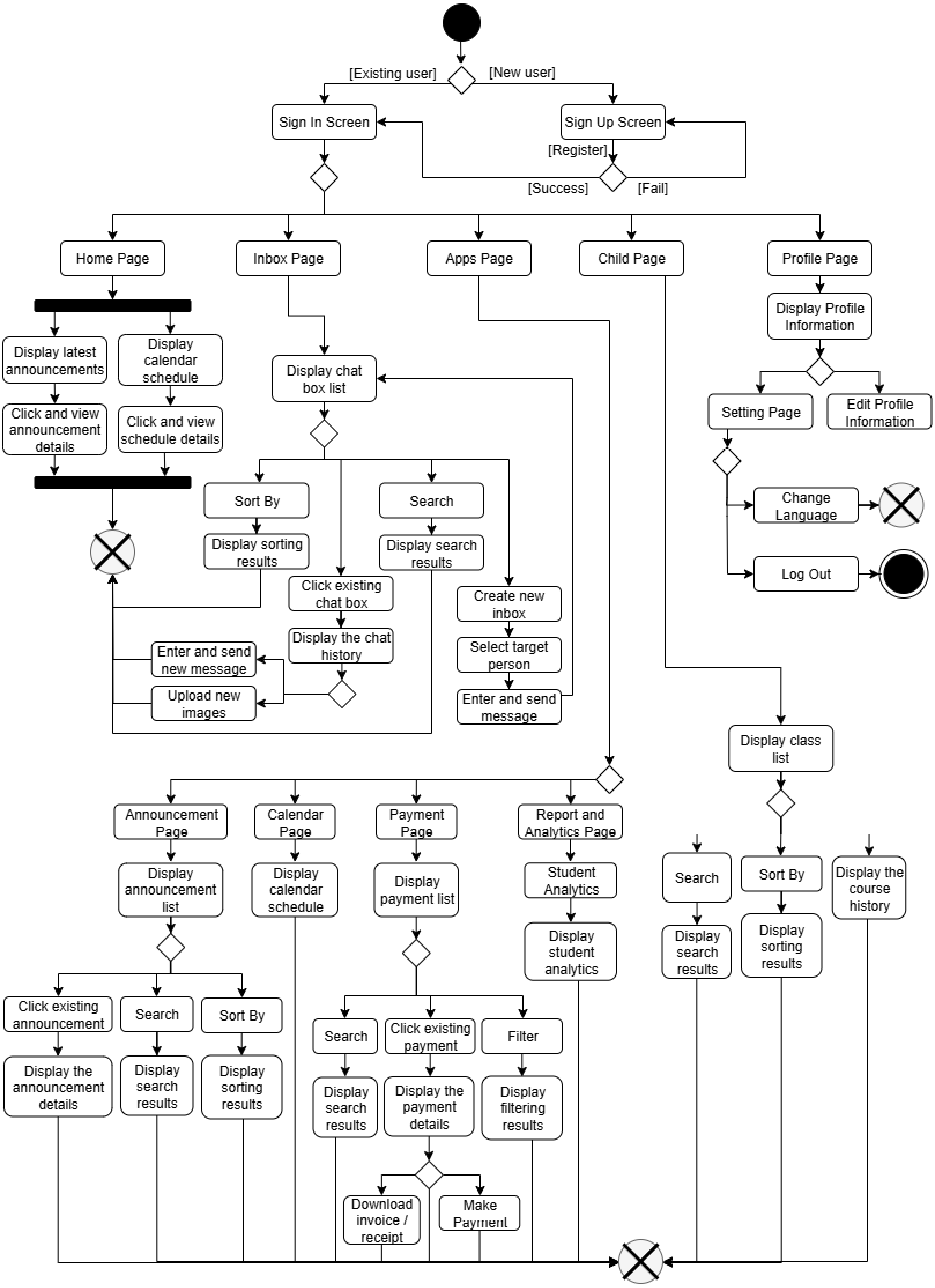


Figure 3.2.3.5 Parent Activity Diagram

3.3 Timeline

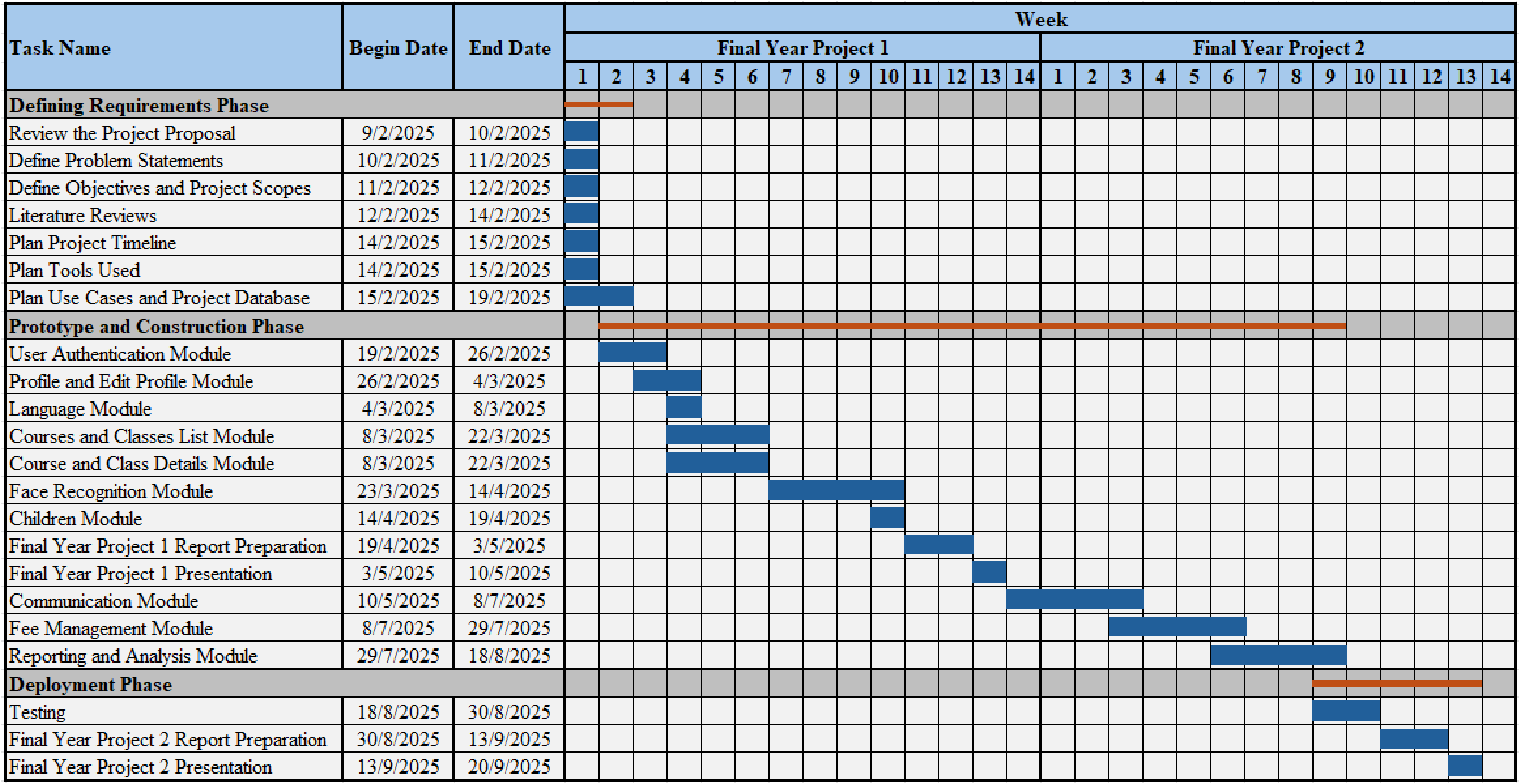
The project timeline is determined by the project methodology and is divided into three main phases, which are the defining requirements phase, the prototype and construction phase, and the deployment phase. Each of the phases consists of specific tasks to ensure the systematic development of the application.

In the defining requirements phase, this initial phase focuses on gathering and establishing system requirements. During week 1 in Final Year Project (FYP) 1, the project proposal was reviewed. Besides, the problem statements, objectives, and project scope are defined. The literature reviews are conducted from 12 February 2025 to 14 February 2025 to analyse similar projects and existing systems, which helps to provide insights into the project. The phase continued by planning the project timeline, planning the tools used, planning the use cases, and planning the project database.

Moreover, the prototype and construction phase involves developing and refining the system through iterative prototyping and construction. The user authentication module, such as sign up, sign in, and sign out, was built from 19 February 2025 to 26 February 2025, and it is followed by the profile, edit profile, and language modules. Between 8 March 2025 and 22 March 2025, the courses and classes list module and the course and class details module were developed. The face recognition module was constructed by integrating features such as biometric attendance tracking using the face_recognition library and Flask. The children's module for parents was built from 14 April to 19 April 2025. From week 11 to week 13, the FYP 1 report and the presentation were prepared.

In the Final Year Project 2, the modules, including the communication module, the fee management module, and the reporting and analysis module, will be developed from 10 May to 18 August 2025. Lastly, the deployment phase focuses on testing and finalising the system for real-world use. From 18 August to 30 August 2025, the system undergoes rigorous testing to identify and fix bugs to ensure all functionality works. After the testing process, the report and presentation of Final Year Project 2 are prepared, and the system is showcased to the supervisor and moderator for evaluation.

3.3.1 Gantt Chart



Task Name	Begin Date	End Date
Defining Requirements Phase		
Review the Project Proposal	9/2/2025	10/2/2025
Define Problem Statements	10/2/2025	11/2/2025
Define Objectives and Project Scopes	11/2/2025	12/2/2025
Literature Reviews	12/2/2025	14/2/2025
Plan Project Timeline	14/2/2025	15/2/2025
Plan Tools Used	14/2/2025	15/2/2025
Plan Use Cases and Project Database	15/2/2025	19/2/2025
Prototype and Construction Phase		
User Authentication Module	19/2/2025	26/2/2025
Profile and Edit Profile Module	26/2/2025	4/3/2025
Language Module	4/3/2025	8/3/2025
Courses and Classes List Module	8/3/2025	22/3/2025
Course and Class Details Module	8/3/2025	22/3/2025
Face Recognition Module	23/3/2025	14/4/2025
Children Module	14/4/2025	19/4/2025
Final Year Project 1 Report Preparation	19/4/2025	3/5/2025
Final Year Project 1 Presentation	3/5/2025	10/5/2025
Communication Module	10/5/2025	8/7/2025
Fee Management Module	8/7/2025	29/7/2025
Reporting and Analysis Module	29/7/2025	18/8/2025
Deployment Phase		
Testing	18/8/2025	30/8/2025
Final Year Project 2 Report Preparation	30/8/2025	13/9/2025
Final Year Project 2 Presentation	13/9/2025	20/9/2025

Figure 3.3.1.1 Gantt Chart

3.4 Summary

In conclusion, this chapter outlines the proposed methodology by focusing on the Rapid Application Development (RAD) approach to develop a comprehensive management system for small-sized tuition centres. Besides, it also contains a use case diagram and various use case descriptions that describe different use cases. Furthermore, it includes the activity diagram for different roles, including administrators, tutors, students, and parents. Lastly, a Gantt Chart, which illustrates the timeline, is included.

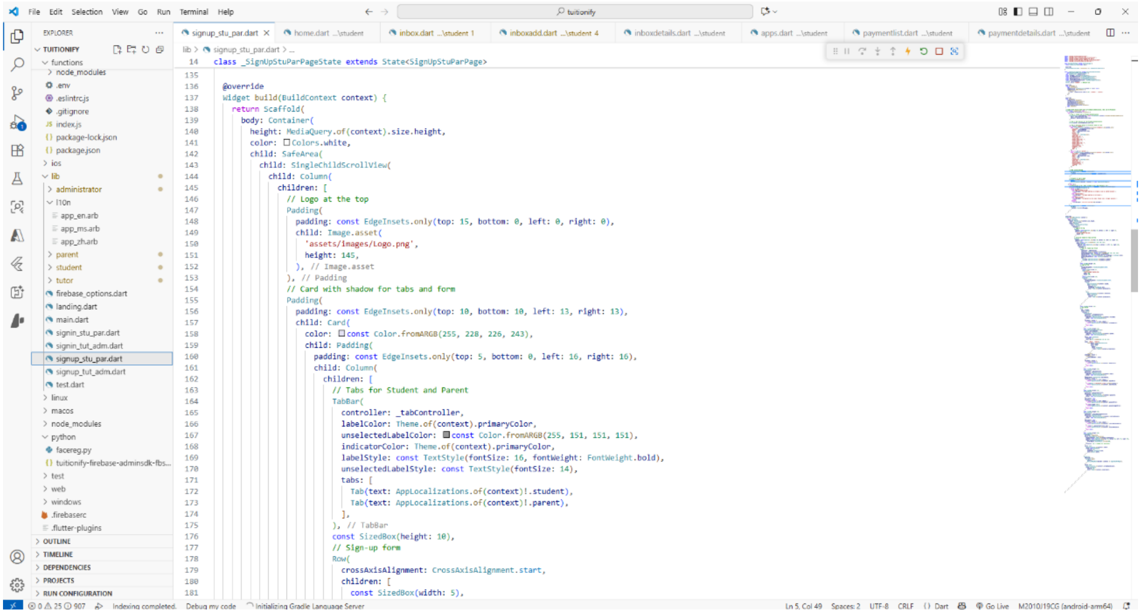
Chapter 4
System Design

In this chapter, fifteen essential modules are developed for the tuition centre management system. The core modules include the user authentication module that allows role-based access control, the home module that displays the latest announcements and schedules, the inbox module that allow private and group communication, the announcement module that display all the announcements, the calendar module that display all class schedules, the payment module that automated the bill tracking and direct payment, the report and analytics module that provide data visualisations, the courses and classes module that for educational management, the students, tutors, and children module that for user management, the profile module that allow update the latest personal information, the face recognition module that for biometric attendance tracking, and the language preferences module with English, Malay, and Chinese options.

4.1 Program Development

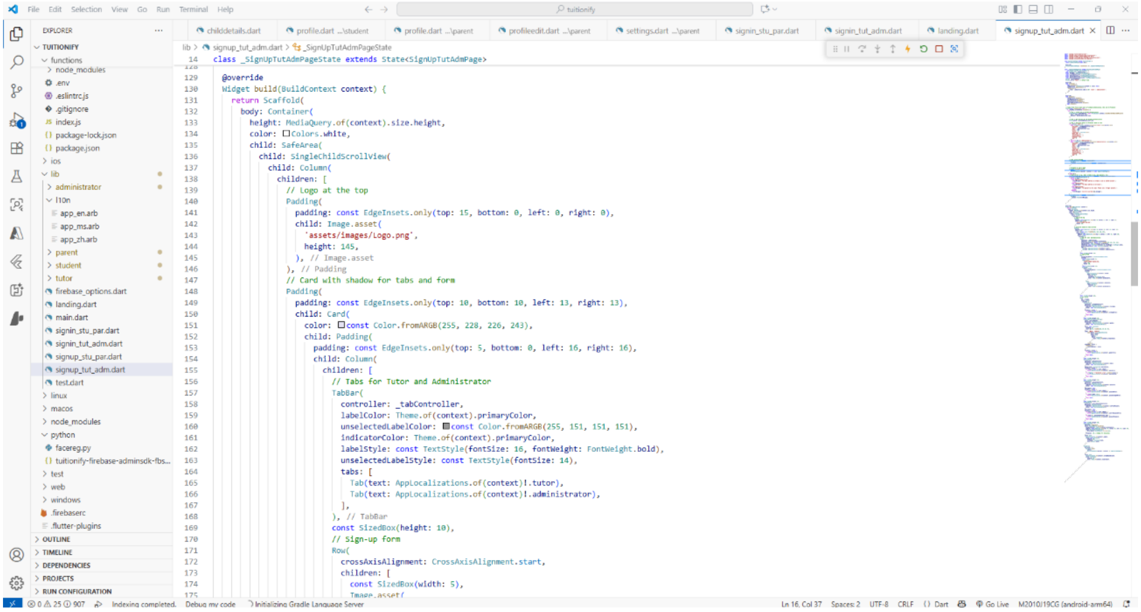
4.1.1 User Authentication Module Development

The Sign Up and Sign In modules are developed by leveraging Firebase Authentication to ensure safe and effective user authentication. Based on Figure 4.1.1.1 and Figure 4.1.1.2, the Sign Up module allows administrators, tutors, students, and parents to create a new account by selecting their role, such as “Student & Parent” or “Staff”. They need to enter their personal information, such as full name, gender, email address, passwords, and confirm password. When they click the “Submit” button, the system will verify and ensure that no required fields are empty without any information. Besides, the system will also validate the input, such as the passwords match, the length of the password meets the requirements of the minimum length of 6 characters, and the email is not already in use in the Firebase Authentication. Once the account is created, a message “Sign up successfully” will be displayed.



```dart
class _SignUpStuParPageState extends State<SignUpStuParPage>

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        height: MediaQuery.of(context).size.height,
        color: Colors.white,
        child: SafeArea(
          child: SingleChildScrollView(
            child: Column(
              children: [
                // Logo at the top
                Padding(
                  padding: const EdgeInsets.only(top: 15, bottom: 0, left: 0, right: 0),
                  child: Image.asset(
                    'assets/images/Logo.png',
                    height: 145,
                  ), // Image.asset
                ), // Padding
                // Card with shadow for tabs and form
                Padding(
                  padding: const EdgeInsets.only(top: 10, bottom: 10, left: 13, right: 13),
                  child: Card(
                    color: const Color.fromARGB(255, 228, 226, 243),
                    child: Padding(
                      padding: const EdgeInsets.only(top: 5, bottom: 0, left: 16, right: 16),
                      child: Column(
                        children: [
                          // Tabs for Student and Parent
                          TabBar(
                            controller: _tabController,
                            labelColor: Theme.of(context).primaryColor,
                            unselectedLabelColor: const Color.fromARGB(255, 151, 151, 151),
                            indicatorColor: Theme.of(context).primaryColor,
                            labelStyle: const TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
                            unselectedLabelStyle: const TextStyle(fontSize: 14),
                            tabs: [
                              Tab(text: AppLocalizations.of(context)!.student),
                              Tab(text: AppLocalizations.of(context)!.parent),
                            ],
                          ), // TabBar
                          const SizedBox(height: 10),
                          // Sign-up form
                          Row(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              const SizedBox(width: 5),
```

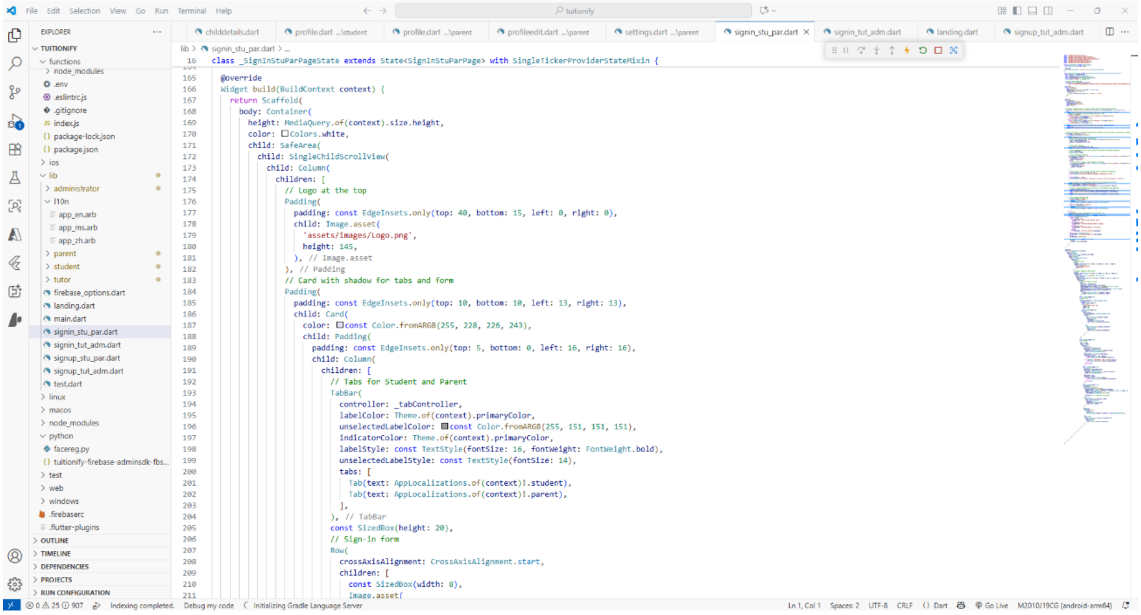
Figure 4.1.1.1 Screenshot of signup_stu_par.dart



```dart
class _SignUpTutAdmPageState extends State<SignUpTutAdmPage>

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        height: MediaQuery.of(context).size.height,
        color: Colors.white,
        child: SafeArea(
          child: SingleChildScrollView(
            child: Column(
              children: [
                // Logo at the top
                Padding(
                  padding: const EdgeInsets.only(top: 15, bottom: 0, left: 0, right: 0),
                  child: Image.asset(
                    'assets/images/Logo.png',
                    height: 145,
                  ), // Image.asset
                ), // Padding
                // Card with shadow for tabs and form
                Padding(
                  padding: const EdgeInsets.only(top: 10, bottom: 10, left: 13, right: 13),
                  child: Card(
                    color: const Color.fromARGB(255, 228, 226, 243),
                    child: Padding(
                      padding: const EdgeInsets.only(top: 5, bottom: 0, left: 16, right: 16),
                      child: Column(
                        children: [
                          // Tabs for Tutor and Administrator
                          TabBar(
                            controller: _tabController,
                            labelColor: Theme.of(context).primaryColor,
                            unselectedLabelColor: const Color.fromARGB(255, 151, 151, 151),
                            indicatorColor: Theme.of(context).primaryColor,
                            labelStyle: const TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
                            unselectedLabelStyle: const TextStyle(fontSize: 14),
                            tabs: [
                              Tab(text: AppLocalizations.of(context)!.tutor),
                              Tab(text: AppLocalizations.of(context)!.administrator),
                            ],
                          ), // TabBar
                          const SizedBox(height: 10),
                          // Sign-up form
                          Row(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              const SizedBox(width: 5),
                              Image.asset(
```

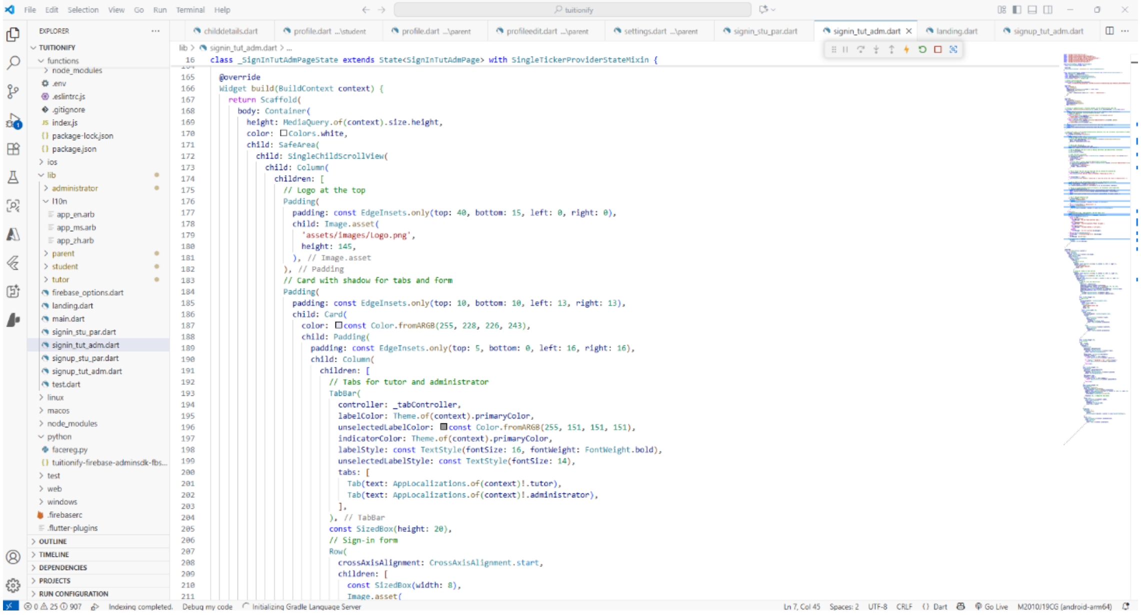
Figure 4.1.1.2 Screenshot of signup_tut_adm.dart

Furthermore, the Sign In module allows administrators, tutors, students, and parents to access their personal account by selecting their role, such as “Student & Parent” or “Staff”, as shown in Figure 4.1.1.3 and Figure 4.1.1.4. They need to enter their email address and password, and Firebase Authentication will verify the credentials against the database by ensuring the email address exists, the password is correct, and the selected role matches. Once the account is verified, a message “Sign in successfully” will be displayed; otherwise, error messages such as “Role mismatch” and “The supplied auth credential is incorrect” will be shown.



```
class _SignInStuParPageState extends State<SignInStuParPage> with SingleTickerProviderStateMixin {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        height: MediaQuery.of(context).size.height,
        color: Colors.white,
        child: SafeArea(
          child: SingleChildScrollView(
            child: Column(
              children: [
                // Logo at the top
                Padding(
                  padding: const EdgeInsets.only(top: 40, bottom: 15, left: 0, right: 0),
                  child: Image.asset(
                    'assets/images/Logo.png',
                    height: 145,
                  ), // Image.asset
                ), // Padding
                // Card with shadow for tabs and form
                Padding(
                  padding: const EdgeInsets.only(top: 10, bottom: 10, left: 13, right: 13),
                  child: Card(
                    color: const Color.fromARGB(255, 228, 226, 243),
                    child: Padding(
                      padding: const EdgeInsets.only(top: 5, bottom: 0, left: 16, right: 16),
                      child: Column(
                        children: [
                          // Tabs for Student and Parent
                          TabBar(
                            controller: _tabController,
                            labelColor: Theme.of(context).primaryColor,
                            unselectedLabelColor: const Color.fromARGB(255, 151, 151, 151),
                            indicatorColor: Theme.of(context).primaryColor,
                            labelStyle: const TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
                            unselectedLabelStyle: const TextStyle(fontSize: 14),
                            tabs: [
                              Tab(text: AppLocalizations.of(context)!.student),
                              Tab(text: AppLocalizations.of(context)!.parent),
                            ],
                          ), // TabBar
                          const SizedBox(height: 20),
                          // Sign-in form
                          Row(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              const SizedBox(width: 8),
                              Image.asset(
```

Figure 4.1.1.3 Screenshot of signin_stu_par.dart

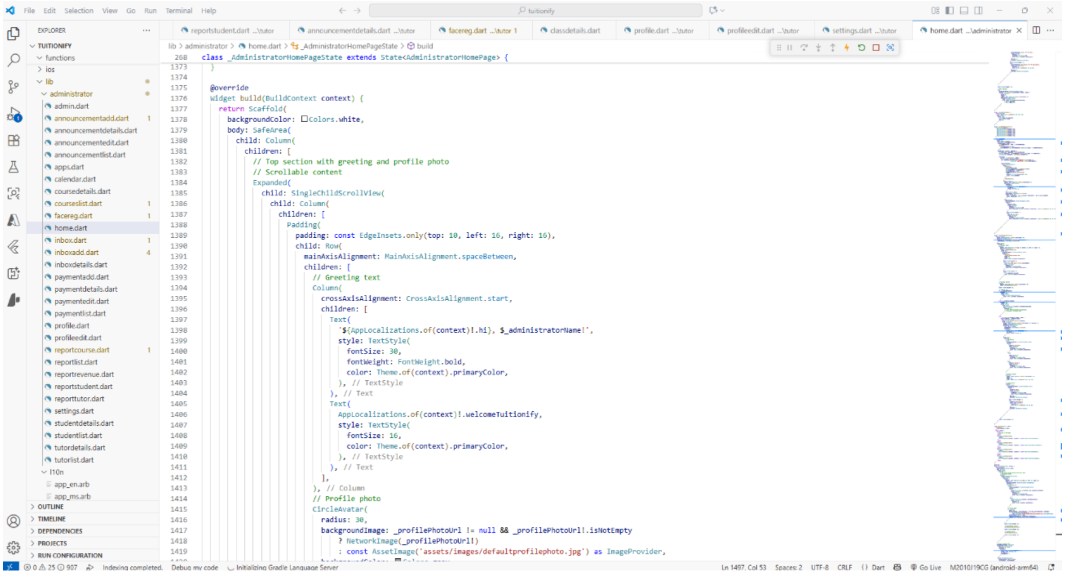


```
class _SignInTutAdmPageState extends State<SignInTutAdmPage> with SingleTickerProviderStateMixin {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        height: MediaQuery.of(context).size.height,
        color: Colors.white,
        child: SafeArea(
          child: SingleChildScrollView(
            child: Column(
              children: [
                // Logo at the top
                Padding(
                  padding: const EdgeInsets.only(top: 40, bottom: 15, left: 0, right: 0),
                  child: Image.asset(
                    'assets/images/Logo.png',
                    height: 145,
                  ), // Image.asset
                ), // Padding
                // Card with shadow for tabs and form
                Padding(
                  padding: const EdgeInsets.only(top: 10, bottom: 10, left: 13, right: 13),
                  child: Card(
                    color: const Color.fromARGB(255, 228, 226, 243),
                    child: Padding(
                      padding: const EdgeInsets.only(top: 5, bottom: 0, left: 16, right: 16),
                      child: Column(
                        children: [
                          // Tabs for tutor and administrator
                          TabBar(
                            controller: _tabController,
                            labelColor: Theme.of(context).primaryColor,
                            unselectedLabelColor: const Color.fromARGB(255, 151, 151, 151),
                            indicatorColor: Theme.of(context).primaryColor,
                            labelStyle: const TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
                            unselectedLabelStyle: const TextStyle(fontSize: 14),
                            tabs: [
                              Tab(text: AppLocalizations.of(context)!.tutor),
                              Tab(text: AppLocalizations.of(context)!.administrator),
                            ],
                          ), // TabBar
                          const SizedBox(height: 20),
                          // Sign-in form
                          Row(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              const SizedBox(width: 8),
                              Image.asset(
```

Figure 4.1.1.4 Screenshot of signin_tut_adm.dart

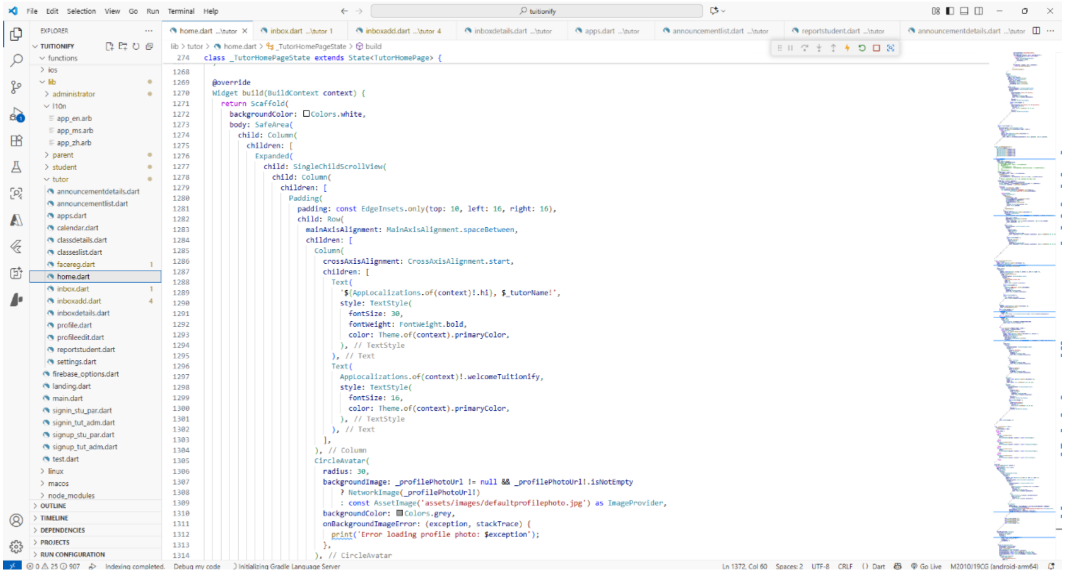
4.1.2 Home Module Development

The home.dart file, as shown in Figures 4.1.2.1 to 4.1.2.4, serves as the central dashboard for the application, which provides all users, including administrators, tutors, students, and parents, with important information. This module starts with an announcement section that will display the five latest announcements from the Cloud Firestore Database. Each announcement is clickable and displays with a photo, title, and description. Below the announcements section, the calendar schedule section will display the daily class schedules, which will be retrieved from Cloud Firestore. The class schedules are smartly grouped in chronological order depending on the start time to allow user to understand their daily timeline. Each class schedule is clickable to view the detailed information, such as course name, course description, tutor name, duration, date, etc. Furthermore, the module ends with a role-specific bottom navigation bar with 5 different bars. For example, students and tutors navigate through Home, Inbox, Apps, Class, and Profile; Parents access Home, Inbox, Apps, Child, and Profile; while Administrators utilise Home, Inbox, Apps, Admins, and Profile.



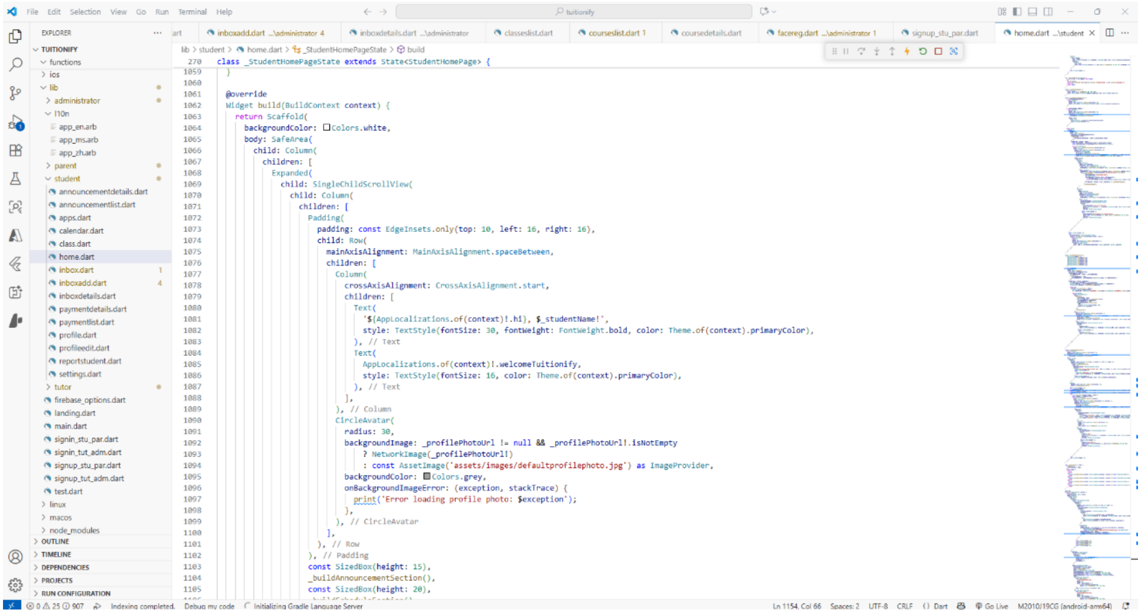
```dart
class _AdministratorHomePageState extends State<AdministratorHomePage> {
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          children: [
            // Top section with greeting and profile photo
            // Scrollable content
            Expanded(
              child: SingleChildScrollView(
                child: Column(
                  children: [
                    Padding(
                      padding: const EdgeInsets.only(top: 10, left: 16, right: 16),
                      child: Row(
                        mainAxisAlignment: MainAxisAlignment.spaceBetween,
                        children: [
                          // Greeting text
                          Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              Text(
                                '${AppLocalizations.of(context)!.hi}, $_administratorName!',
                                style: TextStyle(
                                  fontSize: 30,
                                  fontWeight: FontWeight.bold,
                                  color: Theme.of(context).primaryColor,
                                ), // TextStyle
                              ), // Text
                              Text(
                                AppLocalizations.of(context)!.welcomeTuitionify,
                                style: TextStyle(
                                  fontSize: 16,
                                  color: Theme.of(context).primaryColor,
                                ), // TextStyle
                              ), // Text
                            ],
                          ), // Column
                          // Profile photo
                          CircleAvatar(
                            radius: 30,
                            backgroundImage: _profilePhotoUrl != null && _profilePhotoUrl!.isNotEmpty
                                ? NetworkImage(_profilePhotoUrl!)
                                : const AssetImage('assets/images/defaultprofilephoto.jpg') as ImageProvider,
```

Figure 4.1.2.1 Screenshot of administrator/home.dart



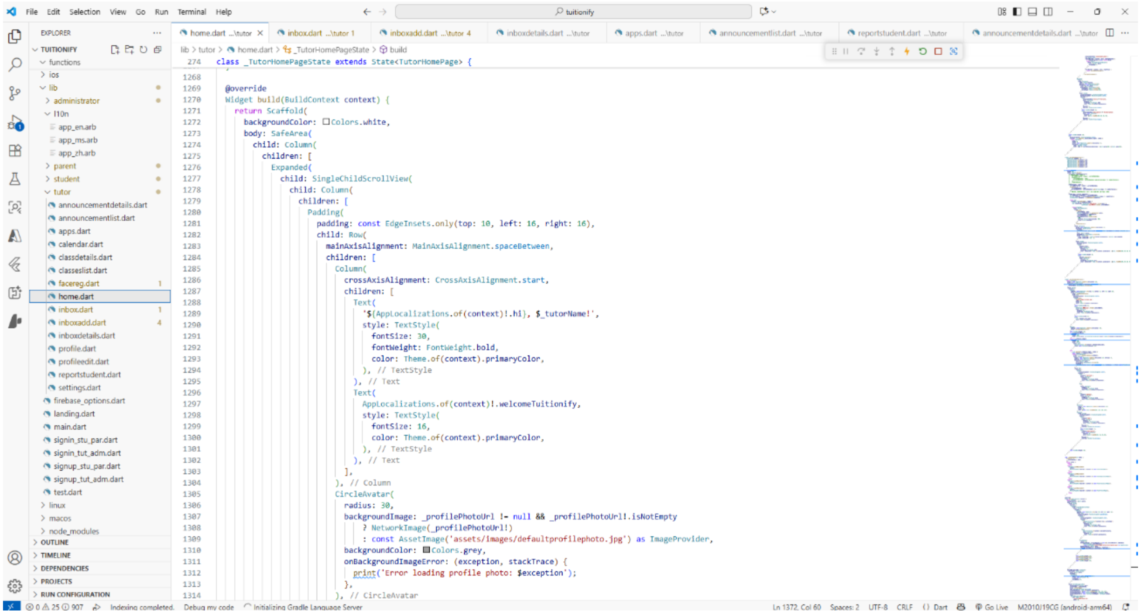
```dart
class _TutorHomePageState extends State<TutorHomePage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          children: [
            Expanded(
              child: SingleChildScrollView(
                child: Column(
                  children: [
                    Padding(
                      padding: const EdgeInsets.only(top: 10, left: 16, right: 16),
                      child: Row(
                        mainAxisAlignment: MainAxisAlignment.spaceBetween,
                        children: [
                          Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              Text(
                                '${AppLocalizations.of(context)!.hi}, $_tutorName!',
                                style: TextStyle(
                                  fontSize: 30,
                                  fontWeight: FontWeight.bold,
                                  color: Theme.of(context).primaryColor,
                                ), // TextStyle
                              ), // Text
                              Text(
                                AppLocalizations.of(context)!.welcomeTuitionify,
                                style: TextStyle(
                                  fontSize: 16,
                                  color: Theme.of(context).primaryColor,
                                ), // TextStyle
                              ), // Text
                            ],
                          ), // Column
                          CircleAvatar(
                            radius: 30,
                            backgroundImage: _profilePhotoUrl != null && _profilePhotoUrl!.isNotEmpty
                                ? NetworkImage(_profilePhotoUrl!)
                                : const AssetImage('assets/images/defaultprofilephoto.jpg') as ImageProvider,
                            backgroundColor: Colors.grey,
                            onBackgroundImageError: (exception, stackTrace) {
                              print('Error loading profile photo: $exception');
                            },
                          ), // CircleAvatar
```

Figure 4.1.2.2 Screenshot of tutor/home.dart



```
class _StudentHomePageState extends State<StudentHomePage> {
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: □Colors.white,
      body: SafeArea(
        child: Column(
          children: [
            Expanded(
              child: SingleChildScrollView(
                child: Column(
                  children: [
                    Padding(
                      padding: const EdgeInsets.only(top: 10, left: 16, right: 16),
                      child: Row(
                        mainAxisAlignment: MainAxisAlignment.spaceBetween,
                        children: [
                          Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              Text(
                                '${AppLocalizations.of(context)!.hi}, $_studentName!',
                                style: TextStyle(fontSize: 30, fontWeight: FontWeight.bold, color: Theme.of(context).primaryColor),
                              ), // Text
                              Text(
                                AppLocalizations.of(context)!.welcomeTuitionify,
                                style: TextStyle(fontSize: 16, color: Theme.of(context).primaryColor),
                              ), // Text
                            ],
                          ), // Column
                          CircleAvatar(
                            radius: 30,
                            backgroundImage: _profilePhotoUrl != null && _profilePhotoUrl!.isNotEmpty
                                ? NetworkImage(_profilePhotoUrl!)
                                : const AssetImage('assets/images/defaultprofilephoto.jpg') as ImageProvider,
                            backgroundColor: ■Colors.grey,
                            onBackgroundImageError: (exception, stackTrace) {
                              print('Error loading profile photo: $exception');
                            },
                          ), // CircleAvatar
                        ],
                      ), // Row
                    ), // Padding
                    const SizedBox(height: 15),
                    _buildAnnouncementSection(),
                    const SizedBox(height: 20),
```

Figure 4.1.2.3 Screenshot of student/home.dart

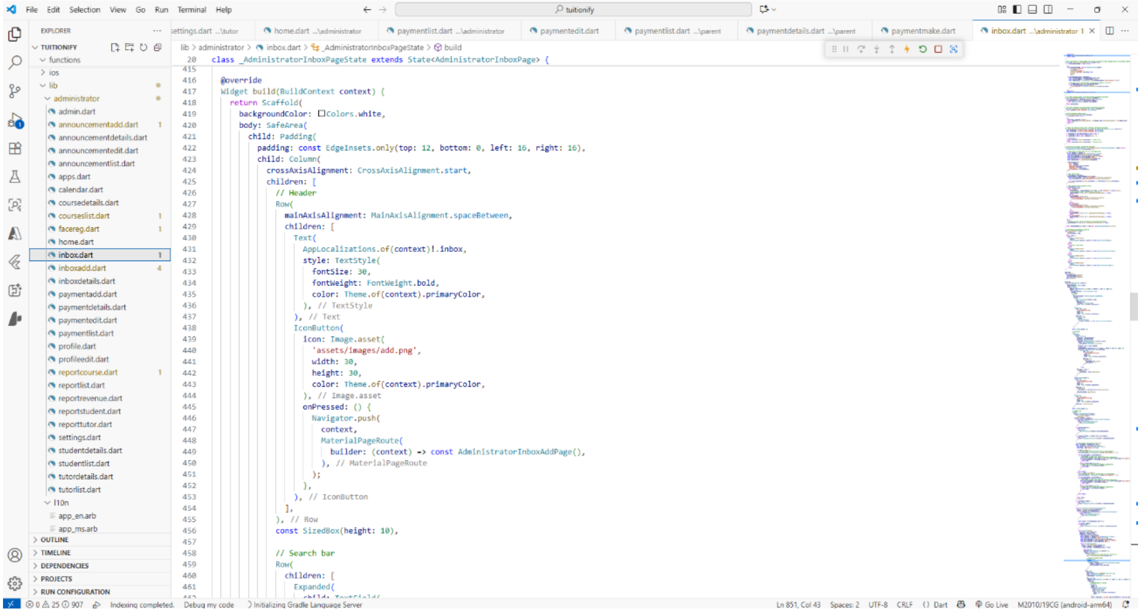


```
class _TutorHomePageState extends State<TutorHomePage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: □Colors.white,
      body: SafeArea(
        child: Column(
          children: [
            Expanded(
              child: SingleChildScrollView(
                child: Column(
                  children: [
                    Padding(
                      padding: const EdgeInsets.only(top: 10, left: 16, right: 16),
                      child: Row(
                        mainAxisAlignment: MainAxisAlignment.spaceBetween,
                        children: [
                          Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              Text(
                                '${AppLocalizations.of(context)!.hi}, $_tutorName!',
                                style: TextStyle(
                                  fontSize: 30,
                                  fontWeight: FontWeight.bold,
                                  color: Theme.of(context).primaryColor,
                                ), // TextStyle
                              ), // Text
                              Text(
                                AppLocalizations.of(context)!.welcomeTuitionify,
                                style: TextStyle(
                                  fontSize: 16,
                                  color: Theme.of(context).primaryColor,
                                ), // TextStyle
                              ), // Text
                            ],
                          ), // Column
                          CircleAvatar(
                            radius: 30,
                            backgroundImage: _profilePhotoUrl != null && _profilePhotoUrl!.isNotEmpty
                                ? NetworkImage(_profilePhotoUrl!)
                                : const AssetImage('assets/images/defaultprofilephoto.jpg') as ImageProvider,
                            backgroundColor: ■Colors.grey,
                            onBackgroundImageError: (exception, stackTrace) {
                              print('Error loading profile photo: $exception');
                            },
                          ), // CircleAvatar
```

Figure 4.1.2.4 Screenshot of parent/home.dart

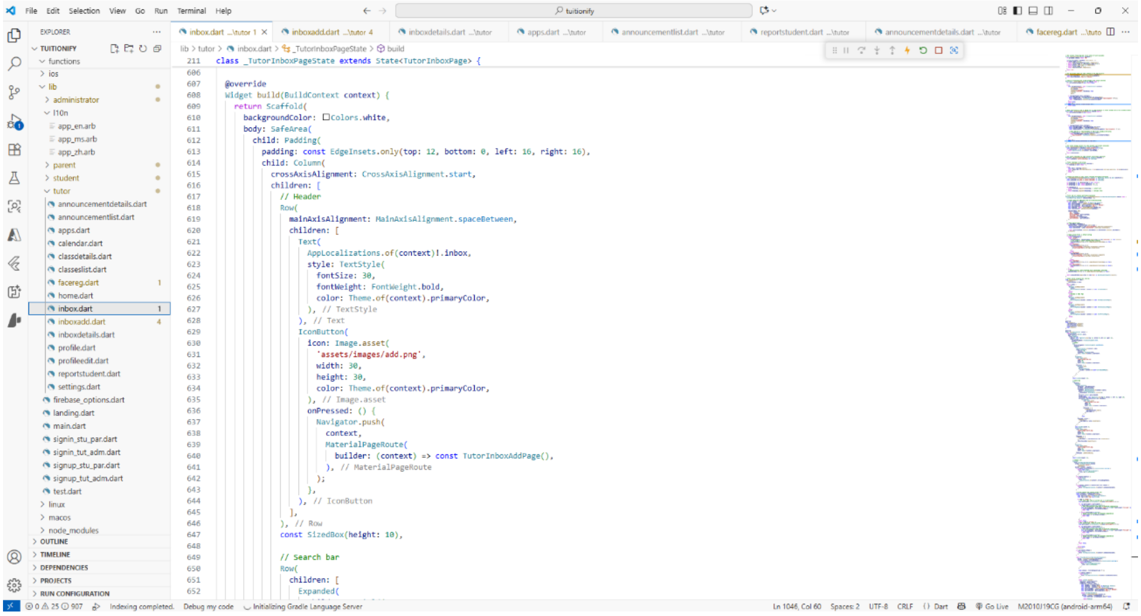
4.1.3 Inbox Module Development

According to Figures 4.1.3.1 to 4.1.3.4, the inbox module development serves as a comprehensive communication hub that brings together all messaging activities, including private chat and group chat. The module displays the chat conversation sorted in descending chronological order, allowing administrators, tutors, students, and parents to find the most recent conversation at the top of the list. When users select any of the chat boxes, they are seamlessly navigated to the complete chat conversation history and allowed to send new text messages or upload new images. To maintain an effective communication flow, the system will send an email notification to all respective chat participants when a new message is received, as shown in Figure 4.1.3.5. This module also provides robust search and sorting capabilities, which allow users to find specific chats using keyword searches or organize the chats using various sorting options, such as Unread, Ascending Order, Descending Order, Latest Date, and Earliest Date. Additionally, it supports the initiation of new private conversations that allow users to start new conversations with specific individuals by clicking on their names. Once the message is successfully sent, the recipient will receive an email notification to alert them of incoming messages.



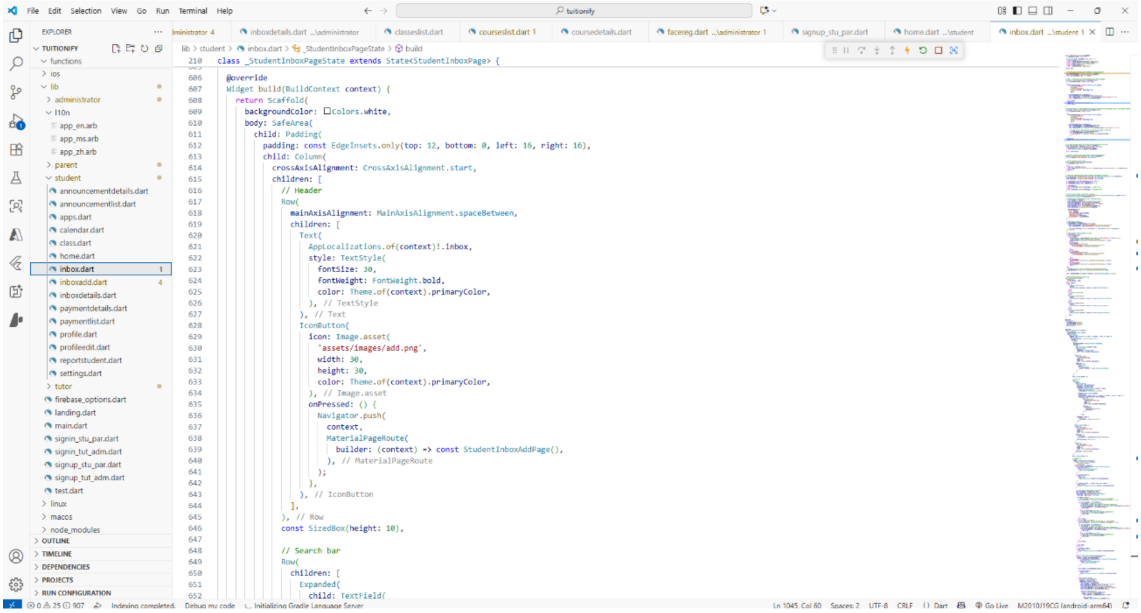
```
class _AdministratorInboxPageState extends State<AdministratorInboxPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 0, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Text(
                    AppLocalizations.of(context)!.inbox,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                  IconButton(
                    icon: Image.asset(
                      'assets/images/add.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const AdministratorInboxAddPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                ],
              ), // Row
              const SizedBox(height: 10),

              // Search bar
              Row(
                children: [
                  Expanded(
                    child: TextField(
```

Figure 4.1.3.1 Screenshot of administrator/inbox.dart



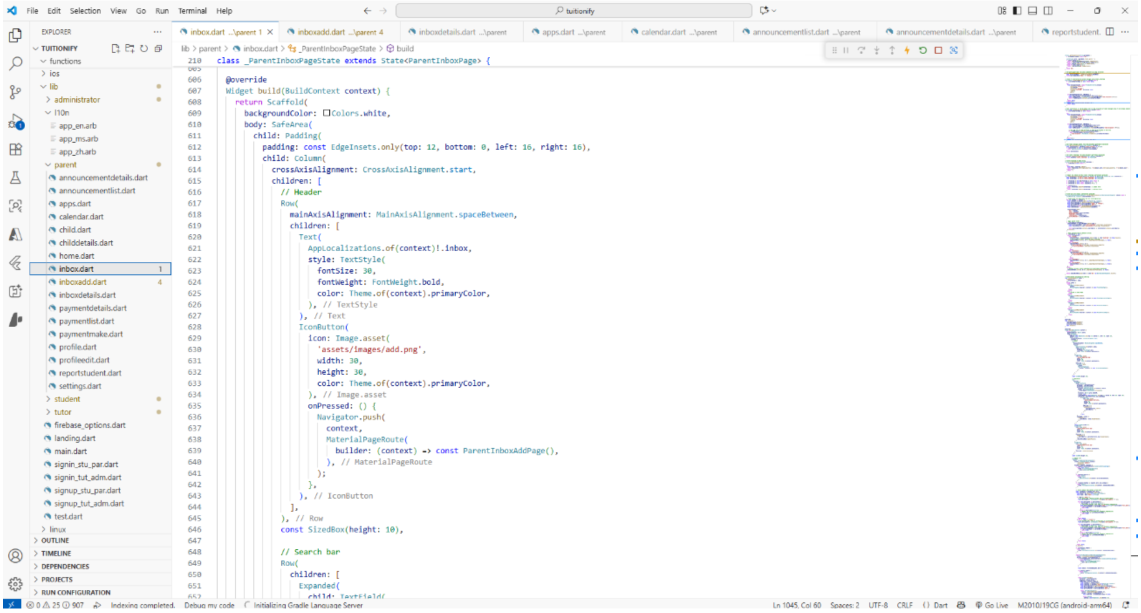
```
class _TutorInboxPageState extends State<TutorInboxPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 0, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Text(
                    AppLocalizations.of(context)!.inbox,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                  IconButton(
                    icon: Image.asset(
                      'assets/images/add.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const TutorInboxAddPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                ],
              ), // Row
              const SizedBox(height: 10),

              // Search bar
              Row(
                children: [
                  Expanded(
```

Figure 4.1.3.2 Screenshot of tutor/inbox.dart



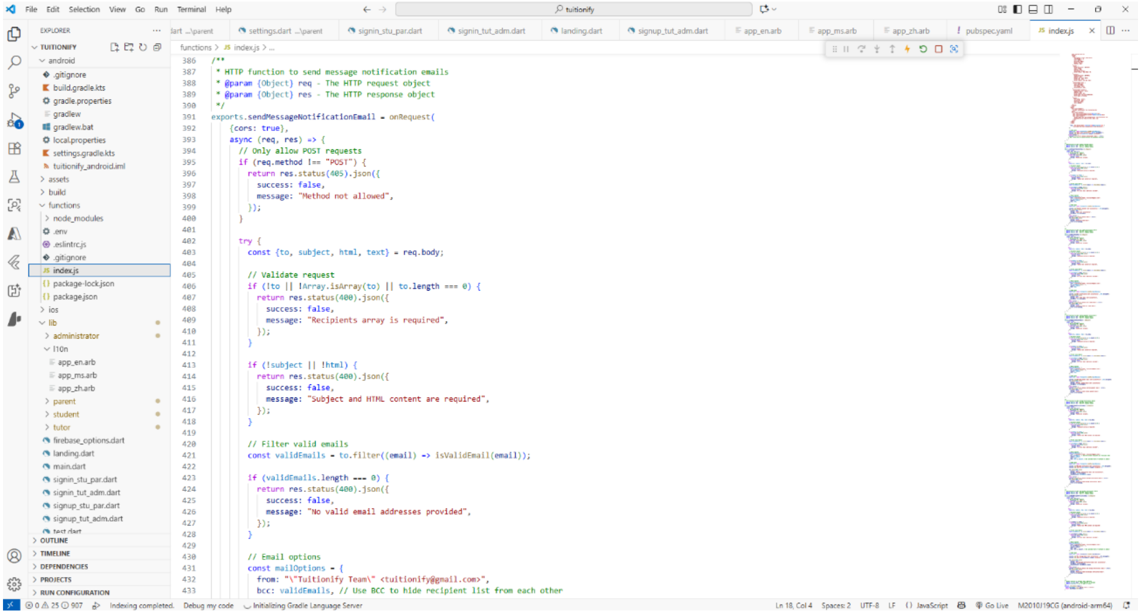
```
class _StudentInboxPageState extends State<StudentInboxPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 0, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Text(
                    AppLocalizations.of(context)!.inbox,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                  IconButton(
                    icon: Image.asset(
                      'assets/images/add.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const StudentInboxAddPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                ],
              ), // Row
              const SizedBox(height: 10),

              // Search bar
              Row(
                children: [
                  Expanded(
                    child: TextField(
```

Figure 4.1.3.3 Screenshot of student/inbox.dart



```dart
class _ParentInboxPageState extends State<ParentInboxPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 0, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Text(
                    AppLocalizations.of(context)!.inbox,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                  IconButton(
                    icon: Image.asset(
                      'assets/images/add.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const ParentInboxAddPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                ],
              ), // Row
              const SizedBox(height: 10),

              // Search bar
              Row(
                children: [
                  Expanded(
                    child: TextField(
```

Figure 4.1.3.4 Screenshot of parent/inbox.dart



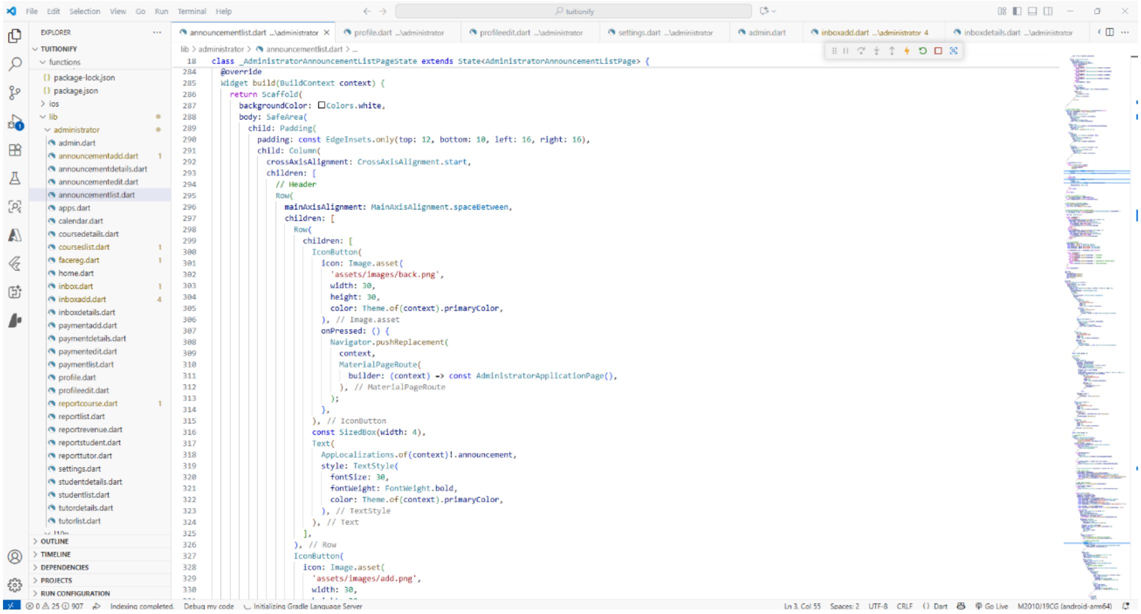
```javascript
/**
 * HTTP function to send message notification emails
 * @param {Object} req - The HTTP request object
 * @param {Object} res - The HTTP response object
 */
exports.sendMessageNotificationEmail = onRequest(
    {cors: true},
    async (req, res) => {
      // Only allow POST requests
      if (req.method !== "POST") {
        return res.status(405).json({
          success: false,
          message: "Method not allowed",
        });
      }

      try {
        const {to, subject, html, text} = req.body;

        // Validate request
        if (!to || !Array.isArray(to) || to.length === 0) {
          return res.status(400).json({
            success: false,
            message: "Recipients array is required",
          });
        }

        if (!subject || !html) {
          return res.status(400).json({
            success: false,
            message: "Subject and HTML content are required",
          });
        }

        // Filter valid emails
        const validEmails = to.filter((email) => isValidEmail(email));

        if (validEmails.length === 0) {
          return res.status(400).json({
            success: false,
            message: "No valid email addresses provided",
          });
        }

        // Email options
        const mailOptions = {
          from: "\"Tuitionify Team\" <tuitionify@gmail.com>",
          bcc: validEmails, // Use BCC to hide recipient list from each other
```

Figure 4.1.3.5 Screenshot of sendMessageNotificationEmail in functions/index.js

4.1.4 Announcement Module Development

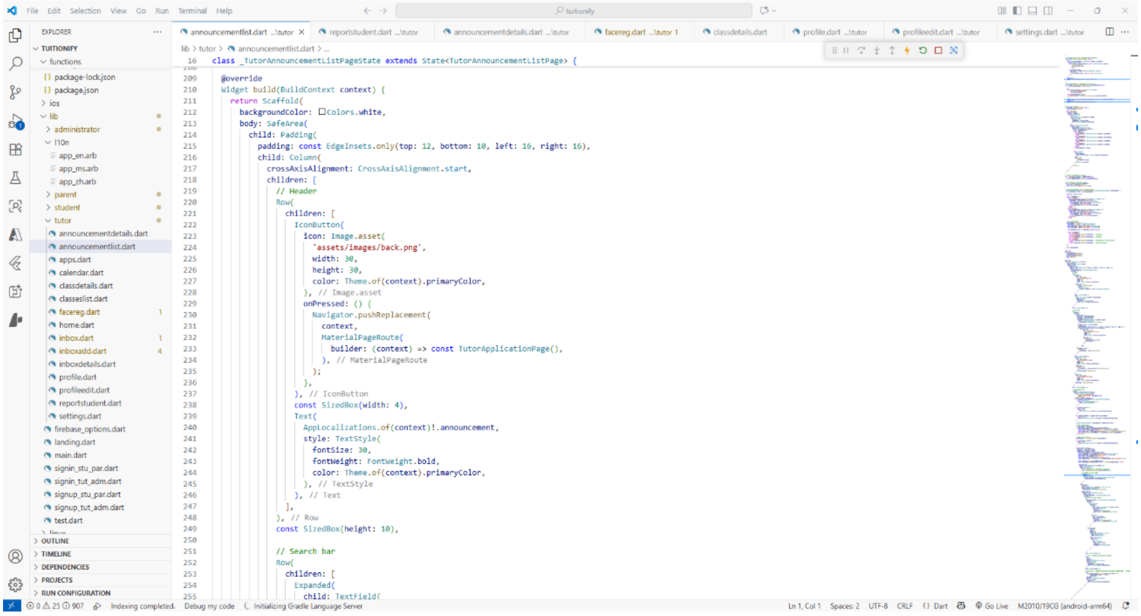
The announcementlist.dart file shown in Figures 4.1.4.1 to 4.1.4.4 is a centralised information distribution system that displays all announcements in a descending chronological order, so all the users, such as administrators, tutors, students, and parents, can always access the most recent updates. Each announcement is presented as an interactive element that users may click to access comprehensive details, including full title, description, photos, and publication date.

This module also includes powerful search and filtering capabilities, which allow users to search for specific announcements using keyword searches or sort the announcements using various sorting options, such as Latest Date, Earliest Date, Last 30 Days, and Last 7 Days, based on their specific needs. Furthermore, administrators are allowed to access comprehensive management capabilities, which allow them to create new announcements and store them in the Cloud Firestore or edit existing announcements to ensure all information is accurate and up-to-date. An email notification will also be sent to all users when a new announcement has been created, as illustrated in Figure 4.1.4.5.



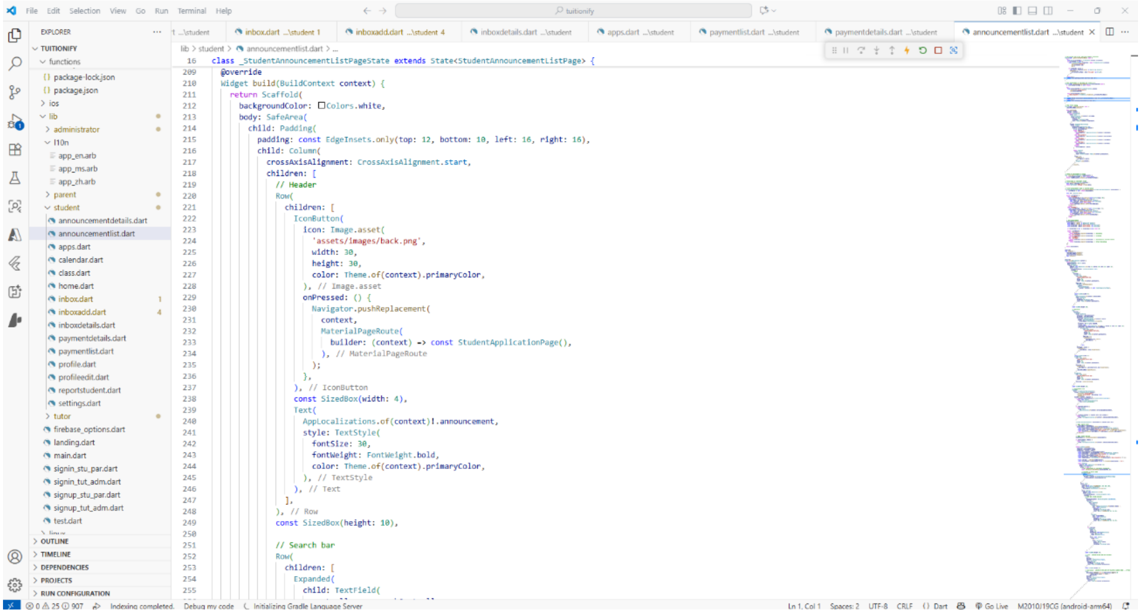
```dart
class _AdministratorAnnouncementListPageState extends State<AdministratorAnnouncementListPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 10, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Row(
                    children: [
                      IconButton(
                        icon: Image.asset(
                          'assets/images/back.png',
                          width: 30,
                          height: 30,
                          color: Theme.of(context).primaryColor,
                        ), // Image.asset
                        onPressed: () {
                          Navigator.pushReplacement(
                            context,
                            MaterialPageRoute(
                              builder: (context) => const AdministratorApplicationPage(),
                            ), // MaterialPageRoute
                          );
                        },
                      ), // IconButton
                      const SizedBox(width: 4),
                      Text(
                        AppLocalizations.of(context)!.announcement,
                        style: TextStyle(
                          fontSize: 30,
                          fontWeight: FontWeight.bold,
                          color: Theme.of(context).primaryColor,
                        ), // TextStyle
                      ), // Text
                    ],
                  ), // Row
                  IconButton(
                    icon: Image.asset(
                      'assets/images/add.png',
                      width: 30,
```

Figure 4.1.4.1 Screenshot of administrator/announcementlist.dart



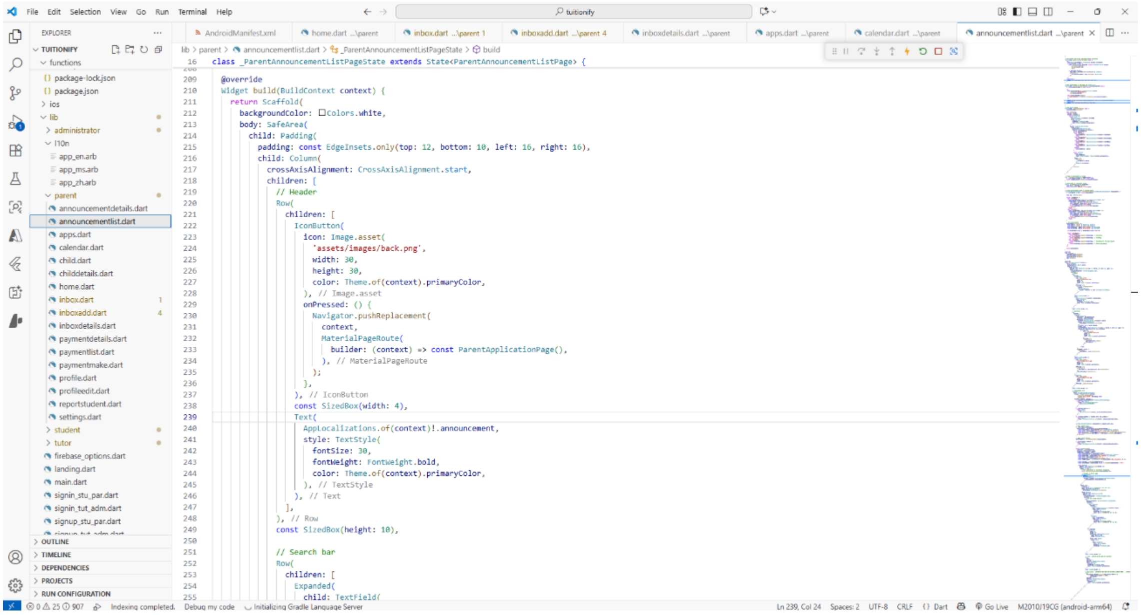
```
class _TutorAnnouncementListPageState extends State<TutorAnnouncementListPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 10, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const TutorApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Text(
                    AppLocalizations.of(context)!.announcement,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                ],
              ), // Row
              const SizedBox(height: 10),

              // Search bar
              Row(
                children: [
                  Expanded(
                    child: TextField(
```

Figure 4.1.4.2 Screenshot of tutor/announcementlist.dart



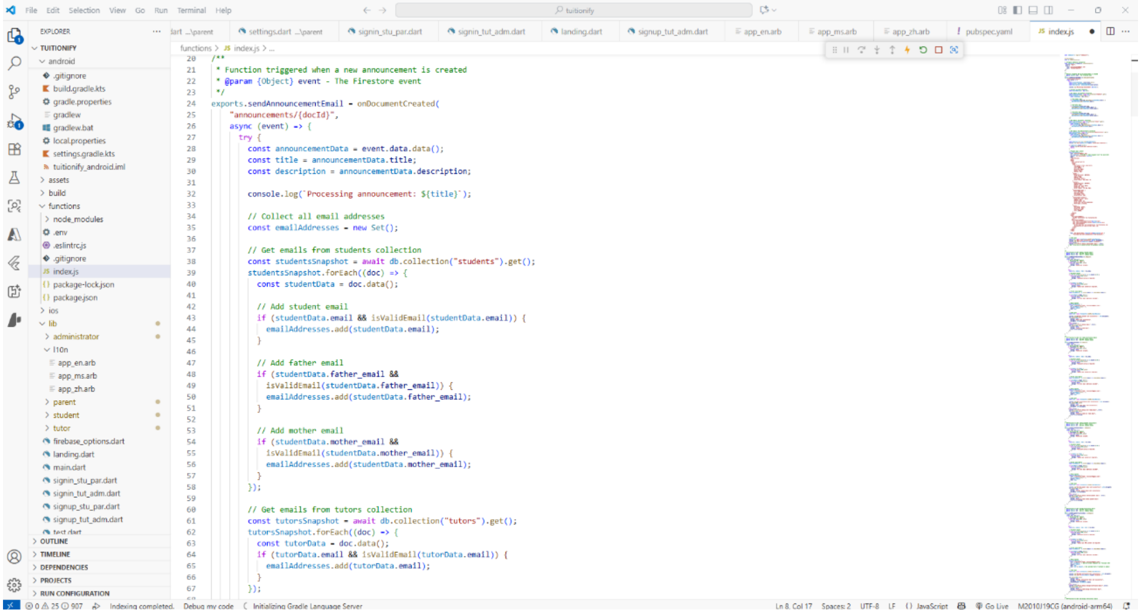
```
class _StudentAnnouncementListPageState extends State<StudentAnnouncementListPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 10, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const StudentApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Text(
                    AppLocalizations.of(context)!.announcement,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                ],
              ), // Row
              const SizedBox(height: 10),

              // Search bar
              Row(
                children: [
                  Expanded(
                    child: TextField(
```

Figure 4.1.4.3 Screenshot of student/announcementlist.dart


File Edit Selection View Go Run Terminal Help
tuitionify
EXPLORER
TUITIONIFY
AndroidManifest.xml
home.dart ...\parent
inbox.dart ...\parent 1
inboxadd.dart ...\parent 4
inboxdetails.dart ...\parent
apps.dart ...\parent
calendar.dart ...\parent
announcementlist.dart ...\parent
lib > parent > announcementlist.dart > _ParentAnnouncementListPageState > build
Ln 239, Col 24 Spaces: 2 UTF-8 CRLF {} Dart Go Live
Indexing completed. Debug my code Initializing Gradle Language Server

```dart
class _ParentAnnouncementListPageState extends State<ParentAnnouncementListPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 10, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const ParentApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Text(
                    AppLocalizations.of(context)!.announcement,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                ],
              ), // Row
              const SizedBox(height: 10),

              // Search bar
              Row(
                children: [
                  Expanded(
                    child: TextField(
```

Figure 4.1.4.4 Screenshot of parent/announcementlist.dart

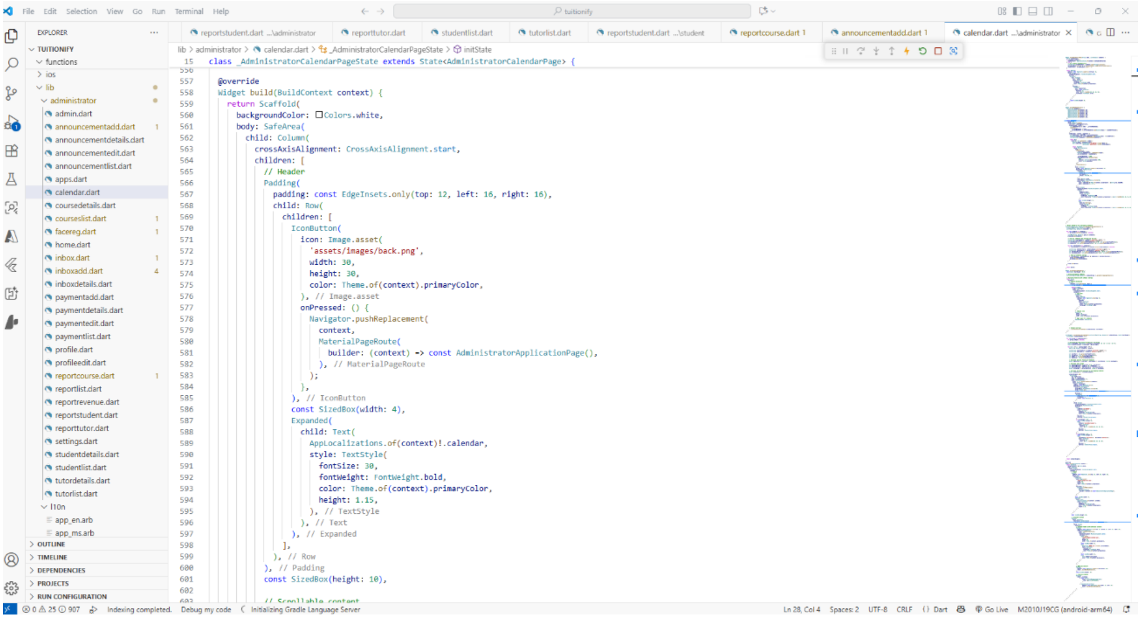

File Edit Selection View Go Run Terminal Help
tuitionify
EXPLORER
TUITIONIFY
dart ...\parent
settings.dart ...\parent
signin_stu_par.dart
signin_tut_adm.dart
landing.dart
signup_tut_adm.dart
app_en.arb
app_ms.arb
app_zh.arb
pubspec.yaml
index.js
functions > index.js > ...
Ln 8, Col 17 Spaces: 2 UTF-8 LF {} JavaScript Go Live
Indexing completed. Debug my code Initializing Gradle Language Server

```javascript
/**
 * Function triggered when a new announcement is created
 * @param {Object} event - The Firestore event
 */
exports.sendAnnouncementEmail = onDocumentCreated(
    "announcements/{docId}",
    async (event) => {
      try {
        const announcementData = event.data.data();
        const title = announcementData.title;
        const description = announcementData.description;

        console.log(`Processing announcement: ${title}`);

        // Collect all email addresses
        const emailAddresses = new Set();

        // Get emails from students collection
        const studentsSnapshot = await db.collection("students").get();
        studentsSnapshot.forEach((doc) => {
          const studentData = doc.data();

          // Add student email
          if (studentData.email && isValidEmail(studentData.email)) {
            emailAddresses.add(studentData.email);
          }

          // Add father email
          if (studentData.father_email &&
            isValidEmail(studentData.father_email)) {
            emailAddresses.add(studentData.father_email);
          }

          // Add mother email
          if (studentData.mother_email &&
            isValidEmail(studentData.mother_email)) {
            emailAddresses.add(studentData.mother_email);
          }
        });

        // Get emails from tutors collection
        const tutorsSnapshot = await db.collection("tutors").get();
        tutorsSnapshot.forEach((doc) => {
          const tutorData = doc.data();
          if (tutorData.email && isValidEmail(tutorData.email)) {
            emailAddresses.add(tutorData.email);
          }
        });
```

Figure 4.1.4.5 Screenshot of sendAnnouncementEmail in functions/index.js

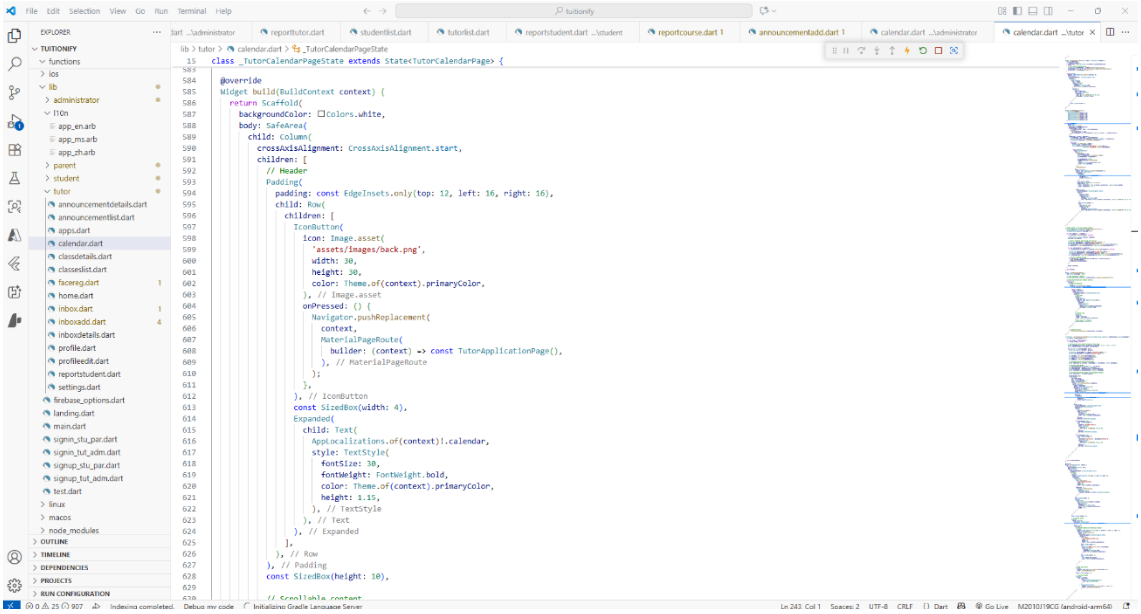
4.1.5 Calendar Module Development

This module is designed for administrators, tutors, students, and parents to have a clear visual depiction of their academic timeline, which will organise all class schedules by specified dates, as shown in Figures 4.1.5.1 to 4.1.5.4. The module improves the user experience by including horizontal scrolling functionality or a calendar icon for date navigation to allow users to navigate through different dates seamlessly. The calendar icon allows direct navigation to specific dates for viewing targeted class schedules without extensive scrolling. Each class schedule is clickable to reveal detailed information, such as course name, course descriptions, assigned tutor name, duration, dates, etc. The system employs role-based access control, which allows administrators to have comprehensive access to all existing class schedules, tutors can access class schedules specific to the courses they are assigned, students have access to view class schedules for all courses they have registered, and parents can monitor all class schedules related to their children’s registered courses. This module allows users access to all the information they need for proper class preparation.



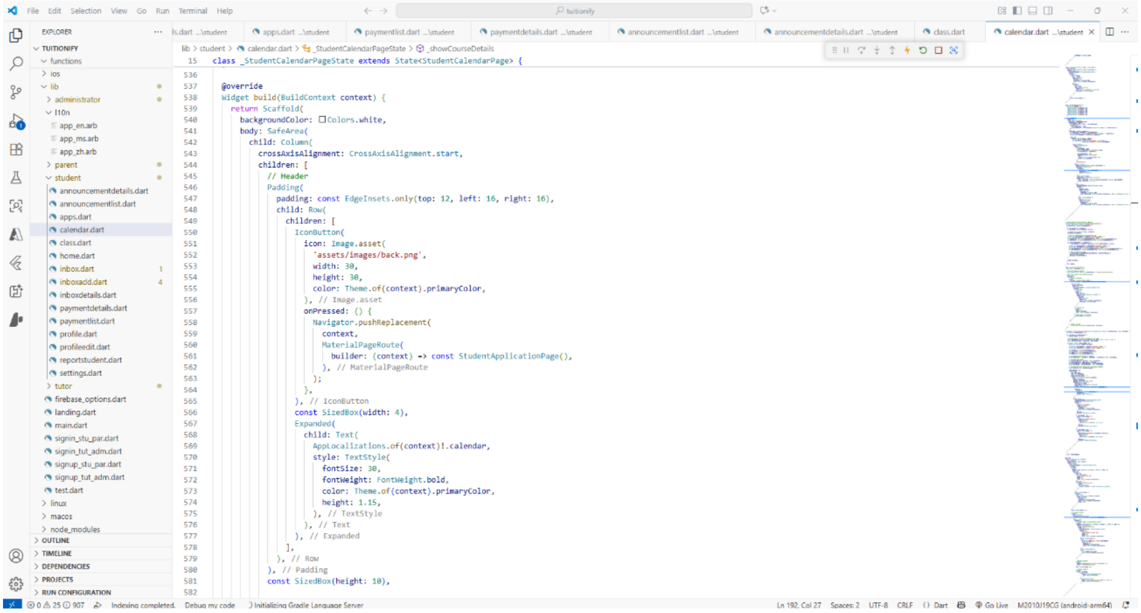
```
class _AdministratorCalendarPageState extends State<AdministratorCalendarPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Header
            Padding(
              padding: const EdgeInsets.only(top: 12, left: 16, right: 16),
              child: Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const AdministratorApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Expanded(
                    child: Text(
                      AppLocalizations.of(context)!.calendar,
                      style: TextStyle(
                        fontSize: 30,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                        height: 1.15,
                      ), // TextStyle
                    ), // Text
                  ), // Expanded
                ],
              ), // Row
            ), // Padding
            const SizedBox(height: 10),

            // Scrollable content
```

Figure 4.1.5.1 Screenshot of administrator/calendar.dart



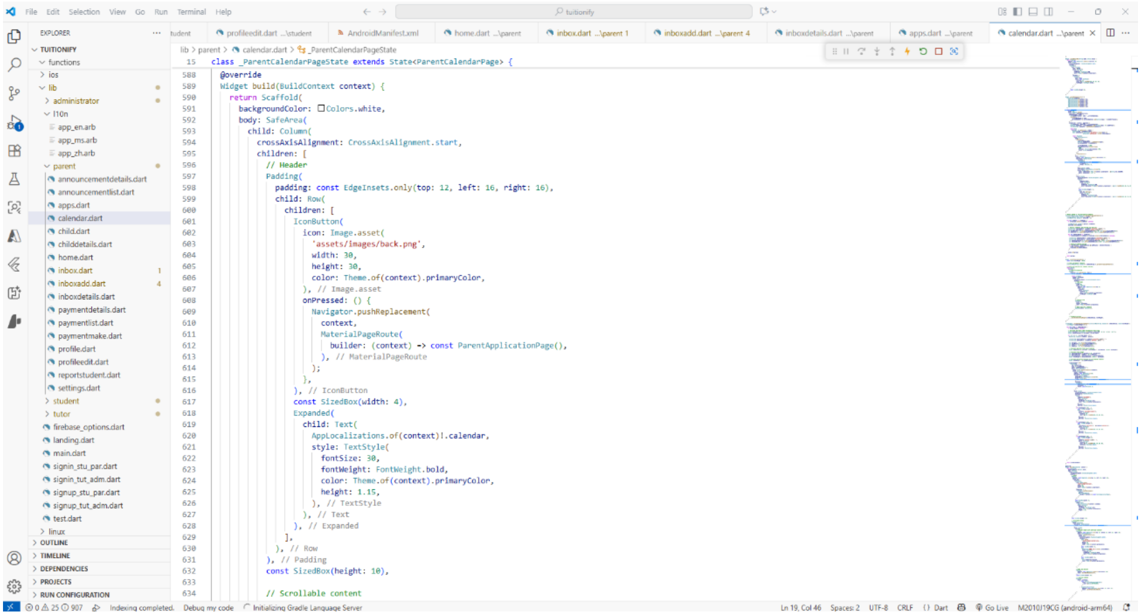
```
class _TutorCalendarPageState extends State<TutorCalendarPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Header
            Padding(
              padding: const EdgeInsets.only(top: 12, left: 16, right: 16),
              child: Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const TutorApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Expanded(
                    child: Text(
                      AppLocalizations.of(context)!.calendar,
                      style: TextStyle(
                        fontSize: 30,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                        height: 1.15,
                      ), // TextStyle
                    ), // Text
                  ), // Expanded
                ],
              ), // Row
            ), // Padding
            const SizedBox(height: 10),

            // Scrollable content
```

Figure 4.1.5.2 Screenshot of tutor/calendar.dart



```
class _StudentCalendarPageState extends State<StudentCalendarPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Header
            Padding(
              padding: const EdgeInsets.only(top: 12, left: 16, right: 16),
              child: Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const StudentApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Expanded(
                    child: Text(
                      AppLocalizations.of(context)!.calendar,
                      style: TextStyle(
                        fontSize: 30,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                        height: 1.15,
                      ), // TextStyle
                    ), // Text
                  ), // Expanded
                ],
              ), // Row
            ), // Padding
            const SizedBox(height: 10),
```

Figure 4.1.5.3 Screenshot of student/calendar.dart



```
class _ParentCalendarPageState extends State<ParentCalendarPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Header
            Padding(
              padding: const EdgeInsets.only(top: 12, left: 16, right: 16),
              child: Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const ParentApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Expanded(
                    child: Text(
                      AppLocalizations.of(context)!.calendar,
                      style: TextStyle(
                        fontSize: 30,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                        height: 1.15,
                      ), // TextStyle
                    ), // Text
                  ), // Expanded
                ],
              ), // Row
            ), // Padding
            const SizedBox(height: 10),

            // Scrollable content
```

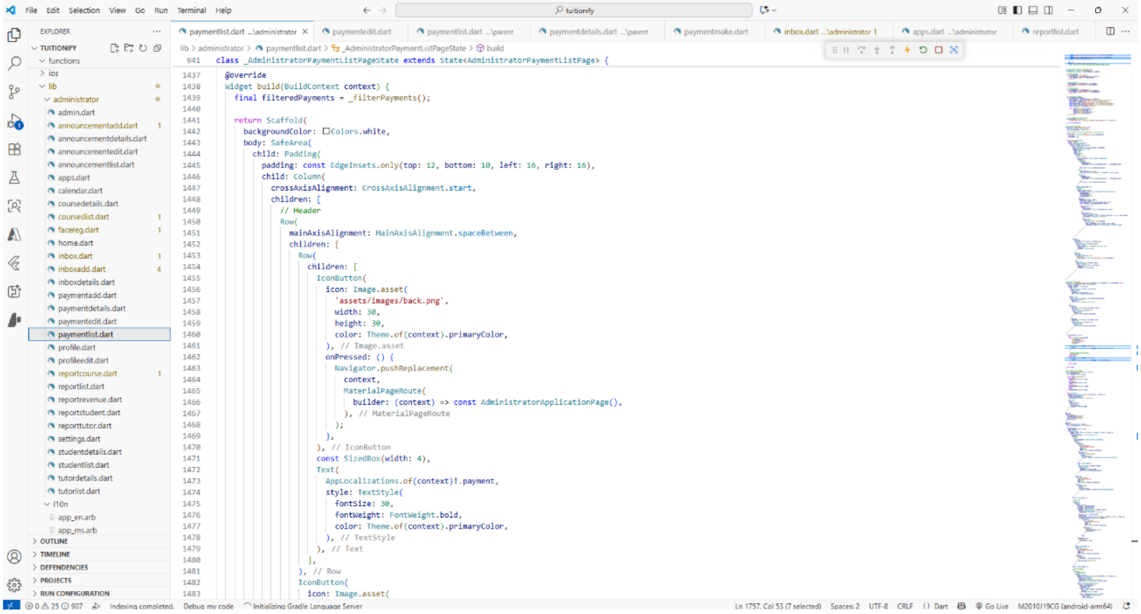
Figure 4.1.5.4 Screenshot of parent/calendar.dart

4.1.6 Payment Module Development

According to Figures 4.1.6.1 to 4.1.6.3, the payment module will display all existing student bills alphabetically by student name that can be retrieved from Cloud Firestore. Each student bill is clickable and allows administrators, students, and parents to access detailed billing information, such as billing month, payment status, list of registered courses, individual course fees, total amount, any additional notes or charges, etc. Besides, this module incorporates document management features that allow administrators, students, and parents to download invoices or receipts to their local devices for record-keeping and personal financial management. To facilitate efficient bill management, the system includes robust search and filtering capabilities that allow users to locate specific bills using keyword searches or apply filters based on criteria such as grade level, billing month, and payment status. The module also includes automated payment monitoring functionality, which tracks bill creation dates and automatically updates the payment status from "Unpaid" to "Overdue" after seven days, triggering an overdue reminder email notification to the respective students' parents to ensure timely payment follow-up, as shown in Figure 4.1.6.4.

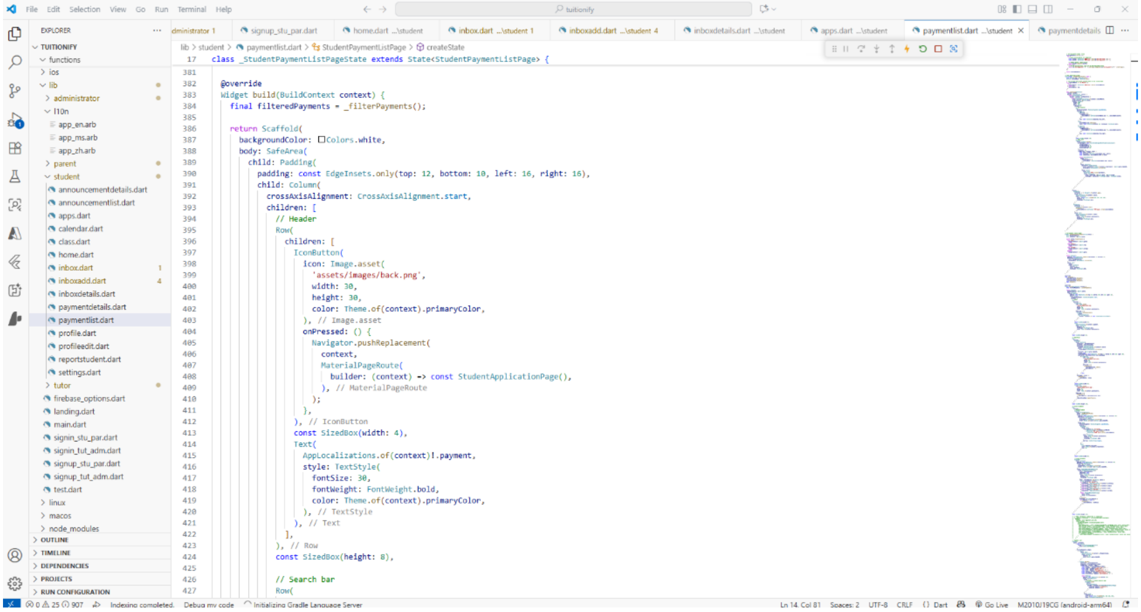
Moreover, administrators have comprehensive access to the bill creation and editing capabilities. Administrators are allowed to create new student bills and send automatic email notifications to students' parents once the bills are created, as shown in Figure 4.1.6.5.

Administrators are also allowed to modify existing bills by updating payment status, adjusting course fees, adding notes, etc. Furthermore, the module allows parents to make payments immediately upon accessing their children's student bills. Students are enabled to send a reminder email to their parents by clicking the "Notify Parents" button when the payment status is Unpaid or Overdue, as depicted in Figure 4.1.6.6.



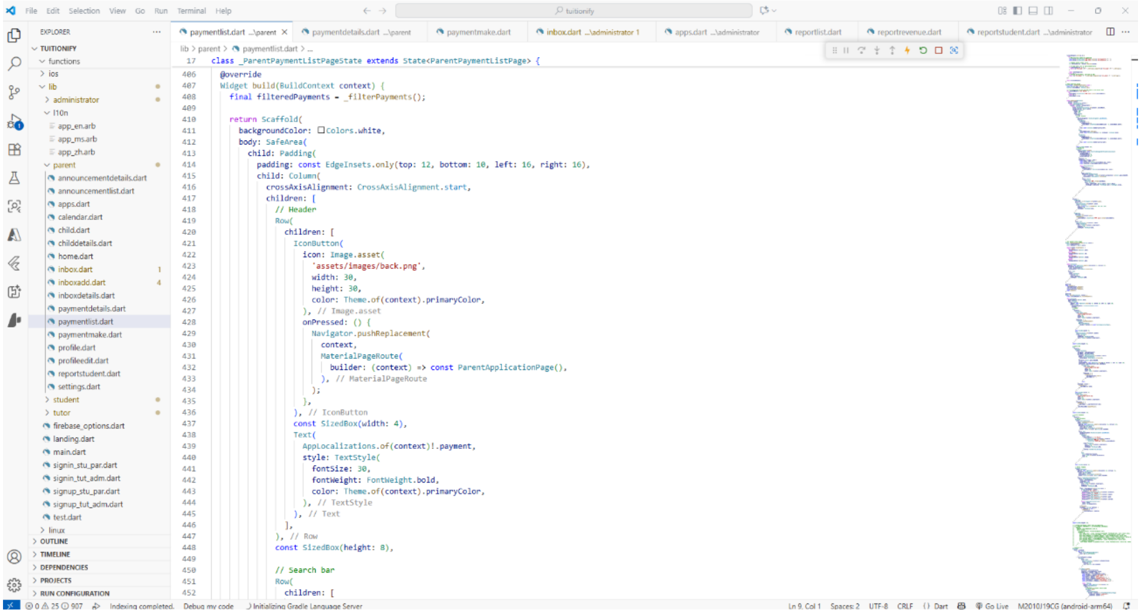
```
class _AdministratorPaymentListPageState extends State<AdministratorPaymentListPage> {
  @override
  Widget build(BuildContext context) {
    final filteredPayments = _filterPayments();

    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 10, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Row(
                    children: [
                      IconButton(
                        icon: Image.asset(
                          'assets/images/back.png',
                          width: 30,
                          height: 30,
                          color: Theme.of(context).primaryColor,
                        ), // Image.asset
                        onPressed: () {
                          Navigator.pushReplacement(
                            context,
                            MaterialPageRoute(
                              builder: (context) => const AdministratorApplicationPage(),
                            ), // MaterialPageRoute
                          );
                        },
                      ), // IconButton
                      const SizedBox(width: 4),
                      Text(
                        AppLocalizations.of(context)!.payment,
                        style: TextStyle(
                          fontSize: 30,
                          fontWeight: FontWeight.bold,
                          color: Theme.of(context).primaryColor,
                        ), // TextStyle
                      ), // Text
                    ],
                  ), // Row
                  IconButton(
                    icon: Image.asset(
```

Figure 4.1.6.1 Screenshot of administrator/paymentlist.dart



```
class _StudentPaymentListPageState extends State<StudentPaymentListPage> {

  @override
  Widget build(BuildContext context) {
    final filteredPayments = _filterPayments();

    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 10, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const StudentApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Text(
                    AppLocalizations.of(context)!.payment,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                ],
              ), // Row
              const SizedBox(height: 8),

              // Search bar
              Row(
```

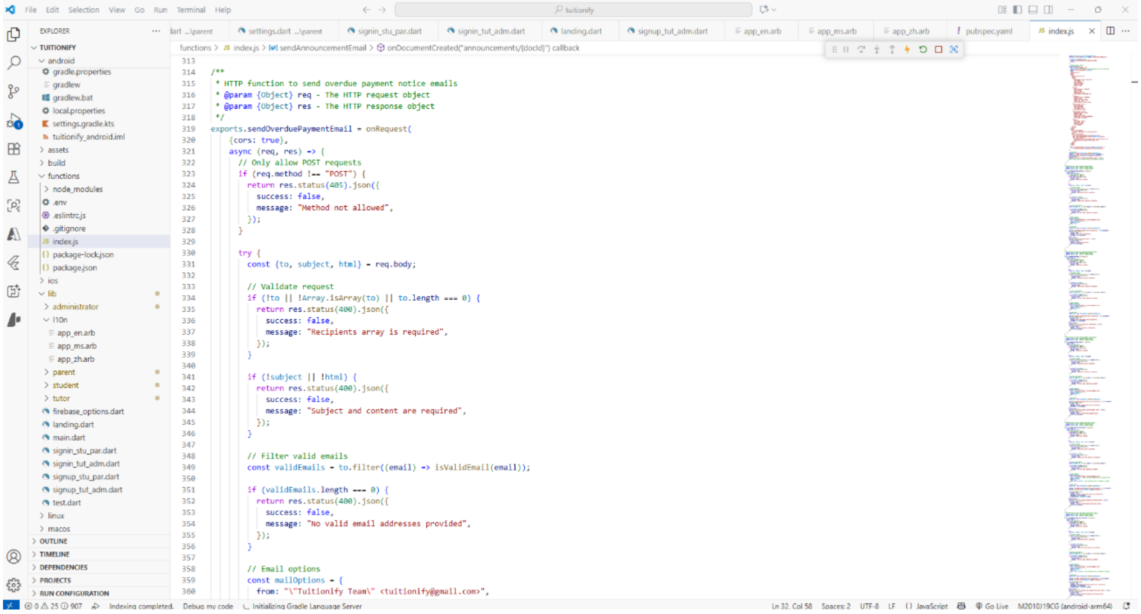
Figure 4.1.6.2 Screenshot of student/paymentlist.dart



paymentlist.dart ...\parent
paymentdetails.dart ...\parent
paymentmake.dart
inbox.dart ...\administrator 1
apps.dart ...\administrator
reportlist.dart
reportrevenue.dart
reportstudent.dart ...\administrator
lib > parent > paymentlist.dart > ...
EXPLORER
TUITIONIFY

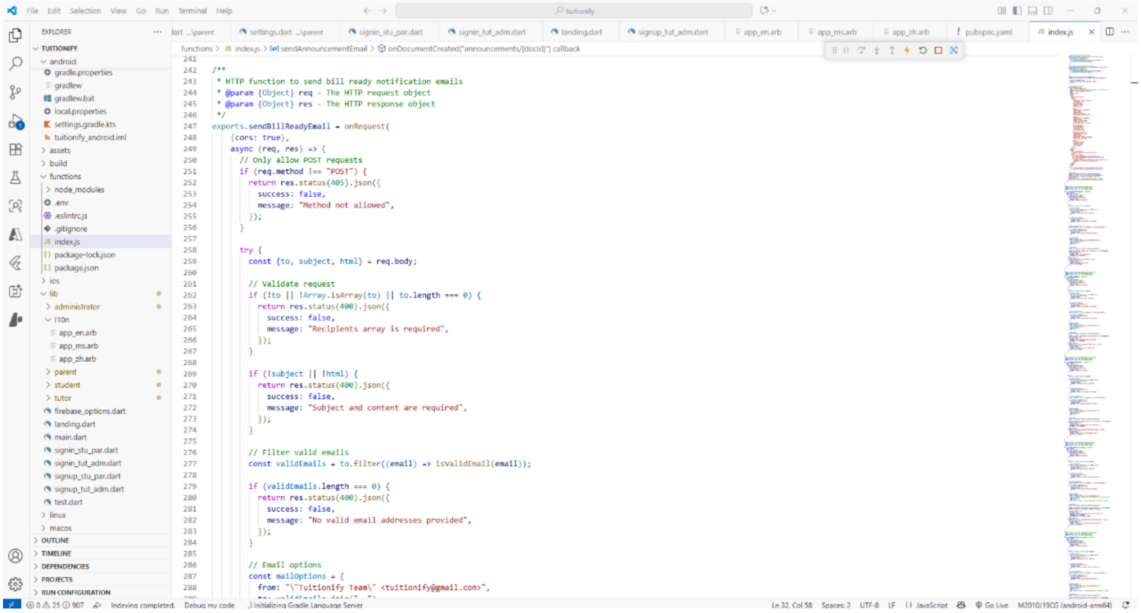
```dart
class _ParentPaymentListPageState extends State<ParentPaymentListPage> {
  @override
  Widget build(BuildContext context) {
    final filteredPayments = _filterPayments();

    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 10, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const ParentApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Text(
                    AppLocalizations.of(context)!.payment,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                ],
              ), // Row
              const SizedBox(height: 8),

              // Search bar
              Row(
                children: [
```

Figure 4.1.6.3 Screenshot of parent/paymentlist.dart



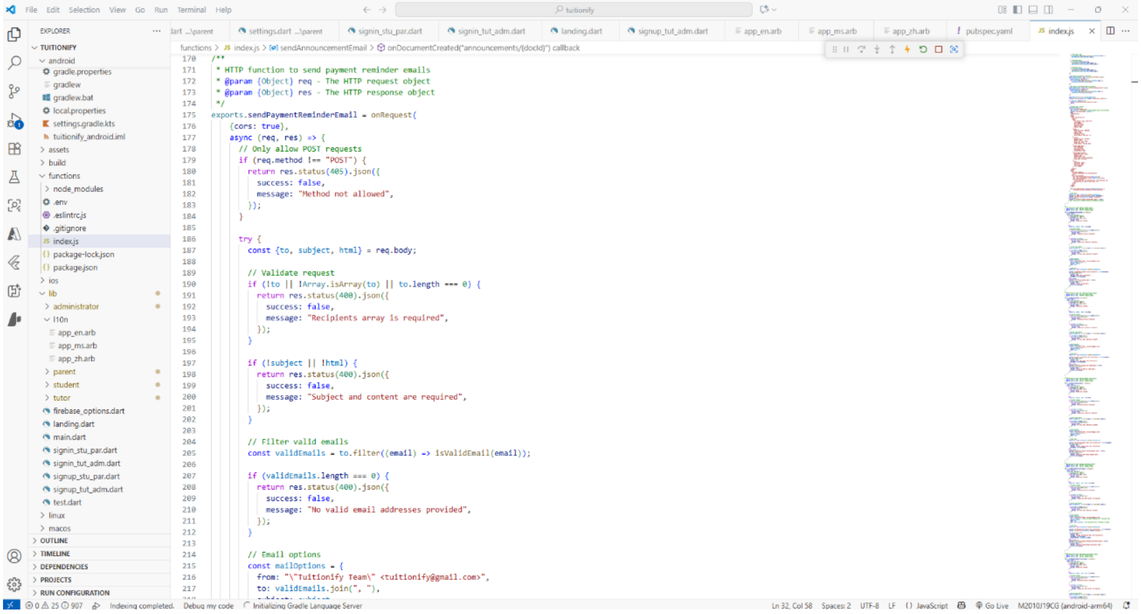
```javascript

/**
 * HTTP function to send overdue payment notice emails
 * @param {Object} req - The HTTP request object
 * @param {Object} res - The HTTP response object
 */
exports.sendOverduePaymentEmail = onRequest(
    {cors: true},
    async (req, res) => {
      // Only allow POST requests
      if (req.method !== "POST") {
        return res.status(405).json({
          success: false,
          message: "Method not allowed",
        });
      }

      try {
        const {to, subject, html} = req.body;

        // Validate request
        if (!to || !Array.isArray(to) || to.length === 0) {
          return res.status(400).json({
            success: false,
            message: "Recipients array is required",
          });
        }

        if (!subject || !html) {
          return res.status(400).json({
            success: false,
            message: "Subject and content are required",
          });
        }

        // Filter valid emails
        const validEmails = to.filter((email) => isValidEmail(email));

        if (validEmails.length === 0) {
          return res.status(400).json({
            success: false,
            message: "No valid email addresses provided",
          });
        }

        // Email options
        const mailOptions = {
          from: "\"Tuitionify Team\" <tuitionify@gmail.com>",
```

Figure 4.1.6.4 Screenshot of sendOverduePaymentEmail in functions/index.js


functions > index.js > sendAnnouncementEmail > onDocumentCreated("announcements/{docId}") callback

```
/**
 * HTTP function to send bill ready notification emails
 * @param {Object} req - The HTTP request object
 * @param {Object} res - The HTTP response object
 */
exports.sendBillReadyEmail = onRequest(
    {cors: true},
    async (req, res) => {
      // Only allow POST requests
      if (req.method !== "POST") {
        return res.status(405).json({
          success: false,
          message: "Method not allowed",
        });
      }

      try {
        const {to, subject, html} = req.body;

        // Validate request
        if (!to || !Array.isArray(to) || to.length === 0) {
          return res.status(400).json({
            success: false,
            message: "Recipients array is required",
          });
        }

        if (!subject || !html) {
          return res.status(400).json({
            success: false,
            message: "Subject and content are required",
          });
        }

        // Filter valid emails
        const validEmails = to.filter((email) => isValidEmail(email));

        if (validEmails.length === 0) {
          return res.status(400).json({
            success: false,
            message: "No valid email addresses provided",
          });
        }

        // Email options
        const mailOptions = {
          from: "\"Tuitionify Team\" <tuitionify@gmail.com>",
```

Figure 4.1.6.5 Screenshot of sendBillReadyEmail in functions/index.js


functions > index.js > sendAnnouncementEmail > onDocumentCreated("announcements/{docId}") callback

```
/**
 * HTTP function to send payment reminder emails
 * @param {Object} req - The HTTP request object
 * @param {Object} res - The HTTP response object
 */
exports.sendPaymentReminderEmail = onRequest(
    {cors: true},
    async (req, res) => {
      // Only allow POST requests
      if (req.method !== "POST") {
        return res.status(405).json({
          success: false,
          message: "Method not allowed",
        });
      }

      try {
        const {to, subject, html} = req.body;

        // Validate request
        if (!to || !Array.isArray(to) || to.length === 0) {
          return res.status(400).json({
            success: false,
            message: "Recipients array is required",
          });
        }

        if (!subject || !html) {
          return res.status(400).json({
            success: false,
            message: "Subject and content are required",
          });
        }

        // Filter valid emails
        const validEmails = to.filter((email) => isValidEmail(email));

        if (validEmails.length === 0) {
          return res.status(400).json({
            success: false,
            message: "No valid email addresses provided",
          });
        }

        // Email options
        const mailOptions = {
          from: "\"Tuitionify Team\" <tuitionify@gmail.com>",
          to: validEmails.join(", "),
```

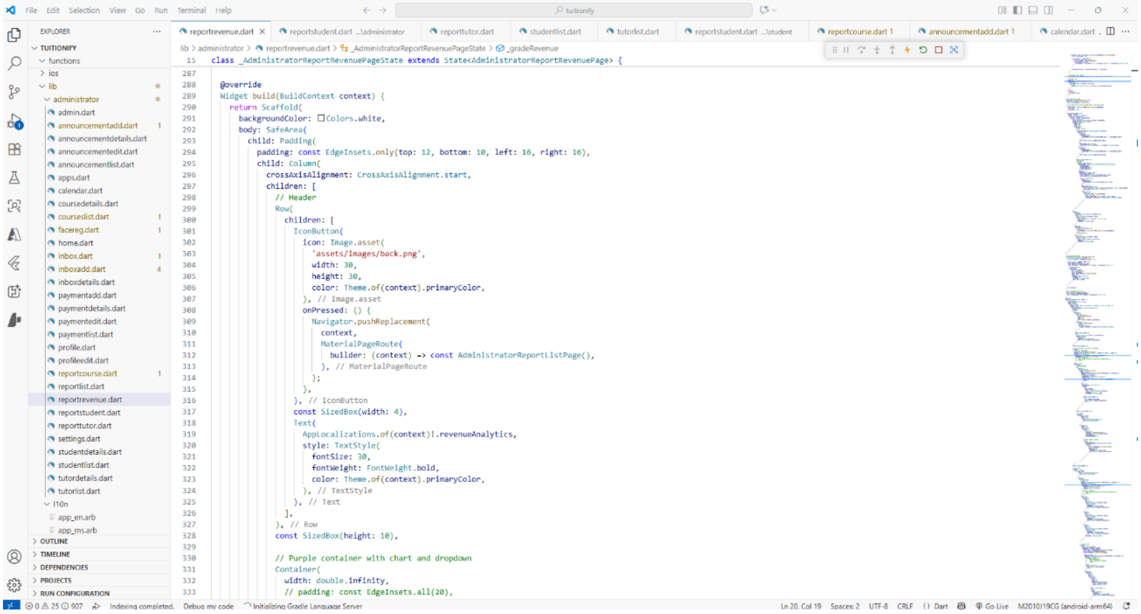
Figure 4.1.6.6 Screenshot of sendPaymentReminderEmail in functions/index.js

4.1.7 Report and Analytics Module Development

The report and analytics module offers four distinct analytics categories, such as revenue analytics, course analytics, tutor analytics, and student analytics. Administrators are granted access to all the analytics, while tutors, students, and parents can only access student analytics for privacy and security reasons.

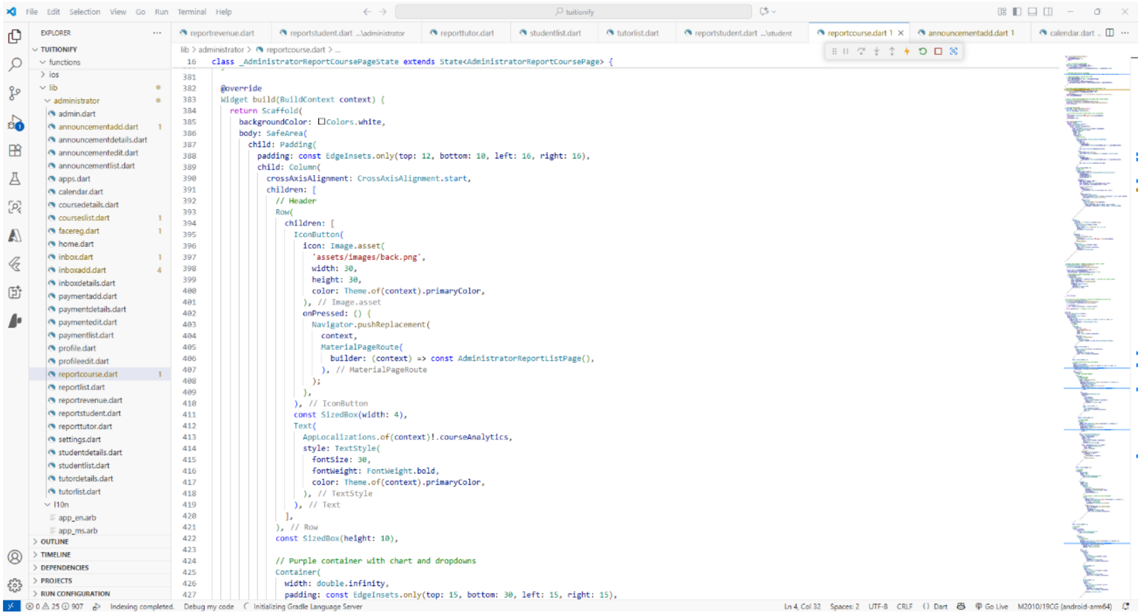
The revenue analytics in Figure 4.1.7.1 displays a dynamic pie chart that depicts the tuition centre's total monthly revenue, categorised by grade level. This is to illustrate the revenue contribution of each grade, and the system will calculate percentages for each segment. Administrators are allowed to utilise the filtering functionality via a pop-up calendar, which allows them to select different months for targeted revenue analysis. Furthermore, course analytics provides a similar visualisation in the form of a pie chart that displays monthly revenue distribution, which is categorised by individual courses, and the system will calculate the percentage of revenue contribution of each course, as illustrated in Figure 4.1.7.2.

Meanwhile, the tutor analytics section uses horizontal bar charts to display tutor attendance performance monthly, with each bar representing an individual tutor by name according to Figure 4.1.7.3. Each bar is clickable to allow administrators to view detailed information, such as the number of courses taught, which are categorised by grade level. The student analytics, as depicted in Figures 4.1.7.4 to 4.1.7.7, also employs horizontal bar chart visualisation to display student attendance performance monthly, with each bar representing a different course. Administrators, tutors, students, and parents can view the detailed attendance information, including lists of absent and present students, and individual attendance percentages, by clicking the horizontal bar. This module creates a comprehensive analytical tool that supports data-driven decision-making.



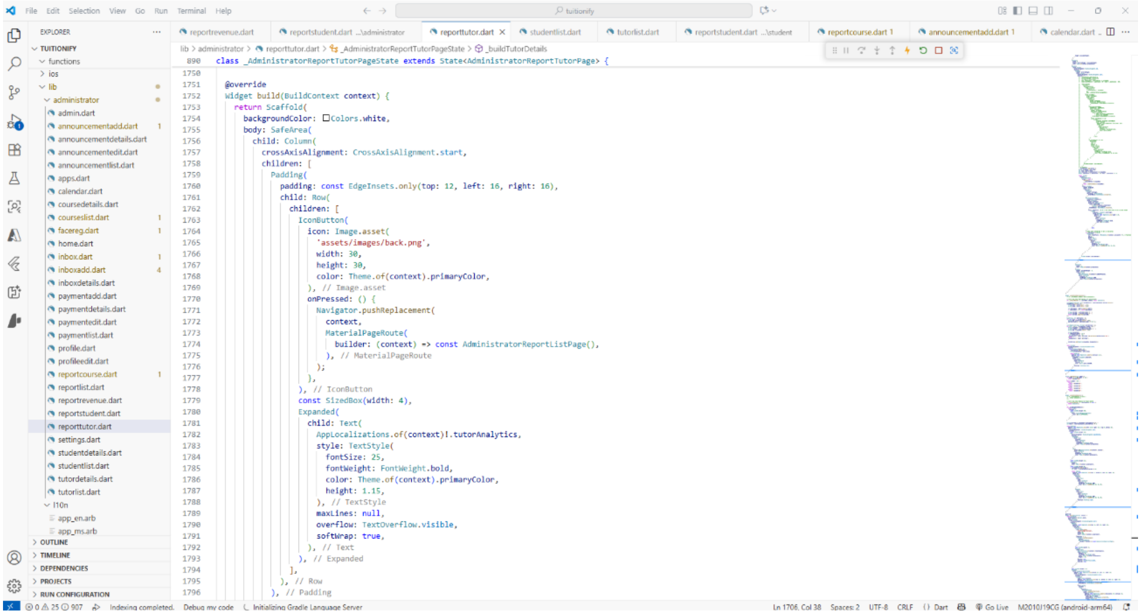
```dart
class _AdministratorReportRevenuePageState extends State<AdministratorReportRevenuePage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 10, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const AdministratorReportListPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Text(
                    AppLocalizations.of(context)!.revenueAnalytics,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                ],
              ), // Row
              const SizedBox(height: 10),

              // Purple container with chart and dropdown
              Container(
                width: double.infinity,
                // padding: const EdgeInsets.all(20),
```

Figure 4.1.7.1 Screenshot of administrator/reportrevenue.dart



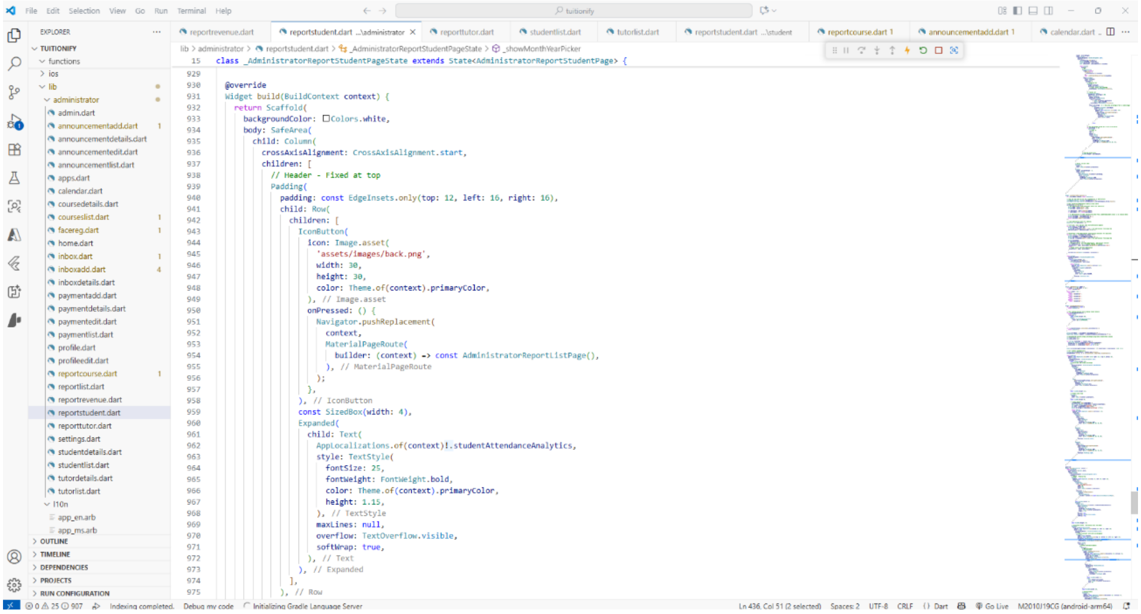
```
class _AdministratorReportCoursePageState extends State<AdministratorReportCoursePage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 10, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const AdministratorReportListPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Text(
                    AppLocalizations.of(context)!.courseAnalytics,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                ],
              ), // Row
              const SizedBox(height: 10),

              // Purple container with chart and dropdowns
              Container(
                width: double.infinity,
                padding: const EdgeInsets.only(top: 15, bottom: 30, left: 15, right: 15),
```

Figure 4.1.7.2 Screenshot of administrator/reportcourse.dart



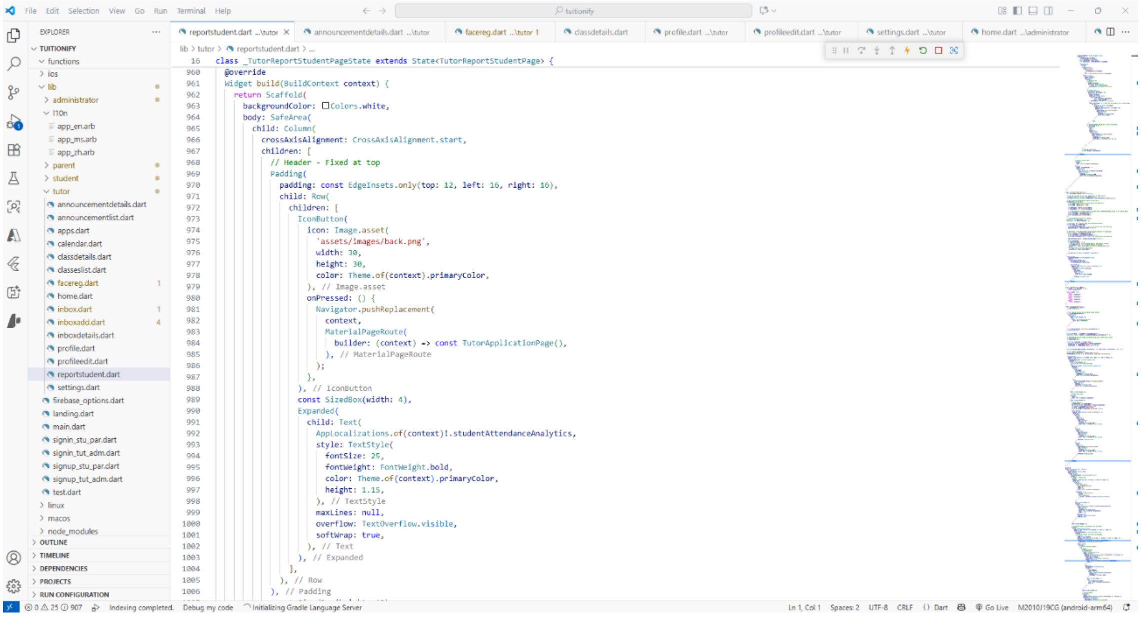
```
class _AdministratorReportTutorPageState extends State<AdministratorReportTutorPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Padding(
              padding: const EdgeInsets.only(top: 12, left: 16, right: 16),
              child: Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const AdministratorReportListPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Expanded(
                    child: Text(
                      AppLocalizations.of(context)!.tutorAnalytics,
                      style: TextStyle(
                        fontSize: 25,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                        height: 1.15,
                      ), // TextStyle
                      maxLines: null,
                      overflow: TextOverflow.visible,
                      softWrap: true,
                    ), // Text
                  ), // Expanded
                ],
              ), // Row
            ), // Padding
```

Figure 4.1.7.3 Screenshot of administrator/reporttutor.dart



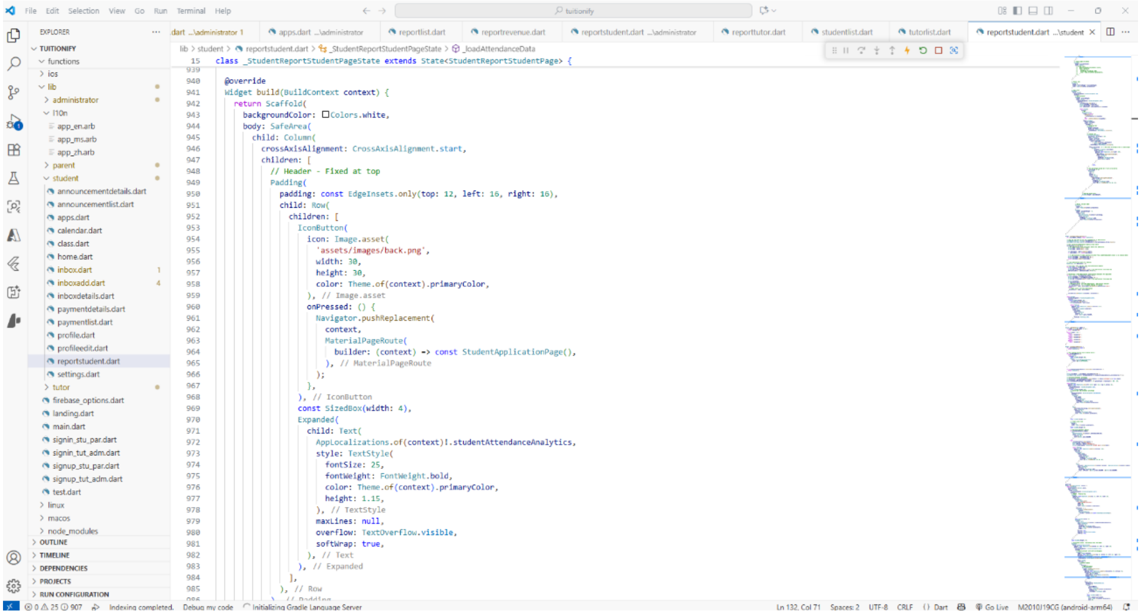
```
class _AdministratorReportStudentPageState extends State<AdministratorReportStudentPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Header - Fixed at top
            Padding(
              padding: const EdgeInsets.only(top: 12, left: 16, right: 16),
              child: Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const AdministratorReportListPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Expanded(
                    child: Text(
                      AppLocalizations.of(context)!.studentAttendanceAnalytics,
                      style: TextStyle(
                        fontSize: 25,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                        height: 1.15,
                      ), // TextStyle
                      maxLines: null,
                      overflow: TextOverflow.visible,
                      softWrap: true,
                    ), // Text
                  ), // Expanded
                ],
              ), // Row
```

Figure 4.1.7.4 Screenshot of administrator/reportstudent.dart



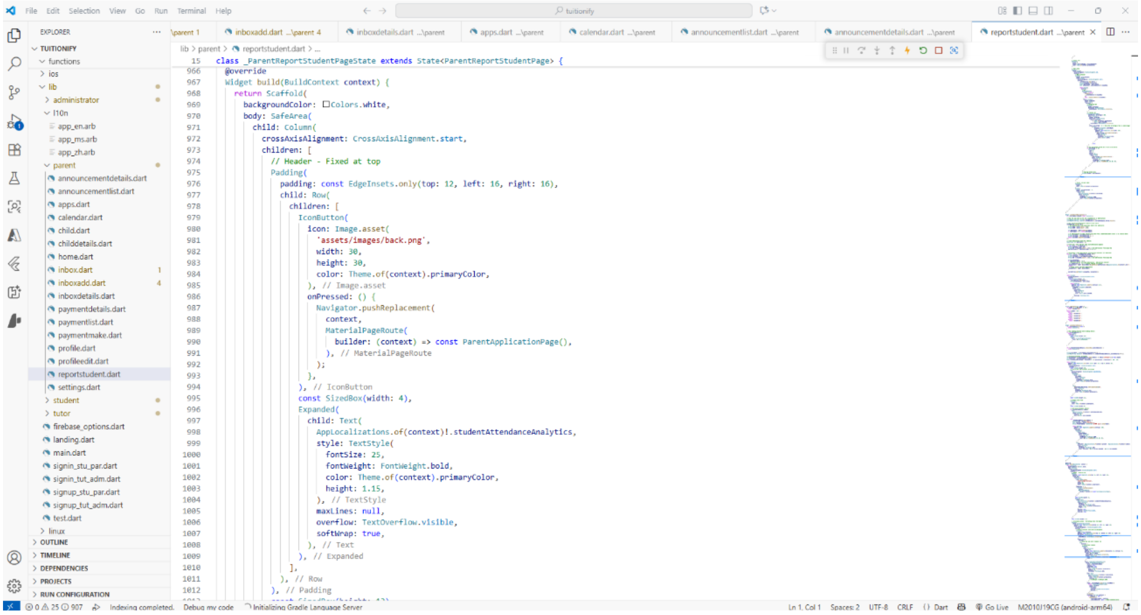
```
class _TutorReportStudentPageState extends State<TutorReportStudentPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Header - Fixed at top
            Padding(
              padding: const EdgeInsets.only(top: 12, left: 16, right: 16),
              child: Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const TutorApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Expanded(
                    child: Text(
                      AppLocalizations.of(context)!.studentAttendanceAnalytics,
                      style: TextStyle(
                        fontSize: 25,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                        height: 1.15,
                      ), // TextStyle
                      maxLines: null,
                      overflow: TextOverflow.visible,
                      softWrap: true,
                    ), // Text
                  ), // Expanded
                ],
              ), // Row
            ), // Padding
```

Figure 4.1.7.5 Screenshot of tutor/reportstudent.dart



```
class _StudentReportStudentPageState extends State<StudentReportStudentPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Header - Fixed at top
            Padding(
              padding: const EdgeInsets.only(top: 12, left: 16, right: 16),
              child: Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const StudentApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Expanded(
                    child: Text(
                      AppLocalizations.of(context)!.studentAttendanceAnalytics,
                      style: TextStyle(
                        fontSize: 25,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                        height: 1.15,
                      ), // TextStyle
                      maxLines: null,
                      overflow: TextOverflow.visible,
                      softWrap: true,
                    ), // Text
                  ), // Expanded
                ],
              ), // Row
```

Figure 4.1.7.6 Screenshot of student/reportstudent.dart

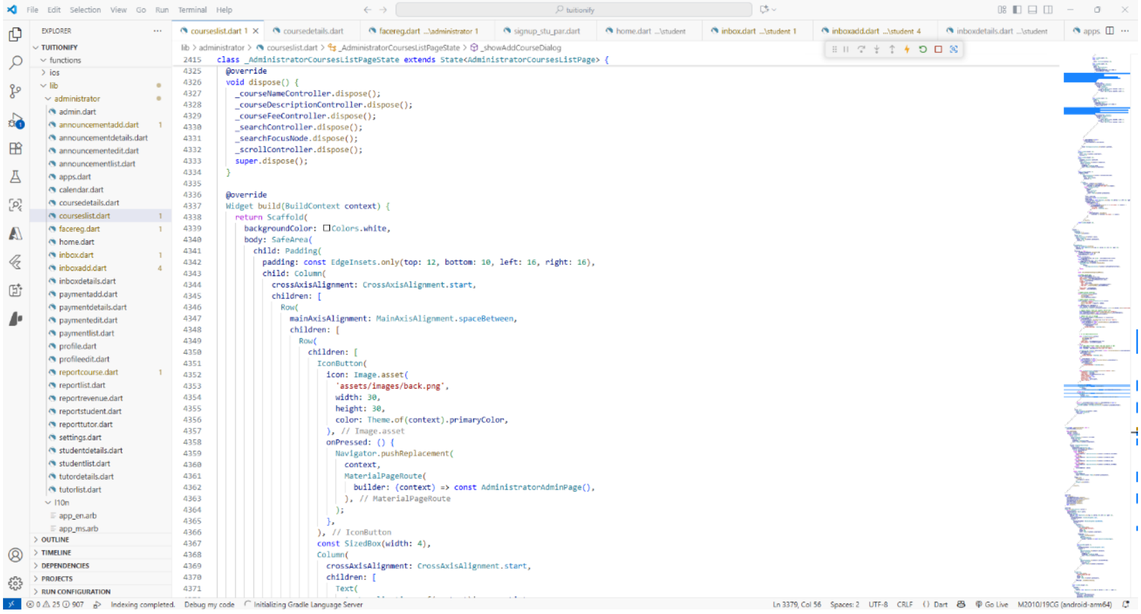


```
class _ParentReportStudentPageState extends State<ParentReportStudentPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Header - Fixed at top
            Padding(
              padding: const EdgeInsets.only(top: 12, left: 16, right: 16),
              child: Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const ParentApplicationPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Expanded(
                    child: Text(
                      AppLocalizations.of(context)!.studentAttendanceAnalytics,
                      style: TextStyle(
                        fontSize: 25,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                        height: 1.15,
                      ), // TextStyle
                      maxLines: null,
                      overflow: TextOverflow.visible,
                      softWrap: true,
                    ), // Text
                  ), // Expanded
                ],
              ), // Row
            ), // Padding
```

Figure 4.1.7.7 Screenshot of parent/reportstudent.dart

4.1.8 Courses Module Development

According to Figure 4.1.8.1, the courses module will display all existing courses from Cloud Firestore in alphabetical order by course name. By clicking each course, administrators will be navigating to the respective course details page, where they can view complete class history records and perform critical administrative tasks, such as updating student attendance records. Besides, administrators are allowed to add new courses by entering comprehensive course details such as course name, description, fee structure, grade level, assigned tutor name, scheduled day and time, and registered student information, and save to Cloud Firestore. Administrators are also allowed to edit the existing courses' information, including the adjustment of any course information to ensure the course details remain accurate.

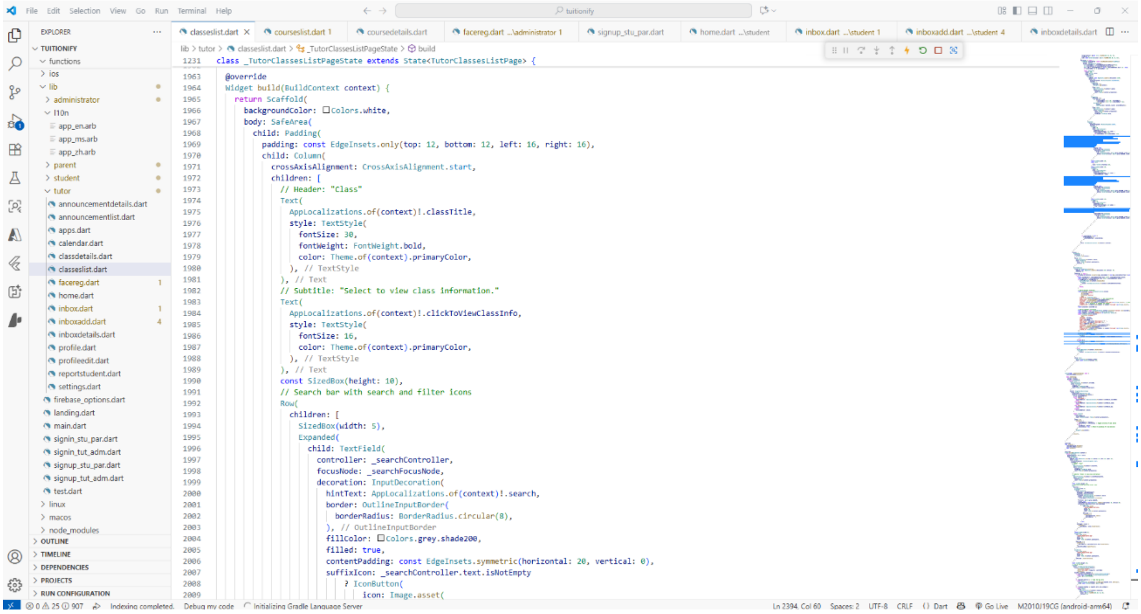


```
class _AdministratorCoursesListPageState extends State<AdministratorCoursesListPage> {
  @override
  void dispose() {
    _courseNameController.dispose();
    _courseDescriptionController.dispose();
    _courseFeeController.dispose();
    _searchController.dispose();
    _searchFocusNode.dispose();
    _scrollController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 10, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Row(
                    children: [
                      IconButton(
                        icon: Image.asset(
                          'assets/images/back.png',
                          width: 30,
                          height: 30,
                          color: Theme.of(context).primaryColor,
                        ), // Image.asset
                        onPressed: () {
                          Navigator.pushReplacement(
                            context,
                            MaterialPageRoute(
                              builder: (context) => const AdministratorAdminPage(),
                            ), // MaterialPageRoute
                          );
                        },
                      ), // IconButton
                      const SizedBox(width: 4),
                      Column(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: [
                          Text(
```

Figure 4.1.8.1 Screenshot of administrator/courseslist.dart

4.1.9 Classes Module Development

The classeslist.dart file shown in Figure 4.1.9.1 is designed specifically for tutors to display all existing courses that are assigned to them by administrators. The course data is retrieved from Cloud Firestore and presented alphabetically by course name to ensure systematic organization. Tutors can click any of the existing courses, and they will be navigated to a comprehensive course details page, where they can review all the class history records. Tutors are allowed to perform essential administrative tasks such as updating student attendance information. This module provides tutors with class scheduling capabilities by allowing them to create new class schedules for their respective courses by inputting specific details such as class date and duration. All class schedules are securely saved to Cloud Firestore to ensure data integrity. Besides, it is also supported by robust search and sorting functionalities, which enable tutors to quickly locate specific courses through keyword searches or organise their course listings according to various sorting criteria, such as Course Name, Grade, and Day.

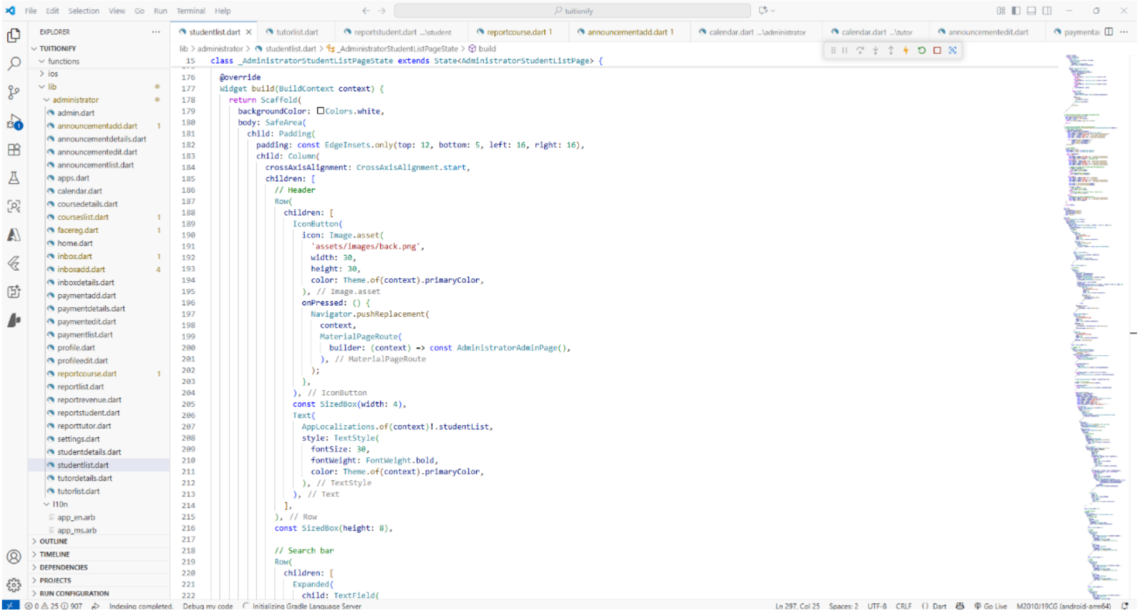


```
class _TutorClassesListPageState extends State<TutorClassesListPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 12, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header: "Class"
              Text(
                AppLocalizations.of(context)!.classTitle,
                style: TextStyle(
                  fontSize: 30,
                  fontWeight: FontWeight.bold,
                  color: Theme.of(context).primaryColor,
                ), // TextStyle
              ), // Text
              // Subtitle: "Select to view class information."
              Text(
                AppLocalizations.of(context)!.clickToViewClassInfo,
                style: TextStyle(
                  fontSize: 16,
                  color: Theme.of(context).primaryColor,
                ), // TextStyle
              ), // Text
              const SizedBox(height: 10),
              // Search bar with search and filter icons
              Row(
                children: [
                  SizedBox(width: 5),
                  Expanded(
                    child: TextField(
                      controller: _searchController,
                      focusNode: _searchFocusNode,
                      decoration: InputDecoration(
                        hintText: AppLocalizations.of(context)!.search,
                        border: OutlineInputBorder(
                          borderRadius: BorderRadius.circular(8),
                        ), // OutlineInputBorder
                        fillColor: Colors.grey.shade200,
                        filled: true,
                        contentPadding: const EdgeInsets.symmetric(horizontal: 20, vertical: 0),
                        suffixIcon: _searchController.text.isNotEmpty
                            ? IconButton(
                                icon: Image.asset(
```

Figure 4.1.9.1 Screenshot of tutor/classeslist.dart

4.1.10 Students Module Development

Referring to Figure 4.1.10.1, the students module will retrieve and display all existing students from Cloud Firestore by sorting them alphabetically by name to ensure a systematic presentation. Each student entry serves as an interactive element that administrators can click to access a detailed student profile page, where they can view extensive personal information, including the student's full name, gender, age, date of birth, parents' full names, parents' email addresses, etc, along with a complete list of registered courses which providing administrators with a holistic view of each student. Furthermore, it also allows administrators to use the robust search and sorting functionalities, which enable administrators to quickly locate specific students through keyword searches or organise their student listings according to various sorting criteria, such as Name, Gender, and Grade. This module helps in streamlining the process of student information retrieval and supporting comprehensive administration workflows.

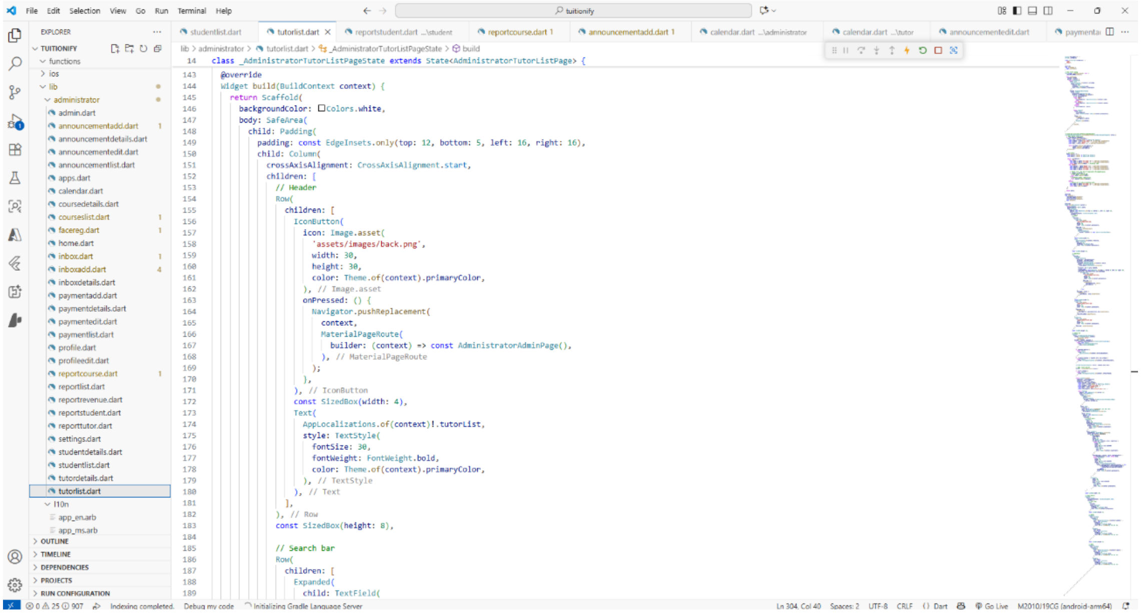


```
class _AdministratorStudentListPageState extends State<AdministratorStudentListPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 5, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const AdministratorAdminPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Text(
                    AppLocalizations.of(context)!.studentList,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                ],
              ), // Row
              const SizedBox(height: 8),

              // Search bar
              Row(
                children: [
                  Expanded(
                    child: TextField(
```

Figure 4.1.10.1 Screenshot of administrator/studentlist.dart

4.1.11 Tutors Module Development

The tutors module development is a comprehensive tutor information management system that retrieves and displays all existing tutors from Cloud Firestore and sorts them alphabetically by name, as shown in Figure 4.1.11.1. Each tutor's name is clickable and administrators will navigate to a detailed tutor profile page, which allows them to view comprehensive personal information, including the tutor’s name, gender, age, date of birth, etc, along with a complete list of assigned courses. This module provides administrators with a comprehensive overview of each tutor’s demographic information. Meanwhile, to improve administrative efficiency and support effective tutor record management, the module is integrated with robust search and sorting capabilities that allow administrators to quickly locate specific tutors using keyword searches or organise tutor listings based on various options, such as Name and Gender.

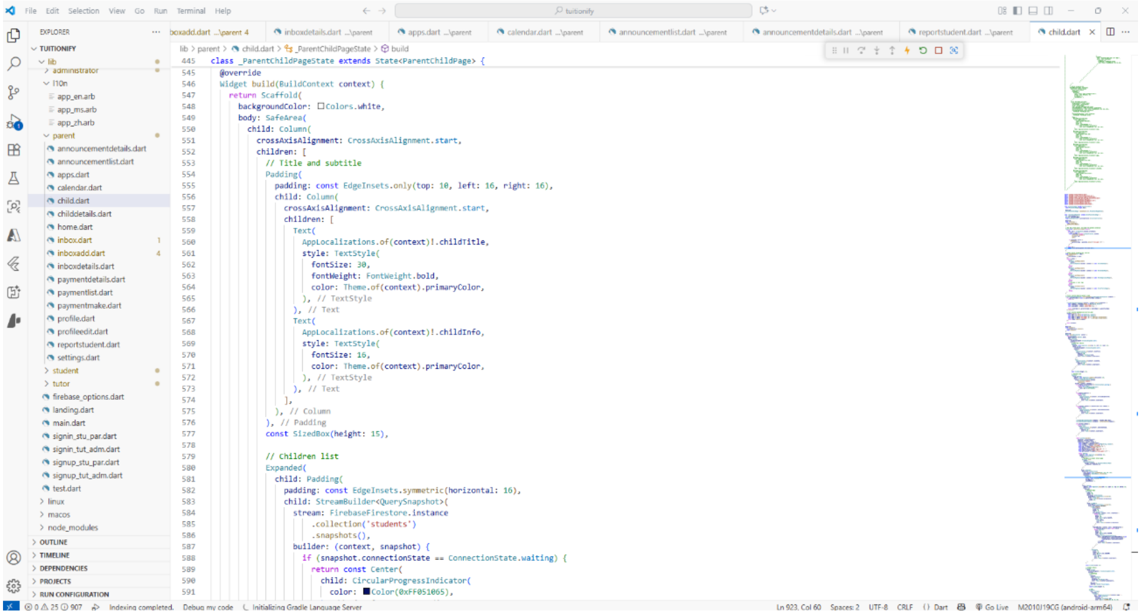


```
class _AdministratorTutorListPageState extends State<AdministratorTutorListPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(top: 12, bottom: 5, left: 16, right: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header
              Row(
                children: [
                  IconButton(
                    icon: Image.asset(
                      'assets/images/back.png',
                      width: 30,
                      height: 30,
                      color: Theme.of(context).primaryColor,
                    ), // Image.asset
                    onPressed: () {
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const AdministratorAdminPage(),
                        ), // MaterialPageRoute
                      );
                    },
                  ), // IconButton
                  const SizedBox(width: 4),
                  Text(
                    AppLocalizations.of(context)!.tutorList,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                ],
              ), // Row
              const SizedBox(height: 8),

              // Search bar
              Row(
                children: [
                  Expanded(
                    child: TextField(
```

Figure 4.1.11.1 Screenshot of administrator/tutorlist.dart

4.1.12 Children Module Development

Based on Figure 4.1.12.1, parents can manage and monitor their children's academic information. This module will display all existing children associated with the parent's account by retrieving the data from Cloud Firestore, and it will organise alphabetically by children's full name. Each child's entry serves as an interactive element that allows parents to access a comprehensive child profile page that contains detailed information about all courses registered by their child. Additionally, the module incorporates robust search and filtering functionalities that allow parents to quickly search specific registered courses by keyword searches or apply various filtering options, such as Course Name, Tutor Name, Day, and Fee, to improve the parent user experience and support effective monitoring of their children's academic enrollments.



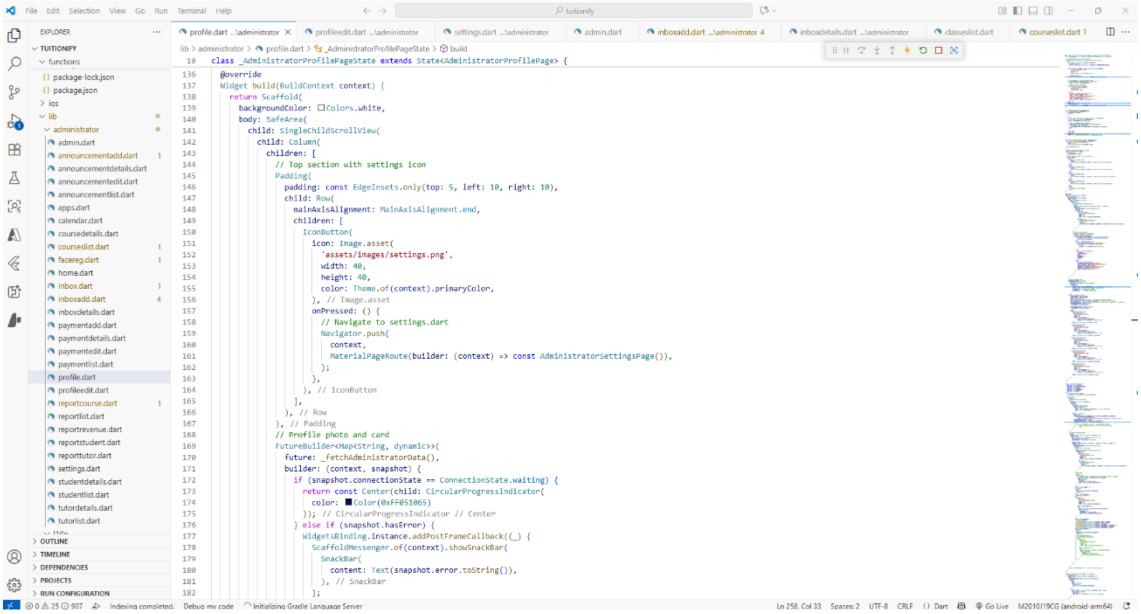
lib > parent > child.dart > _ParentChildPageState > build
child.dart
Initializing Gradle Language Server

```
class _ParentChildPageState extends State<ParentChildPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Title and subtitle
            Padding(
              padding: const EdgeInsets.only(top: 10, left: 16, right: 16),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    AppLocalizations.of(context)!.childTitle,
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                  Text(
                    AppLocalizations.of(context)!.childInfo,
                    style: TextStyle(
                      fontSize: 16,
                      color: Theme.of(context).primaryColor,
                    ), // TextStyle
                  ), // Text
                ],
              ), // Column
            ), // Padding
            const SizedBox(height: 15),

            // Children list
            Expanded(
              child: Padding(
                padding: const EdgeInsets.symmetric(horizontal: 16),
                child: StreamBuilder<QuerySnapshot>(
                  stream: FirebaseFirestore.instance
                      .collection('students')
                      .snapshots(),
                  builder: (context, snapshot) {
                    if (snapshot.connectionState == ConnectionState.waiting) {
                      return const Center(
                        child: CircularProgressIndicator(
                          color: Color(0xFF051065),
```

Figure 4.1.12.1 Screenshot of parent/child.dart

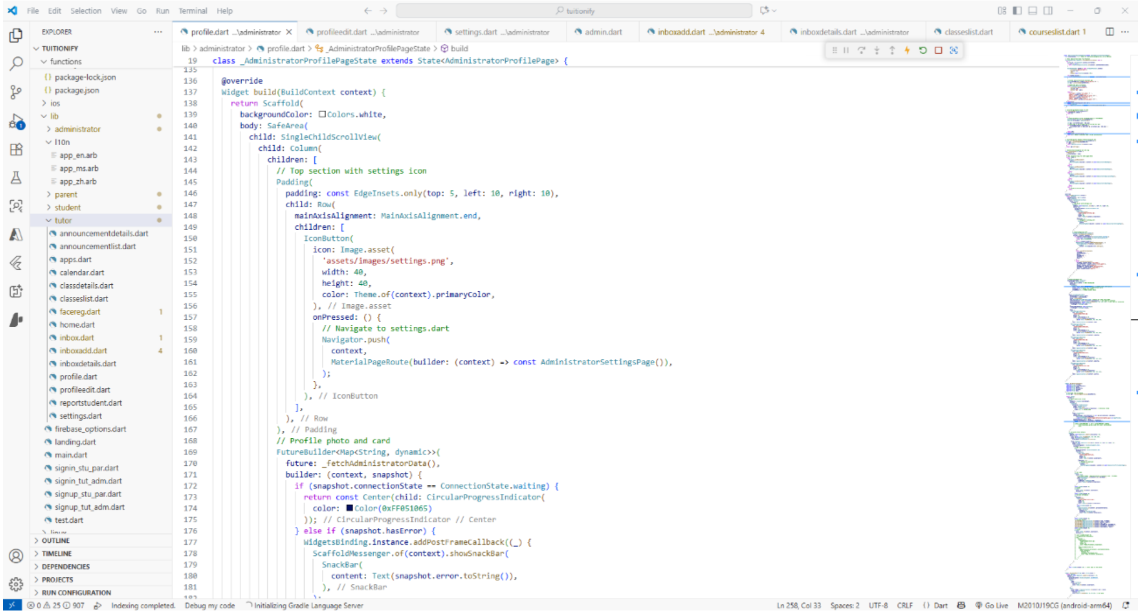
4.1.13 Profile Module Development

The profile module is designed to display administrators’, tutors’, students’, and parents’ personal profile details, which will be retrieved from Cloud Firestore. It will display all essential information, including profile photo, full name, gender, age, date of birth, phone number, and email address, to provide a complete overview of account information. For students’ profiles, the module will expand its functionality by displaying comprehensive parental details such as the father’s name, father’s phone number, father’s email address, mother’s name, mother’s phone number, and mother’s email address. Additionally, this module provides all users with extensive profile editing features to allow them to maintain accurate personal information by updating their latest profile photos or modifying any personal details as needed. This can ensure that user profiles remain up-to-date.



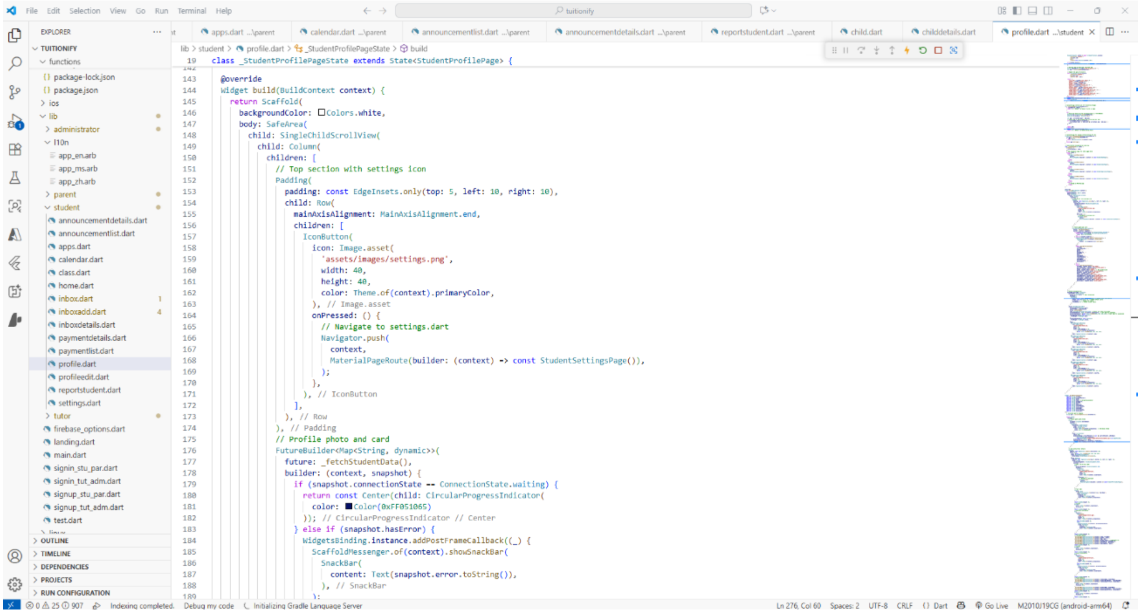
```dart
class _AdministratorProfilePageState extends State<AdministratorProfilePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            children: [
              // Top section with settings icon
              Padding(
                padding: const EdgeInsets.only(top: 5, left: 10, right: 10),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.end,
                  children: [
                    IconButton(
                      icon: Image.asset(
                        'assets/images/settings.png',
                        width: 40,
                        height: 40,
                        color: Theme.of(context).primaryColor,
                      ), // Image.asset
                      onPressed: () {
                        // Navigate to settings.dart
                        Navigator.push(
                          context,
                          MaterialPageRoute(builder: (context) => const AdministratorSettingsPage()),
                        );
                      },
                    ), // IconButton
                  ],
                ), // Row
              ), // Padding
              // Profile photo and card
              FutureBuilder<Map<String, dynamic>>(
                future: _fetchAdministratorData(),
                builder: (context, snapshot) {
                  if (snapshot.connectionState == ConnectionState.waiting) {
                    return const Center(child: CircularProgressIndicator(
                      color: Color(0xFF051065)
                    )); // CircularProgressIndicator // Center
                  } else if (snapshot.hasError) {
                    WidgetsBinding.instance.addPostFrameCallback((_) {
                      ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(
                          content: Text(snapshot.error.toString()),
                        ), // SnackBar
                      );
```

Figure 4.1.13.1 Screenshot of administrator/profile.dart



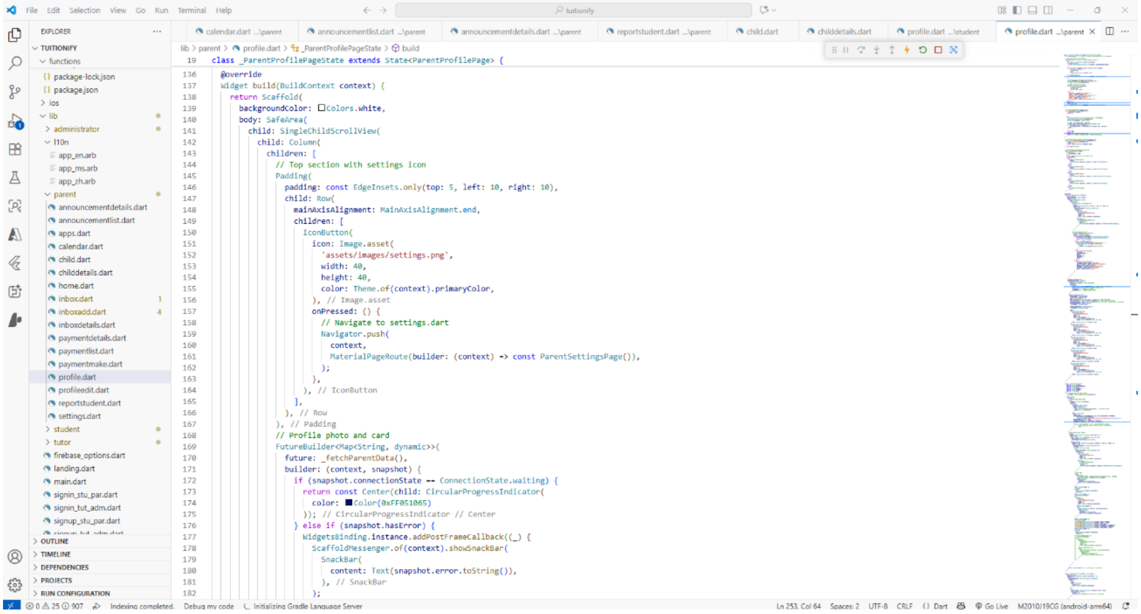
```dart
class _AdministratorProfilePageState extends State<AdministratorProfilePage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            children: [
              // Top section with settings icon
              Padding(
                padding: const EdgeInsets.only(top: 5, left: 10, right: 10),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.end,
                  children: [
                    IconButton(
                      icon: Image.asset(
                        'assets/images/settings.png',
                        width: 40,
                        height: 40,
                        color: Theme.of(context).primaryColor,
                      ), // Image.asset
                      onPressed: () {
                        // Navigate to settings.dart
                        Navigator.push(
                          context,
                          MaterialPageRoute(builder: (context) => const AdministratorSettingsPage()),
                        );
                      },
                    ), // IconButton
                  ],
                ), // Row
              ), // Padding
              // Profile photo and card
              FutureBuilder<Map<String, dynamic>>(
                future: _fetchAdministratorData(),
                builder: (context, snapshot) {
                  if (snapshot.connectionState == ConnectionState.waiting) {
                    return const Center(child: CircularProgressIndicator(
                      color: Color(0xFF051065)
                    )); // CircularProgressIndicator // Center
                  } else if (snapshot.hasError) {
                    WidgetsBinding.instance.addPostFrameCallback((_) {
                      ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(
                          content: Text(snapshot.error.toString()),
                        ), // SnackBar
```

Figure 4.1.13.2 Screenshot of tutor/profile.dart



```
class _StudentProfilePageState extends State<StudentProfilePage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            children: [
              // Top section with settings icon
              Padding(
                padding: const EdgeInsets.only(top: 5, left: 10, right: 10),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.end,
                  children: [
                    IconButton(
                      icon: Image.asset(
                        'assets/images/settings.png',
                        width: 40,
                        height: 40,
                        color: Theme.of(context).primaryColor,
                      ), // Image.asset
                      onPressed: () {
                        // Navigate to settings.dart
                        Navigator.push(
                          context,
                          MaterialPageRoute(builder: (context) => const StudentSettingsPage()),
                        );
                      },
                    ), // IconButton
                  ],
                ), // Row
              ), // Padding
              // Profile photo and card
              FutureBuilder<Map<String, dynamic>>(
                future: _fetchStudentData(),
                builder: (context, snapshot) {
                  if (snapshot.connectionState == ConnectionState.waiting) {
                    return const Center(child: CircularProgressIndicator(
                      color: Color(0xFF051065)
                    )); // CircularProgressIndicator // Center
                  } else if (snapshot.hasError) {
                    WidgetsBinding.instance.addPostFrameCallback((_) {
                      ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(
                          content: Text(snapshot.error.toString()),
                        ), // SnackBar
                      );
```

Figure 4.1.13.3 Screenshot of student/profile.dart

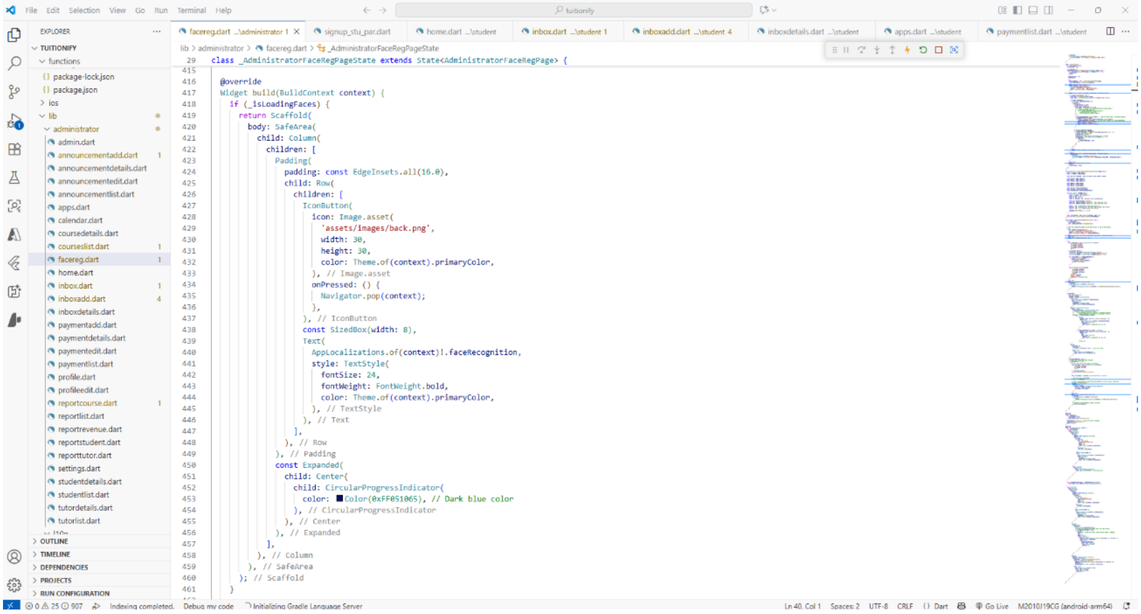


```
class _ParentProfilePageState extends State<ParentProfilePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            children: [
              // Top section with settings icon
              Padding(
                padding: const EdgeInsets.only(top: 5, left: 10, right: 10),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.end,
                  children: [
                    IconButton(
                      icon: Image.asset(
                        'assets/images/settings.png',
                        width: 40,
                        height: 40,
                        color: Theme.of(context).primaryColor,
                      ), // Image.asset
                      onPressed: () {
                        // Navigate to settings.dart
                        Navigator.push(
                          context,
                          MaterialPageRoute(builder: (context) => const ParentSettingsPage()),
                        );
                      },
                    ), // IconButton
                  ],
                ), // Row
              ), // Padding
              // Profile photo and card
              FutureBuilder<Map<String, dynamic>>(
                future: _fetchParentData(),
                builder: (context, snapshot) {
                  if (snapshot.connectionState == ConnectionState.waiting) {
                    return const Center(child: CircularProgressIndicator(
                      color: Color(0xFF051065)
                    )); // CircularProgressIndicator // Center
                  } else if (snapshot.hasError) {
                    WidgetsBinding.instance.addPostFrameCallback((_) {
                      ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(
                          content: Text(snapshot.error.toString()),
                        ), // SnackBar
                      );
```

Figure 4.1.13.4 Screenshot of parent/profile.dart

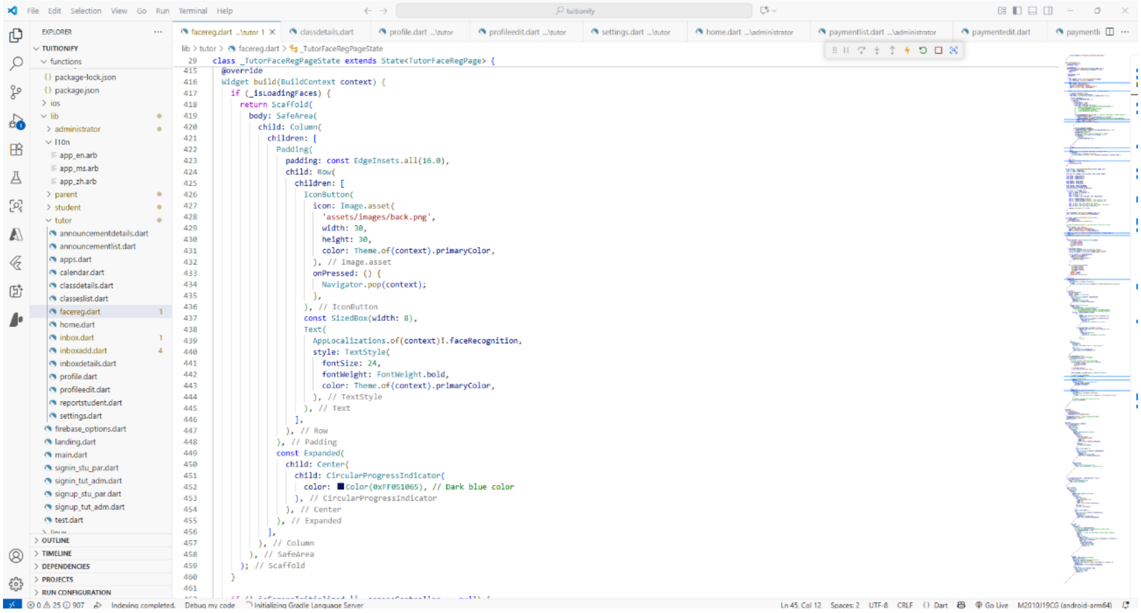
4.1.14 Face Recognition Module Development

Figures 4.1.14.1 to 4.1.14.3 show the facereg.dart for administrators and tutors, which is used for biometric attendance tracking. This module allows administrators and tutors to take attendance by scanning students' faces using a system that is powered by Python code and the face_recognition libraries, and integrated via a Flask server. When this module is initiated, the Flutter application sends a request to the Flask server, which will then process the camera input, detect faces, and match the faces against the stored images in the Firebase Storage by using the face_recognition library. After that, the server will return the recognised students' information to the Flutter application. The latest attendance data is updated to Cloud Firestore and displayed to administrators and tutors. Cloud Firestore helps to guarantee the real-time synchronisation of the attendance records. Moreover, the integration of Flutter, Python, and Flask ensures a seamless and automated attendance tracking process.


facereg.dart ...\administrator 1
signup_stu_par.dart
home.dart ...\student
inbox.dart ...\student 1
lib > administrator > facereg.dart > _AdministratorFaceRegPageState
Indexing completed.
Debug my code
Initializing Gradle Language Server

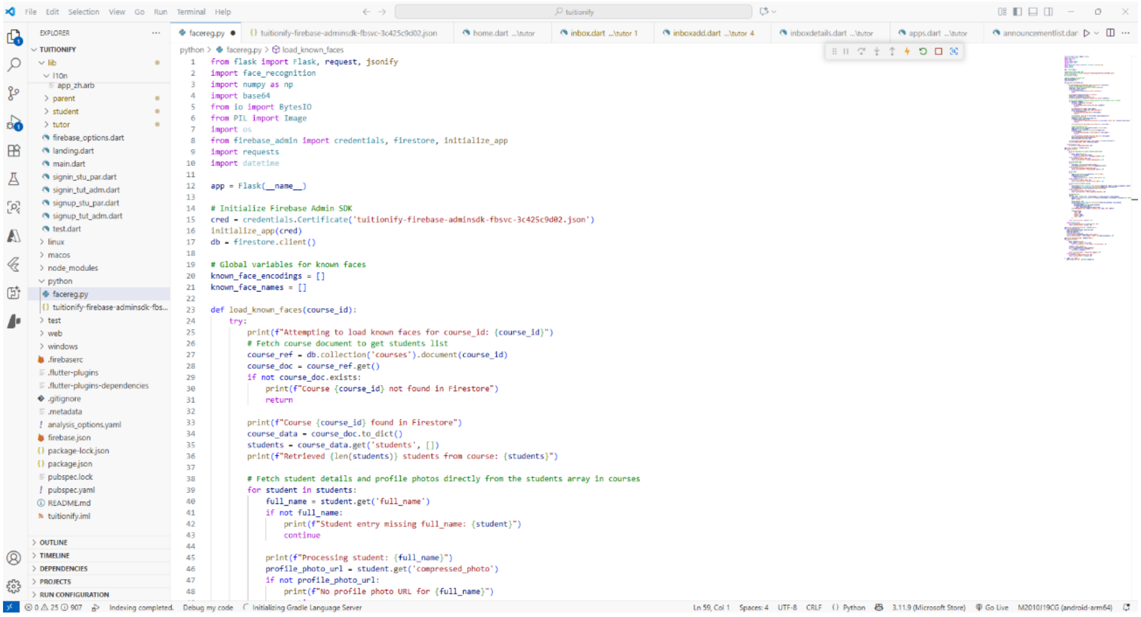
```
class _AdministratorFaceRegPageState extends State<AdministratorFaceRegPage> {

  @override
  Widget build(BuildContext context) {
    if (_isLoadingFaces) {
      return Scaffold(
        body: SafeArea(
          child: Column(
            children: [
              Padding(
                padding: const EdgeInsets.all(16.0),
                child: Row(
                  children: [
                    IconButton(
                      icon: Image.asset(
                        'assets/images/back.png',
                        width: 30,
                        height: 30,
                        color: Theme.of(context).primaryColor,
                      ), // Image.asset
                      onPressed: () {
                        Navigator.pop(context);
                      },
                    ), // IconButton
                    const SizedBox(width: 8),
                    Text(
                      AppLocalizations.of(context)!.faceRecognition,
                      style: TextStyle(
                        fontSize: 24,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                      ), // TextStyle
                    ), // Text
                  ],
                ), // Row
              ), // Padding
              const Expanded(
                child: Center(
                  child: CircularProgressIndicator(
                    color: Color(0xFF051065), // Dark blue color
                  ), // CircularProgressIndicator
                ), // Center
              ), // Expanded
            ],
          ), // Column
        ), // SafeArea
      ); // Scaffold
    }
```

Figure 4.1.14.1 Screenshot of administrator/facereg.dart


facereg.dart ...\tutor 1
classdetails.dart
profile.dart ...\tutor
profileedit.dart ...\tutor
lib > tutor > facereg.dart > _TutorFaceRegPageState
Indexing completed.
Debug my code
Initializing Gradle Language Server

```
class _TutorFaceRegPageState extends State<TutorFaceRegPage> {
  @override
  Widget build(BuildContext context) {
    if (_isLoadingFaces) {
      return Scaffold(
        body: SafeArea(
          child: Column(
            children: [
              Padding(
                padding: const EdgeInsets.all(16.0),
                child: Row(
                  children: [
                    IconButton(
                      icon: Image.asset(
                        'assets/images/back.png',
                        width: 30,
                        height: 30,
                        color: Theme.of(context).primaryColor,
                      ), // Image.asset
                      onPressed: () {
                        Navigator.pop(context);
                      },
                    ), // IconButton
                    const SizedBox(width: 8),
                    Text(
                      AppLocalizations.of(context)!.faceRecognition,
                      style: TextStyle(
                        fontSize: 24,
                        fontWeight: FontWeight.bold,
                        color: Theme.of(context).primaryColor,
                      ), // TextStyle
                    ), // Text
                  ],
                ), // Row
              ), // Padding
              const Expanded(
                child: Center(
                  child: CircularProgressIndicator(
                    color: Color(0xFF051065), // Dark blue color
                  ), // CircularProgressIndicator
                ), // Center
              ), // Expanded
            ],
          ), // Column
        ), // SafeArea
      ); // Scaffold
    }
```

Figure 4.1.14.2 Screenshot of tutor/facereg.dart

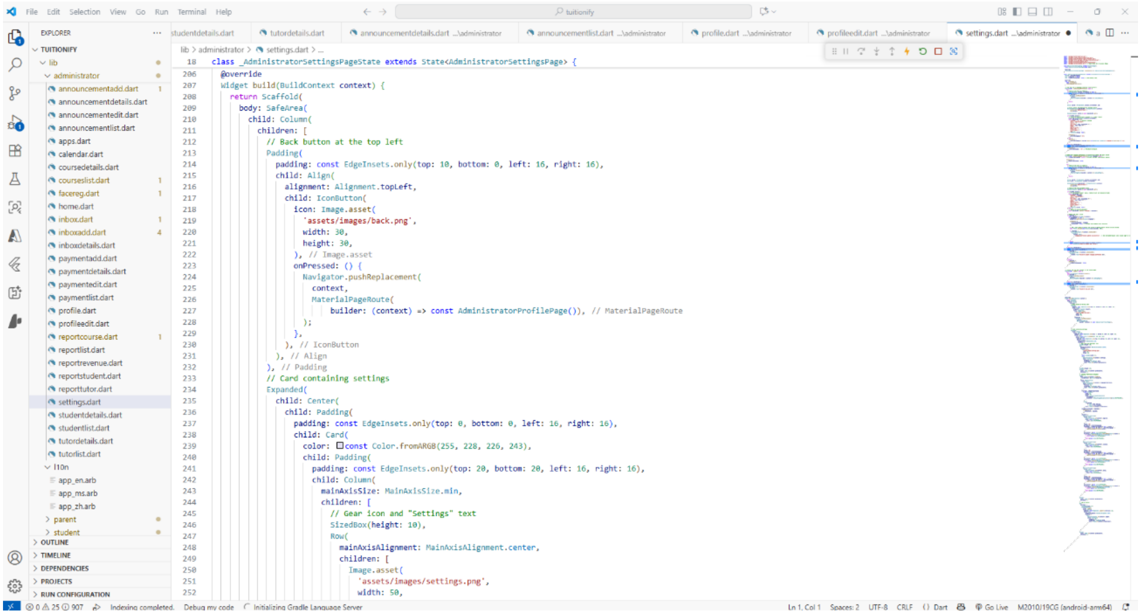


```python
from flask import Flask, request, jsonify
import face_recognition
import numpy as np
import base64
from io import BytesIO
from PIL import Image
import os
from firebase_admin import credentials, firestore, initialize_app
import requests
import datetime

app = Flask(__name__)

# Initialize Firebase Admin SDK
cred = credentials.Certificate('tuitionify-firebase-adminsdk-fbsvc-3c425c9d02.json')
initialize_app(cred)
db = firestore.client()

# Global variables for known faces
known_face_encodings = []
known_face_names = []

def load_known_faces(course_id):
    try:
        print(f"Attempting to load known faces for course_id: {course_id}")
        # Fetch course document to get students list
        course_ref = db.collection('courses').document(course_id)
        course_doc = course_ref.get()
        if not course_doc.exists:
            print(f"Course {course_id} not found in Firestore")
            return

        print(f"Course {course_id} found in Firestore")
        course_data = course_doc.to_dict()
        students = course_data.get('students', [])
        print(f"Retrieved {len(students)} students from course: {students}")

        # Fetch student details and profile photos directly from the students array in courses
        for student in students:
            full_name = student.get('full_name')
            if not full_name:
                print(f"Student entry missing full_name: {student}")
                continue

            print(f"Processing student: {full_name}")
            profile_photo_url = student.get('compressed_photo')
            if not profile_photo_url:
                print(f"No profile photo URL for {full_name}")
```

Figure 4.1.14.3 Screenshot of python/facereg.py

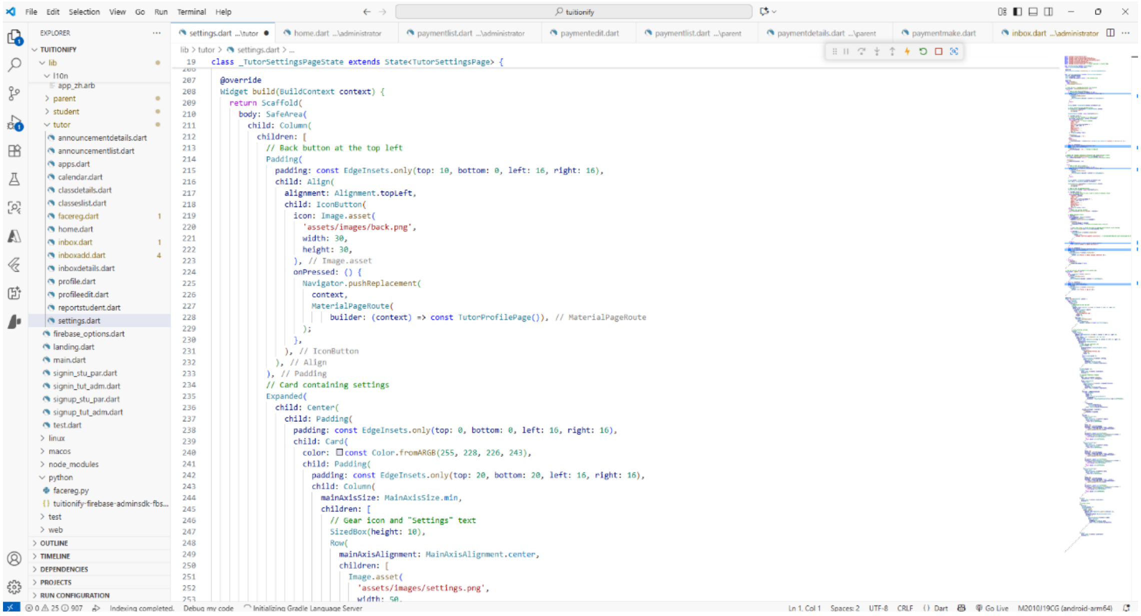
4.1.15 Language Preferences Module Development

According to Figures 4.1.15.1 to 4.1.15.4, the language preferences in settings.dart file is designed to support multilingual functionality. This module enables administrators, tutors, students, and parents to change the application language to English, Malay, or Chinese. This is achieved by implementing Flutter's localization features, integrating the flutter_localizations package, and defining the language-specific resource files, such as "app_en.arb", "app_ms.arb", and "app_zh.arb" in the "pubspec.yaml" file as shown in Figures 4.1.15.5 to 4.1.15.7. Users can switch between different languages by using the expanded dropdown list in the settings.dart page. Then, the localizations widget will modify the user interface dynamically to reflect the selected language.



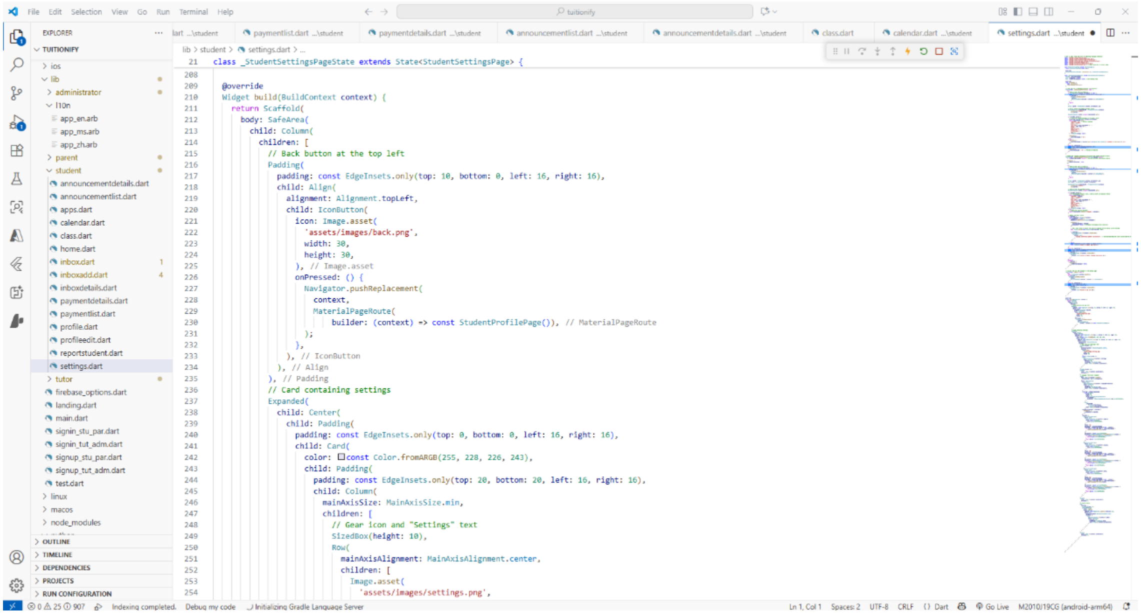
```
class _AdministratorSettingsPageState extends State<AdministratorSettingsPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Column(
          children: [
            // Back button at the top left
            Padding(
              padding: const EdgeInsets.only(top: 10, bottom: 0, left: 16, right: 16),
              child: Align(
                alignment: Alignment.topLeft,
                child: IconButton(
                  icon: Image.asset(
                    'assets/images/back.png',
                    width: 30,
                    height: 30,
                  ), // Image.asset
                  onPressed: () {
                    Navigator.pushReplacement(
                      context,
                      MaterialPageRoute(
                          builder: (context) => const AdministratorProfilePage()), // MaterialPageRoute
                    );
                  },
                ), // IconButton
              ), // Align
            ), // Padding
            // Card containing settings
            Expanded(
              child: Center(
                child: Padding(
                  padding: const EdgeInsets.only(top: 0, bottom: 0, left: 16, right: 16),
                  child: Card(
                    color: const Color.fromARGB(255, 228, 226, 243),
                    child: Padding(
                      padding: const EdgeInsets.only(top: 20, bottom: 20, left: 16, right: 16),
                      child: Column(
                        mainAxisSize: MainAxisSize.min,
                        children: [
                          // Gear icon and "Settings" text
                          SizedBox(height: 10),
                          Row(
                            mainAxisAlignment: MainAxisAlignment.center,
                            children: [
                              Image.asset(
                                'assets/images/settings.png',
                                width: 50,
```

Figure 4.1.15.1 Screenshot of administrator/settings.dart



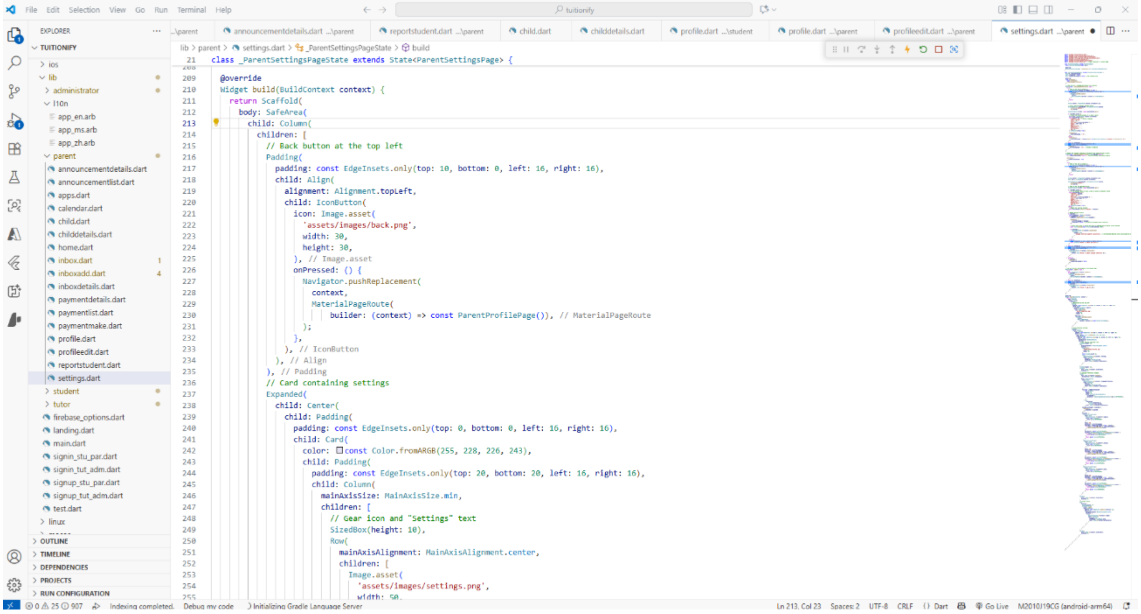
```
class _TutorSettingsPageState extends State<TutorSettingsPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Column(
          children: [
            // Back button at the top left
            Padding(
              padding: const EdgeInsets.only(top: 10, bottom: 0, left: 16, right: 16),
              child: Align(
                alignment: Alignment.topLeft,
                child: IconButton(
                  icon: Image.asset(
                    'assets/images/back.png',
                    width: 30,
                    height: 30,
                  ), // Image.asset
                  onPressed: () {
                    Navigator.pushReplacement(
                      context,
                      MaterialPageRoute(
                          builder: (context) => const TutorProfilePage()), // MaterialPageRoute
                    );
                  },
                ), // IconButton
              ), // Align
            ), // Padding
            // Card containing settings
            Expanded(
              child: Center(
                child: Padding(
                  padding: const EdgeInsets.only(top: 0, bottom: 0, left: 16, right: 16),
                  child: Card(
                    color: const Color.fromARGB(255, 228, 226, 243),
                    child: Padding(
                      padding: const EdgeInsets.only(top: 20, bottom: 20, left: 16, right: 16),
                      child: Column(
                        mainAxisSize: MainAxisSize.min,
                        children: [
                          // Gear icon and "Settings" text
                          SizedBox(height: 10),
                          Row(
                            mainAxisAlignment: MainAxisAlignment.center,
                            children: [
                              Image.asset(
                                'assets/images/settings.png',
                                width: 50,
```

Figure 4.1.15.2 Screenshot of tutor/settings.dart



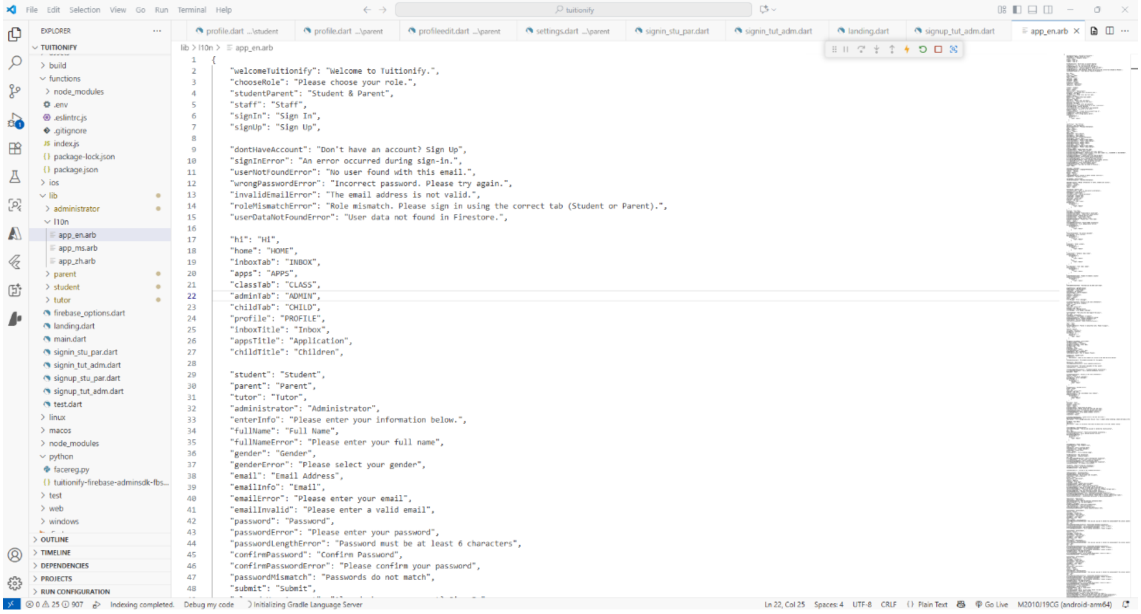
```dart
class _StudentSettingsPageState extends State<StudentSettingsPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Column(
          children: [
            // Back button at the top left
            Padding(
              padding: const EdgeInsets.only(top: 10, bottom: 0, left: 16, right: 16),
              child: Align(
                alignment: Alignment.topLeft,
                child: IconButton(
                  icon: Image.asset(
                    'assets/images/back.png',
                    width: 30,
                    height: 30,
                  ), // Image.asset
                  onPressed: () {
                    Navigator.pushReplacement(
                      context,
                      MaterialPageRoute(
                          builder: (context) => const StudentProfilePage()), // MaterialPageRoute
                    );
                  },
                ), // IconButton
              ), // Align
            ), // Padding
            // Card containing settings
            Expanded(
              child: Center(
                child: Padding(
                  padding: const EdgeInsets.only(top: 0, bottom: 0, left: 16, right: 16),
                  child: Card(
                    color: const Color.fromARGB(255, 228, 226, 243),
                    child: Padding(
                      padding: const EdgeInsets.only(top: 20, bottom: 20, left: 16, right: 16),
                      child: Column(
                        mainAxisSize: MainAxisSize.min,
                        children: [
                          // Gear icon and "Settings" text
                          SizedBox(height: 10),
                          Row(
                            mainAxisAlignment: MainAxisAlignment.center,
                            children: [
                              Image.asset(
                                'assets/images/settings.png',
```

Figure 4.1.15.3 Screenshot of student/settings.dart



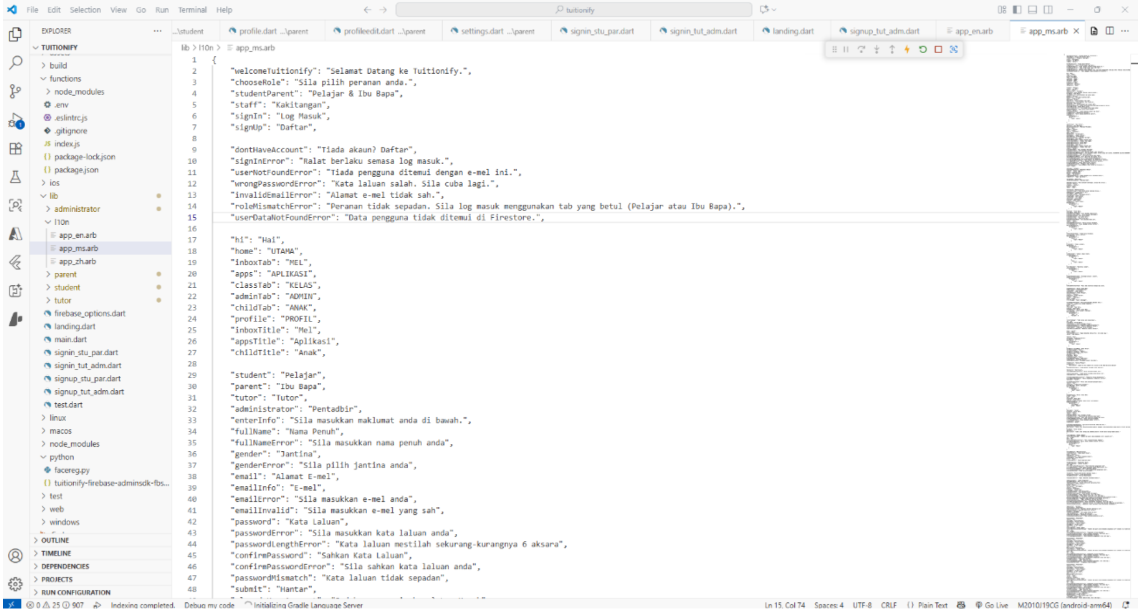
```dart
class _ParentSettingsPageState extends State<ParentSettingsPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Column(
          children: [
            // Back button at the top left
            Padding(
              padding: const EdgeInsets.only(top: 10, bottom: 0, left: 16, right: 16),
              child: Align(
                alignment: Alignment.topLeft,
                child: IconButton(
                  icon: Image.asset(
                    'assets/images/back.png',
                    width: 30,
                    height: 30,
                  ), // Image.asset
                  onPressed: () {
                    Navigator.pushReplacement(
                      context,
                      MaterialPageRoute(
                          builder: (context) => const ParentProfilePage()), // MaterialPageRoute
                    );
                  },
                ), // IconButton
              ), // Align
            ), // Padding
            // Card containing settings
            Expanded(
              child: Center(
                child: Padding(
                  padding: const EdgeInsets.only(top: 0, bottom: 0, left: 16, right: 16),
                  child: Card(
                    color: const Color.fromARGB(255, 228, 226, 243),
                    child: Padding(
                      padding: const EdgeInsets.only(top: 20, bottom: 20, left: 16, right: 16),
                      child: Column(
                        mainAxisSize: MainAxisSize.min,
                        children: [
                          // Gear icon and "Settings" text
                          SizedBox(height: 10),
                          Row(
                            mainAxisAlignment: MainAxisAlignment.center,
                            children: [
                              Image.asset(
                                'assets/images/settings.png',
                                width: 50,
```

Figure 4.1.15.4 Screenshot of parent/settings.dart



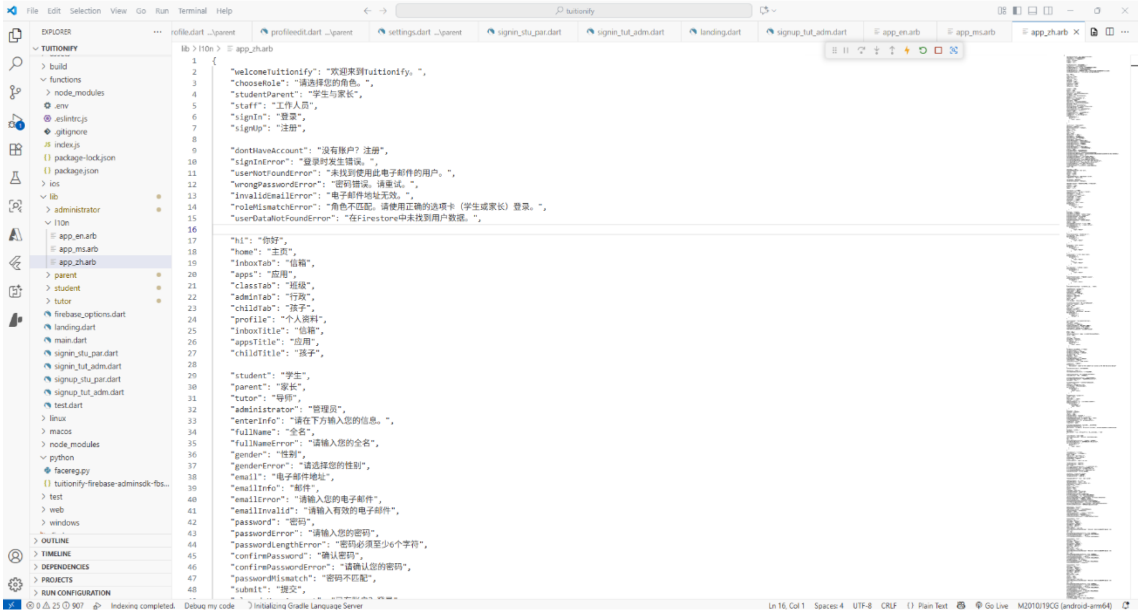
```
{
    "welcomeTuitionify": "Welcome to Tuitionify.",
    "chooseRole": "Please choose your role.",
    "studentParent": "Student & Parent",
    "staff": "Staff",
    "signIn": "Sign In",
    "signUp": "Sign Up",

    "dontHaveAccount": "Don't have an account? Sign Up",
    "signInError": "An error occurred during sign-in.",
    "userNotFoundError": "No user found with this email.",
    "wrongPasswordError": "Incorrect password. Please try again.",
    "invalidEmailError": "The email address is not valid.",
    "roleMismatchError": "Role mismatch. Please sign in using the correct tab (Student or Parent).",
    "userDataNotFoundError": "User data not found in Firestore.",

    "hi": "Hi",
    "home": "HOME",
    "inboxTab": "INBOX",
    "apps": "APPS",
    "classTab": "CLASS",
    "adminTab": "ADMIN",
    "childTab": "CHILD",
    "profile": "PROFILE",
    "inboxTitle": "Inbox",
    "appsTitle": "Application",
    "childTitle": "Children",

    "student": "Student",
    "parent": "Parent",
    "tutor": "Tutor",
    "administrator": "Administrator",
    "enterInfo": "Please enter your information below.",
    "fullName": "Full Name",
    "fullNameError": "Please enter your full name",
    "gender": "Gender",
    "genderError": "Please select your gender",
    "email": "Email Address",
    "emailInfo": "Email",
    "emailError": "Please enter your email",
    "emailInvalid": "Please enter a valid email",
    "password": "Password",
    "passwordError": "Please enter your password",
    "passwordLengthError": "Password must be at least 6 characters",
    "confirmPassword": "Confirm Password",
    "confirmPasswordError": "Please confirm your password",
    "passwordMismatch": "Passwords do not match",
    "submit": "Submit",
```

Figure 4.1.15.5 Screenshot of l10n/app_en.arb



```
{
    "welcomeTuitionify": "Selamat Datang ke Tuitionify.",
    "chooseRole": "Sila pilih peranan anda.",
    "studentParent": "Pelajar & Ibu Bapa",
    "staff": "Kakitangan",
    "signIn": "Log Masuk",
    "signUp": "Daftar",

    "dontHaveAccount": "Tiada akaun? Daftar",
    "signInError": "Ralat berlaku semasa log masuk.",
    "userNotFoundError": "Tiada pengguna ditemui dengan e-mel ini.",
    "wrongPasswordError": "Kata laluan salah. Sila cuba lagi.",
    "invalidEmailError": "Alamat e-mel tidak sah.",
    "roleMismatchError": "Peranan tidak sepadan. Sila log masuk menggunakan tab yang betul (Pelajar atau Ibu Bapa).",
    "userDataNotFoundError": "Data pengguna tidak ditemui di Firestore.",

    "hi": "Hai",
    "home": "UTAMA",
    "inboxTab": "MEL",
    "apps": "APLIKASI",
    "classTab": "KELAS",
    "adminTab": "ADMIN",
    "childTab": "ANAK",
    "profile": "PROFIL",
    "inboxTitle": "Mel",
    "appsTitle": "Aplikasi",
    "childTitle": "Anak",

    "student": "Pelajar",
    "parent": "Ibu Bapa",
    "tutor": "Tutor",
    "administrator": "Pentadbir",
    "enterInfo": "Sila masukkan maklumat anda di bawah.",
    "fullName": "Nama Penuh",
    "fullNameError": "Sila masukkan nama penuh anda",
    "gender": "Jantina",
    "genderError": "Sila pilih jantina anda",
    "email": "Alamat E-mel",
    "emailInfo": "E-mel",
    "emailError": "Sila masukkan e-mel anda",
    "emailInvalid": "Sila masukkan e-mel yang sah",
    "password": "Kata Laluan",
    "passwordError": "Sila masukkan kata laluan anda",
    "passwordLengthError": "Kata laluan mestilah sekurang-kurangnya 6 aksara",
    "confirmPassword": "Sahkan Kata Laluan",
    "confirmPasswordError": "Sila sahkan kata laluan anda",
    "passwordMismatch": "Kata laluan tidak sepadan",
    "submit": "Hantar",
```

Figure 4.1.15.6 Screenshot of l10n/app_ms.arb



```
{
    "welcomeTuitionify": "欢迎来到Tuitionify。",
    "chooseRole": "请选择您的角色。",
    "studentParent": "学生与家长",
    "staff": "工作人员",
    "signIn": "登录",
    "signUp": "注册",

    "dontHaveAccount": "没有账户？注册",
    "signInError": "登录时发生错误。",
    "userNotFoundError": "未找到使用此电子邮件的用户。",
    "wrongPasswordError": "密码错误。请重试。",
    "invalidEmailError": "电子邮件地址无效。",
    "roleMismatchError": "角色不匹配。请使用正确的选项卡（学生或家长）登录。",
    "userDataNotFoundError": "在Firestore中未找到用户数据。",

    "hi": "你好",
    "home": "主页",
    "inboxTab": "信箱",
    "apps": "应用",
    "classTab": "班级",
    "adminTab": "行政",
    "childTab": "孩子",
    "profile": "个人资料",
    "inboxTitle": "信箱",
    "appsTitle": "应用",
    "childTitle": "孩子",

    "student": "学生",
    "parent": "家长",
    "tutor": "导师",
    "administrator": "管理员",
    "enterInfo": "请在下方输入您的信息。",
    "fullName": "全名",
    "fullNameError": "请输入您的全名",
    "gender": "性别",
    "genderError": "请选择您的性别",
    "email": "电子邮件地址",
    "emailInfo": "邮件",
    "emailError": "请输入您的电子邮件",
    "emailInvalid": "请输入有效的电子邮件",
    "password": "密码",
    "passwordError": "请输入您的密码",
    "passwordLengthError": "密码必须至少6个字符",
    "confirmPassword": "确认密码",
    "confirmPasswordError": "请确认您的密码",
    "passwordMismatch": "密码不匹配",
    "submit": "提交",
```

Figure 4.1.15.7 Screenshot of l10n/app_zh.arb

4.2 Summary

In conclusion, this chapter outlines the fifteen interconnected modules that create a complete tuition centre management system. The fifteen interconnected modules, including the user authentication module, home module, inbox module, announcement module, calendar module, payment module, report and analytics module, courses module, classes module, students module, tutors module, children module, profile module, face recognition module, and language preferences module. Together, these modules form a cohesive environment that meets the different demands of administrators, tutors, students, and parents while retaining role-based access control.

Chapter 5
System Implementation

This chapter outlines five different sections, which consist of hardware and software requirements, configuration procedures, system operational workflows, development issues and challenges, and a summary.

5.1 System Requirements

To support the development and operation of the projects, the hardware and software requirements are specified. The hardware requirements include a laptop for development and a mobile device for running the mobile application. Moreover, the software requirements include using Visual Studio Code, Firebase, Node.js, Dart, JavaScript, Python, and Flutter. These requirements and components ensure a robust and scalable system that can support the project's modules.

5.1.1 Hardware Requirements

The hardware involved in this project are a laptop and an Android mobile device. The laptop is required to build the system, while the mobile device is required to operate the system.

Table 5.1.1.1 Table of laptop specifications

Components	**Specifications**
Model	ASUS TUF Gaming A15 FA506NCR
Processor	AMD Ryzen 7 7435HS @ 3.10GHz
System Type	64-bit operating system, x64-based processor
Windows	Windows 11 Home Single Language
Graphic Card	NVIDIA GeForce RTX 3050
Storage	512GB
RAM	16.0 GB (15.8 GB usable)

Table 5.1.1.2 Table of mobile device specifications

Components	**Specifications**
Model	M2010J19CG (POCO M3)
CPU	Octa-core Max 2.0GHz
Storage	64.00 GB
RAM	4.00 + 1.00 GB

5.1.2 Software Requirements

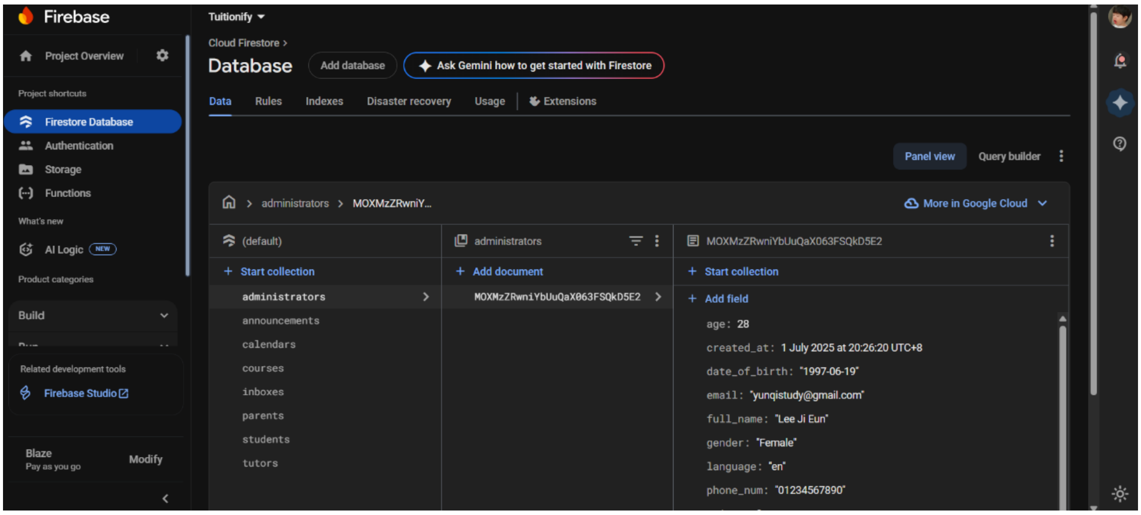
Table 5.1.2.1 Table of software requirements

Components	**Requirements**
Tools	**Visual Studio Code** Visual Studio Code (VS Code) is a comprehensive integrated development environment (IDE) [9]. Besides, it is also a multi-operating system and multi-language programming editor [9]. It supports a huge number of programming languages, such as Dart, Python, C++, JavaScript, etc. It is a strong tool for source code management and a debugging code editor.
	Firebase Database Firebase is a Backend-as-a-Service (BaaS) and is based on the infrastructure of Google [10]. It offers a vast array of services, features, and application development tools for developers. Firebase is also classified as one of the NoSQL database applications for storing data. It provides key features such as authentication, real-time database, Firestore database, etc.
	Node.js Node.js is a cost-free and open-source JavaScript runtime that works on Linux, Mac, Windows, and other platforms. It allows running JavaScript programs outside of a browser on the internet, and makes server-side JavaScript development possible [11].
Languages, Libraries, and Framework	**Dart** Dart is an object-oriented programming language that is used to build software applications, such as mobile, web, and desktop applications,

	with the Flutter framework [12]. With Flutter, developers can create cross-platform applications for iOS and Android by using the same code.
	Python Python is a high-level and object-oriented programming language that can be used for software development, server-side web development, etc [13]. Due to its high-level and built-in data structures, it is widely used for rapid application development (RAD).
	JavaScript JavaScript is an interpreted scripting language. It is generally used to create dynamic front-ends for online applications, and it may also be used in back-end development alongside Node.js [14]. It can be used in various situations, including mobile and web development, artificial intelligence (AI), virtual reality (VR), etc.
	Flutter Flutter is a user interface (UI) framework that is used to develop native mobile applications that facilitate the rapid development of iOS and Android apps by utilising a single codebase [15]. It is powered by Dart, and developers may utilise it to build powerful and high-performance applications across different platforms.

5.2 Setting and Configuration

The “Smart Management System for Tuition Centre Operations” leverages a robust set of tools for application development and data management. The system is created by utilising Visual Studio Code and Flutter with the Dart programming language to create a cross-platform and responsive mobile application that works on Android and iOS. In this project, the focus is on Android mobile application development. Besides, this system makes the connection with the Firebase database, which is a NoSQL cloud-based database. Cloud Firestore database helps to store and manage structured data securely and with real-time synchronization, as demonstrated in Figure 5.2.1. Moreover, Firebase Storage offers a scalable and secure cloud storage for images such as profile photos, which are needed for biometric facial recognition, as illustrated in Figure 5.2.2. Furthermore, the system integrates with Firebase Authentication to handle user authentication safely with its role-based access control, as depicted in Figure 5.2.3. According to Figure 5.2.4, the Firebase Functions service is used to provide email notification functions for the system.

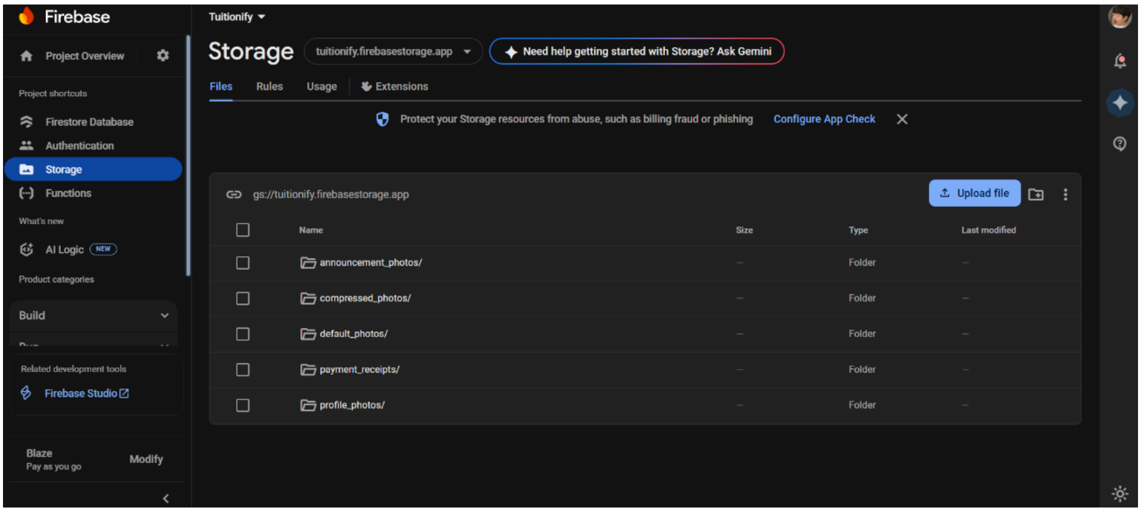


administrators
announcements
calendars
courses
inboxes
parents
students
tutors

MOXMzZRwniYbUuQaX063FSQkD5E2

age: 28
created_at: 1 July 2025 at 20:26:20 UTC+8
date_of_birth: "1997-06-19"
email: "yunqistudy@gmail.com"
full_name: "Lee Ji Eun"
gender: "Female"
language: "en"
phone_num: "01234567890"

Figure 5.2.1 Screenshot of Cloud Firestore

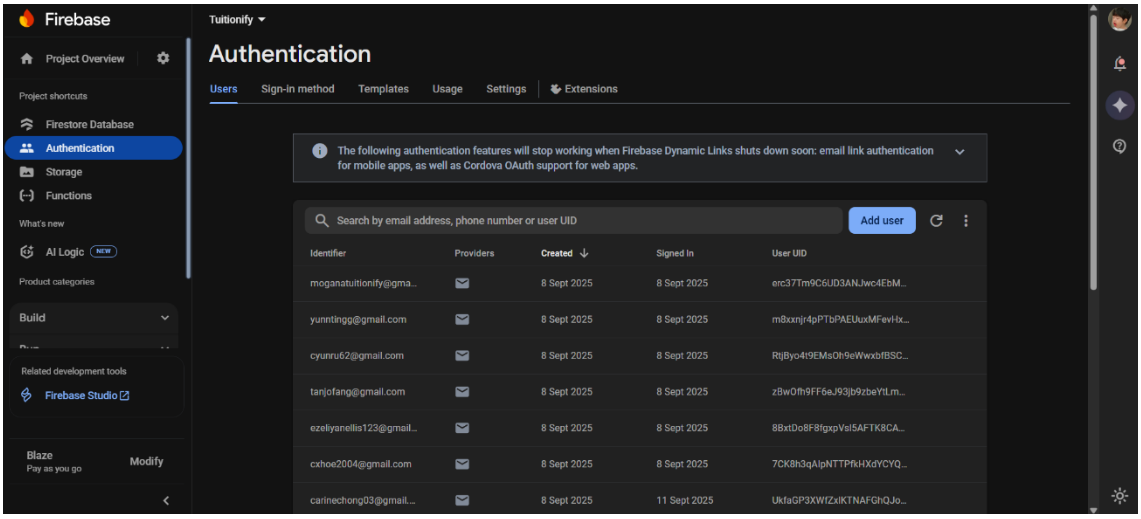


Protect your Storage resources from abuse, such as billing fraud or phishing

gs://tuitionify.firebasestorage.app

□	Name	Size	Type	Last modified
□	announcement_photos/	–	Folder	–
□	compressed_photos/	–	Folder	–
□	default_photos/	–	Folder	–
□	payment_receipts/	–	Folder	–
□	profile_photos/	–	Folder	–

Figure 5.2.2 Screenshot of Firebase Storage

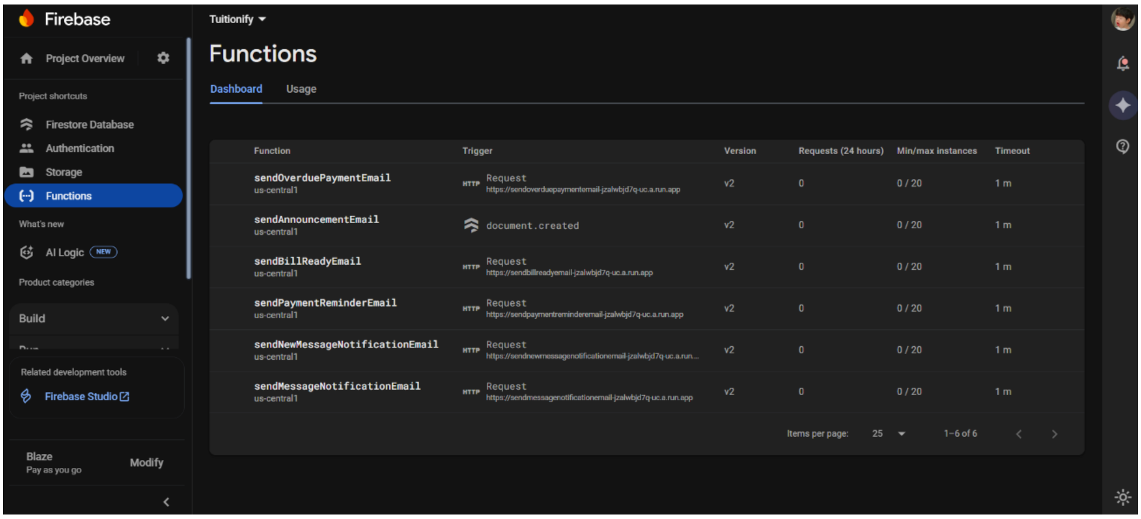


The following authentication features will stop working when Firebase Dynamic Links shuts down soon: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.

Search by email address, phone number or user UID

Identifier	Providers	Created	Signed In	User UID
moganatuitionify@gma...	✉	8 Sept 2025	8 Sept 2025	erc37Tm9C6UD3ANJwc4EbM...
yunntingg@gmail.com	✉	8 Sept 2025	8 Sept 2025	m8xxnjr4pPTbPAEUuxMFevHx...
cyunru62@gmail.com	✉	8 Sept 2025	8 Sept 2025	RtjByo4t9EMsOh9eWwxbfBSC...
tanjofang@gmail.com	✉	8 Sept 2025	8 Sept 2025	zBwOfh9FF6eJ93jb9zbeYtLm...
ezeliyanellis123@gmail...	✉	8 Sept 2025	8 Sept 2025	8BxtDo8F8fgxpVsI5AFTK8CA...
cxhoe2004@gmail.com	✉	8 Sept 2025	8 Sept 2025	7CK8h3qAlpNTTPfkHXdYCYQ...
carinechong03@gmail....	✉	8 Sept 2025	11 Sept 2025	UkfaGP3XWfZxlKTNAFGhQJo...

Figure 5.2.3 Screenshot of Firebase Authentication

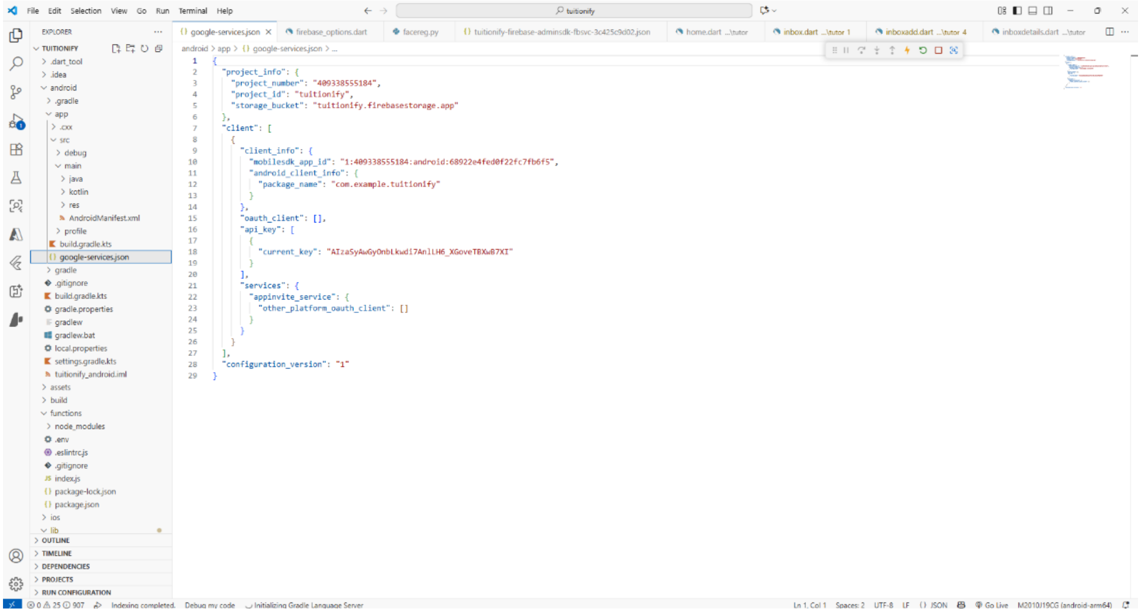


Function	Trigger	Version	Requests (24 hours)	Min/max instances	Timeout
sendOverduePaymentEmail us-central1	HTTP Request https://sendoverduepaymentemail-jzalwbjd7q-uc.a.run.app	v2	0	0 / 20	1 m
sendAnnouncementEmail us-central1	document.created	v2	0	0 / 20	1 m
sendBillReadyEmail us-central1	HTTP Request https://sendbillreadyemail-jzalwbjd7q-uc.a.run.app	v2	0	0 / 20	1 m
sendPaymentReminderEmail us-central1	HTTP Request https://sendpaymentreminderemail-jzalwbjd7q-uc.a.run.app	v2	0	0 / 20	1 m
sendNewMessageNotificationEmail us-central1	HTTP Request https://sendnewmessagenotificationemail-jzalwbjd7q-uc.a.run...	v2	0	0 / 20	1 m
sendMessageNotificationEmail us-central1	HTTP Request https://sendmessagenotificationemail-jzalwbjd7q-uc.a.run.app	v2	0	0 / 20	1 m

Items per page: 25 1–6 of 6

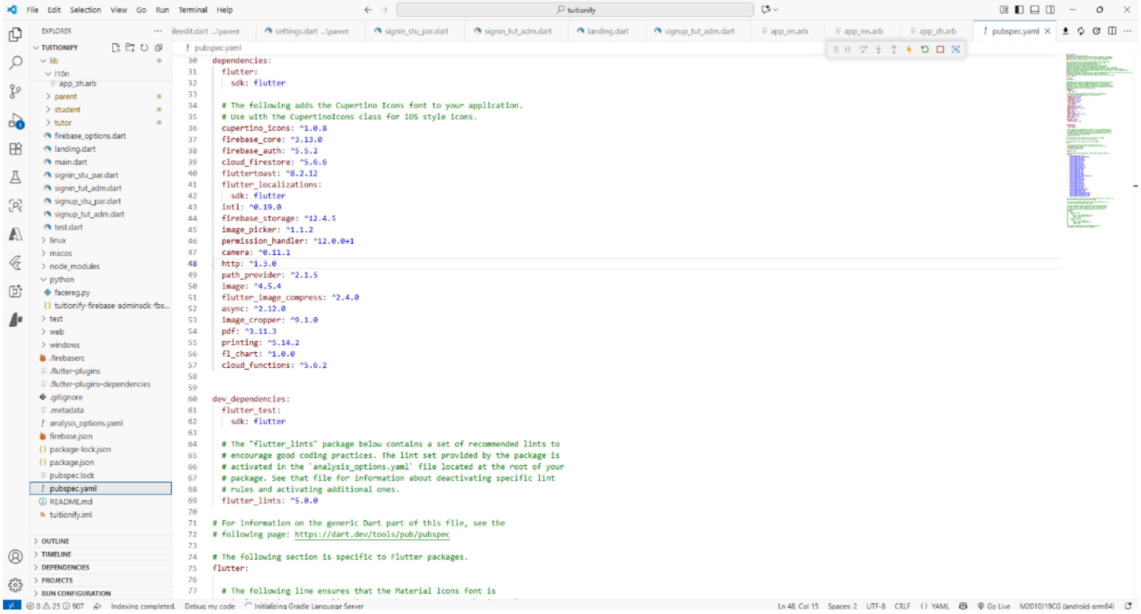
Figure 5.2.4 Screenshot of Firebase Functions

Besides, the “google-services.json” file, which includes the configuration information, is used to link the application to the Firebase database, as shown in Figure 5.2.5. According to Figure 5.2.6, the “pubspec.yaml" file manages the project’s dependencies to ensure the smooth integration of all required tools and packages. The tools need to be set up correctly in order to support the development of the comprehensive and scalable tuition centre management system.



```json
{
  "project_info": {
    "project_number": "409338555184",
    "project_id": "tuitionify",
    "storage_bucket": "tuitionify.firebasestorage.app"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:409338555184:android:68922e4fed0f22fc7fb6f5",
        "android_client_info": {
          "package_name": "com.example.tuitionify"
        }
      },
      "oauth_client": [],
      "api_key": [
        {
          "current_key": "AIzaSyAwGyOnbLkwdi7AnlLH6_XGoveTBXwB7XI"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": []
        }
      }
    }
  ],
  "configuration_version": "1"
}
```

Figure 5.2.5 Screenshot of google-services.json



```yaml
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.8
  firebase_core: ^3.13.0
  firebase_auth: ^5.5.2
  cloud_firestore: ^5.6.6
  fluttertoast: ^8.2.12
  flutter_localizations:
    sdk: flutter
  intl: ^0.19.0
  firebase_storage: ^12.4.5
  image_picker: ^1.1.2
  permission_handler: ^12.0.0+1
  camera: ^0.11.1
  http: ^1.3.0
  path_provider: ^2.1.5
  image: ^4.5.4
  flutter_image_compress: ^2.4.0
  async: ^2.12.0
  image_cropper: ^9.1.0
  pdf: ^3.11.3
  printing: ^5.14.2
  fl_chart: ^1.0.0
  cloud_functions: ^5.6.2


dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of recommended lints to
  # encourage good coding practices. The lint set provided by the package is
  # activated in the `analysis_options.yaml` file located at the root of your
  # package. See that file for information about deactivating specific lint
  # rules and activating additional ones.
  flutter_lints: ^5.0.0

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.
flutter:

  # The following line ensures that the Material Icons font is
```

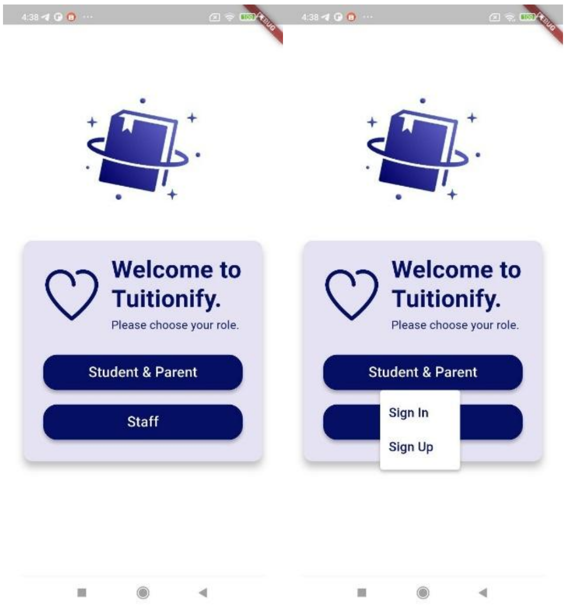
Figure 5.2.6 Screenshot of pubspec.yaml

5.3 System Operation

5.3.1 User Authentication Module

The user authentication module consists of signup, signin, and signout features. First of all, the user will be navigated to the landing page to choose their roles, such as “Student & Parent” or “Staff”, as shown in Figure 5.3.1.1. When the user selects the role, the system will ask for “Sign

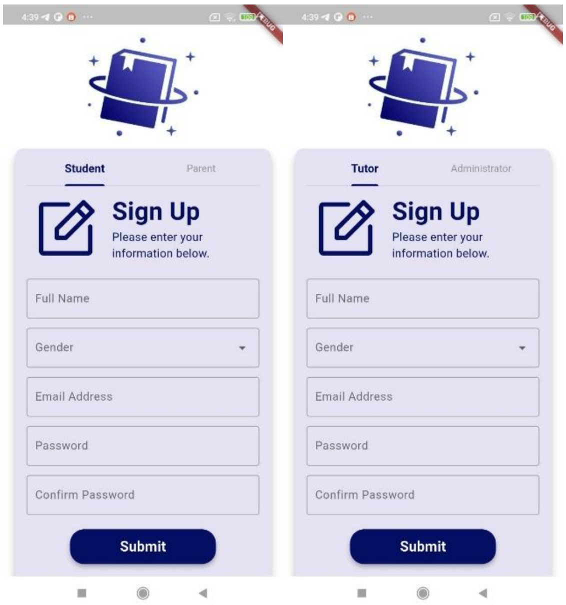
In" or "Sign Up". In Figure 5.3.1.2, the user is navigated to the "Sign Up" page, where they need to select their roles and enter their personal information, such as full name, gender, email address, passwords, and confirm password. When they click the "Submit" button, the system will verify and validate the inputs to ensure no empty fields, passwords match, the email is not in use, and the length of the password meets the requirements of a minimum length of 6 characters. Once the system verifies and creates the new account, the user will be navigated to the "Sign In" page.



Welcome to Tuitionify.
Please choose your role.

Welcome to Tuitionify.
Please choose your role.

Figure 5.3.1.1 Screenshot of Landing Page

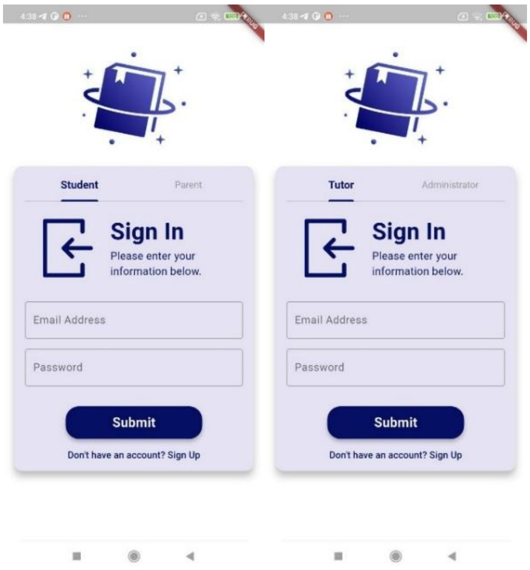


Sign Up
Please enter your information below.

Sign Up
Please enter your information below.

Figure 5.3.1.2 Screenshot of Sign Up Page

Figure 5.3.1.3 shows the “Sign In” page, where the user is required to enter their email address and password. Furthermore, the system will evaluate the credentials against the Firebase Authentication by ensuring the email address exists, the password is correct, and the selected role matches. Once the account is verified, a message “Sign in successfully” will be displayed. If there are errors, then the error messages will be displayed, such as “Role mismatch” and “The supplied auth credential is incorrect”. In Figure 5.3.1.4, the “Sign Out” button will be located in the settings of the profile page. When the user clicks on the button, they will be signed out.



Student Parent

Sign In

Please enter your information below.

Email Address

Password

Submit

Don't have an account? Sign Up

Tutor Administrator

Sign In

Please enter your information below.

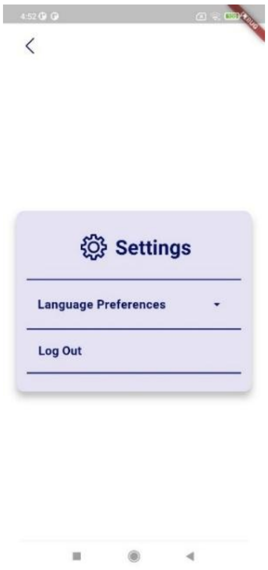
Email Address

Password

Submit

Don't have an account? Sign Up

Figure 5.3.1.3 Screenshot of Sign In Page



Settings

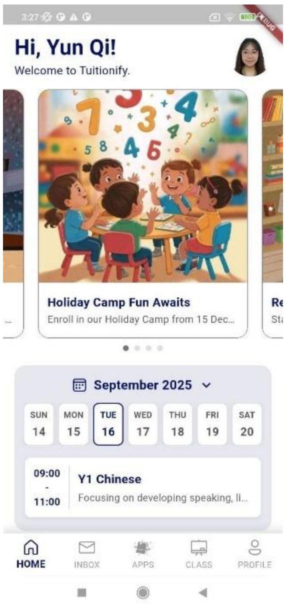
Language Preferences

Log Out

Figure 5.3.1.4 Screenshot of Sign Out Page

5.3.2 Home Module

The home page serves as the main interface for administrators, tutors, students, and parents and provides role-specific functionalities, as shown in Figure 5.3.2.1. Each user is greeted at the top of the screen with their name, profile photo, and a welcome text message, "Welcome to Tuitionify". The main content area has a scrollable interface that adapts to phone screen dimensions and guarantees accessibility on a variety of device dimensions. At the top of the main content area will be the announcement section, which will retrieve and show the five most recent announcements from the Cloud Firestore Database. Each announcement is shown in a horizontally swipeable carousel format, with associated photo, title, and description. Users are allowed to navigate between announcements with left and right swipe gestures. Users can click on any announcement to get detailed information, including the complete title and full description, as depicted in Figure 5.3.2.2.


3:27

Hi, Yun Qi!

Welcome to Tuitionify.

Holiday Camp Fun Awaits

Enroll in our Holiday Camp from 15 Dec...

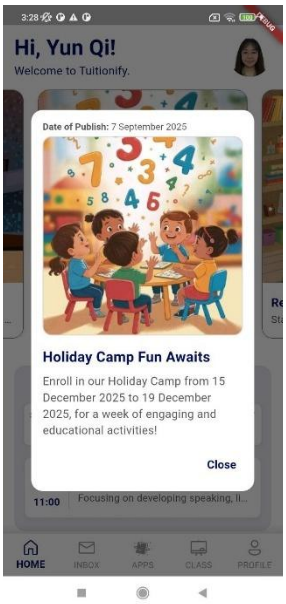
September 2025

SUN	MON	TUE	WED	THU	FRI	SAT
14	15	16	17	18	19	20

09:00 - 11:00 Y1 Chinese

Focusing on developing speaking, li...

Figure 5.3.2.1 Screenshot of Home Page



Hi, Yun Qi!

Welcome to Tuitionify.

Date of Publish: 7 September 2025

Holiday Camp Fun Awaits

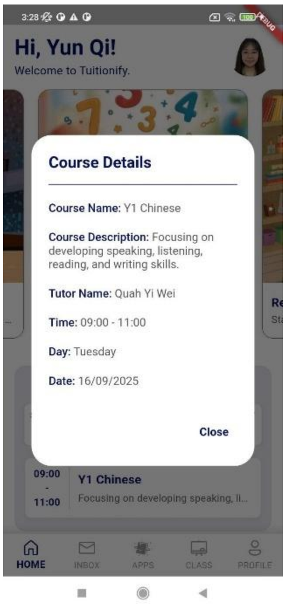
Enroll in our Holiday Camp from 15 December 2025 to 19 December 2025, for a week of engaging and educational activities!

Close

11:00 Focusing on developing speaking, li...

Figure 5.3.2.2 Screenshot of Announcement Details in Home Page

Below the announcement section, it will be the calendar schedule section sourced from Cloud Firestore. This section features horizontal scrolling functionality for date navigation and also the calendar icon that allows users to navigate to specific dates for viewing targeted class schedules. The schedules are ordered chronologically by start time. Each class schedule is clickable to show the detailed information, such as course name, course description, day, date, duration, etc, as illustrated in Figure 5.3.2.3. The home page will end with a role-specific bottom navigation bar with five distinct tabs.



Hi, Yun Qi!

Welcome to Tuitionify.

Course Details

Course Name: Y1 Chinese

Course Description: Focusing on developing speaking, listening, reading, and writing skills.

Tutor Name: Quah Yi Wei

Time: 09:00 - 11:00

Day: Tuesday

Date: 16/09/2025

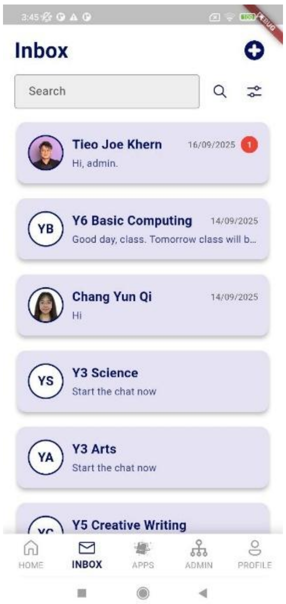
Close

09:00 - 11:00 **Y1 Chinese** Focusing on developing speaking, li...

Figure 5.3.2.3 Screenshot of Schedule Details in Home Page

5.3.3 Inbox Module

The inbox module displays a full message interface that can manage both private and course group conversations, as shown in Figure 5.3.3.1. The content of the inbox page will be scrollable if the content exceeds the height of the phone screen to guarantee all inbox chats are accessible. All the chat boxes are arranged in descending chronological order, and each box is clickable to navigate administrators, tutors, students, or parents to the extensive chat history, as illustrated in Figure 5.3.3.2. All users can send new messages or upload new images to the chat, and the new messages will be reflected in the chat interface. To prevent notification spam, the system includes an intelligent email notification system that uses a 30-second countdown timer to batch several messages before delivering a single email notification to all chat participants, as depicted in Figure 5.3.3.3.



Tieo Joe Khern 16/09/2025 1
Hi, admin.

YB Y6 Basic Computing 14/09/2025
Good day, class. Tomorrow class will b...

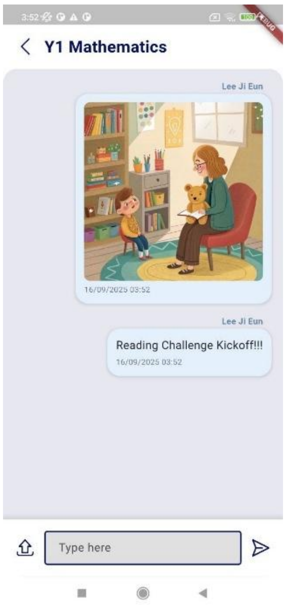
Chang Yun Qi 14/09/2025
Hi

YS Y3 Science
Start the chat now

YA Y3 Arts
Start the chat now

YC Y5 Creative Writing

Figure 5.3.3.1 Screenshot of Inbox Page



3:52

Y1 Mathematics

Lee Ji Eun

16/09/2025 03:52

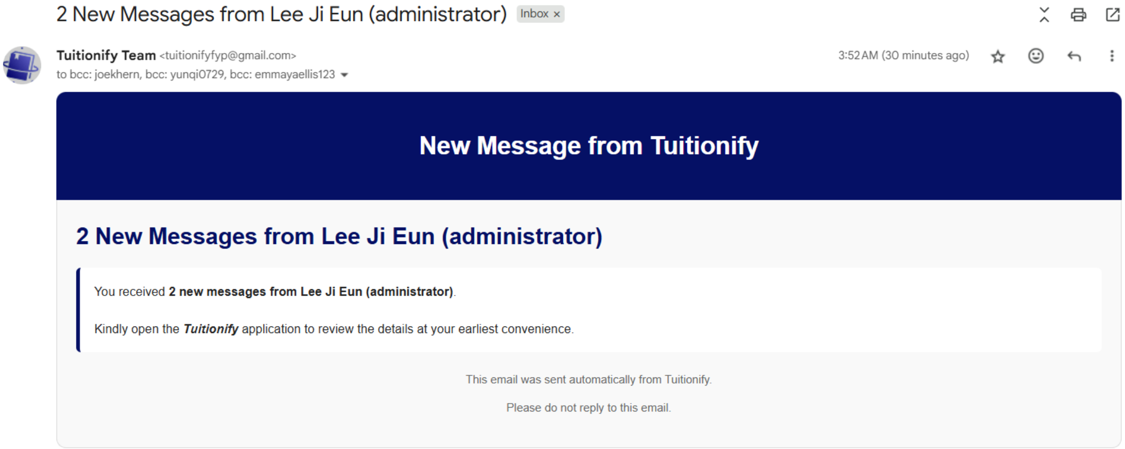
Lee Ji Eun

Reading Challenge Kickoff!!!

16/09/2025 03:52

Type here

Figure 5.3.3.2 Screenshot of Inbox Details Page



2 New Messages from Lee Ji Eun (administrator) Inbox ×

Tuitionify Team <tuitionifyfyp@gmail.com>

to bcc: joekhern, bcc: yunqi0729, bcc: emmayaellis123

3:52 AM (30 minutes ago)

New Message from Tuitionify

2 New Messages from Lee Ji Eun (administrator)

You received **2 new messages from Lee Ji Eun (administrator)**.

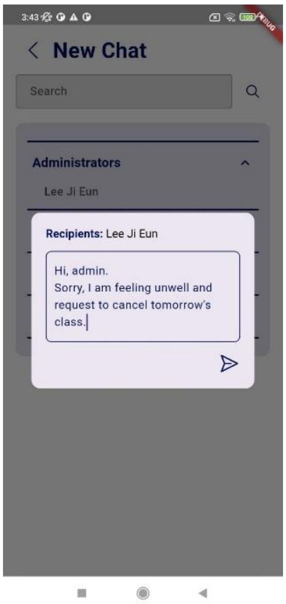
Kindly open the ***Tuitionify*** application to review the details at your earliest convenience.

This email was sent automatically from Tuitionify.

Please do not reply to this email.

Figure 5.3.3.3 Screenshot of Batch Chat Email Notification

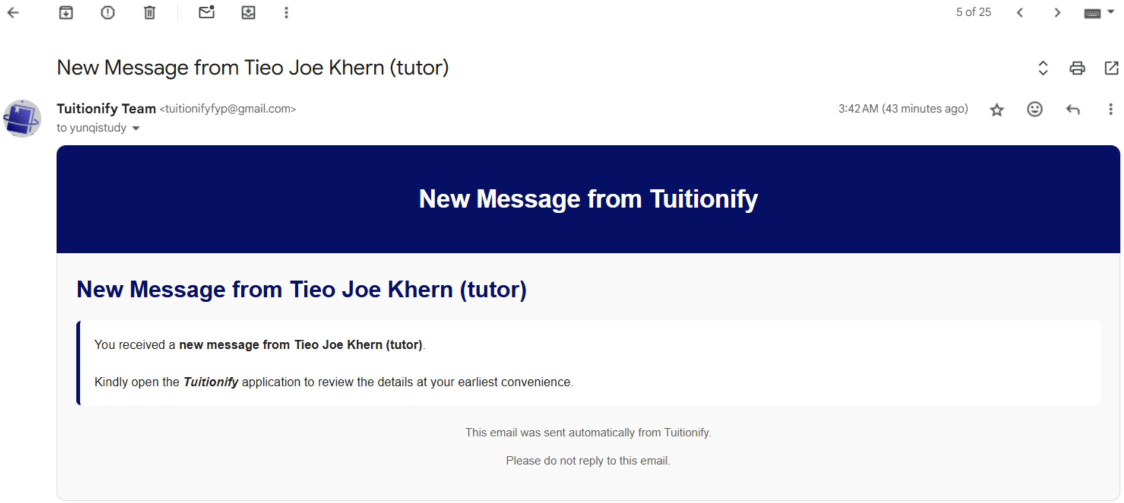
All users may start new private conversations by clicking the “add” icon at the top right of the page, where selecting a contact immediately establishes a private chat box and sends an instant email notification to the recipient, as shown in Figure 5.3.3.4 and Figure 5.3.3.5. Furthermore, the module also provides search and sorting options that allow users to search chat by keywords or organise the chat boxes, based on Figure 5.3.3.6.



Recipients: Lee Ji Eun

Hi, admin.
Sorry, I am feeling unwell and request to cancel tomorrow's class.

Figure 5.3.3.4 Screenshot of Create New Private Chat Page



New Message from Tieo Joe Khern (tutor)

Tuitionify Team <tuitionifyfyp@gmail.com>
to yunqistudy

3:42 AM (43 minutes ago)

New Message from Tuitionify

New Message from Tieo Joe Khern (tutor)

You received a **new message from Tieo Joe Khern (tutor)**.

Kindly open the ***Tuitionify*** application to review the details at your earliest convenience.

This email was sent automatically from Tuitionify.

Please do not reply to this email.

Figure 5.3.3.5 Screenshot of Chat Email Notification

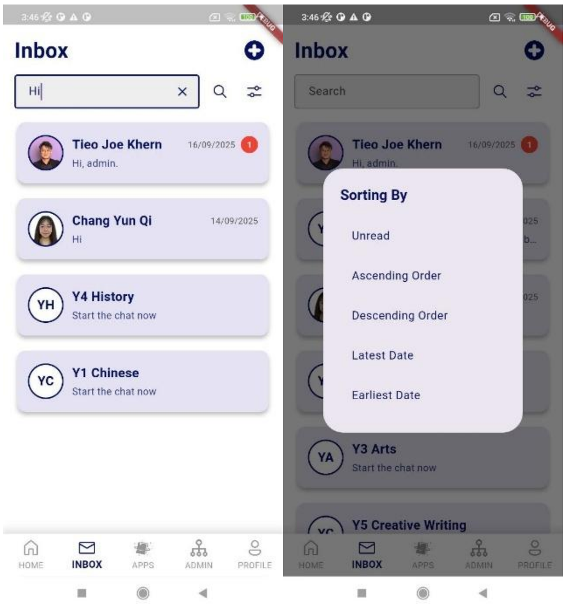
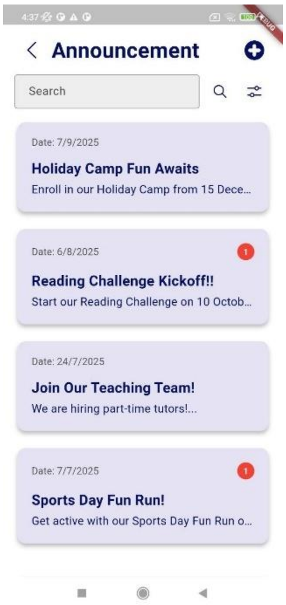


Figure 5.3.3.6 Screenshot of Search and Sorting Functions in Inbox Page

5.3.4 Announcement Module

The announcement page employs a scrollable container architecture that handles content overflow beyond the phone screen size. According to Figure 5.3.4.1, all announcements are organised in descending chronological order. Each announcement is clickable, and the user will be navigated to a comprehensive announcement details page, which will display the title, description, and date of publication, as shown in Figure 5.3.4.2. Besides, administrators are allowed to add new announcements, and an email notification will be sent to all the users, shown in Figures 5.3.4.3 and 5.3.4.4. Administrators also allow editing existing announcements shown in Figure 5.3.4.5. Based on Figures 5.3.4.6, all users are allow to search specific announcements using keyword searches or sort content by various sorting options.



4:37

Announcement

Search

Date: 7/9/2025

Holiday Camp Fun Awaits

Enroll in our Holiday Camp from 15 Dece...

Date: 6/8/2025

1

Reading Challenge Kickoff!!

Start our Reading Challenge on 10 Octob...

Date: 24/7/2025

Join Our Teaching Team!

We are hiring part-time tutors!...

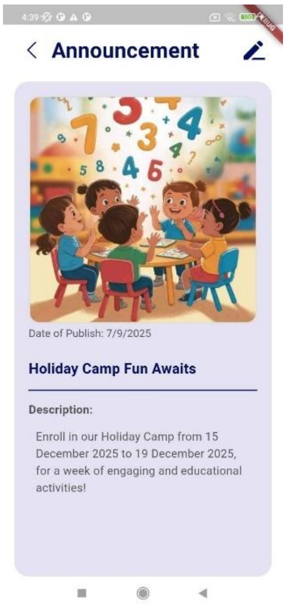
Date: 7/7/2025

1

Sports Day Fun Run!

Get active with our Sports Day Fun Run o...

Figure 5.3.4.1 Screenshot of Announcement Page



4:39

Announcement

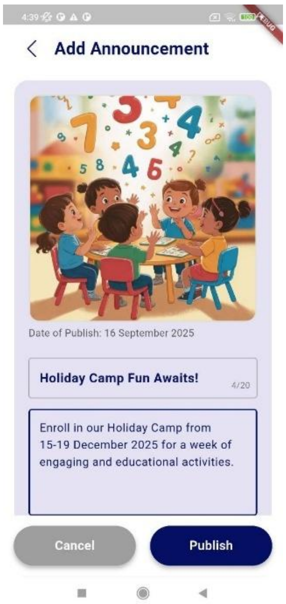
Date of Publish: 7/9/2025

Holiday Camp Fun Awaits

Description:

Enroll in our Holiday Camp from 15 December 2025 to 19 December 2025, for a week of engaging and educational activities!

Figure 5.3.4.2 Screenshot of Announcement Details Page

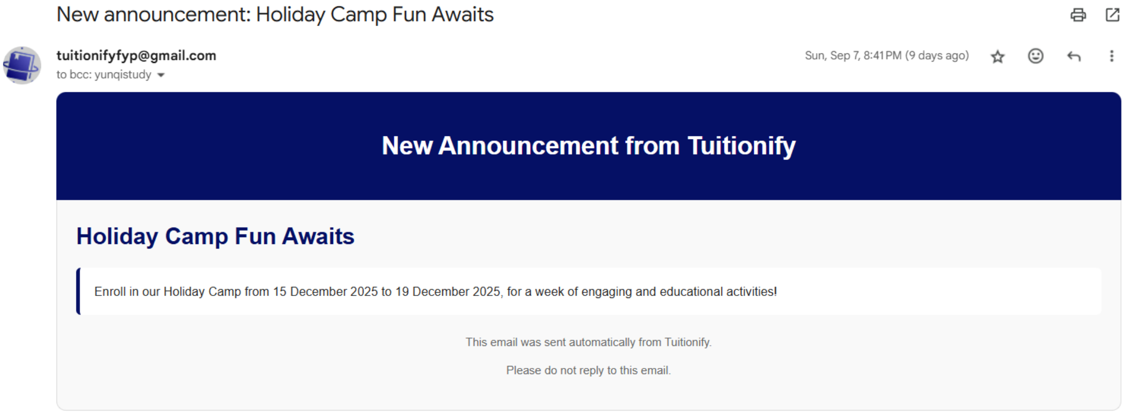


Date of Publish: 16 September 2025

Holiday Camp Fun Awaits! 4/20

Enroll in our Holiday Camp from 15-19 December 2025 for a week of engaging and educational activities.

Figure 5.3.4.3 Screenshot of Add New Announcement Page


New announcement: Holiday Camp Fun Awaits

tuitionifyfyp@gmail.com
to bcc: yunqistudy

Sun, Sep 7, 8:41 PM (9 days ago)

New Announcement from Tuitionify

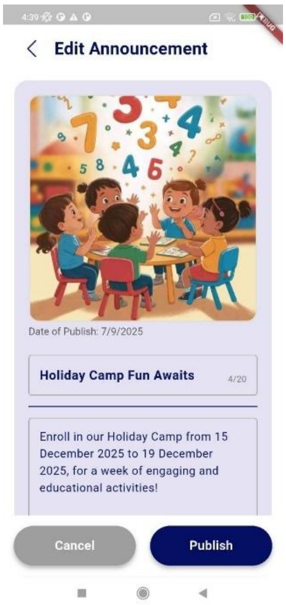
Holiday Camp Fun Awaits

Enroll in our Holiday Camp from 15 December 2025 to 19 December 2025, for a week of engaging and educational activities!

This email was sent automatically from Tuitionify.

Please do not reply to this email.

Figure 5.3.4.4 Screenshot of Announcement Email Notification

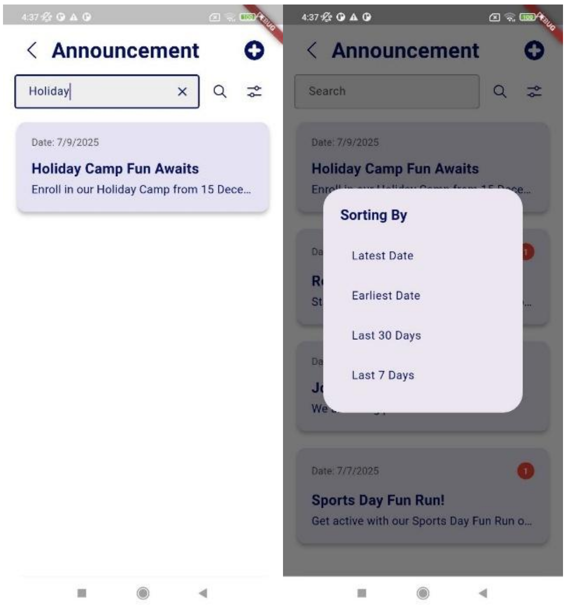


Date of Publish: 7/9/2025

Holiday Camp Fun Awaits 4/20

Enroll in our Holiday Camp from 15 December 2025 to 19 December 2025, for a week of engaging and educational activities!

Figure 5.3.4.5 Screenshot of Edit Existing Announcement Page



Holiday

Date: 7/9/2025

Holiday Camp Fun Awaits

Enroll in our Holiday Camp from 15 Dece...

Date: 7/7/2025

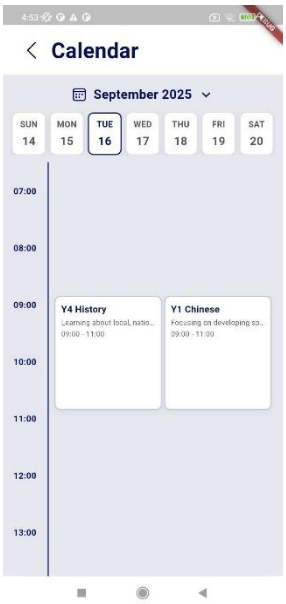
Sports Day Fun Run!

Get active with our Sports Day Fun Run o...

Figure 5.3.4.6 Screenshot of Search and Sorting Functions in Announcement Page

5.3.5 Calendar Module

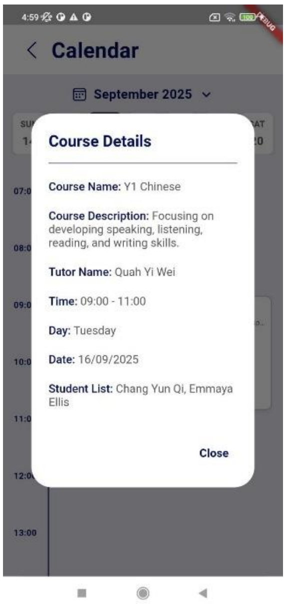
The calendar module offers role-based access to class schedules via a time-oriented display system. The calendar page will display a vertically scrollable timeline that spans the entire day from 00:00 to 23:59, which allows users to traverse through all available time slots and view class schedules, as shown in Figure 5.3.5.1. In Figure 5.3.5.2, when the user clicks the respective schedule, it will display the details of the schedule, including course name, course description, tutor name, time, day, date, and student list. The interface has two navigation mechanisms, which are horizontal scrolling for easy date browsing and a dedicated calendar icon that allows users to move directly to certain dates for targeted schedule viewing, as depicted in Figure 5.3.5.3.



Y4 History
Learning about local, natio...
09:00 - 11:00

Y1 Chinese
Focusing on developing sp...
09:00 - 11:00

Figure 5.3.5.1 Screenshot of Calendar Page



Course Details

Course Name: Y1 Chinese

Course Description: Focusing on developing speaking, listening, reading, and writing skills.

Tutor Name: Quah Yi Wei

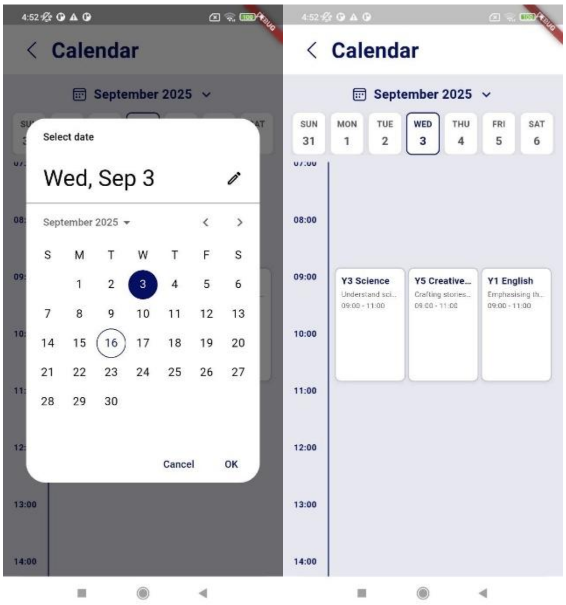
Time: 09:00 - 11:00

Day: Tuesday

Date: 16/09/2025

Student List: Chang Yun Qi, Emmaya Ellis

Figure 5.3.5.2 Screenshot of Schedule Details Page



Y3 Science
Understand sci...
09:00 - 11:00

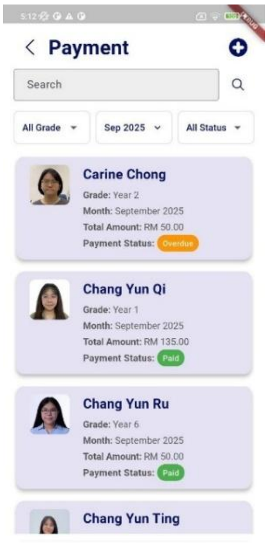
Y5 Creative...
Crafting stories...
09:00 - 11:00

Y1 English
Emphasising th...
09:00 - 11:00

Figure 5.3.5.3 Screenshot of Pop-up Calendar Page

5.3.6 Payment Module

Based on Figure 5.3.6.1, the payment page will display all the existing student bills, and they will be arranged in alphabetical order based on student names. When the bill is clicked, users will be navigated to a detailed billing page with comprehensive information, such as billing month, payment status, list of registered courses, individual course fees, total amount, etc, as shown in Figure 5.3.6.2. The module also allows users to download and save the invoices and receipts directly to their local devices for record-keeping purposes, as depicted in Figure 5.3.6.3.



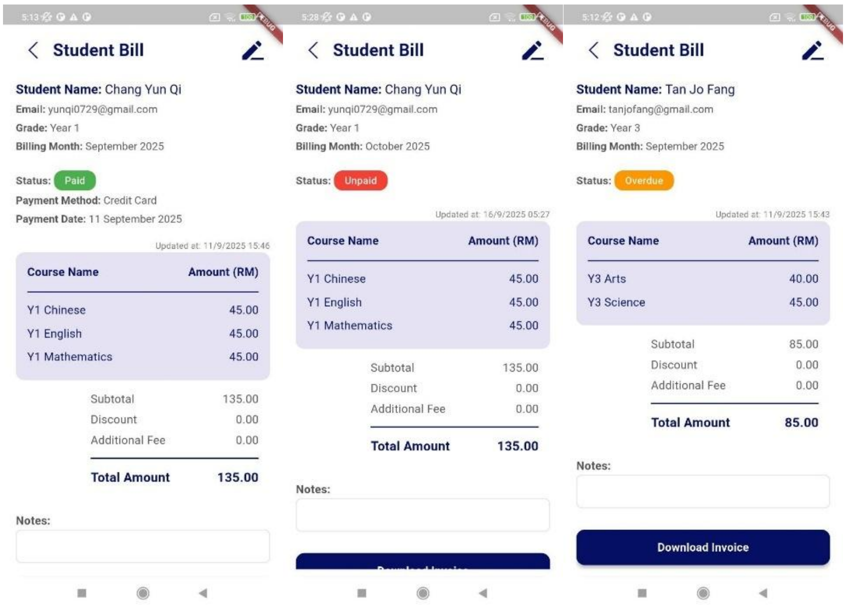
Carine Chong
Grade: Year 2
Month: September 2025
Total Amount: RM 50.00
Payment Status: Overdue

Chang Yun Qi
Grade: Year 1
Month: September 2025
Total Amount: RM 135.00
Payment Status: Paid

Chang Yun Ru
Grade: Year 6
Month: September 2025
Total Amount: RM 50.00
Payment Status: Paid

Chang Yun Ting

Figure 5.3.6.1 Screenshot of Payment Page



Student Bill

Student Name: Chang Yun Qi
Email: yunqi0729@gmail.com
Grade: Year 1
Billing Month: September 2025

Status: Paid
Payment Method: Credit Card
Payment Date: 11 September 2025

Updated at: 11/9/2025 15:46

Course Name	Amount (RM)
Y1 Chinese	45.00
Y1 English	45.00
Y1 Mathematics	45.00

Subtotal	135.00
Discount	0.00
Additional Fee	0.00
Total Amount	135.00

Notes:

Student Bill

Student Name: Chang Yun Qi
Email: yunqi0729@gmail.com
Grade: Year 1
Billing Month: October 2025

Status: Unpaid

Updated at: 16/9/2025 05:27

Course Name	Amount (RM)
Y1 Chinese	45.00
Y1 English	45.00
Y1 Mathematics	45.00

Subtotal	135.00
Discount	0.00
Additional Fee	0.00
Total Amount	135.00

Notes:

Student Bill

Student Name: Tan Jo Fang
Email: tanjofang@gmail.com
Grade: Year 3
Billing Month: September 2025

Status: Overdue

Updated at: 11/9/2025 15:43

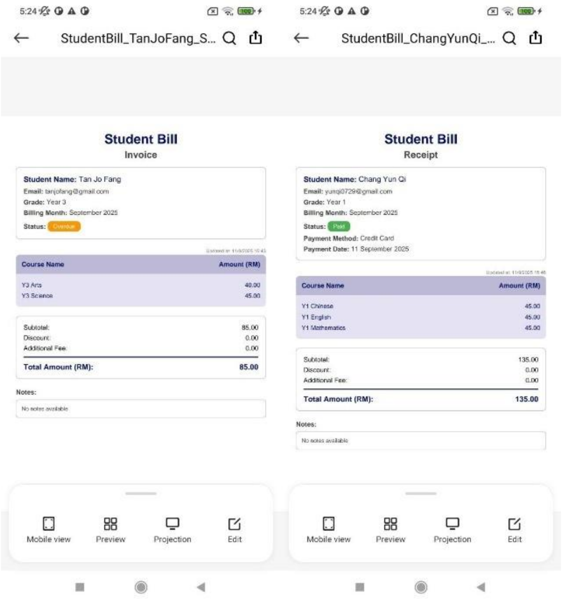
Course Name	Amount (RM)
Y3 Arts	40.00
Y3 Science	45.00

Subtotal	85.00
Discount	0.00
Additional Fee	0.00
Total Amount	85.00

Notes:

Download Invoice

Figure 5.3.6.2 Screenshot of Payment Details Page



StudentBill_TanJoFang_S...

Student Bill
Invoice

Student Name: Tan Jo Fang
Email: tanjofang@gmail.com
Grade: Year 3
Billing Month: September 2025
Status: Overdue

Updated at: 11/9/2025 15:43

Course Name	Amount (RM)
Y3 Arts	40.00
Y3 Science	45.00

Subtotal:	85.00
Discount:	0.00
Additional Fee:	0.00
Total Amount (RM):	85.00

Notes:
No notes available

StudentBill_ChangYunQi_...

Student Bill
Receipt

Student Name: Chang Yun Qi
Email: yunqi0729@gmail.com
Grade: Year 1
Billing Month: September 2025
Status: Paid
Payment Method: Credit Card
Payment Date: 11 September 2025

Updated at: 11/9/2025 15:46

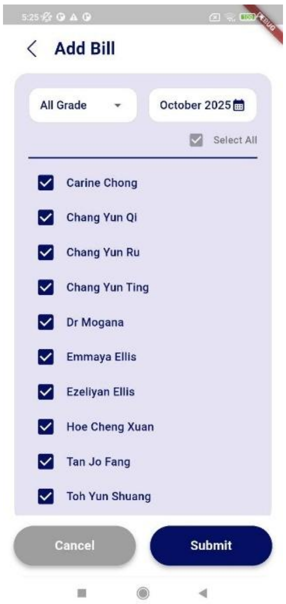
Course Name	Amount (RM)
Y1 Chinese	45.00
Y1 English	45.00
Y1 Mathematics	45.00

Subtotal:	135.00
Discount:	0.00
Additional Fee:	0.00
Total Amount (RM):	135.00

Notes:
No notes available

Figure 5.3.6.3 Screenshot of Downloaded Invoice and Receipt

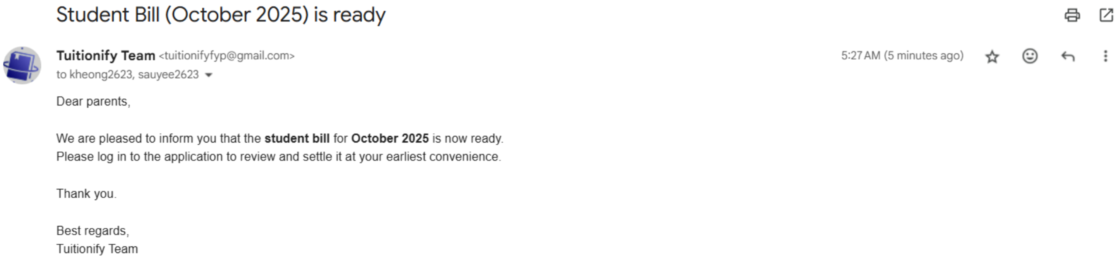
Besides, administrators are allowed to create new student bills and send automated email notifications to parents when bills are generated, as shown in Figures 5.3.6.4 and 5.3.6.5. Administrators are also able to edit the existing student bill, such as update payment status, modify course fees, etc, as illustrated in Figure 5.3.6.6. Moreover, the system supports direct financial transactions by allowing parents to access and pay their children's student invoices directly via the application in Figure 5.3.6.7. Lastly, students are given communication options through a “Notify Parents” button, which sends email reminders to parents when the bill payment status is Unpaid or Overdue, as depicted in Figure 5.3.6.8.



☑ Select All

☑ Carine Chong
☑ Chang Yun Qi
☑ Chang Yun Ru
☑ Chang Yun Ting
☑ Dr Mogana
☑ Emmaya Ellis
☑ Ezeliyan Ellis
☑ Hoe Cheng Xuan
☑ Tan Jo Fang
☑ Toh Yun Shuang

Figure 5.3.6.4 Screenshot of Add New Payment Page



Student Bill (October 2025) is ready

Tuitionify Team <tuitionifyfyp@gmail.com>
to kheong2623, sauyee2623

5:27 AM (5 minutes ago)

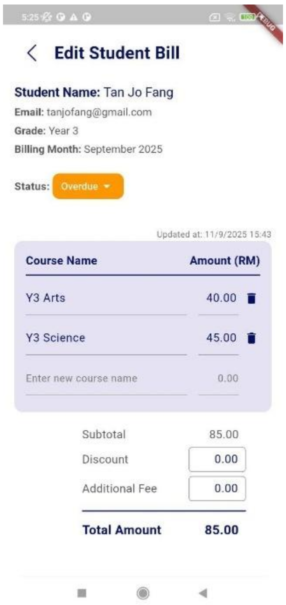
Dear parents,

We are pleased to inform you that the **student bill** for **October 2025** is now ready.
Please log in to the application to review and settle it at your earliest convenience.

Thank you.

Best regards,
Tuitionify Team

Figure 5.3.6.5 Screenshot of New Student Bill Email Notification



Edit Student Bill

Student Name: Tan Jo Fang
Email: tanjofang@gmail.com
Grade: Year 3
Billing Month: September 2025

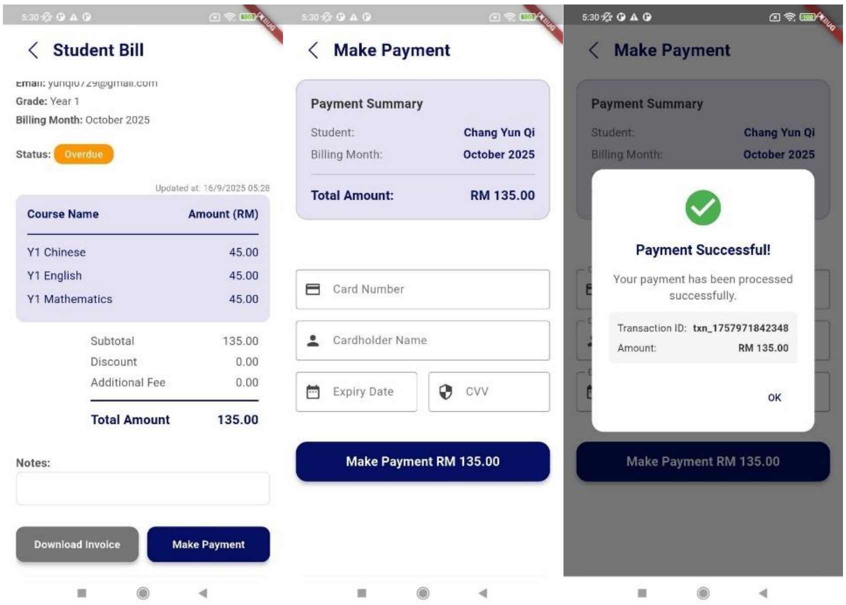
Status: Overdue

Updated at: 11/9/2025 15:43

Course Name	Amount (RM)
Y3 Arts	40.00
Y3 Science	45.00
Enter new course name	0.00

Subtotal	85.00
Discount	0.00
Additional Fee	0.00
Total Amount	**85.00**

Figure 5.3.6.6 Screenshot of Edit Existing Payment Page



Student Bill

Email: yunqi0729@gmail.com
Grade: Year 1
Billing Month: October 2025

Status: Overdue

Updated at: 16/9/2025 05:28

Course Name	Amount (RM)
Y1 Chinese	45.00
Y1 English	45.00
Y1 Mathematics	45.00

Subtotal	135.00
Discount	0.00
Additional Fee	0.00
Total Amount	**135.00**

Notes:

Make Payment

Payment Summary

Student:	**Chang Yun Qi**
Billing Month:	**October 2025**
Total Amount:	**RM 135.00**

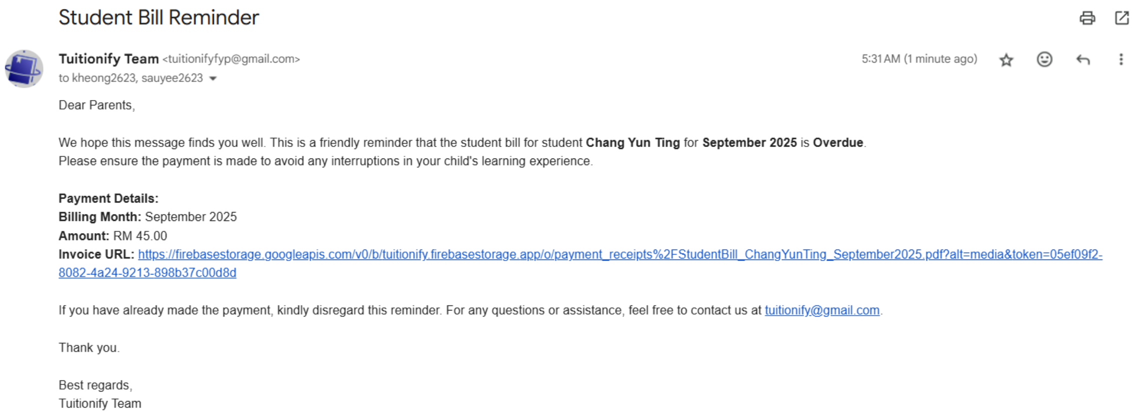
Make Payment

Payment Successful!

Your payment has been processed successfully.

Transaction ID:	**txn_1757971842348**
Amount:	**RM 135.00**

Figure 5.3.6.7 Screenshot of Make Payment Page



Student Bill Reminder

Tuitionify Team <tuitionifyfyp@gmail.com>
to kheong2623, sauyee2623

5:31 AM (1 minute ago)

Dear Parents,

We hope this message finds you well. This is a friendly reminder that the student bill for student **Chang Yun Ting** for **September 2025** is **Overdue**.
Please ensure the payment is made to avoid any interruptions in your child's learning experience.

Payment Details:
Billing Month: September 2025
Amount: RM 45.00
Invoice URL: https://firebasestorage.googleapis.com/v0/b/tuitionify.firebasestorage.app/o/payment_receipts%2FStudentBill_ChangYunTing_September2025.pdf?alt=media&token=05ef09f2-8082-4a24-9213-898b37c00d8d

If you have already made the payment, kindly disregard this reminder. For any questions or assistance, feel free to contact us at tuitionify@gmail.com.

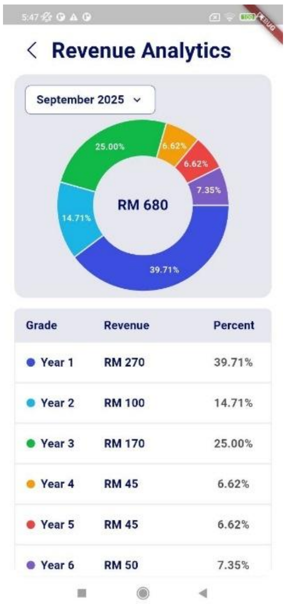
Thank you.

Best regards,
Tuitionify Team

Figure 5.3.6.8 Screenshot of Notify Parents Email Notification

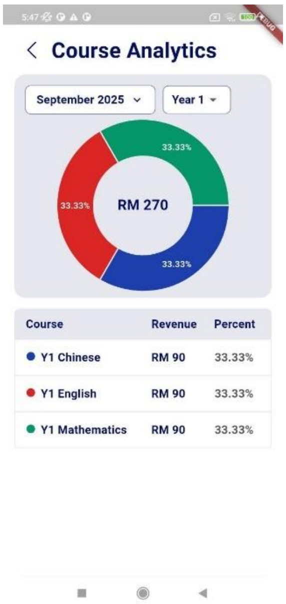
5.3.7 Report and Analytics Module

The report and analytics module consists of four unique analytical components, which are revenue analytics, course analytics, tutor analytics, and student analytics. Administrators have full access to all analytical components, while tutors, students, and parents are limited to student analytics only. Figure 5.3.7.1 shows the revenue analytics visualising financial performance using a pie chart that shows total monthly income, categorised by grade level. Figure 5.3.7.2 depicts the course analytics visualising financial performance using a pie chart that shows monthly revenue, categorised by individual courses. Figure 5.3.7.3 illustrates the tutor analytics by using a horizontal bar chart to display tutor attendance performance measures every month. Furthermore, Figures 5.3.7.4 to 5.3.7.7 show the student analytics through horizontal bar charts that display student attendance performance with each horizontal bar representing a different course.



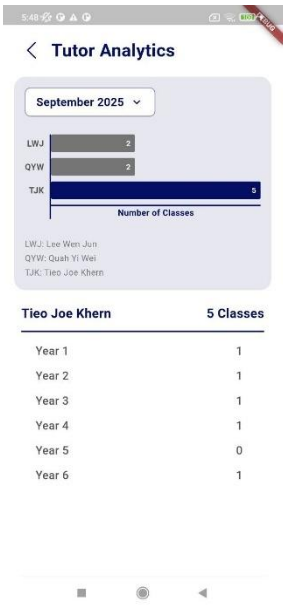
Grade	Revenue	Percent
● Year 1	RM 270	39.71%
● Year 2	RM 100	14.71%
● Year 3	RM 170	25.00%
● Year 4	RM 45	6.62%
● Year 5	RM 45	6.62%
● Year 6	RM 50	7.35%

Figure 5.3.7.1 Screenshot of Administrators' Revenue Analytics Page



Course	Revenue	Percent
● Y1 Chinese	RM 90	33.33%
● Y1 English	RM 90	33.33%
● Y1 Mathematics	RM 90	33.33%

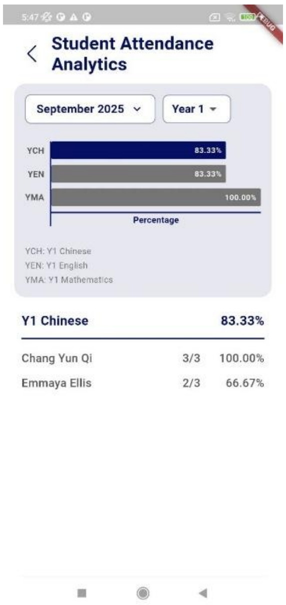
Figure 5.3.7.2 Screenshot of Administrators' Course Analytics Page



LWJ: Lee Wen Jun
QYW: Quah Yi Wei
TJK: Tieo Joe Khern

Tieo Joe Khern	5 Classes
Year 1	1
Year 2	1
Year 3	1
Year 4	1
Year 5	0
Year 6	1

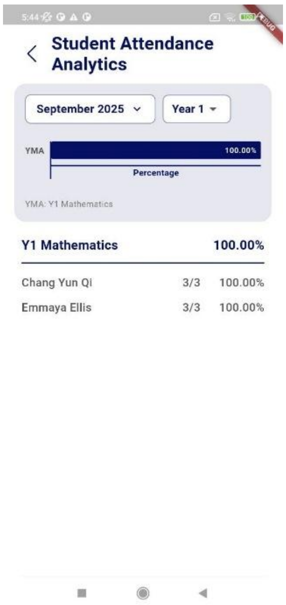
Figure 5.3.7.3 Screenshot of Administrators' Tutor Analytics Page



YCH: Y1 Chinese
YEN: Y1 English
YMA: Y1 Mathematics

Y1 Chinese		83.33%
Chang Yun Qi	3/3	100.00%
Emmaya Ellis	2/3	66.67%

Figure 5.3.7.4 Screenshot of Administrators' Student Analytics Page

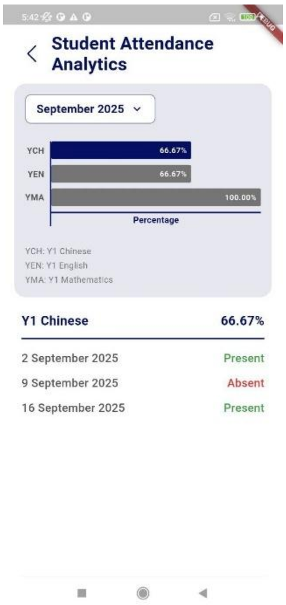


Student Attendance Analytics

YMA: Y1 Mathematics

Y1 Mathematics		100.00%
Chang Yun Qi	3/3	100.00%
Emmaya Ellis	3/3	100.00%

Figure 5.3.7.5 Screenshot of Tutors' Student Analytics Page

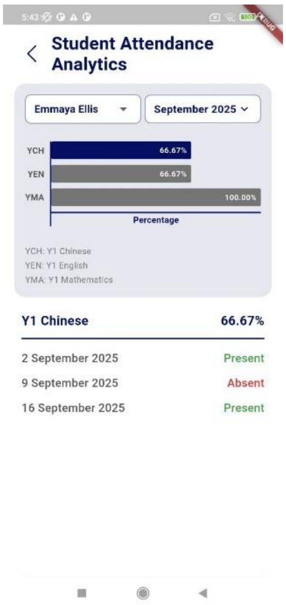


Student Attendance Analytics

YCH: Y1 Chinese
YEN: Y1 English
YMA: Y1 Mathematics

Y1 Chinese	66.67%
2 September 2025	Present
9 September 2025	Absent
16 September 2025	Present

Figure 5.3.7.6 Screenshot of Students' Student Analytics Page



Student Attendance Analytics

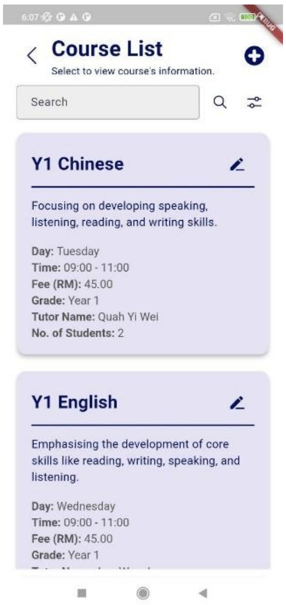
YCH: Y1 Chinese
YEN: Y1 English
YMA: Y1 Mathematics

Y1 Chinese	66.67%
2 September 2025	Present
9 September 2025	Absent
16 September 2025	Present

Figure 5.3.7.7 Screenshot of Parents' Student Analytics Page

5.3.8 Courses Module

Based on Figure 5.3.8.1, the courses page displays all existing courses alphabetically by course name. Each course is clickable; when it is clicked by administrators, they will be navigated to the course details page to view complete class history records and update student attendance records manually or use the facial recognition attendance tracking by clicking the “camera” icon, as shown in Figure 5.3.8.2. This module provides administrators with powerful course management abilities, which allow administrators to add new courses by clicking the “add” icon at the top right of the screen and entering the course information, such as course name, description, fee, date, time, etc, as depicted in Figure 5.3.8.3. According to Figure 5.3.8.4, administrators are also allowed to edit the existing course information, including updating course data, reassigning tutors, managing student enrollments, etc.



Course List

Select to view course's information.

Search

Y1 Chinese

Focusing on developing speaking, listening, reading, and writing skills.

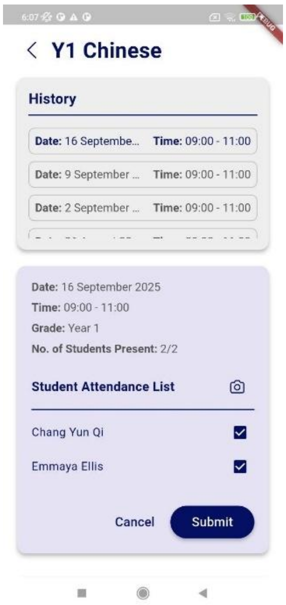
Day: Tuesday
Time: 09:00 - 11:00
Fee (RM): 45.00
Grade: Year 1
Tutor Name: Quah Yi Wei
No. of Students: 2

Y1 English

Emphasising the development of core skills like reading, writing, speaking, and listening.

Day: Wednesday
Time: 09:00 - 11:00
Fee (RM): 45.00
Grade: Year 1

Figure 5.3.8.1 Screenshot of Courses Page



Y1 Chinese

History

Date: 16 Septembe...	**Time:** 09:00 - 11:00
Date: 9 September ...	**Time:** 09:00 - 11:00
Date: 2 September ...	**Time:** 09:00 - 11:00

Date: 16 September 2025
Time: 09:00 - 11:00
Grade: Year 1
No. of Students Present: 2/2

Student Attendance List

Chang Yun Qi ☑
Emmaya Ellis ☑

Cancel Submit

Figure 5.3.8.2 Screenshot of Course Details Page

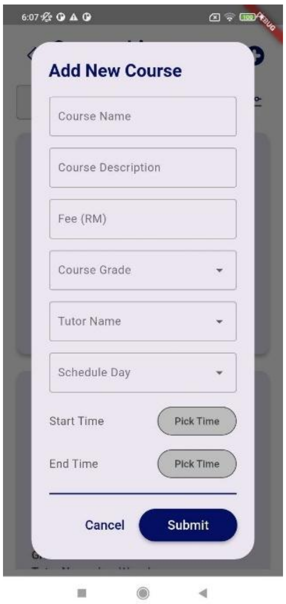
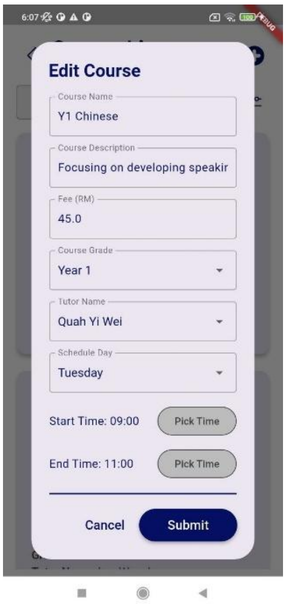


Figure 5.3.8.3 Screenshot of Add New Course Page


Edit Course

Course Name: Y1 Chinese

Course Description: Focusing on developing speakir

Fee (RM): 45.0

Course Grade: Year 1

Tutor Name: Quah Yi Wei

Schedule Day: Tuesday

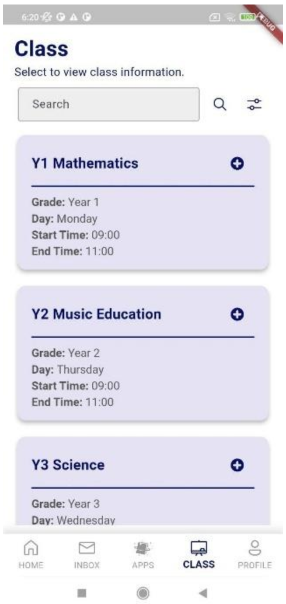
Start Time: 09:00

End Time: 11:00

Figure 5.3.8.4 Screenshot of Edit Existing Course Page

5.3.9 Classes Module

The classes module generates a tutor-focused interface that lists all existing courses that are assigned by administrators in alphabetical order by course name, as illustrated in Figure 5.3.9.1. When tutors click the course, they will be navigated to the respective class details page, as shown in Figure 5.3.9.2. Tutors can view all the class history records and update the student attendance records manually or use the facial recognition attendance tracking by clicking the “camera” icon. Besides, Figure 5.3.9.3 shows tutors are allowed to create new class schedules by clicking the “add” icon of the respective course and entering crucial scheduling data, such as the exact date and class duration.



Class

Select to view class information.

Search

Y1 Mathematics

Grade: Year 1
Day: Monday
Start Time: 09:00
End Time: 11:00

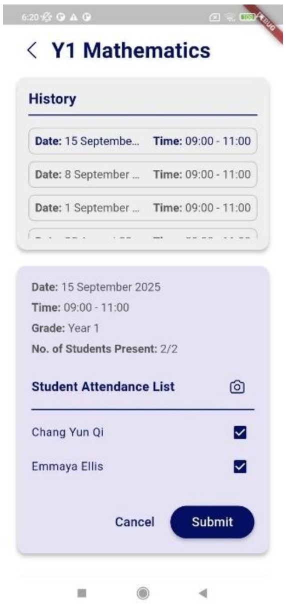
Y2 Music Education

Grade: Year 2
Day: Thursday
Start Time: 09:00
End Time: 11:00

Y3 Science

Grade: Year 3
Day: Wednesday

Figure 5.3.9.1 Screenshot of Classes Page



Y1 Mathematics

History

Date: 15 Septembe... **Time:** 09:00 - 11:00
Date: 8 September ... **Time:** 09:00 - 11:00
Date: 1 September ... **Time:** 09:00 - 11:00

Date: 15 September 2025
Time: 09:00 - 11:00
Grade: Year 1
No. of Students Present: 2/2

Student Attendance List

Chang Yun Qi ☑
Emmaya Ellis ☑

Figure 5.3.9.2 Screenshot of Class Details Page

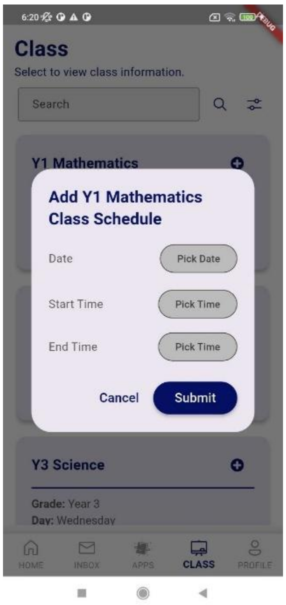
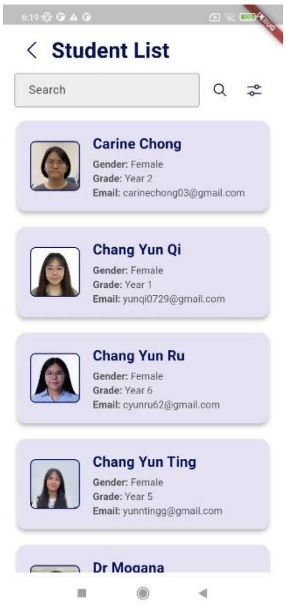


Figure 5.3.9.3 Screenshot of Add New Class Schedule Page

5.3.10 Students Module

The students module displays all the enrolled students alphabetically by student's name to provide a systematically organised directory, as shown in Figure 5.3.10.1. Each student is clickable, and when clicked, administrators will be directed to the detailed student page with extensive personal, parental, and academic information, as depicted in Figure 5.3.10.2. The profile interface displays essential demographic data, such as the student's full name, gender, age, date of birth, father's name, father's email address, mother's name, mother's email address, etc, for effective communication and record-keeping. Additionally, the student profile incorporates academic information by displaying a complete list of registered courses



Student List

Search

Carine Chong
Gender: Female
Grade: Year 2
Email: carinechong03@gmail.com

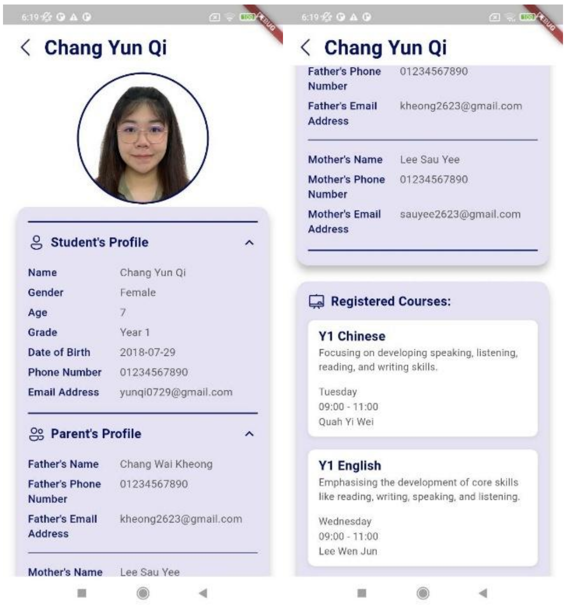
Chang Yun Qi
Gender: Female
Grade: Year 1
Email: yunqi0729@gmail.com

Chang Yun Ru
Gender: Female
Grade: Year 6
Email: cyunru62@gmail.com

Chang Yun Ting
Gender: Female
Grade: Year 5
Email: yunntingg@gmail.com

Dr Mogana

Figure 5.3.10.1 Screenshot of Students Page



Chang Yun Qi

Student's Profile

Name	Chang Yun Qi
Gender	Female
Age	7
Grade	Year 1
Date of Birth	2018-07-29
Phone Number	01234567890
Email Address	yunqi0729@gmail.com

Parent's Profile

Father's Name	Chang Wai Kheong
Father's Phone Number	01234567890
Father's Email Address	kheong2623@gmail.com
Mother's Name	Lee Sau Yee
Mother's Phone Number	01234567890
Mother's Email Address	sauyee2623@gmail.com

Registered Courses:

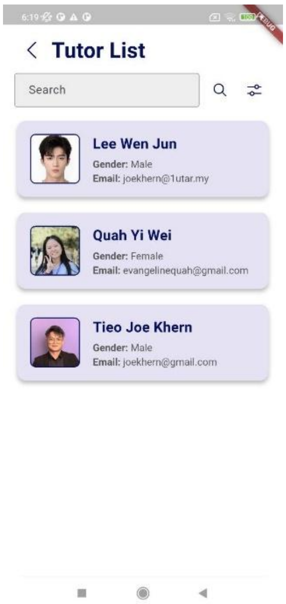
Y1 Chinese
Focusing on developing speaking, listening, reading, and writing skills.
Tuesday
09:00 - 11:00
Quah Yi Wei

Y1 English
Emphasising the development of core skills like reading, writing, speaking, and listening.
Wednesday
09:00 - 11:00
Lee Wen Jun

Figure 5.3.10.2 Screenshot of Student Details Page

5.3.11 Tutors Module

Figure 5.3.11.1 shows the list of all existing tutors, which are arranged alphabetically by name. Administrators are allowed to search for a specific tutor by keywords and filter the tutor list with the sorting options. Each tutor is clickable, and when clicked, administrators will be directed to the detailed tutor page with extensive personal and academic information, as depicted in Figure 5.3.11.2. The profile screen will display key demographic information such as the tutor's full name, gender, age, date of birth, etc, which are required for proper record-keeping and administrative paperwork. Furthermore, the tutor profile also provides a comprehensive list of responsible courses, which allows administrators to have a clear picture of each tutor's teaching obligations and workload allocation.



Tutor List

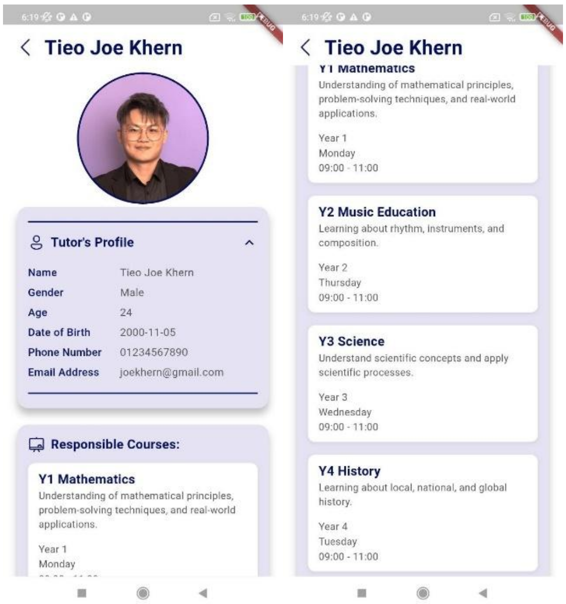
Search

Lee Wen Jun
Gender: Male
Email: joekhern@1utar.my

Quah Yi Wei
Gender: Female
Email: evangelinequah@gmail.com

Tieo Joe Khern
Gender: Male
Email: joekhern@gmail.com

Figure 5.3.11.1 Screenshot of Tutors Page



Tieo Joe Khern

Tutor's Profile

Name	Tieo Joe Khern
Gender	Male
Age	24
Date of Birth	2000-11-05
Phone Number	01234567890
Email Address	joekhern@gmail.com

Responsible Courses:

Y1 Mathematics
Understanding of mathematical principles, problem-solving techniques, and real-world applications.
Year 1
Monday
09:00 - 11:00

Y2 Music Education
Learning about rhythm, instruments, and composition.
Year 2
Thursday
09:00 - 11:00

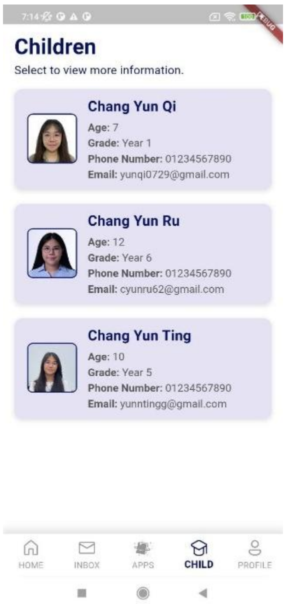
Y3 Science
Understand scientific concepts and apply scientific processes.
Year 3
Wednesday
09:00 - 11:00

Y4 History
Learning about local, national, and global history.
Year 4
Tuesday
09:00 - 11:00

Figure 5.3.11.2 Screenshot of Tutor Details Page

5.3.12 Children Module

Figure 5.3.12.1 shows the parent-centric interface that displays all existing children in alphabetical order by name. Parents are allowed to access their children's registered courses' information. Each child is clickable, and when clicked, parents will be navigated to the children's registered courses page, which will list all registered courses related to the specific child. The page will display the course information, including the course name, description, fee, day, time, and tutor name, as illustrated in Figure 5.3.12.2. Besides, parents are also allowed to search for a specific course by keywords and filter the course list with the sorting options, such as Course Name, Tutor Name, Day, and Fee.



Children

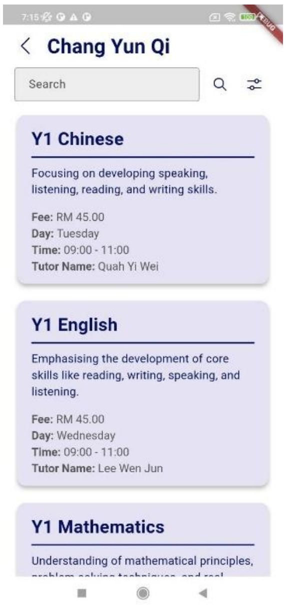
Select to view more information.

Chang Yun Qi
Age: 7
Grade: Year 1
Phone Number: 01234567890
Email: yunqi0729@gmail.com

Chang Yun Ru
Age: 12
Grade: Year 6
Phone Number: 01234567890
Email: cyunru62@gmail.com

Chang Yun Ting
Age: 10
Grade: Year 5
Phone Number: 01234567890
Email: yunntingg@gmail.com

Figure 5.3.12.1 Screenshot of Children Page



Chang Yun Qi

Y1 Chinese

Focusing on developing speaking, listening, reading, and writing skills.

Fee: RM 45.00
Day: Tuesday
Time: 09:00 - 11:00
Tutor Name: Quah Yi Wei

Y1 English

Emphasising the development of core skills like reading, writing, speaking, and listening.

Fee: RM 45.00
Day: Wednesday
Time: 09:00 - 11:00
Tutor Name: Lee Wen Jun

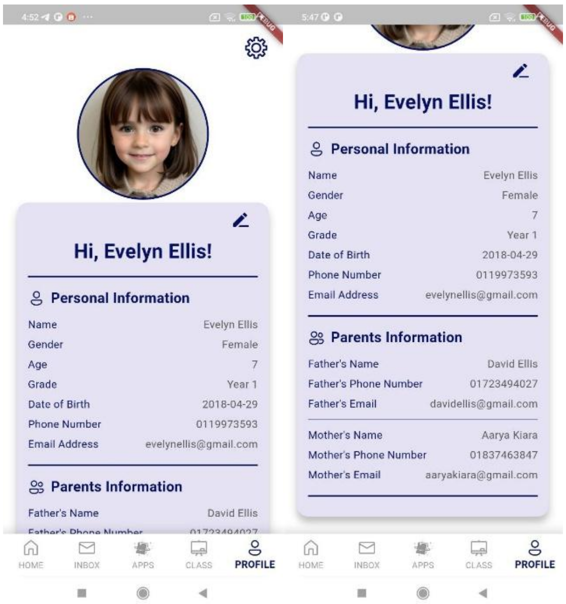
Y1 Mathematics

Understanding of mathematical principles,

Figure 5.3.12.2 Screenshot of Children Registered Courses Page

5.3.13 Profile Module

The profile module enables users to navigate to their profile page via the bottom navigation bar. The profile page prominently displays the profile photo, which is retrieved from Firebase Storage. If no profile photo is stored in Firebase Storage, then a default profile photo will be displayed. Below the profile photo, the personal information, such as name, gender, age, date of birth, phone number, and email address, will be retrieved from Cloud Firestore and displayed as shown in Figures 5.3.13.1 to 5.3.13.4 For the student’s profile page, it will additionally include a section named “Parents' Information” to store their parents’ personal information, such as the name, phone number, and email address, as illustrated in Figure 5.3.13.1.



Hi, Evelyn Ellis!

Personal Information

Name	Evelyn Ellis
Gender	Female
Age	7
Grade	Year 1
Date of Birth	2018-04-29
Phone Number	0119973593
Email Address	evelynellis@gmail.com

Parents Information

Father's Name	David Ellis
Father's Phone Number	01723494027

Hi, Evelyn Ellis!

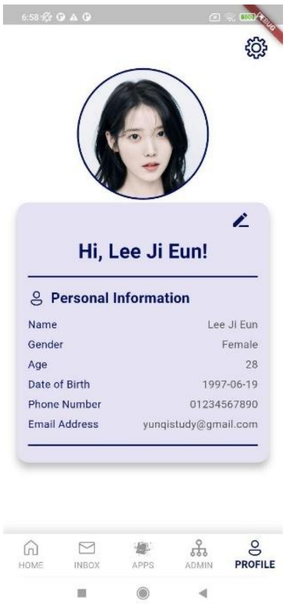
Personal Information

Name	Evelyn Ellis
Gender	Female
Age	7
Grade	Year 1
Date of Birth	2018-04-29
Phone Number	0119973593
Email Address	evelynellis@gmail.com

Parents Information

Father's Name	David Ellis
Father's Phone Number	01723494027
Father's Email	davidellis@gmail.com
Mother's Name	Aarya Kiara
Mother's Phone Number	01837463847
Mother's Email	aaryakiara@gmail.com

Figure 5.3.13.1 Screenshot of Students' Profile Page

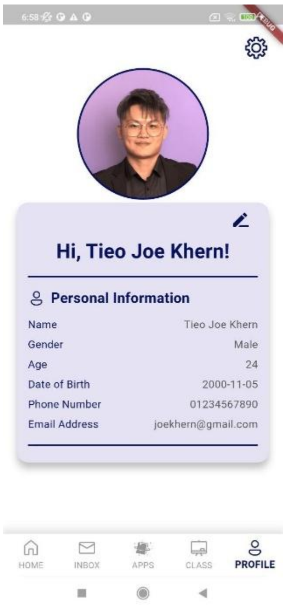


Hi, Lee Ji Eun!

Personal Information

Name	Lee Ji Eun
Gender	Female
Age	28
Date of Birth	1997-06-19
Phone Number	01234567890
Email Address	yunqistudy@gmail.com

Figure 5.3.13.2 Screenshot of Administrators' Profile Page

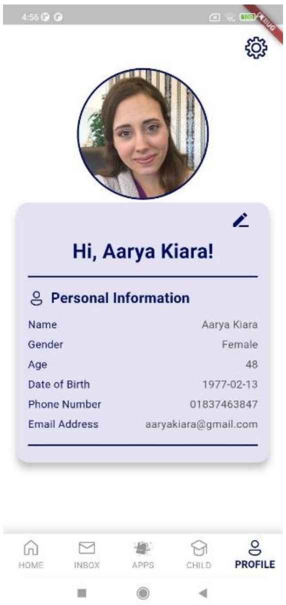


Hi, Tieo Joe Khern!

Personal Information

Name	Tieo Joe Khern
Gender	Male
Age	24
Date of Birth	2000-11-05
Phone Number	01234567890
Email Address	joekhern@gmail.com

Figure 5.3.13.3 Screenshot of Tutors' Profile Page



Hi, Aarya Kiara!

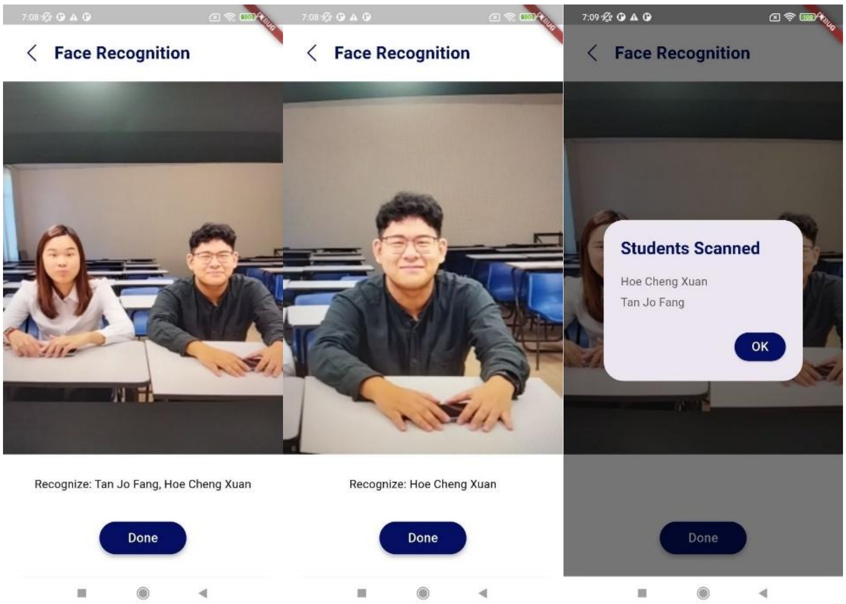
Personal Information

Name	Aarya Kiara
Gender	Female
Age	48
Date of Birth	1977-02-13
Phone Number	01837463847
Email Address	aaryakiara@gmail.com

Figure 5.3.13.4 Screenshot of Parents' Profile Page

5.3.14 Face Recognition Module

The face recognition module allows administrators and tutors to take attendance by scanning students' faces using a system that is powered by Python code and the face_recognition libraries, and it is integrated via a Flask server. When this module is initiated, the Flutter application sends a request to the Flask server, which will then process the camera input, detect faces, and match the faces against the stored images in the Firebase Storage by using the face_recognition library, as depicted in Figure 5.3.14.1. After administrators or tutors click the “Done” button, it will display a dialogue message about the names of the scanned students. In Figure 5.3.14.2, the attendance of the scanned students is updated in the history page.



Face Recognition

Recognize: Tan Jo Fang, Hoe Cheng Xuan

Done

Face Recognition

Recognize: Hoe Cheng Xuan

Done

Face Recognition

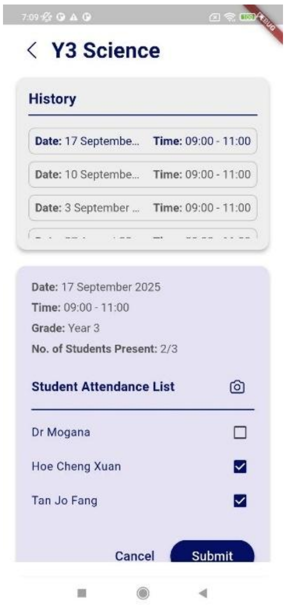
Students Scanned

Hoe Cheng Xuan
Tan Jo Fang

OK

Done

Figure 5.3.14.1 Screenshot of Face Recognition Page



Y3 Science

History

Date: 17 Septembe... **Time:** 09:00 - 11:00

Date: 10 Septembe... **Time:** 09:00 - 11:00

Date: 3 September ... **Time:** 09:00 - 11:00

Date: 17 September 2025
Time: 09:00 - 11:00
Grade: Year 3
No. of Students Present: 2/3

Student Attendance List

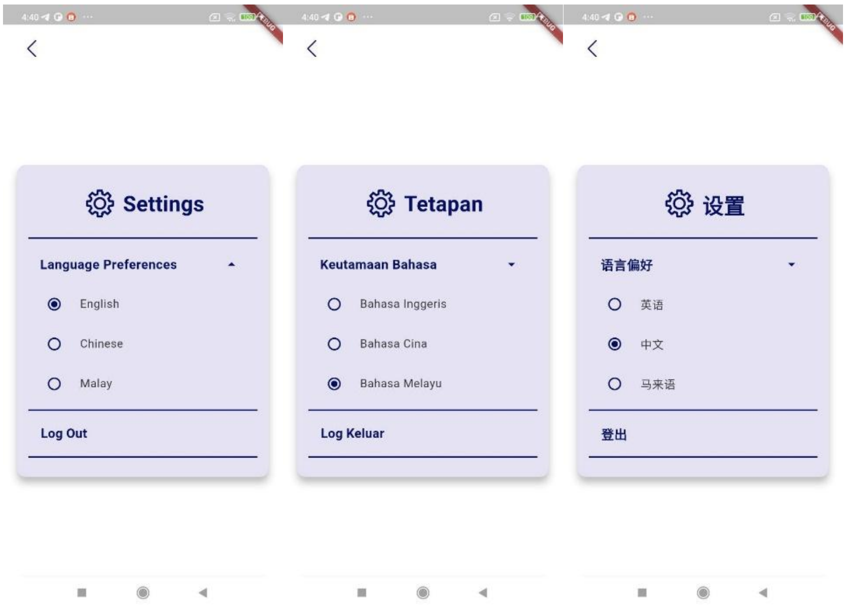
Dr Mogana □
Hoe Cheng Xuan ☑
Tan Jo Fang ☑

Cancel Submit

Figure 5.3.14.2 Screenshot of Course Details Page

5.3.15 Language Preferences Module

In the language module, the user is allowed to change the application's language preferences in the settings of the profile page. Based on Figure 5.3.15.1, the options consist of English, Malay, and Chinese. Once the user shifts the language preferences, the UI of the applications will change to the selected language, as shown in Figure 5.3.15.2.



Settings

Language Preferences

● English

○ Chinese

○ Malay

Log Out

Tetapan

Keutamaan Bahasa

○ Bahasa Inggeris

○ Bahasa Cina

● Bahasa Melayu

Log Keluar

设置

语言偏好

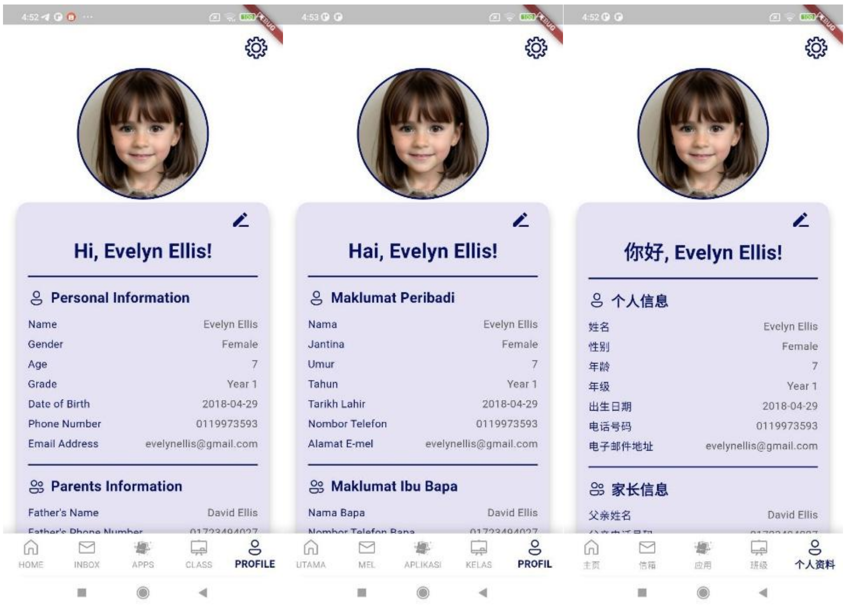
○ 英语

● 中文

○ 马来语

登出

Figure 5.3.15.1 Screenshot of Settings Page



Hi, Evelyn Ellis!

Personal Information

Name	Evelyn Ellis
Gender	Female
Age	7
Grade	Year 1
Date of Birth	2018-04-29
Phone Number	0119973593
Email Address	evelynellis@gmail.com

Parents Information

Father's Name	David Ellis
Father's Phone Number	01723494027

Hai, Evelyn Ellis!

Maklumat Peribadi

Nama	Evelyn Ellis
Jantina	Female
Umur	7
Tahun	Year 1
Tarikh Lahir	2018-04-29
Nombor Telefon	0119973593
Alamat E-mel	evelynellis@gmail.com

Maklumat Ibu Bapa

Nama Bapa	David Ellis
Nombor Telefon Bapa	01723494027

你好, Evelyn Ellis!

个人信息

姓名	Evelyn Ellis
性别	Female
年龄	7
年级	Year 1
出生日期	2018-04-29
电话号码	0119973593
电子邮件地址	evelynellis@gmail.com

家长信息

父亲姓名	David Ellis

Figure 5.3.15.2 Screenshot of the UI of the application

5.4 Implementing Issues and Challenges

The development and deployment of the “Smart Management System for Tuition Centre Operations” encountered several issues and challenges, such as the profile photo download performance and automated email notification implementation.

i. **Profile Photo Download Performance Issue in Face Recognition Module**

The face recognition module posed a big performance barrier in this project, with initialising the face recognition function requiring a lengthy loading time of around 13 to 15 seconds to retrieve user profile photos from Firebase Storage. The system was designed to get students’ profile photos from Firebase Storage for biometric attendance tracking. However, lengthy download times caused user experience concerns and operational inefficiencies. Initial attempts to address this performance bottleneck included converting profile photos to grayscale format to reduce file size. This attempt successfully reduced loading times but compromised face recognition accuracy due to the loss of color information that aids in facial feature recognition. After extensive optimisation efforts, the issue was successfully resolved by implementing image compression techniques that reduced profile photos to a standardized 128x128 pixel size. This compression method maintained adequate image quality for accurate face recognition while reducing the download time to around 5 to 7 seconds. This approach efficiently balanced system performance and identification accuracy.

ii. **Automated Email Notification Implementation Issue in Firebase Functions**

A significant challenge emerged during the implementation of the automated email notification feature using Node.js and Firebase Functions. The system repeatedly failed to send emails, such as new message notifications, new billing notifications, and new announcement notifications, to the target recipients. The initial attempts were to trigger email notifications, which resulted in function failures and connectivity issues, preventing successful email delivery to target recipients. The implementation obstacles included configuration errors in the functions/index.js email service integration and deployment concerns with Firebase Functions. However, after extensive troubleshooting and debugging efforts, including evaluating Firebase Functions logs and updating Node.js dependencies, the automated email notification module was successfully fixed and ensured consistent email delivery to target recipients.

5.5 Summary

In conclusion, this chapter outlines the system requirements, including specific hardware requirements, such as an ASUS TUF Gaming laptop and a POCO M3 Android device, and software requirements, including Visual Studio Code, Firebase database, Node.js, Dart, JavaScript, Python, and Flutter. Besides, it also covers the settings and configuration methods, system operations with real screenshots, and issues and challenges when implementing this project. The issues and challenges include the profile photo download performance issue in the face recognition module and the automated email notification implementation issue in Firebase Functions.

Chapter 6
System Evaluation and Discussion

This chapter highlights a complete evaluation of the developed system in four different sections, which consist of system testing that outlines the testing strategies, testing setup and results that present the technical configuration and actual outcomes of various test scenarios, objective evaluation that analyses the system's performance, and summary. This chapter acts as a critical assessment step, which is used to validate the system's functionality, dependability, and effectiveness in satisfying the project objectives.

6.1 System Testing

6.1.1 Formal Testing

The formal testing process employed a three-tiered approach, including unit testing, integration testing, and system testing, to ensure the reliability and functionality of the tuition centre management system across all developed modules.

Unit testing concentrated on individual module parts. It is investigating isolated functionalities, such as user authentication mechanisms in the sign-in and sign-up module, message delivery features within the inbox module, new announcement creation processes in the announcement module, calendar scheduling functions, payment processing features in the payment module, data visualisation algorithm in the report and analytics module, and courses and classes management features. Each unit test evaluated specific functions, methods, and components to ensure they worked properly under a variety of input conditions.

Moreover, integration testing analysed the connection and data flow across multiple modules. It helps to ensure smooth communication and functionality across different module borders. Critical integration scenarios in this project included evaluating the connectivity between the inbox module and automated email notification systems, validating data synchronisation between the calendar module and classes module, ensuring proper data flow between the payment module and student billing records, and verifying the integration of the face recognition attendance tracking system and class attendance records management. The integration testing also validates Firebase database functionality to ensure consistent data retrieval and storage across all modules.

Furthermore, system testing offered end-to-end validation of the entire application's functionality by mimicking real-world usage scenarios across all user roles and devices. This testing phase covered the entire user journey from sign-in authentication to module navigation, task completion, and data persistence verification. System tests included comprehensive role-based access validation to ensure all the roles could access specific functionalities. For example, administrators could access all functionalities, while tutors, students, and parents can only access specific functionalities to maintain security.

6.1.2 Informal Testing

Informal testing involved spontaneous and unstructured exploration of the system's capabilities. It relies on the tester's intuition and expertise to identify the flaws and usability concerns that would have been missed in the formal testing techniques. Informal testing allowed random browsing across multiple modules without prepared test scripts and allowed testers to engage with the system naturally and discover unexpected user behaviors or interface inconsistencies.

For example, during informal testing sessions, testers investigated the inbox module by quickly switching between private and group chats, testing message sending under various circumstances, and attempting unusual user interactions like sending multiple rapid messages or uploading oversized images to observe system behavior. Besides, testers also experimented with the calendar module by trying out different date navigation patterns, testing the horizontal scrolling feature at different usage speeds, etc. The face recognition module conducted extensive informal testing, including random face recognition attempts under various lighting conditions, facial angles, and user positioning scenarios. The testers performed informal evaluations of the courses, classes, students, tutors, and children modules using intuitive exploration and real-world usage simulations.

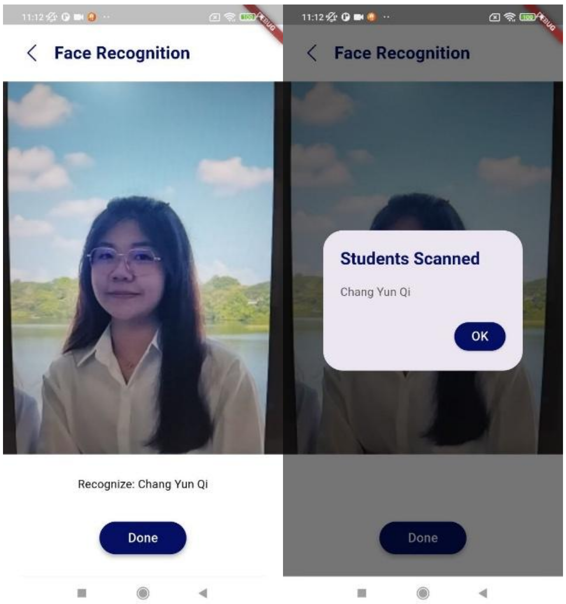
In conclusion, this unstructured informal testing approach helps to reveal subtle flaws, such as minor data discrepancies, UI alignment issues, or irregular connectivity issues with Firebase services. These informal testing insights proved invaluable in identifying real-world flaws and usability issues. Hence, it results in a more robust and user-friendly tuition centre management system.

6.2 Testing Setup and Result

The testing setup focused specifically on evaluating biometric facial attendance tracking functionality under diverse environmental and technical conditions to ensure reliable and consistent performance across real-world settings. Testing was conducted across multiple circumstances, including varying lighting conditions, multiple faces in one frame, multiple facial angles, without wearing glasses, and different hairstyles.

6.2.1 Varying lighting conditions

Figure 6.2.1.1 demonstrates successful facial recognition performance under suboptimal lighting environments, specifically in dimmer natural lighting environments. The system accurately identified the individual and displayed the confirmation text “Recognize: Chang Yun Qi” indicating effective biometric processing. However, Figure 6.2.1.2 illustrates recognition failure attributed to insufficient facial illumination, resulting in the system’s inability to match the captured image with stored profile data and consequently displaying the text “Recognize: Unknown”.

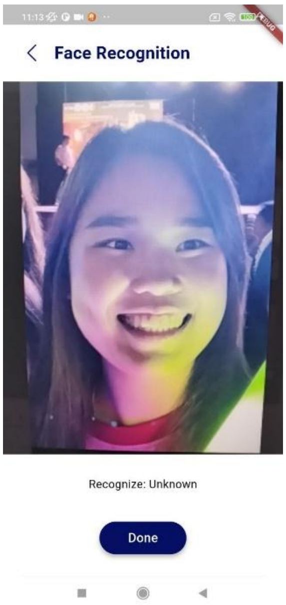


Recognize: Chang Yun Qi

Students Scanned

Chang Yun Qi

Figure 6.2.1.1 Screenshot of Case 1 in 6.2.1

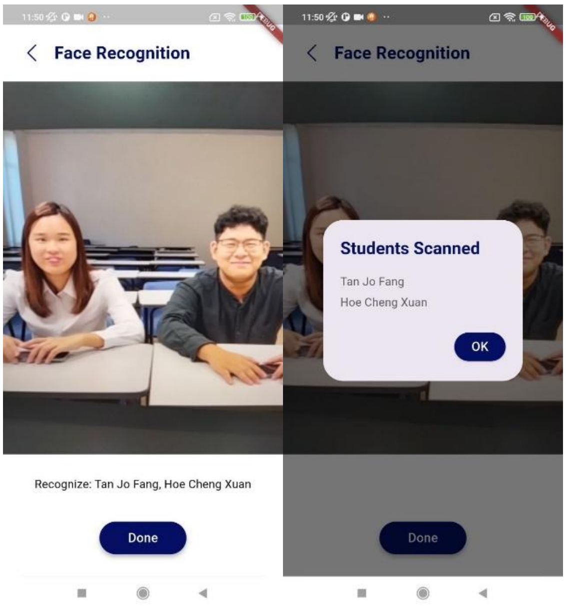


Recognize: Unknown

Figure 6.2.1.2 Screenshot of Case 2 in 6.2.1

6.2.2 Multiple faces in one frame

Figure 6.2.2.1 shows successful multi-face recognition capabilities within a single frame. The face recognition system accurately identifies both individuals and displays the confirmation text "Recognize: Tan Jo Fang, Hoe Cheng Xuan", which demonstrates the system's ability to process multiple biometric profiles concurrently. Besides, Figure 6.2.2.2 illustrates comparable multi-face recognition performance, accurately detecting both people and displaying "Recognize: Carine Chong, Toh Yun Shuang".



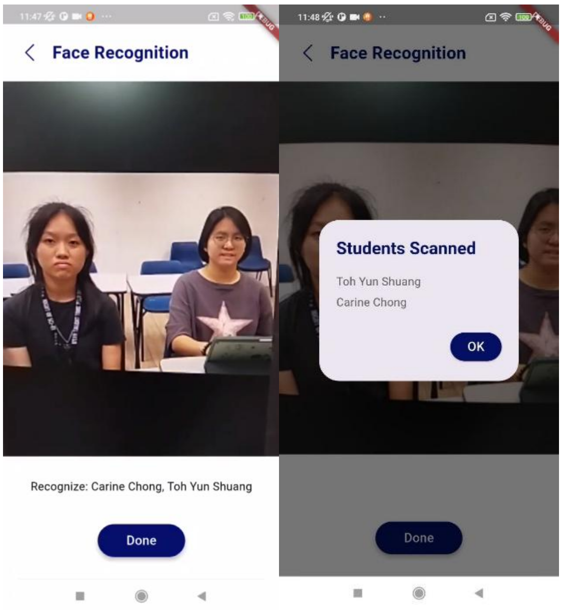
Face Recognition

Recognize: Tan Jo Fang, Hoe Cheng Xuan

Students Scanned

Tan Jo Fang
Hoe Cheng Xuan

Figure 6.2.2.1 Screenshot of Case 1 in 6.2.2



Face Recognition

Recognize: Carine Chong, Toh Yun Shuang

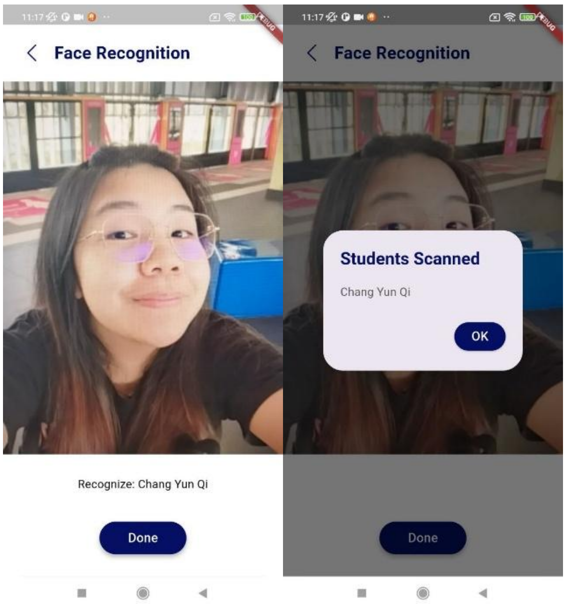
Students Scanned

Toh Yun Shuang
Carine Chong

Figure 6.2.2.2 Screenshot of Case 2 in 6.2.2

6.2.3 Multiple facial angles

Figure 6.2.3.1 depicts effective facial recognition performance despite raised camera positioning. The system correctly detects the individual from an above angle and displays “Recognize: Chang Yun Qi”. This demonstrates the system’s tolerance for a variety of camera placement configurations. In contrast, Figure 6.2.3.2 indicates recognition failure caused by incomplete facial data capture, where only partial facial features were visible within the frame boundaries, resulting in insufficient biometric information for accurate identification.

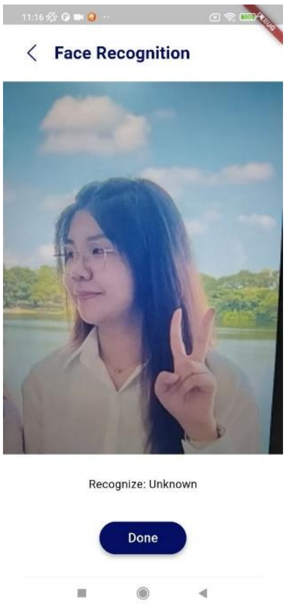


Recognize: Chang Yun Qi

Students Scanned

Chang Yun Qi

Figure 6.2.3.1 Screenshot of Case 1 in 6.2.3

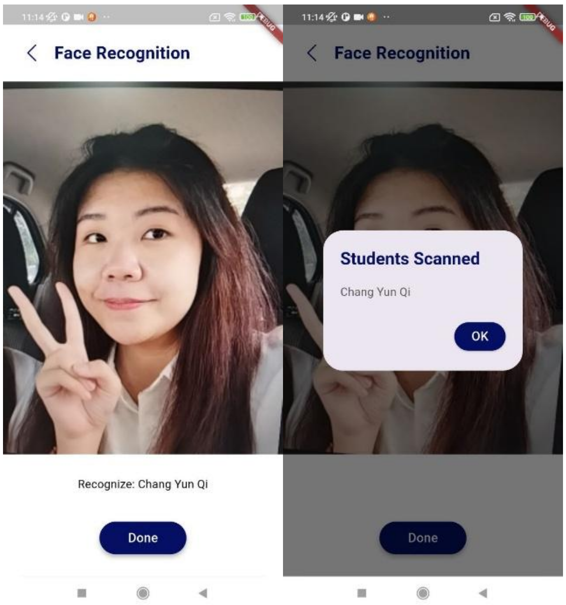


Recognize: Unknown

Figure 6.2.3.2 Screenshot of Case 2 in 6.2.3

6.2.4 Without wearing glasses

Figures 6.2.4.1 and 6.2.4.2 illustrate consistent facial recognition accuracy under standard conditions and without facial accessories. The system successfully identifies the individual in both cases and displays “Recognize: Chang Yun Qi”, which establishes baseline recognition performance parameters for unobstructed facial features.

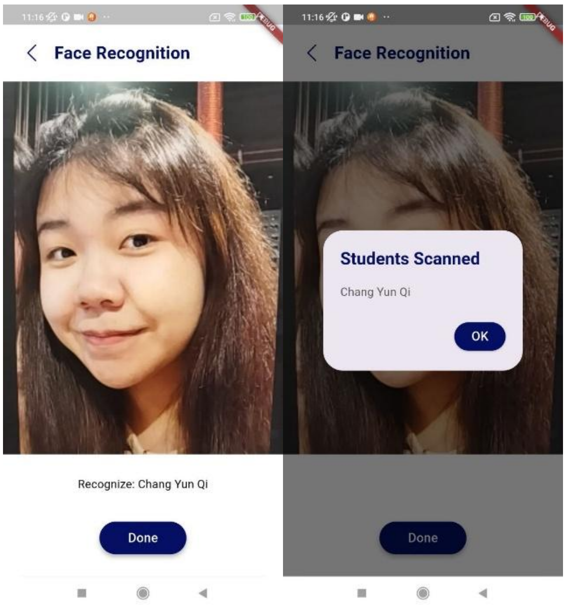


Recognize: Chang Yun Qi

Students Scanned

Chang Yun Qi

Figure 6.2.4.1 Screenshot of Case 1 in 6.2.4



Recognize: Chang Yun Qi

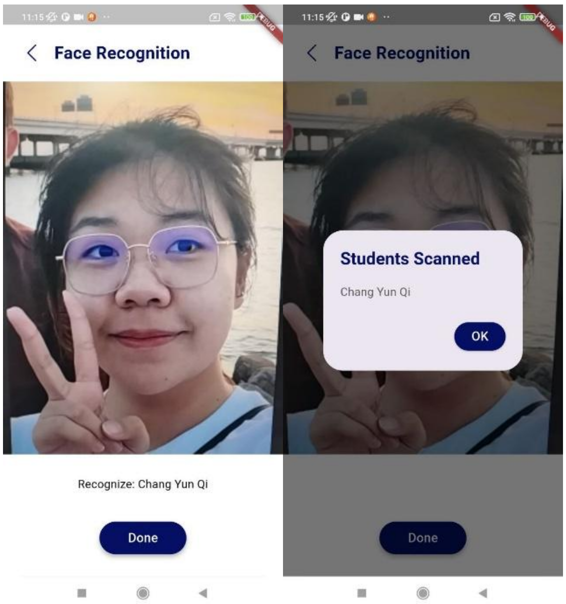
Students Scanned

Chang Yun Qi

Figure 6.2.4.2 Screenshot of Case 2 in 6.2.4

6.2.5 Different hairstyles

Figures 6.2.5.1 and 6.2.5.2 depict the system's robustness against appearance variations, specifically hairstyle changes in which the individual's long hair was tied up. Both test scenarios successfully identified the individual and displayed "Recognize: Chang Yun Qi". This shows the system's ability to maintain recognition accuracy despite changes in hair presentation that do not obscure primary facial features.

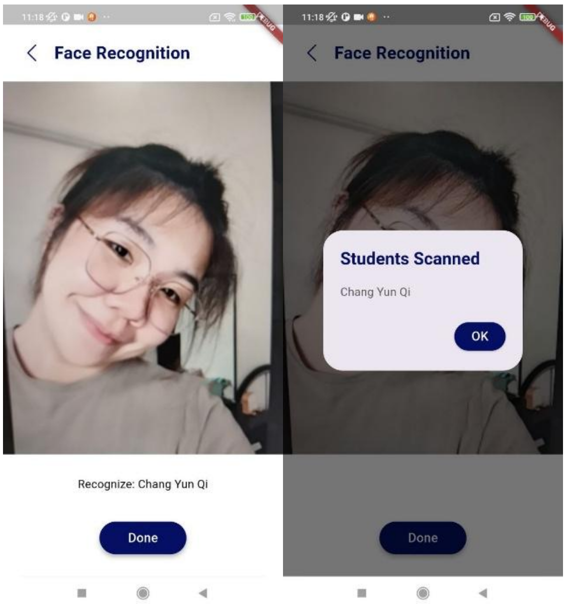


Students Scanned

Chang Yun Qi

Recognize: Chang Yun Qi

Figure 6.2.5.1 Screenshot of Case 1 in 6.2.5



Students Scanned

Chang Yun Qi

Recognize: Chang Yun Qi

Figure 6.2.5.2 Screenshot of Case 2 in 6.2.5

6.3 Objective Evaluation

The primary objective of this project is to develop a user-friendly smart management system for tuition centre operations, which has been successfully achieved by effectively digitalising and automating traditional paper-based tuition centre operations. The implementation of multiple interconnected modules, such as the courses module, classes module, students module, tutors module, and children module, indicates a complete shift from manual administrative operations to automated digital workflows. The system's alphabetical data and information arrangement, straightforward clickable interfaces, and role-based access controls for different roles support the user-friendly design concept.

Besides, the biometric attendance tracking objective has been significantly accomplished through the successful implementation of the face recognition module. The system successfully reduces the issues of friend impersonation and manual attendance by utilising automated facial recognition technology. The reduction of Firebase Storage download latency from 13-15 seconds to 5-7 seconds via image compression indicates the system's ability to provide immediate and real-time attendance monitoring without sacrificing recognition accuracy. The integration of the face recognition module allows tutors and administrators to update class attendance records smoothly and eliminates the time-consuming procedure of manually calling names.

The goal of communication and transparency between parents and tutors has been fully achieved through the integrated deployment of numerous communication-focused modules that ensure seamless information flow. The announcement module allows administrators to publish real-time updates with multimedia content and ensure that parents are informed about tuition centre activities and vital announcements. Besides, the advanced inbox module also allows parents and tutors to communicate directly through a private chat box, and it also integrates with intelligent email notification systems to ensure timely message delivery. The children module significantly improves transparency by providing parents with comprehensive visibility into their children's course schedules and academic activities. By integrating various communication-focused modules, it creates a robust communication ecosystem that bridges traditional communication gaps.

6.4 Summary

In conclusion, Chapter 6 conducts a complete review of the tuition centre management system using systematic testing procedures and objective evaluation of project outcomes. The system testing used both formal testing methods that encompass structured unit testing, integration testing, and comprehensive system testing, and informal testing methods by allowing for spontaneous exploration based on the tester's experience. Besides, the testing setup and result section focused extensively on face recognition module evaluation under diverse environmental conditions. Furthermore, the objective evaluation verified the successful achievement of all three primary project goals.

Chapter 7
Conclusion and Recommendation

7.1 Conclusion

The “Smart Management System for Tuition Centre Operations” plays a crucial role in transforming small-sized tuition centres’ operational and administrative effectiveness. By addressing the inefficiencies of manual processes, such as time-consuming paperwork and inaccurate attendance tracking, the system provides a digital solution to enhance productivity and reduce human errors. Besides, it also fosters better engagement between administrators, tutors, parents, and students by ensuring seamless communication and transparency.

This project aims to deliver a comprehensive, user-friendly, and cost-effective solution to the small-sized tuition centres by utilising the Rapid Application Development (RAD) methodology. RAD methodology ensures rapid prototyping, iterative feedback, and continuous refinement. The system integrates several key modules, including user authentication, student management, course management, class management, fee management, communication, and reporting and analysis. These modules improve the user experience and streamline operational tasks.

In this report, all three key project objectives were met successfully through extensive system development and implementation. Through systematic testing and evaluation, the system demonstrated reliable performance across multiple modules, including user authentication, course management, student administration, payment processing, analytics reporting, and automated notifications. As an outcome, this project builds a solid technological foundation for transforming traditional tuition centre operations into an efficient, automated, and transparent tuition centre management system.

7.2 Recommendation

To improve the overall performance and productivity of the system, various significant recommendations are suggested. The face recognition module offers numerous potential for improvement, which would significantly improve the system’s biometric attendance tracking capabilities and dependability. Enhancing recognition accuracy in low-light conditions is an essential improvement that could be accomplished by implementing adaptive brightness algorithms or enhanced image preprocessing techniques that automatically adjust for the various illumination scenarios. The integration of anti-spoofing detection mechanism is critical to ensure attendance integrity by preventing fraudulent attempts at fooling the biometric system using photographs, videos, or digital displays. This could be accomplished through liveness detection algorithms that analyse facial movement or depth sensing.

Furthermore, the second recommendation is to improve user experience with a focus on interface design and user guidance. This would significantly increase system adoption and operational effectiveness among all users. For example, implementing a dark mode theme option would provide users with visual comfort alternatives that reduce eye strain during extended periods of use. The addition of extensive tutorial and onboarding screens is also a critical feature that will assist smoother system adoption by offering step-by-step instructions for new users to navigate the various modules and functionalities. This can help in reducing the learning curve and minimising user confusion and frustration.

REFERENCES

[1] Ayra, “Tuition Centre Management System Malaysia 2024,” Apr. 19, 2023. https://yuran.my/tuition-centre-management-system-malaysia/

[2] “Tuition Management System - A Complete Guide & Best Provider,” *smartclasses.in*, Mar. 16, 2022. https://smartclasses.in/Tuition-Management-System-Complete-Guide

[3] A. Kumar, S. Samal, M. S. Saluja, and A. Tiwari, “Automated Attendance System Based on Face Recognition Using Opencv | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*, May 05, 2023. https://ieeexplore.ieee.org/document/10112665

[4] “MCPLUS - The Biggest Online Tuition in Malaysia,” *MCPlus*, Sep. 19, 2021. https://mcplus.my/

[5] iKEY TECHNOLOGY Sdn Bhd, “iKEY Technology Sdn Bhd - Borderless Learning,” *Ikey.my*, 2024. https://ikey.my/

[6] “Private Tuition is one stop solution for Maths Tuition ,GCSE Tuition, 11 plus exams,Tuition Centres,” *Primetuition.co.uk*, 2024. https://primetuition.co.uk/

[7] A. Altvater, “What is SDLC? Understand the Software Development Life Cycle,” *Stackify*, 2024. https://stackify.com/what-is-sdlc/

[8] Kissflow, “Rapid Application Development (RAD) | Definition, Steps & Full Guide,” *kissflow.com*, Oct. 31, 2022. https://kissflow.com/application-development/rad/rapid-application-development/

[9] “Smart Answers,” *InfoWorld*, 2024. https://www.infoworld.com/smart-answers/?q=What%20is%20Visual%20Studio%20Code%20used%20for%3F&qs=article_infoworld_2335960 (accessed Apr. 27, 2025).

[10] “What is Firebase?,” *HowDev*, 2025. https://how.dev/answers/what-is-firebase

[11] W3Schools, “Node.js Introduction,” *W3schools.com*, 2019. https://www.w3schools.com/nodejs/nodejs_intro.asp

[12] A. Badkar, “What is Dart Programming - A Paradigm Shift in Coding,” *Simplilearn.com*, Jul. 13, 2023. https://www.simplilearn.com/what-is-dart-programming-article

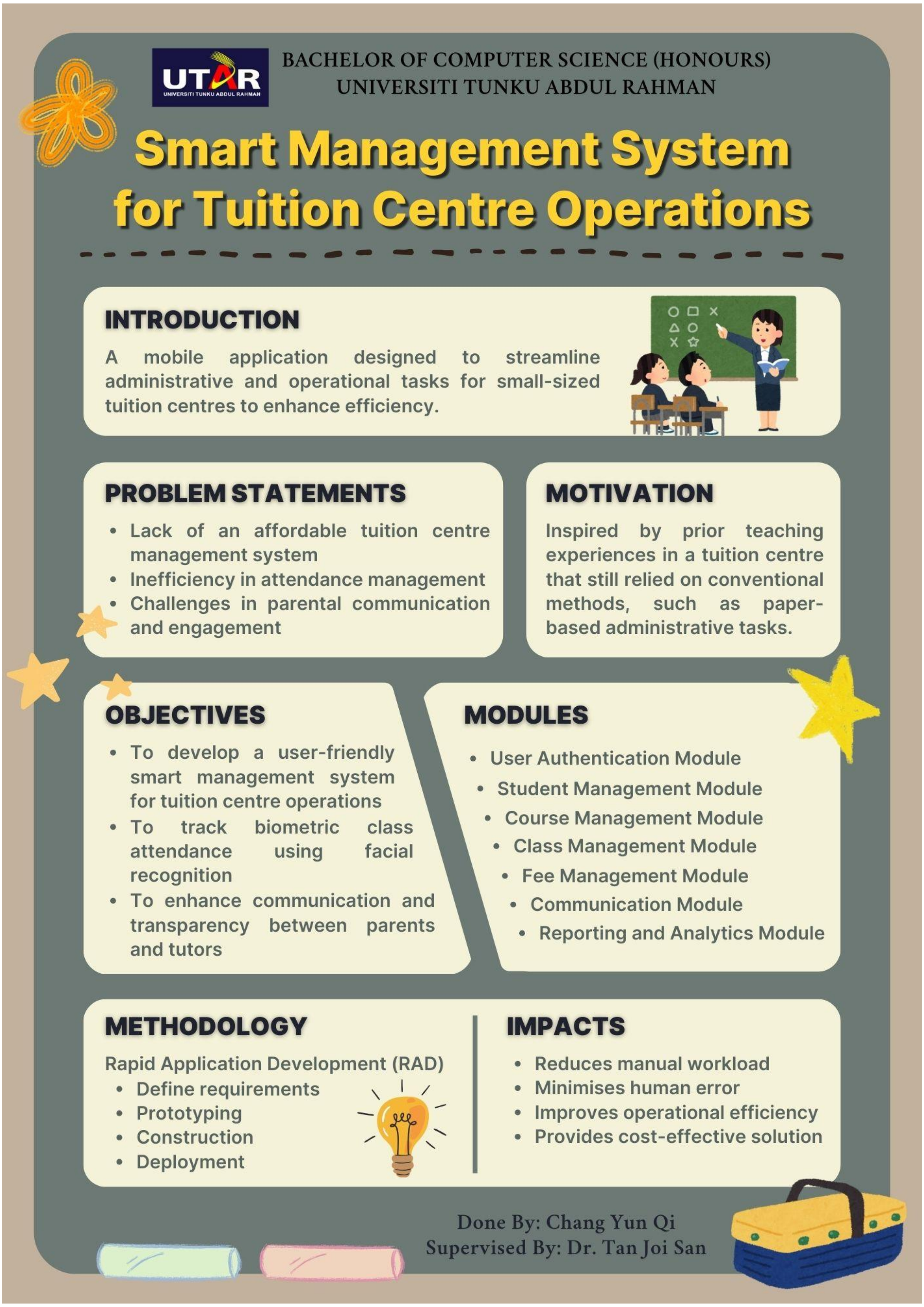
[13] “What is Python Programming Language? | Teradata,” *www.teradata.com*, Dec. 05, 2023. https://www.teradata.com/insights/data-platform/what-is-python-programming-language

[14] C. Staff, “What Is JavaScript Used For?,” *Coursera*, 2024. https://www.coursera.org/articles/what-is-javascript-used-for

[15] G. Andrades, “What Is Flutter Framework? A Guide to Flutter App Testing,” *ACCELQ Inc*, Jun. 03, 2022. https://www.accelq.com/blog/flutter-framework/

APPENDIX A

A.1 Poster



BACHELOR OF COMPUTER SCIENCE (HONOURS)
UNIVERSITI TUNKU ABDUL RAHMAN

Smart Management System for Tuition Centre Operations

INTRODUCTION

A mobile application designed to streamline administrative and operational tasks for small-sized tuition centres to enhance efficiency.

PROBLEM STATEMENTS

- Lack of an affordable tuition centre management system
- Inefficiency in attendance management
- Challenges in parental communication and engagement

MOTIVATION

Inspired by prior teaching experiences in a tuition centre that still relied on conventional methods, such as paper-based administrative tasks.

OBJECTIVES

- To develop a user-friendly smart management system for tuition centre operations
- To track biometric class attendance using facial recognition
- To enhance communication and transparency between parents and tutors

MODULES

- User Authentication Module
- Student Management Module
- Course Management Module
- Class Management Module
- Fee Management Module
- Communication Module
- Reporting and Analytics Module

METHODOLOGY

Rapid Application Development (RAD)

- Define requirements
- Prototyping
- Construction
- Deployment

IMPACTS

- Reduces manual workload
- Minimises human error
- Improves operational efficiency
- Provides cost-effective solution

Done By: Chang Yun Qi
Supervised By: Dr. Tan Joi San