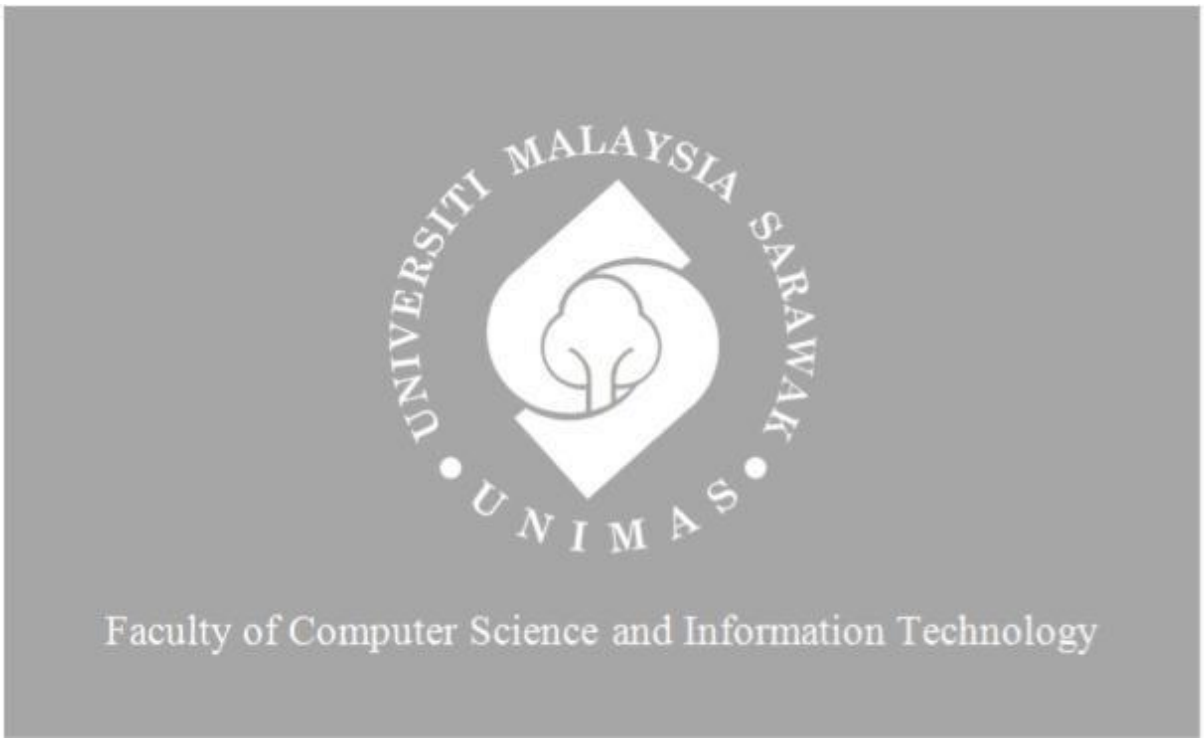


BUS TRACKER MOBILE APPLICATION AT UNIMAS

Nisa Nur Alia Binti Sukri

Bachelor of Software Engineering with Honours

(Software Engineering)

Bus Tracker Mobile Application at UNIMAS

Nisa Nur Alia Binti Sukri

78320

This project is submitted in partial fulfilment of

requirements for the degree of

Bachelor of Software Engineering with Honours

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2025

Aplikasi Penjejak Bas di UNIMAS

Nisa Nur Alia Binti Sukri

78320

Projek ini merupakan salah satu keperluan untuk

Ijazah Sarjana Muda Kejuruteraan Perisian dengan Kepujian

Fakulti Sains Komputer dan Teknologi Maklumat

UNIVERSITI MALAYSIA SARAWAK

2025

UNIVERSITI MALAYSIA SARAWAK

THESIS STATUS ENDORSEMENT FORM

TITLE Bus Tracker Mobile Application at UNIMAS

ACADEMIC SESSION: 2024/2025

NISA NUR ALIA BINTI SUKRI

(CAPITAL LETTERS)

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order to develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [or for the purpose of interlibrary loan between HLI]
5. ** Please tick (√)

☐	CONFIDENTIAL	(Contains classified information bounded by the OFFICIAL SECRETS ACT 1972)
☐	RESTRICTED	(Contains restricted information as dictated by the body or organization where the research was conducted)
√	UNRESTRICTED	

(AUTHOR'S SIGNATURE)

Permanent Address

98150 Bekenu, Sarawak

Date: 17/1/2025

Validated by

(SUPERVISOR'S SIGNATURE)

Date:

Note * Thesis refers to PhD, Master, and Bachelor Degree

** For Confidential or Restricted materials, please attach relevant documents from relevant organizations / authorities

DECLARATION

I hereby declare that the project is my original work. I have not copied from any other student's work or from any other sources except due to reference or acknowledgment is not made explicitly in the text, nor has any part had been written for me by another person.

(NISA NUR ALIA BINTI SUKRI, 78320)

ACKNOWLEDGMENT

First and foremost, I would like to express my deepest gratitude to Allah for granting me the strength, patience, and perseverance to successfully complete my Final Year Project 1. I am also thankful to my supervisor Assoc. Prof. Dr. Kartinah, for her invaluable guidance, feedback, and continuous support throughout the development of this project. I am deeply grateful to my parents and siblings for their support, understanding, and encouragement throughout this journey, which motivated me to strive for excellence. A special and heartfelt thanks to my mom, whose unwavering support during my hard times, constant encouragement, always listen to my problems, helped me stay motivated and focused. I also want to extend my gratitude to my friends and colleagues, especially Hafizah, Azizah, Ainul, Amira, June, Adriana, Afiqah, and Zalina, for their encouragement, emotional support, collaboration, and valuable input that contributed significantly to my project. Lastly, I would like to thank all individuals who participated in the user surveys for this project and helped to share my survey, as their responses were crucial to the success of my project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

UNIMAS students rely on bus services as one of their main modes of transportation to attend classes. However, the students face challenges related to bus schedules, particularly mismatched timings between the provided schedule and actual bus arrivals. The Bus Tracker Mobile Application aims to address these issues by providing real-time bus locations along with estimated time of arrival (ETA) information for students. Additionally, the app offers features such as bus schedule updates, bus status updates, and integration with students' class schedules. This project seeks to improve students' time management, reduce waiting times, and enhance their overall commuting experience. The application is designed based on insights gathered from student surveys and research on existing systems to ensure it effectively meets user needs. The collected data ensures that the proposed app provides impactful solutions to address the current bus-related issues at UNIMAS.

ABSTRAK

Pelajar UNIMAS bergantung kepada perkhidmatan bas sebagai salah satu cara utama untuk ke kelas. Namun begitu, pelajar menghadapi cabaran berkaitan jadual bas, terutamanya ketidakpadanan waktu antara jadual yang disediakan dengan masa sebenar ketibaan bas. Aplikasi Mudah Alih Penjejak Bas ini bertujuan untuk menangani isu-isu tersebut dengan menyediakan lokasi bas secara masa nyata bersama anggaran masa ketibaan (ETA) untuk pelajar. Selain itu, aplikasi ini menawarkan ciri-ciri seperti jadual bas terkini, status bas terkini, dan integrasi dengan jadual kelas pelajar. Projek ini bertujuan untuk membantu meningkatkan pengurusan masa pelajar, mengurangkan masa menunggu, dan memperbaiki pengalaman perjalanan mereka. Aplikasi ini direka berdasarkan data yang diperoleh daripada tinjauan pelajar dan penyelidikan terhadap sistem sedia ada bagi memastikan ia memenuhi keperluan pengguna dengan berkesan. Data yang dikumpulkan dapat memastikan bahawa aplikasi yang dicadangkan ini menyediakan penyelesaian yang berkesan untuk menangani isu-isu berkaitan bas di UNIMAS.

CHAPTER 1: INTRODUCTION

1.1 Introduction

For several decades, bus tracking systems have been essential for improving the reliability of public transport. The bus tracking system works by determining the real-time position of the bus on its route (Public Transport Victoria, 2024). Typically, the bus tracking system uses Global Positioning System (GPS) technology as a logistical database to determine the real-time location of the bus (Mara Labs Inc, 2024). Then, the data collected by the system is linked to several devices, providing information about the bus's location and Estimated Time of Arrival (ETA) to customers (Public Transport Victoria, 2024). GPS technology has become important in transforming how we navigate and track locations. For instance, transportation industry in USA, undergo significant transformation with real time insights, cost reduction, data-driven decision making and increased productivity, due to the use of GPS tracking software (Lowry Solutions, 2024).

Campus bus and external bus services have become the primary transportation option for university students due to their affordability. These buses are critical to student daily life, providing essential transport to classes, hostel and other university facilities. For example, at Universiti Malaysia Sarawak (UNIMAS), the campus bus is called the UNIMAS Shuttle Bus, which is free for every ride. The external bus service at UNIMAS is called the PanBorneo Bus, which costs RM1 per ride. This price is cheaper than using e-hailing services, which can become costly, especially during peak hours.

Even though bus services are very convenient for university students, frequent issues arise because of bus arrival times. Many factors contribute to this problem, such as traffic, weather conditions, bus speed, and driver time management. Some of these factors are beyond control, making it difficult for students to predict when a bus will arrive, especially when the bus schedule does not match the actual arrival times.

This project aims to provide a mobile application for bus tracking that displays real-time locations and anticipated arrival times based on the GPS position of the bus driver's device. Using the driver's mobile device location as GPS can provide real-time data, allowing the application to accurately track and update the bus's location in the app. The bus driver will also be a user of this mobile app, enabling them to share updates about bus availability and any changes to bus information. Additionally, the app will include a feature allowing students to

input their class schedules, enabling the system to provide tailored notifications about bus status and bus availability.

1.2 Problem Statement

University students living outside the campus rely on buses for transportation to the university. However, they often face challenges in tracking both the UNIMAS Shuttle Bus and the PanBorneo Bus. The buses frequently do not adhere to their schedules, causing students to either wait longer or miss their buses altogether, which can lead to missed classes or students arriving late (Lonescu, 2021). Sometimes the other factor of irregular bus schedule when one bus does not arrive on time delays subsequent bus trips. This is particularly problematic for students with late-night classes that finish at 10 PM, some students are unaware of the updated bus schedules and assume that service has ended, forcing them to use e-hailing services, which defeats the purpose of seeking more economical transportation.

It is essential to build a real-time bus tracking to improve student and driver time management. The availability of effective and affordable transportation greatly influences students' ability to stay enrolled and successfully graduate (Lonescu, 2021). Even though there are many real-time bus tracking systems already exist, there are limited research that focuses on bus tracking in university that specifically incorporate features like student class schedule integration including the ability to recommend the best bus based on the nearest bus stop to their class location. Furthermore, the existing bus tracking focus on providing basic real-time location tracking and estimated time of arrival information but it unable to notify the students who struggle to manage their time and are often unaware of the bus schedules.

1.3 Scope

This project was conducted to develop a Bus Tracker Mobile Application that will provide real-time bus tracking and incorporate features related to students' class schedules. The target location for this application is the bus services at Universiti Malaysia Sarawak (UNIMAS). The target users of this project are UNIMAS students and bus drivers. The bus drivers play an important role because they use their devices as GPS, allowing them to provide updates about bus availability and any changes to bus information. Meanwhile, students can track the bus's location, get estimated time arrival(ETA), estimated walking time to pickup point, receive notifications about bus availability, bus status and bus schedule. However, there may be some limitations that this Bus Tracker App may face, such as connectivity problems and the battery life of devices, as these depend on the mobile devices of the bus drivers.

1.4 Objectives

The objectives of this Bus Tracker Mobile Application are:

i. To analyze existing bus tracking systems to develop a mobile application that provides real-time tracking of buses and estimates time arrival(ETA).
ii. To design and implement a mobile application that offers nearest bus recommendations and optimal pickup points based on students' class schedules.
iii. To evaluate the usability and functionality of the application in checking bus availability, particularly during late hours.

1.5 Brief Methodology

Methodology is a procedure to manage the project for overall validity and reliability of the project (Mural, 2024). Agile Methodology was chosen for this project because it allows adaptability, regular adjustments, gathers feedback continuously and makes it easier to identify problems through testing at each iteration. Agile project management has six phases, Figure 1.1 illustrates the Agile process and below is the breakdown of each phase along with the related practices that define the Agile process.

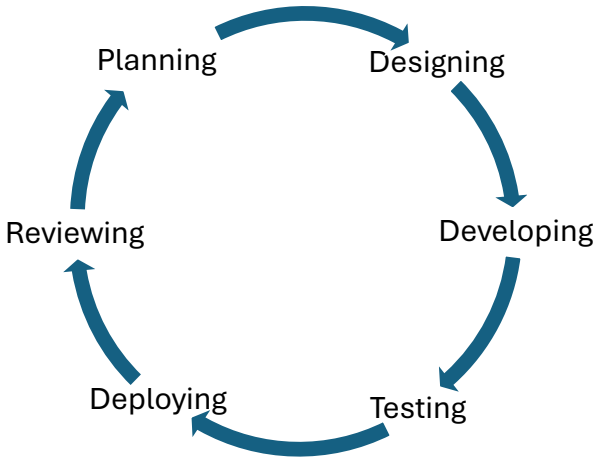


Figure 1.1: Agile Development Model (Mural, 2024)

i. **Planning Phase**

In the planning phase, initial functionalities, problems, and solutions were identified through research on existing systems and survey. This phase focused on outlining deliverables, setting objectives, and preparing the sprint backlog. Each deliverable in

this phase was associated with an estimated timeline to guide the development process (Mural, 2024).

ii. **Designing Phase**

The design phase involves creating the user interface (UI) and establishing the software architecture for the Bus Tracker mobile application. During this phase, the UI is tailored to ensure ease of use for users, incorporating essential features. Additionally, this phase involves organising tasks and the establishment of a clear development framework to guide the project (Mural, 2024).

iii. **Developing Phase**

The development phase focuses on developing and adapting the project as it progresses. This phase ensures that the project stays aligned with the plan while adjusting priorities and timelines as needed (Mural, 2024).

iv. **Testing Phase**

After the development phase, the testing phase takes place. This is the time to evaluate the system and ensure it aligns with the defined requirements (Mural, 2024). If the system does not meet the requirements, steps three and four will need to be redone and adapted before moving to the next phase (Mural, 2024).

v. **Deploying Phase**

The deployment phase is where the system is delivered to the customer or user (Mural, 2024). This is the phase that ensures all the requirements and deliverables have been successfully submitted (Mural, 2024).

vi. **Reviewing Phase**

The review phase is the phase that encourages continuous review and feedback during the entire development process (Mural, 2024).

1.6 Significant Project

i. Provide students with essential information about the bus such as bus location, bus schedule, and bus availability.
ii. It can minimize wasted time by helping students optimize travel plans.

iii. Offers a cost-effective and convenient solution for students, reducing reliance on e-hailing services.

1.7 Project Schedule

This final year project 1 starts on 10th October 2024 and is expected to be completed on 25th January 2024 which is 4 months. The details of the project are described in Table 1.1 and represented in Figure 1.2. The final year project 2 starts on 17th March 2025 and is expected to be completed on 12th June 2025 which is 4 months. The details of the project are described in Table 1.1 and represented in Figure 1.3.

Table 1.1: Project Schedule for FYP 1

Task	Start Date	End Date
PROJECT PROPOSAL	10 October 2024	14 November 2024
Brief Proposal Preparation	10 October 2024	19 October 2024
Submission of Brief Proposal	20 October 2024	20 October 2024
Approval and Comments by Examiner	21 October 2024	28 October 2024
Full Proposal Preparation	29 October 2024	13 November 2024
Submission of Full Proposal	14 November 2024	14 November 2024
FYP 1	15 November 2024	25 January 2025
Chapter 1: Introduction	15 November 2024	20 November 2024
Submission of Chapter 1	21 November 2024	21 November 2024
Chapter 2: Literature Review	22 November 2024	12 December 2024
Submission of Chapter 2	13 December 2024	13 December 2024
Chapter 3: Requirement Analysis and Design	14 December 2024	4 January 2025
Submission of Chapter 3	5 January 2025	5 January 2025
Submission of FYP 1 Final Report and Paper Assessment	17 January 2025	17 January 2025
Presentation	24 January 2025	25 January 2025
FYP 2	24 March 2025	28 March 2025
Chapter 4: Implementation and Testing	24 March 2025	13 May 2025
Submission Chapter 4	14 May 2025	15 May 2025
Chapter 5 & 6: Conclusion	16 May 2025	29 May 2025
Submission Chapter 5 & 6	30 May 2025	31 May 2025

Preparation of First Draft for FYP2 Full Report and Paper	1 June 2025	8 June 2025
Submission of First Draft for FYP 2 Full Report and Paper	9 June 2025	10 June 2025
Submission of FYP 2 Final Report and Paper Assessment	23 June 2025	23 June 2025
FYP 2 Symposium	30 June 2025	1 July 2025

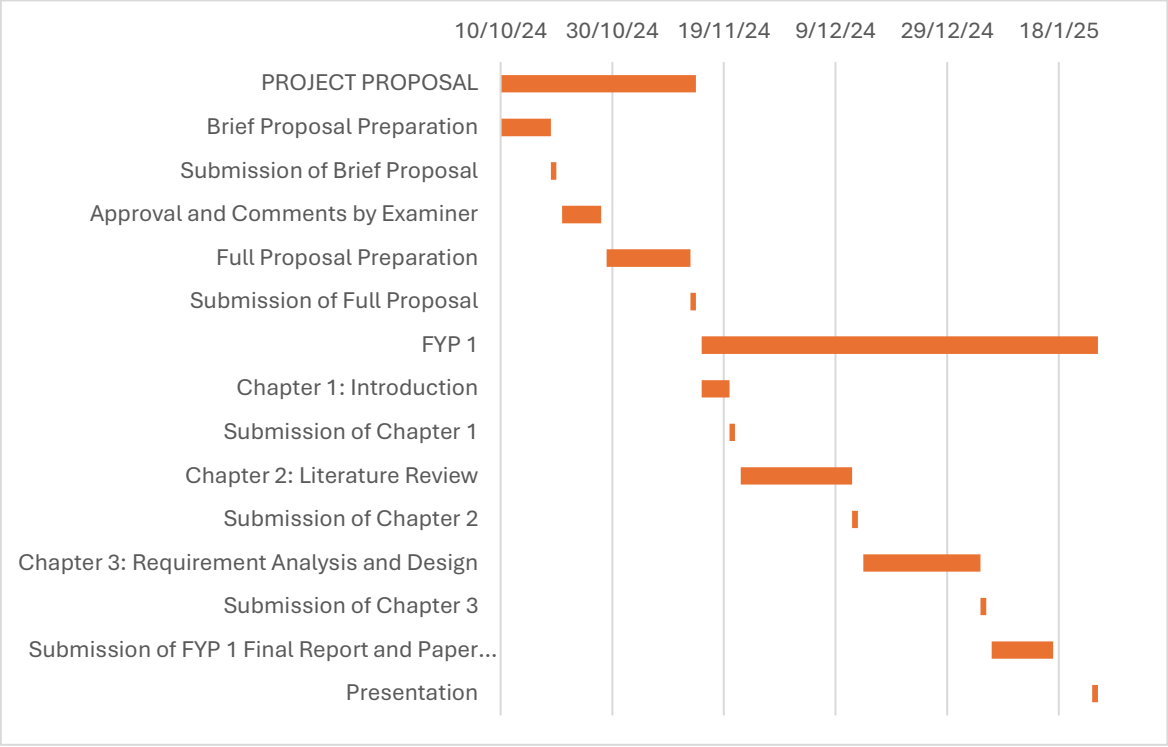


Figure 1.2: Schedule FYP 1

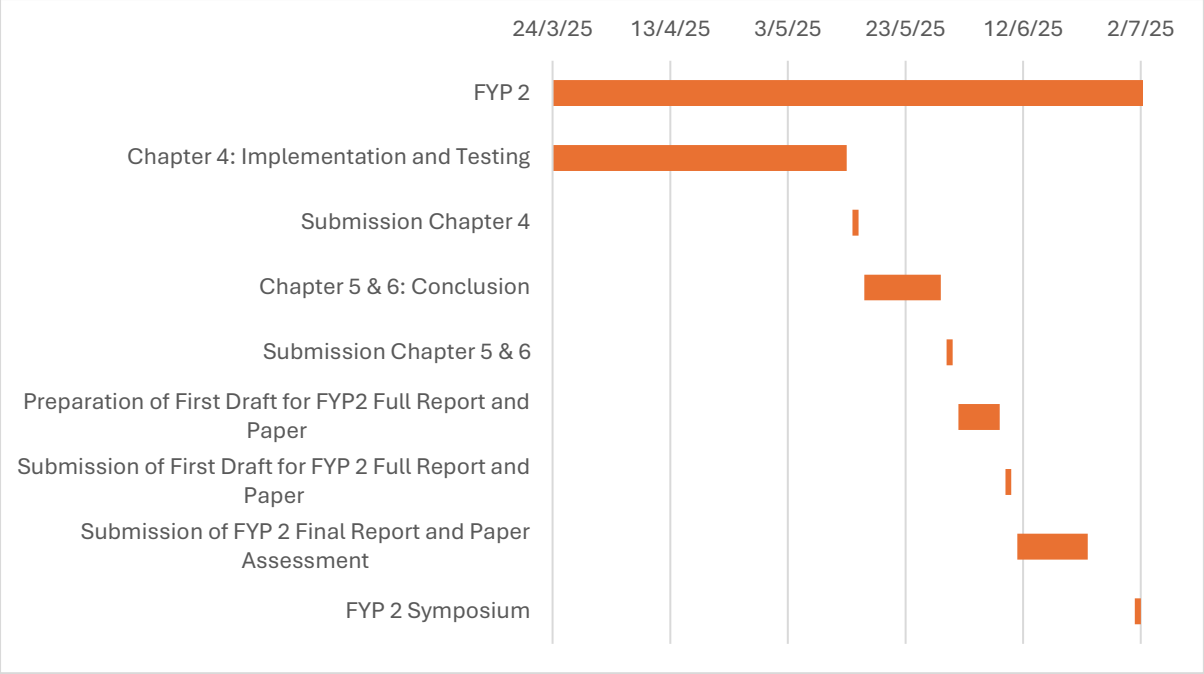


Figure 1.3: Schedule FYP 2

1.8 Expected Outcome

The outcomes that should be expected for this project are:

i. The mobile application displays real-time bus locations and estimate time arrival.
ii. A feature that allows students to view nearest bus and recommended pickup points based on their class location.
iii. A notification system that will notify students about bus availability during late hours and the end of service hours.

1.9 Summary

In this chapter, the purpose, scope, and aims of the Bus Tracker Mobile Application are outlined. The project aims to enhance transportation services for UNIMAS students and bus drivers. The scope also highlights the project's limitations, such as connectivity issues and device battery life. The Agile methodology has been chosen to achieve the expected outcomes, and a project schedule has been set to achieve the objectives within four months. Furthermore, the significance of this project has been identified, including providing students with essential information about bus schedules with timely reminders based on their class schedules, minimizing wasted time by optimizing travel plans, and offering a cost-effective and convenient solution for students. The next chapter will discuss the review and analysis of the existing system in detail.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Based on the objectives of project that mentioned in the previous chapter, this chapter aims to achieve the objectives by doing background research and analyze the mobile application. The chapter focuses on reviewing and comparing 3 existing bus tracker mobile application in Malaysia. Besides, this comparison aims to analyze the system requirements of each existing mobile application, including their features and limitations, while providing an overview of the essential features that will define the proposed mobile application.

i. Moovit
ii. MyRapid PULSE by Prasarana Malaysia Berhad
iii. Malaysia Bus Live GPS Tracker

2.2 Reviewed on Existing Mobile Applications

2.2.1 Moovit

Moovit was created to simplify urban mobility and make it more accessible, efficient, and sustainable (Moovit, 2020). It provides sustainable mobility solutions that encourage shared transportation, reducing pollution and increasing safety (Moovit, 2020). The Moovit app is available in many cities worldwide, including Malaysia. This transport app supports various modes of transportation, and in Malaysia, it provides services for buses, trains, LRT, monorail, cable cars, and ferries. However, it is only available in Bandaraya Ipoh, Kuala Lumpur, Johor Bahru, Kuantan, and Penang.

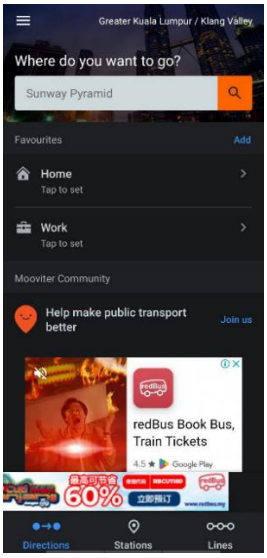


Figure 2.4: Directions Page

In Figure 2.4, the first tab the navigation bar is Directions. Users can set their location on this page, and the location options are limited to Bandaraya Ipoh, Kuala Lumpur/Klang Valley, Johor Bahru, Kuantan, and Penang. Users can enter the destination location in the search bar, and then it will be directed to the next page, as shown in Figure 2.1. Additionally, on the Directions page, users can set their favorite locations, such as home or work, making it easier for them to plan their routes quickly.

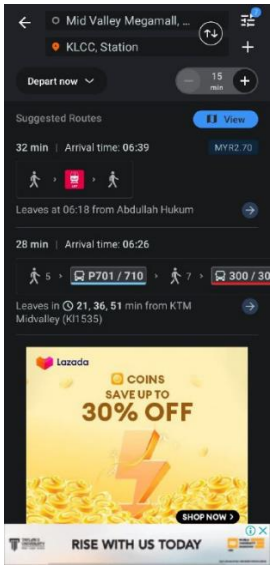


Figure 2.5: Set Location Page

After the user enters the destination location in the search bar, it will be directed to the next page, as shown in Figure 2.5. Users can enter both the current location and destination on this page. The '+' icon beside the destination location bar allows the user to add additional locations they plan to visit. Additionally, the user can set the departure time by tapping on the dropdown icon labeled 'Depart now' and use the '+' icon to add 15 minutes or more to the departure time. The filter icon at the top right corner allows the user to select route preferences and transport type preferences, making it easier since this app not only focuses on buses but also offers other types of transportation. Once the user has entered all locations, set the departure time, and adjusted their preferences, the app provides bus suggestions and plans the journey for the user. This includes details such as the estimated walking time for the user to reach the bus stop, the estimated arrival time, and the duration time to reach the destination. After a journey plan is chosen, the user can view real-time bus tracking updates, which are taken directly from GPS devices on the bus.

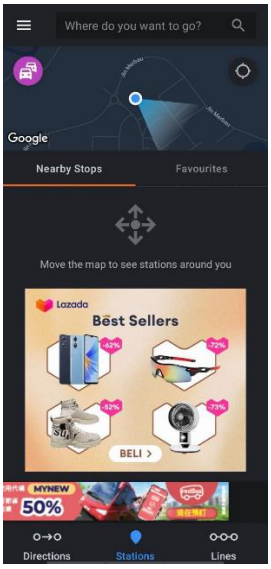


Figure 2.6: Stations Page

In Figure 2.6, the user can enter the destination location, which will be displayed on the map. Additionally, the user can either tap on the 'Nearby Stop' section to view nearby bus stops or tap on the 'Favourites' section to select their favorite lines, stations, and places.

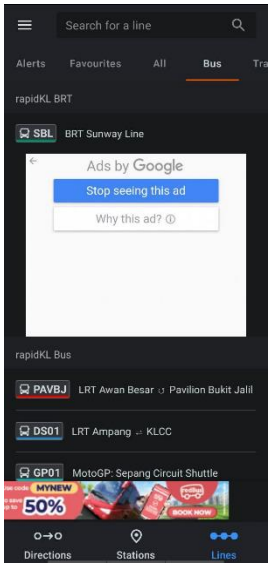


Figure 2.7: Lines Page

Figure 2.7 shows the Lines page. The user can enter a route in the search bar to find a specific route or select one from the Favourites or Bus sections. The different colors on the bus icons indicate various categories or types of bus routes, such as express buses, feeder buses, or shuttle buses. In the 'Alerts' section or the notification center in the sidebar, the user can receive real-time alerts about issues such as unexpected delays, traffic jams, new construction, or emergencies.

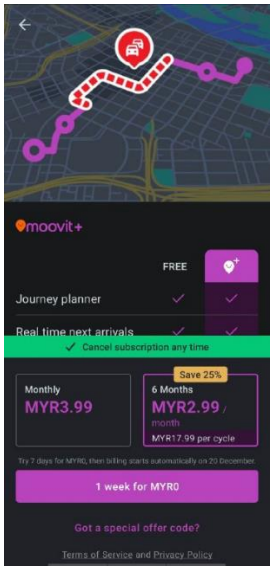


Figure 2.8: Upgrade to Moovit+

The Moovit app offers in-app purchases for RM3.99 monthly. By upgrading to Moovit+, it provides features such as a journey planner, real-time next arrivals, line schedules, live navigation, green rides, an ad-free experience, route comparisons, and more. For users who choose to use the free version, Moovit's features are limited to journey planner, real-time next arrivals, line schedules, and live navigation. In the free version, there are many ads, including pop-up ads that appear every few minutes, which can sometimes lead to user frustration.

2.2.2 MyRapid PULSE

The MyRapid PULSE app, developed by Prasarana Malaysia Berhad that focuses on public transport development and infrastructure projects in the country which also owns several bus services in Peninsular Malaysia (Prasarana Malaysia Berhad, 2020). This app is created to assist public transport commuters in Penang, Klang Valley, and Kuantan, enabling users to efficiently plan their journeys using Rapid Penang, Rapid KL, and Rapid Kuantan routes and networks (Prasarana Malaysia Berhad, 2020). MyRapid PULSE app offers features for bus tracker, allowing users to view real-time bus locations, estimation arrival times, reducing uncertainty in travel planning.

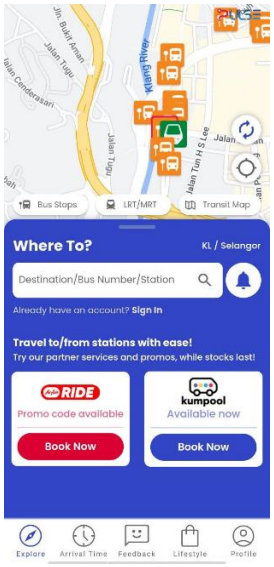


Figure 2.9: Explore Page

In Figure 2.9 is Explore page, the orange bus icon on the map is bus stops that available in that region. The user can set the service region option above bell icon, but its limited to only KL/Selangor, Penang and Kuantan region (Prasarana Malaysia Berhad, 2020). The user can enter destination or bus number or station in search bar to search best route for their journey. Besides, user can either choose to tap on any bus stop in the map and it will provide the bus number, bus stop locations and the routes available at each bus stop.

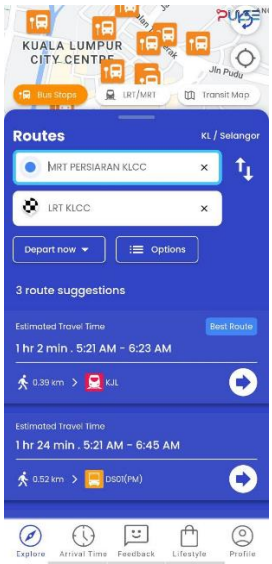


Figure 2.10: Search Routes

When the user enters the current location and destination location as shown in Figure 2.10, the app suggests the available bus services and identifies the best route. It provides details such as the estimated travel time, departure time, arrival time, and the walking distance to the bus stop. Additionally, the user can set their preferred departure or arrival time using the

dropdown menu labeled 'Depart now'. Then, the user can select the most suitable route suggested based on their journey requirements.

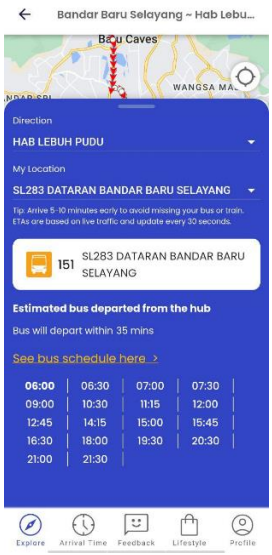


Figure 2.11: Details About the Selected Route

After the user selects the suggested route shown in Figure 2.10, they will be directed to a page in Figure 2.11. This page provides details such as the bus route number, bus number, bus stop location, estimated departure time, and the bus schedule. The map in Figure 2.8 displays the route line along with real-time bus tracking.

2.2.3 Malaysia Bus Live GPS Tracker

Malaysia Bus Live app was release on Play Store on September 2023 (Abdullah, 2023). Bus Live app currently only cover public bus tracking in Klang Valley Region including buses that operated by Go KL, Rapid Bus KL(use MyRapid PULSE app) and selected Smart Selangor bus routes. This app was created to increase the public commuting experience to user by providing real-time bus tracking, provide accurate location updates using GPS and also offer see real-time timing over 170 routes (Abdullah, 2023). The user can plan their trip because this app provide information about bus routes and bus stops. By using this app, no more uncertainty in bus location and long waiting times (Abdullah, 2023), because this app uses differently colored bus icons to display the updated status of each bus.

Figure 2.12: Bus Live Page

Figure 2.12 shows the Bus Live page, where users can search for a specific route by entering the route number in the search bar. The map displays the available buses in the city. The different colors of the bus icons are explained in the 'Legend' box. For example, the green bus icon indicates an active trip, while the yellow bus icon represents a bus off-trip.

Figure 2.13: Search Route

When the user enters the route number in the search bar, the map displays the buses operating on that route. It also provides real-time bus tracking by showing the bus's location along the route (Abdullah, 2023). Additionally, users can view detailed information about the bus by tapping on its icon, which state the bus number, GPS time, GPS date, bus speed, and route number. The app updates bus movements every 30 seconds, which is enabled by GPS devices installed on the buses.

2.3 Comparison on Existing Mobile Applications and Proposed Mobile Application

Table 2.2: Comparison Features Between Mobile Applications and Proposed Mobile Application

Features \ Systems		Moovit	MyRapid PULSE	Malaysia Bus Live GPS Tracker	Proposed Mobile Application
Main Components		GPS (Hardware)	GPS (Hardware)	GPS (Hardware)	GPS (Mobile-based)
Real-Time Bus Tracking		√	√	√	√
Route Line		√	√	√	√
Estimation Waiting/Depart Time		√	√	-	√
Estimated Walking Times		√	-	-	√
Estimation Time Arrival		√	√	-	√
Bus Information	Plate Number & Type	√	√	√	√
	Bus Schedule	√	√	-	√
	Update Bus Availibility	√	-	-	√
Nearby Bus Recommendation		√	√	-	√
Student Schedule Integration		-	-	-	√

Table 2.2 outlines the comparison features between existing mobile applications, namely Moovit, MyRapid PULSE, and Malaysia Bus Live, with the proposed mobile application, Bus Tracker Mobile Application at UNIMAS. In modern vehicle tracking, GPS technology is a standard method for tracking the transport (Sree et al., 2021). The proposed Bus Tracker app utilizes mobile-based GPS by enabling GPS on the driver's phone (Sree et al., 2021), whereas the other three apps rely on hardware GPS installed on buses. The use of mobile-based GPS in the Bus Tracker app is accurate and cost-effective, providing a practical alternative to hardware GPS, which is typically employed in large-scale bus services with extensive routes.

The Malaysia Bus Live GPS Tracker lacks many important features and only includes basic functionalities, such as real-time bus tracking, route lines displayed on maps, and bus information. However, this app does not provide critical features like bus schedules, destination

locations. These features are important for helping users plan their journeys effectively and quickly identify nearby bus stops. The Bus Live app is primarily designed only for tracking real-time bus locations based on route numbers and monitoring the status of buses.

The Moovit app and MyRapid PULSE app, on the other hand, do not offer bus availability status, and MyRapid PULSE also lacks features like estimated walking time. These missing features are crucial for enabling efficient journey planning. Additionally, all the existing apps lack integration with users' personal schedules, which would be a valuable feature. Integrating user schedules could help users get estimated time arrival, estimated walking time and nearest bus recommendation based on their class location and nearest pickup point recommendation.

In a nutshell, based on the comparison in Table 1, the proposed mobile application is beneficial as it addresses the gaps found in existing apps and provides all the necessary features to enhance the user experience. Unlike the existing systems, which offer only basic functionalities like real-time bus tracking and route information, the proposed mobile application includes features such as bus schedules, estimated walking time, bus availability status and notifications. These features make it easier for students to stay updated on bus status, and plan their journeys effectively. Since the proposed app is specifically designed for university students, student schedule integration has been included as a key feature. This feature allows students to input their class schedules, suggest nearby buses, identify nearby pickup points, provide estimated waiting times and estimated walking times. By offering these enhanced features, the proposed mobile application not only improves the overall bus tracking experience but also makes daily commuting more efficient and convenient for UNIMAS students.

2.4 Summary

This chapter reviewed the features of three existing mobile applications and highlighted their limitations. The comparison table illustrates the gaps and differences between the features available in the existing apps and the proposed app. Additionally, this chapter introduces key features that have not been implemented in any existing university bus tracking app. The requirements analysis will be detailed in the Chapter 3.

CHAPTER 3: REQUIREMENT ANALYSIS AND DESIGN

3.1 Introduction

In this chapter, it will cover the requirements analysis and design of the proposed system and discuss various topics. Besides, this chapter will explain the project methodology and break down the steps that are involved in developing the proposed system. The architecture and the flowchart of the system will be focused on.

3.2 Project Methodology

Project methodology for this Bus Tracker Mobile Application as stated in chapter 1 is Agile methodology. Agile project management has six phases, Figure 3.14 illustrates the Agile process and below the breakdown of each phase along with the related practices that define the Agile process.

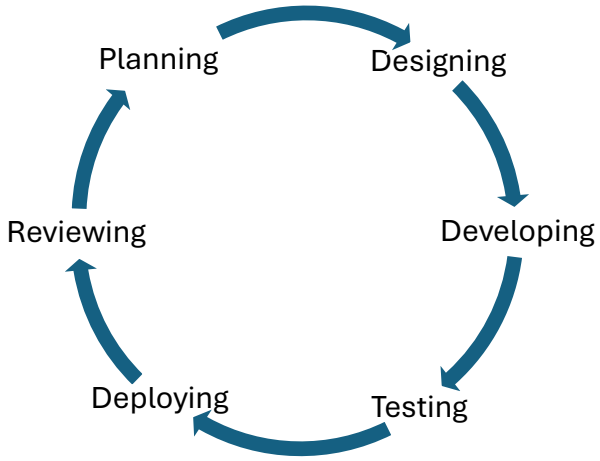


Figure 3.14: Agile Development Model

In the planning phase, initial functionalities, problems, and solutions were identified through research on existing systems and survey. General knowledge about bus services was considered to better understand user needs and define the project's scope. This phase focused on outlining deliverables, setting objectives, and preparing the sprint backlog. Each deliverable in this phase was associated with an estimated timeline to guide the development process (Mural, 2024).

Next, the design phase involves creating the user interface (UI) and establishing the software architecture for the Bus Tracker mobile application. During this phase, the UI is tailored to ensure ease of use for users, incorporating essential features. Additionally, this phase

involves the organization of tasks and the establishment of a clear development framework to guide the project (Mural, 2024).

The development phase focuses on developing and adapting the project as it progresses. This phase ensures that the project stays aligned with the plan while adjusting priorities and timelines as needed (Mural, 2024). The Bus Tracker mobile application will be developed to utilize the built-in GPS feature of the mobile device. This feature allows the app to access the device's GPS system to track and share the bus driver’s location in real-time.

After the development phase, the testing phase takes place. This is the time to evaluate the system and ensure it aligns with the defined requirements (Mural, 2024). If the system does not meet the requirements, previous phases will need to be revisited and adapted before proceeding to the next phase (Mural, 2024). In this phase, fully developed Bus Tracker app will be tested and executed under typical user conditions.

The deployment phase is where the system is delivered to the customer or user (Mural, 2024). This phase ensures that all requirements and deliverables have been successfully met and submitted (Mural, 2024). The fully developed Bus Tracker app will be deployed to a testing environment first, followed by deployment to end users.

The review phase is the phase that encourages continuous review and feedback during the entire development process (Mural, 2024). This process ensures that any issues regarding the Bus Tracker app are identified and addressed to improve the overall user experience. By incorporating feedback from users, the app can enhance its functionality and performance, ensuring it aligns with the project’s goals.

3.3 Requirement Analysis

3.3.1 Identify the user

The target users for this proposed system are students and bus drivers at Universiti Malaysia Sarawak (UNIMAS). These students often face challenges in keeping track of available buses due to unexpected delays. This system is designed to provide real-time bus tracking, helping students manage their daily transportation more effectively. Since the app is specifically tailored for students, an integrated class schedule feature is included to improve usability. By using this app, students can arrive on time for classes and reduce their reliance on costly e-hailing services. Additionally, as bus drivers are also users of the app, they can update bus locations and statuses in real time, as well as modify bus information, ensuring accurate and timely updates are available to students.

3.3.1.1 Survey Analysis and Proposed Features for Bus Tracker Mobile Application

I conducted a survey to identify the problems students faced and determine which features respondents believe are essential for a bus tracker app for UNIMAS students. Below is the analysis of the survey results, along with my proposed solutions:

a) Demographics and Usage

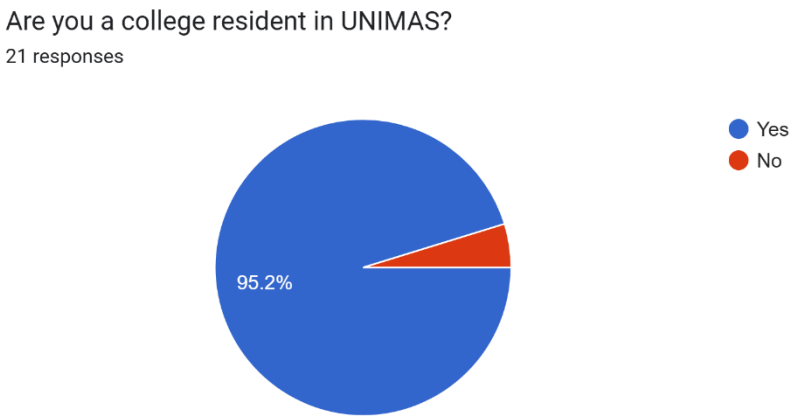


Figure 3.15: College Residents

Based on the pie graph on Figure 3.15, 95.2% of students are college residents, indicating that most bus service users live on campus. These students rely on buses as their primary mode of transportation within UNIMAS.

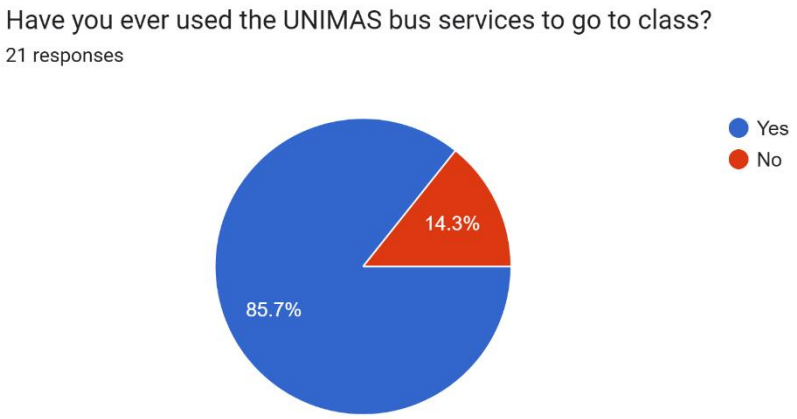


Figure 3.16: UNIMAS Bus Services Usage

According to the pie chart on Figure 3.16, 85.7% of students use UNIMAS bus services to attend classes, while 20% never use the bus. The non-users likely have their own transportation or live close enough to walk to their faculties.

b) Bus Services Usage and Experiences

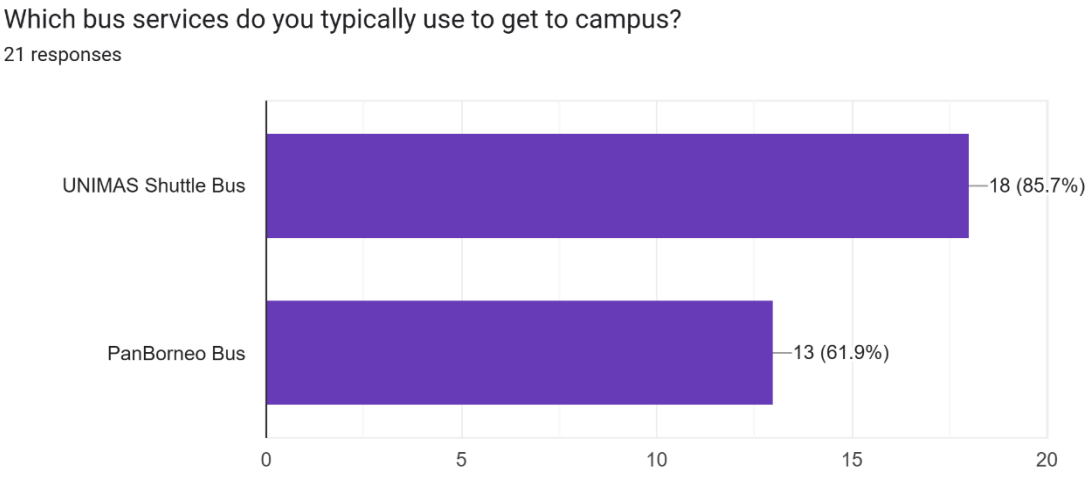


Figure 3.17: Bus Services Preferences

From the bar graph in Figure 3.17, 85.7% of students typically used the UNIMAS Shuttle Bus, compared to 61.9% who used the PanBorneo Bus. The difference is likely because the

UNIMAS Shuttle Bus is free, while the Pan Borneo Bus charges RM1 per ride. This highlights those financial considerations play an important role in bus preference among students.

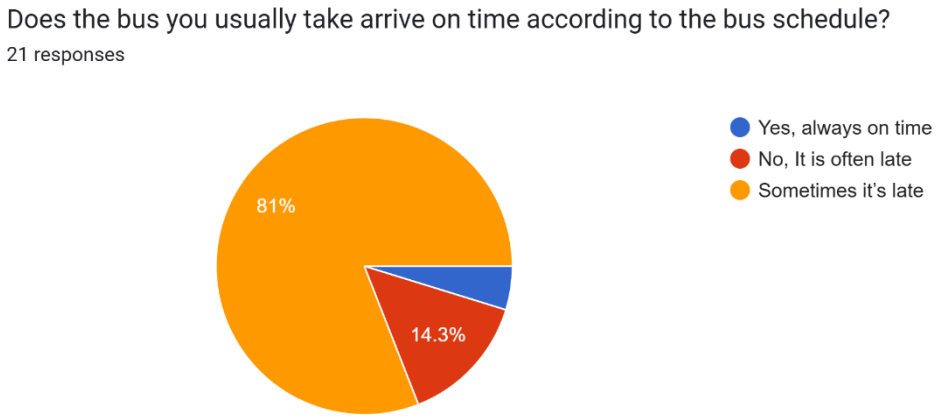


Figure 3.18: Bus Arrival Time According to Bus Schedule

Based on pie chart on Figure 3.18, only 4.8% of students stated that buses arrive on time according to the schedule, while 14.3% reported frequent delays, and 81% experienced occasional delays. This indicates that students often find it difficult to predict bus arrival times, leading to long waiting periods. To address this issue, the proposed system will implement an Estimated Time of Arrival (ETA) feature in the app. This feature will provide real-time arrival updates, allowing students to plan their journeys and reduce waiting times at pickup points.

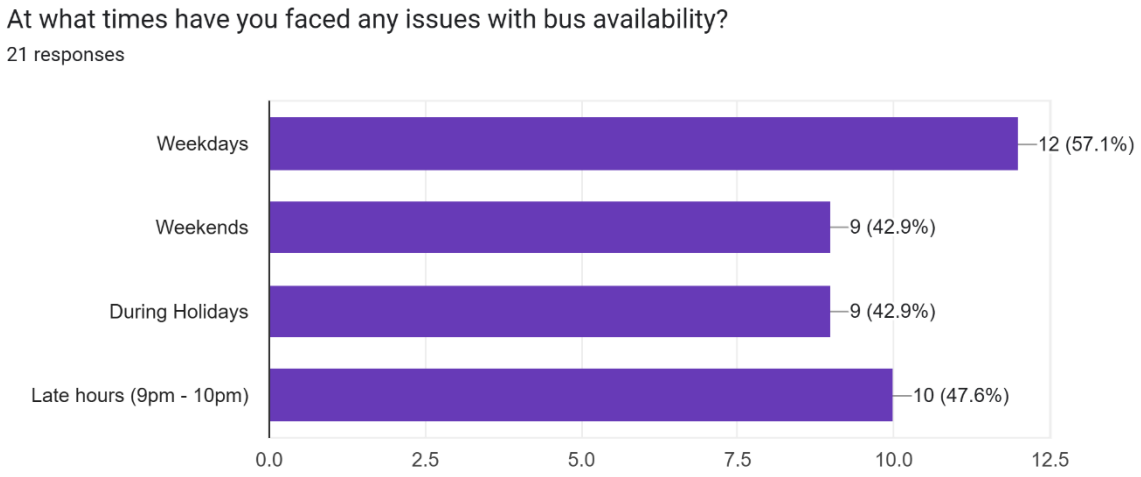


Figure 3.19: Bus Availability Issues

The bar graph in Figure 3.19 indicates that students faced issues with bus availability during 57.1% on weekdays, 47.6% on late hours, 42.9% on weekends and 42.9% on holidays. Bus availability on weekdays is critical as most students attend classes during this time. Late-hour buses are especially important for students with classes ending between 9pm until 10pm, but many students are unsure if the buses are still operating due to delays or lack of updates. On weekends and holidays, students faced difficulties due long breaks taken by driver or missed announcements about bus schedules during holidays. The proposed system will include a bus availability status feature which bus drivers can update their availability in real-time by setting the bus status to "Available," "On Break," or "Not Available" and include notifications about changes in bus availability.

c) Recommendation Features

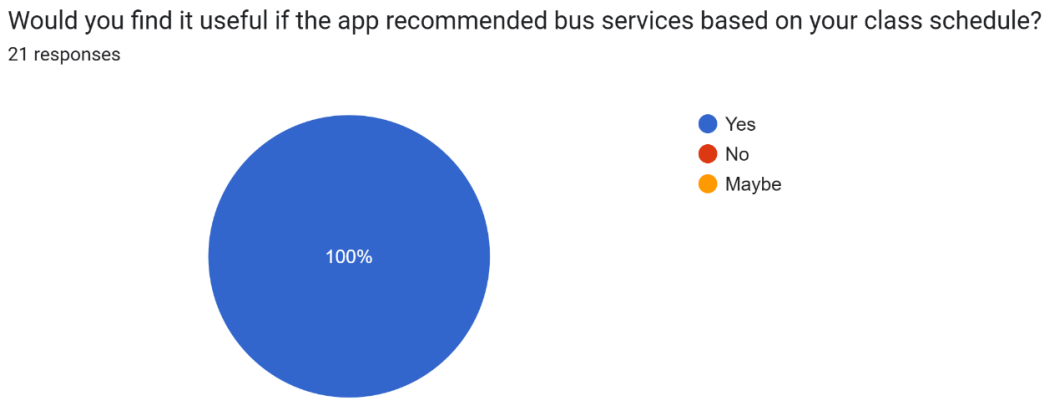


Figure 3.20: Bus Services Recommendation

Based on pie chart in Figure 3.20, 100% of student stated that a bus recommendation feature based on their class schedule would be useful. This feature would help students identify which bus to take, helps their time management decision and financial planning.

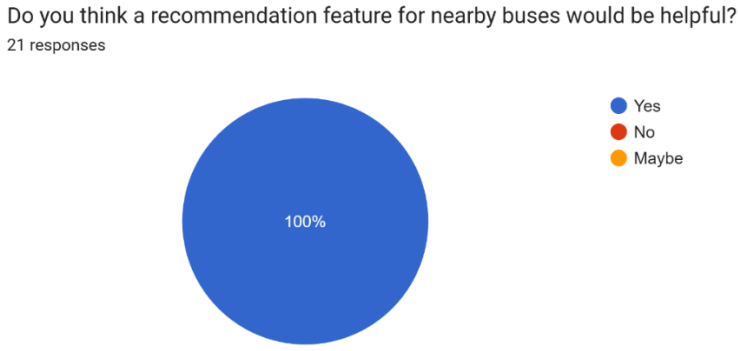


Figure 3.21: Nearby Bus Recommendation

Based on pie chart in Figure 3.21, 100% of the student find the recommendation feature for nearby buses would be helpful. The proposed system will allow students to input their desired pickup and destination points, and the app will recommend nearby available buses and routes.

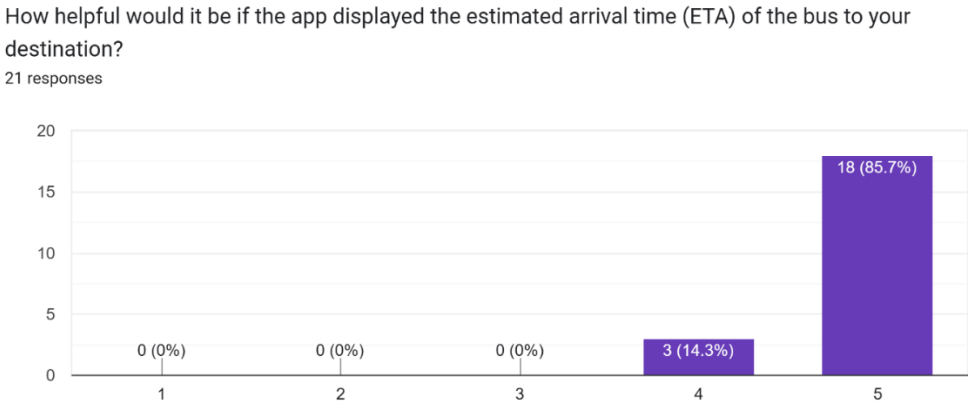


Figure 3.22: Estimated Time of Arrival

Based on bar graph in Figure 3.22, 85.7% of the students stated that if the app displayed the estimate time of arrival of the bus to the destination would be helpful. The proposed system will include ETA feature which allow students to monitor the bus's real-time location and plan their arrival at the pickup point more effectively.

The survey results highlight the experiences and issues faced by UNIMAS students regarding bus services. By incorporating the features that students find useful, the proposed mobile app will provide effective solutions to address these issues. These features align with the survey findings and ensure the app meets the needs and expectation of the users.

3.3.2 Hardware Requirement

The proposed system only requires the mobile devices of bus drivers and students. This bus tracker app does not rely on GPS hardware, which is often expensive and typically used in larger industrial applications. Instead, the system uses the driver's mobile device as a GPS, offering a more cost-effective solution.

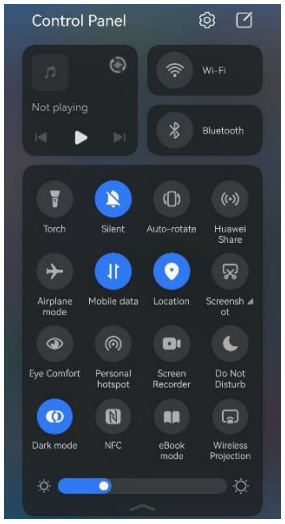


Figure 3.23: Enable the Location Feature

Based on the Figure 3.23, the user must enable the location feature on their mobile device to allow the app to act as a tracking tool by sending location data to the app's tracking system. Mobile devices come with built-in Assisted Global Positioning System (AGPS) receivers that identify the user's location and support location-based services. The AGPS chip in the mobile device receives signals from satellites and using this data to calculate the user's location. The app uses AGPS to determine the user's location in real-time and relies on cellular network data, as shown in the figure, to enhance location accuracy, which is crucial for real-time bus tracking. By integrating GPS signals and cell tower data, the app can provide a reliable and accurate real-time view of the buse's locations, typically within a 50-meter range. The location data is transmitted through existing cellular connections, ensuring minimal additional data usage. Leveraging AGPS allows the app to achieve quick location updates while conserving the device's battery life.

3.3.3 Software Requirement

The software requirements are essential to develop the proposed system. The Table 3.3 below outlines the software requirements for the Bus Tracker app prototype.

Table 3.3: Software Requirements

Software	Specification
Operating System	Windows
Integrated Development Environment (IDE)	• Android Studio
Programming Language	• Dart • Flutter
Backend	Firebase
Location & Map Integration	Google Maps API

3.4 System Design and General Architecture

This project using the Unified Modeling Language (UML) as the design methodology. UML was selected because it is compatible with object-oriented approaches, aligning well with Flutter's object-oriented framework. Additionally, it supports integration with Firebase and Google Maps API, which are important components of the Bus Tracker App. This section focuses on diagram Use Case, Class, Sequence, Component, and State Machine Diagram which describe the system's logic, structure and user interactions.

3.4.1 Use Case Diagram

The Use Case Diagram in the Figure 3.24 provides an overview of interactions in the Bus Tracker App between two actors, the Bus Driver and the Student. Students can perform actions such as viewing real-time bus locations, inputting class schedules, viewing class schedules,bus status, and receiving notifications. Bus drivers are responsible for updating bus schedule, updating bus status, updating bus operational mode and able to view notifications.

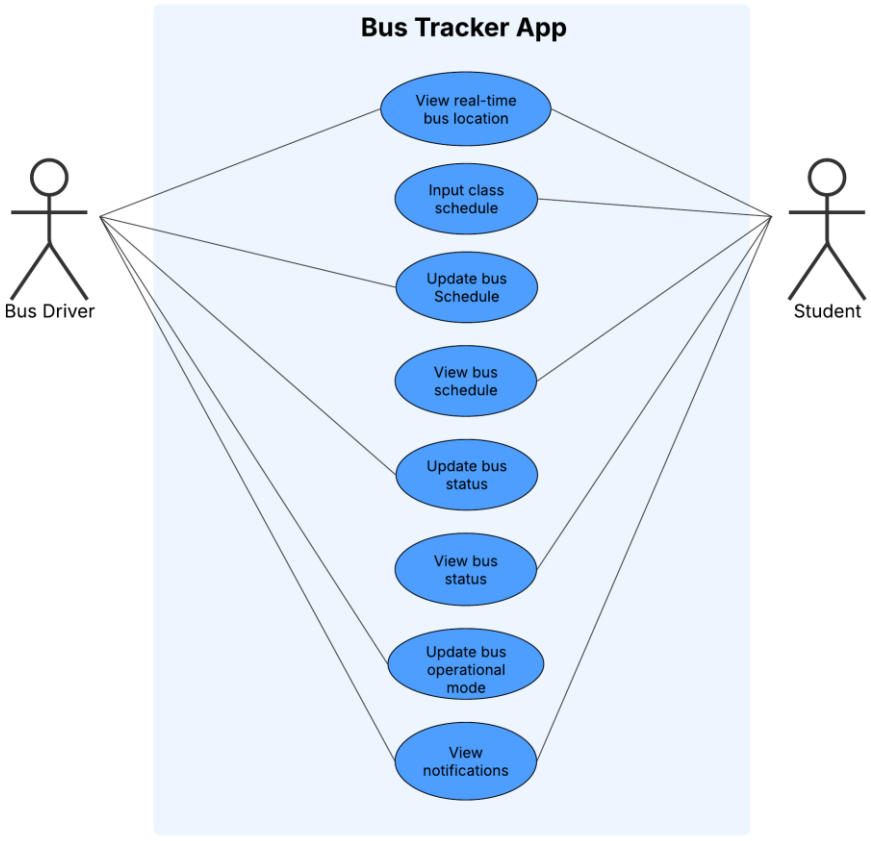


Figure 3.24: Use Case Diagram

3.4.2 Use Case: Functional Requirements

3.4.2.1 Student Can View Real-Time Bus Locations

3.4.2.1.1 Stimulus/Response Sequence

Table 3.4: Response Sequence of view real-time bus locations

Use Case	View Real-Time Bus Locations
Short Description	Allows students to track in real time bus location using GPS through the app interface.
Actor(s)	Student
Pre-condition(s)	The location button is enabled.
Post-condition(s)	The student successfully views the current bus location on the map.
Main Flow	1. Student opens the Bus Tracker App. 2. The system retrieves real-time GPS location data. 3. The system displays the bus location on the map.
Alternative Flow	N/A

3.4.2.2 Student Can Input Class Schedule

3.4.2.2.1 Stimulus/Response Sequence

Table 3.5: Response Sequence of input class schedule.

Use Case	Input Class Schedule
Short Description	Allows students to save their class schedules and nearest pickup points
Actor(s)	Student
Pre-condition(s)	N/A
Post-condition(s)	Class schedule saved successfully.
Main Flow	1. Student logs into the app. 2. Selects "Class Schedule" from the sidebar bar. 3. Inputs class details (location, class start time, class end time). 4. The system saves the schedule and associates it with nearest pickup point.
Alternative Flow	A1: Missing input fields, system prompts the user to complete all required fields.
Exception Flow	N/A

3.4.2.2.2 Direct Functional Requirement

Table 3.6: Input Class Schedule direct funtional requirement

Requirement	Description
REQ-1	Manage and store multiple schedules.

3.4.2.1 Student Can View Bus Schedule

3.4.2.1.1 Stimulus/Response Sequence

Table 3.7: Response Sequence of view bus schedule

Use Case	View Bus Schedule
Short Description	Able to view bus schedule for both UNIMAS Shuttle Bus and PanBorneo Bus.
Actor(s)	Student
Pre-condition(s)	Bus schedules are updated by bus driver.
Post-condition(s)	N/A
Main Flow	1. The student selects either 'UNIMAS Shuttle Bus' or 'PanBorneo Bus' from the sidebar. 2. The student checks the bus schedule image on the page.
Alternative Flow	N/A

3.4.2.1 Student Can View Bus Status

3.4.2.1.1 Stimulus/Response Sequence

Table 3.8: Response Sequence of view bus status

Use Case	View Bus Status
Short Description	Able to check bus status.
Actor(s)	Student
Pre-condition(s)	Bus driver update the bus status.
Post-condition(s)	Able to check unavailable bus because or short break or holiday.
Main Flow	3. The student selects either 'UNIMAS Shuttle Bus' or 'PanBorneo Bus' from the sidebar. 4. The student checks the bus status below the bus schedule image on the page. 5. The student can check the bus information and status whether the bus is available, not available, or on a break.

Alternative Flow	N/A

3.4.2.3 Student Can View Notifications

3.4.2.3.1 Stimulus/Response Sequence

Table 3.9: Response Sequence of view notifications

Use Case	View notifications for bus status update, bus operational time and schedule Changes
Short Description	Ensures the students got the updated information about schedule changes, bus status and operational time.
Actor(s)	Student
Pre-condition(s)	-
Post-condition(s)	Notifications are successfully sent and received.
Main Flow	1. System detects a bus driver update bus schedule, update bus status or update regarding start operational time and the end of operational time of the bus. 2. Notification message generated. 3. Student receives the notification in the notification page.
Alternative Flow	N/A
Exception Flow	N/A

3.4.2.1 Bus Driver Can View Real-Time Bus Locations

3.4.2.1.1 Stimulus/Response Sequence

Table 3.10: : Response Sequence of view real-time bus locations

Use Case	View Real-Time Bus Locations
Short Description	Allows bus driver to view in real time bus location using GPS through the app interface.
Actor(s)	Bus Driver
Pre-condition(s)	The location button is enabled.
Post-condition(s)	The bus driver successfully views their current location on the map.
Main Flow	1. Bus driver opens the Bus Tracker App. 2. The system retrieves real-time GPS location data. 3. The system displays the bus location on the map.
Alternative Flow	N/A

3.4.2.8 Bus Driver Can Update Bus Schedule

3.4.2.8.1 Stimulus/Response Sequence

Table 3.11: Response Sequence of update bus schedule.

Use Case	Update Bus Schedule
Short Description	Ensures accurate bus schedules for students.
Actor(s)	Bus Driver
Pre-condition(s)	The new schedule from the authorized personnel.
Post-condition(s)	Schedule information is updated successfully.
Main Flow	1. Bus driver selects “Bus Schedule” from sidebar bar. 2. Click “Edit Bus Schedule” button. 3. Upload new bus schedule in image format and save the changes. 4. System stores and updates the new schedule.
Alternative Flow	N/A
Exception Flow	N/A

3.4.2.7 Bus Driver Can Update Bus Status

3.4.2.7.1 Stimulus/Response Sequence

Table 3.12: Response Sequence of update bus status.

Use Case	Update Bus Status
Short Description	Allows drivers to set whether the bus is currently on break, available, or Not available.
Actor(s)	Bus Driver
Pre-condition(s)	The bus driver is logged into the system.
Post-condition(s)	The operational status is updated successfully.
Main Flow	1. Bus driver logs into the system. 2. Selects the “Bus Status” from the sidebar. 3. Set the start date, end date, start time, end time, and availability status, which can be set to On Break, Available, or Not Available. 4. Save the bus status by clicking ‘Update’ button. 5. The system updates the bus status in real-time for users.
Alternative Flow	N/A
Exception Flow	N/A

3.4.2.7 Bus Driver Can Update Bus Operational Mode

3.4.2.7.1 Stimulus/Response Sequence

Table 3.13: Response Sequence of update.

Use Case	Update Bus Operational Mode
Short Description	Allows drivers to mark the start and end of their shifts, so students receive updates on whether the bus driver has begun their journey or completed their duty for the day.
Actor(s)	Bus Driver
Pre-condition(s)	The bus driver is logged into the system.
Post-condition(s)	The bus operational mode is updated successfully.
Main Flow	1. Bus driver logs into the system. 2. Selects the “Operational Mode ” from the sidebar. 3. The bus driver selects the starting point (bus stop) of their route. 4. The bus driver taps the "Start Shift" button to begin their shift, and the "End Shift" button when they complete their service. 5. A summary report of the trip is generated.

Alternative Flow	N/A
Exception Flow	N/A

3.4.2.10 Bus Driver Can View Notifications

3.4.2.10.1 Stimulus/Response Sequence

Table 3.14: Response Sequence of view notifications.

Use Case	View Notifications.
Short Description	Notify bus driver regarding updates they made on the bus status, new bus schedule upload and start and end of their operational time.
Actor(s)	Bus Driver
Pre-condition(s)	Notifications are enabled for bus drivers.
Post-condition(s)	Notifications viewed and acknowledged by the driver.
Main Flow	1. Bus driver logs into the app. 2. Selects "Notification" from navigation bar 3. System displays notifications regarding updates they made on the bus status, new bus schedule upload and start and end of their operational time.
Alternative Flow	N/A
Exception Flow	N/A

3.4.3 Class Diagram

The class diagram in Figure 3.26 show the key components and their relationships within the Bus Tracker App. It provides a high-level view of the system's structure and helps in understanding how different parts interact with one another (Geeksforgeek, 2025).

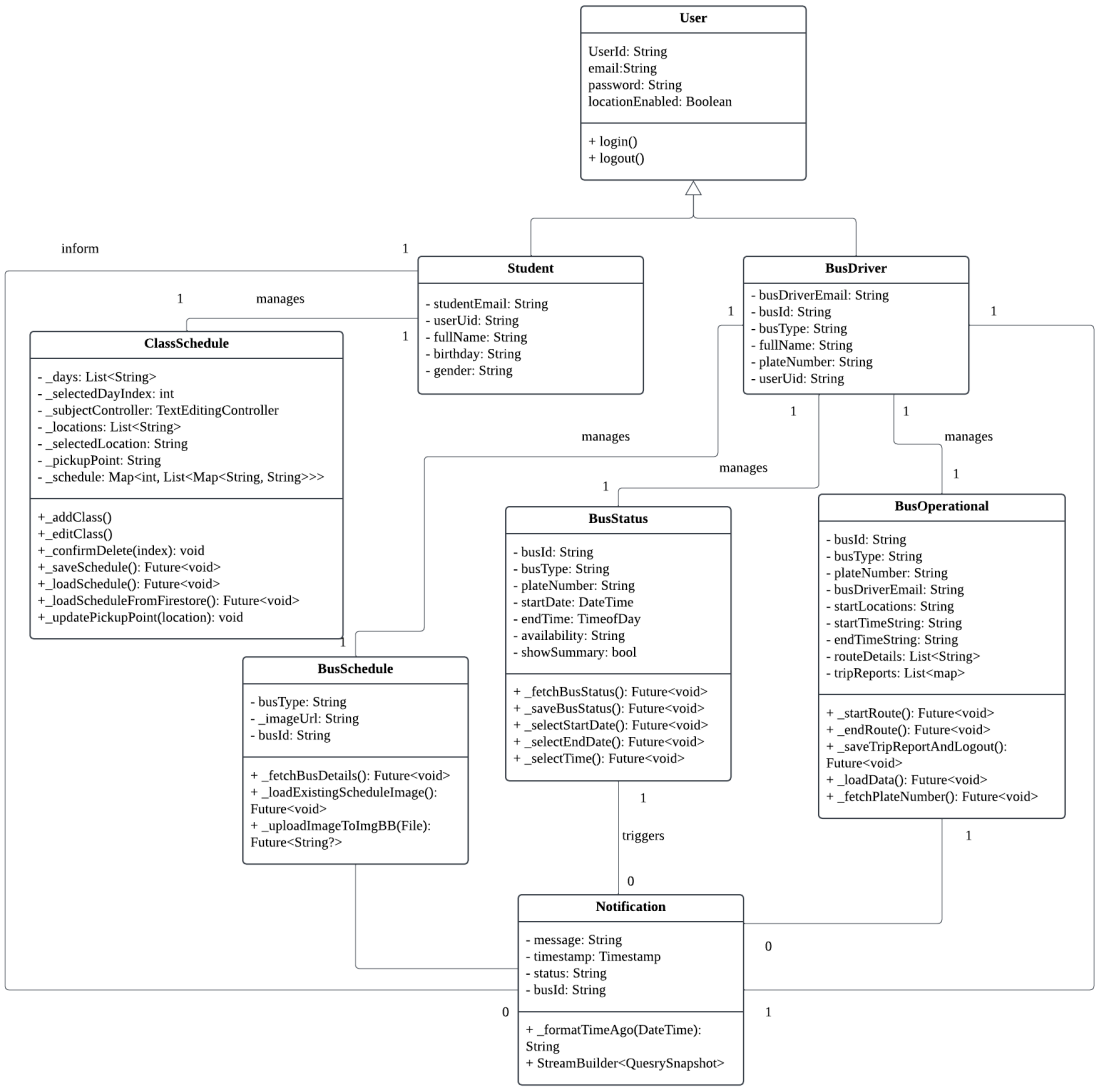


Figure 3.25: Class Diagram

Based on the Figure 3.25, there are 8 classes. The User class represents system users which are categorized into two subclasses, students and bus drivers. The BusSchedule class manages uploaded bus schedule images associated with each bus type. The ClassSchedule class stores each student's class schedule and determines the nearest pickup points based on class locations. The BusStatus class tracks the bus availability and operational state of each bus. The BusOperational class handles real-time operational details such as trip timing, routes, and shift information. Finally, the Notification class manages notification messages for both students and bus drivers.

3.4.4 Sequence Diagram

Sequence diagrams represent the flow of interactions between actors and the system for specific use cases (Creately, 2022). These diagrams illustrate the system's behavior, including data flow and system responses to user actions. The order of operations is highlighted to ensure a better understanding of system functionality and communication between components.

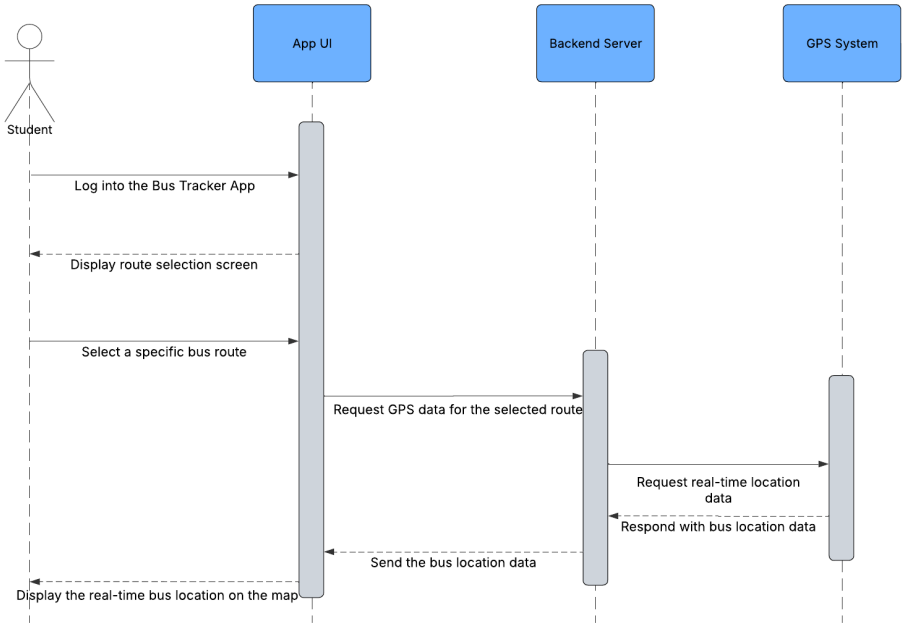


Figure 3.26: Sequence Diagram for View Real-Time Bus Locations

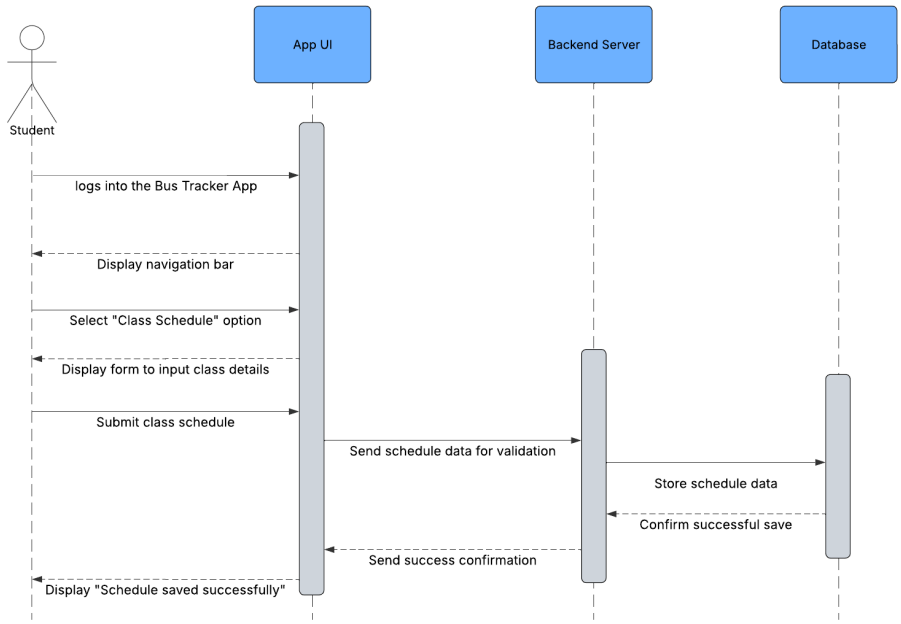


Figure 3.27: Sequence Diagram for Input Class Schedule

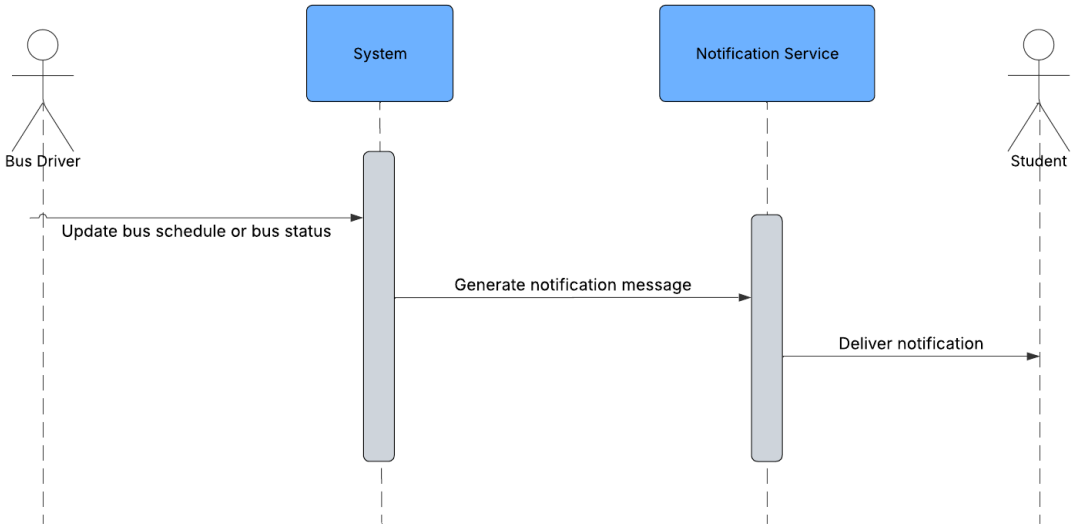


Figure 3.28: Sequence Diagram for Receive Notifications

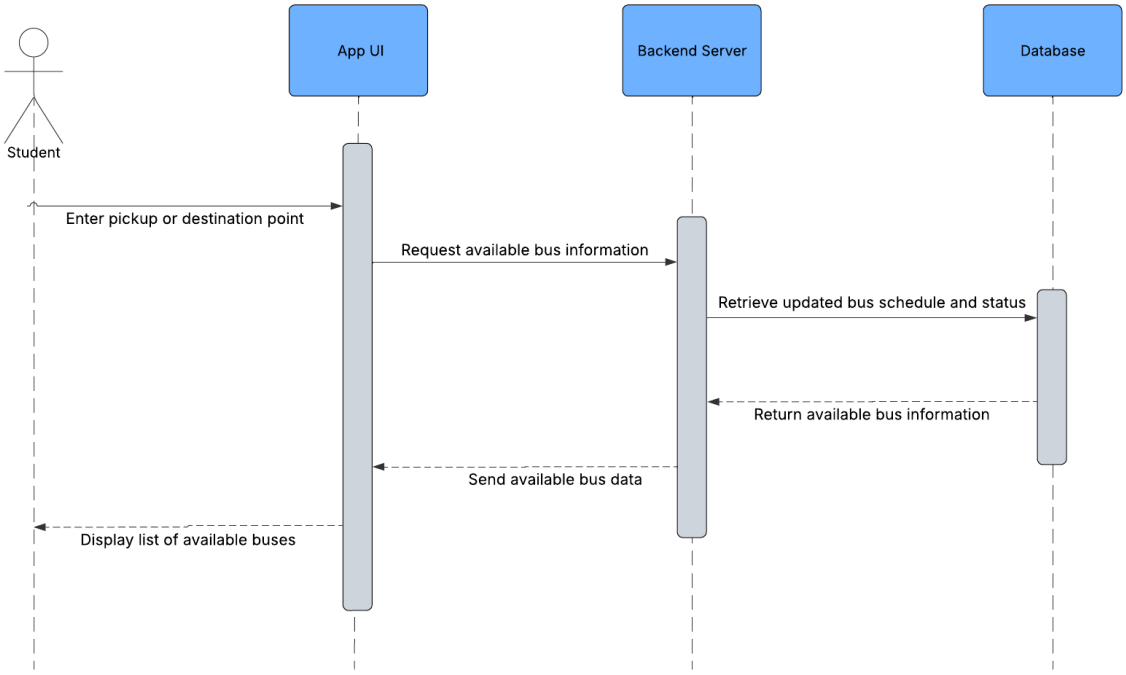


Figure 3.29: Sequence Diagram for View Bus Availability

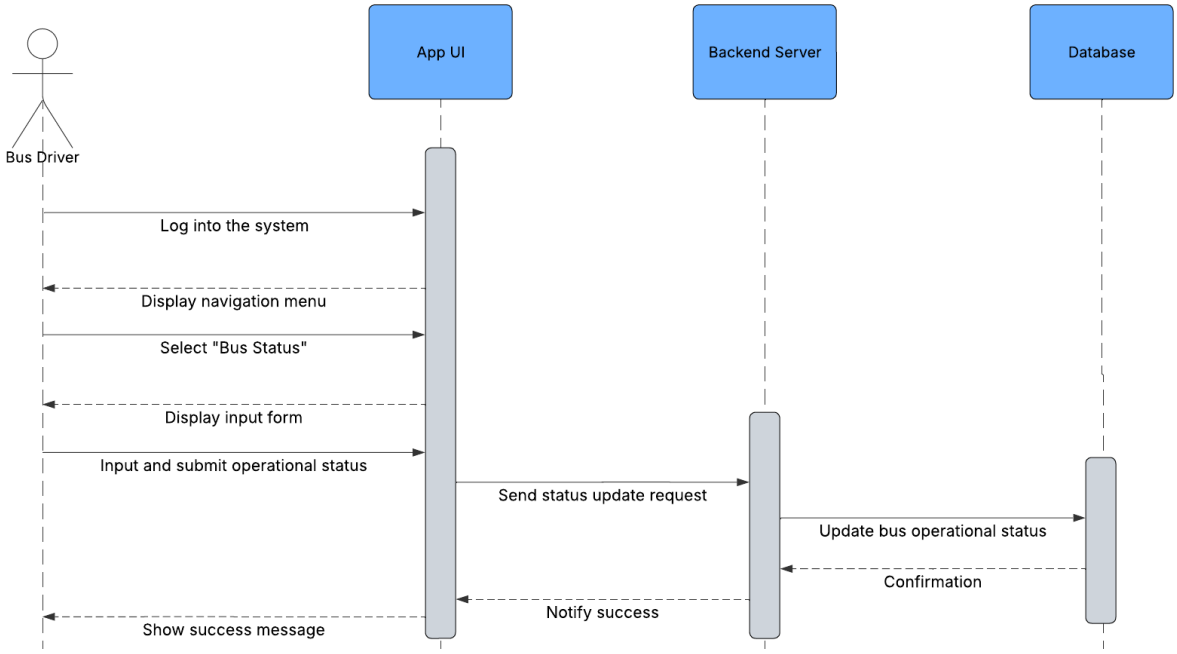


Figure 3.30: Sequence Diagram for Update Bus Status

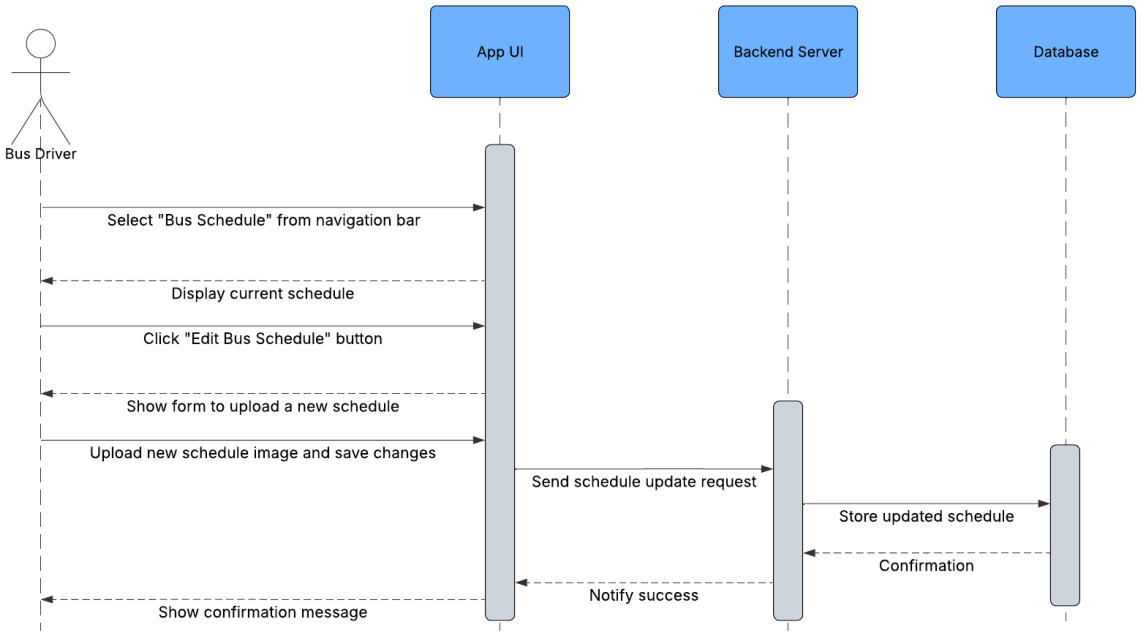


Figure 3.31: Sequence Diagram for Update Bus Schedule

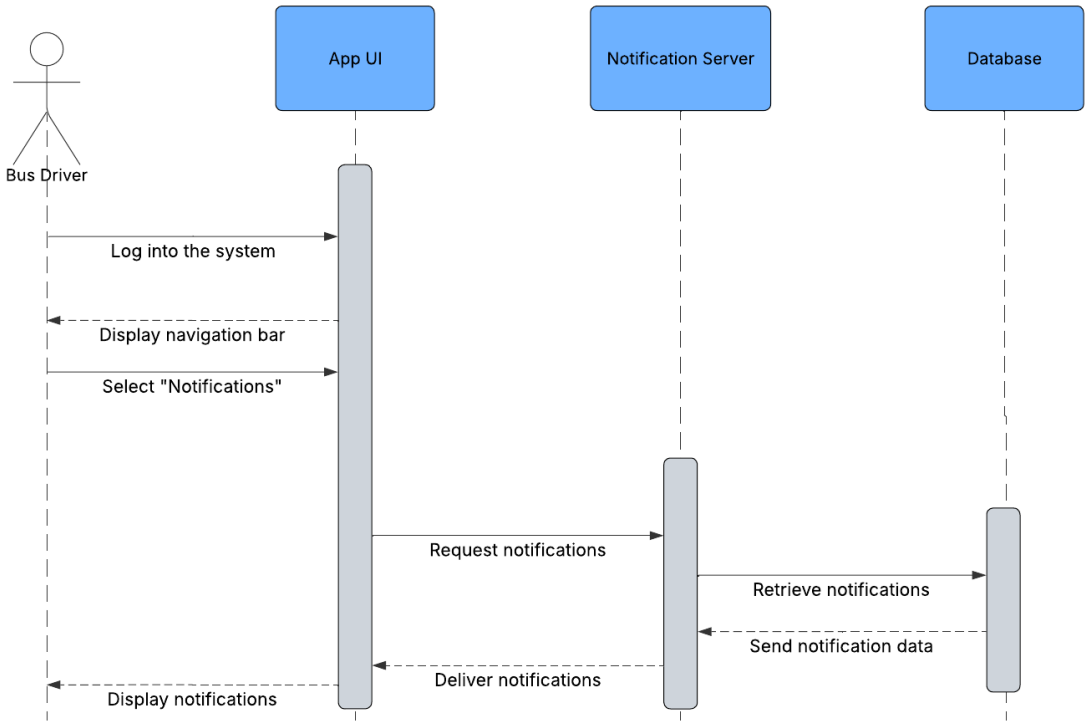


Figure 3.32: Sequence Diagram for View Notifications

3.4.5 Component Diagram

The component diagram in Figure 3.33 illustrates the structural components of the application and their interactions in the bus tracking system (Nishadha, 2025). It shows how system modules, data storage, services, and external dependencies are organized.

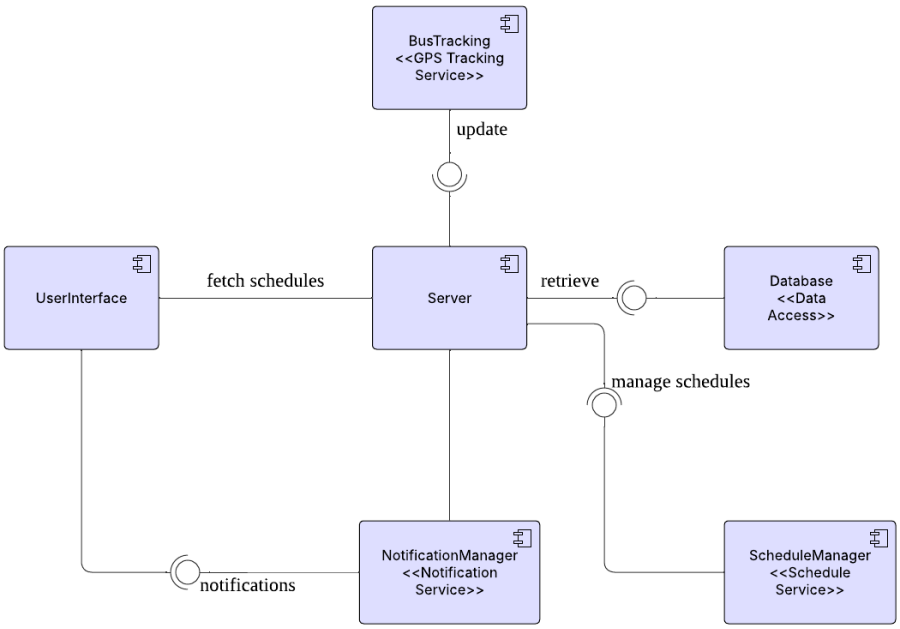


Figure 3.33: Component Diagram

Based on the Figure 3.33:

- The User Interface Component represents the graphical interface for interaction with students and bus drivers. It handles tasks such as real-time tracking, schedule viewing, and route selection.
- The Server Component acts as the central controller that processes user requests, fetches schedule information, and manages data interactions between the database and the user interface.
- The Database Component stores system data, including bus schedules, route information, favourite routes, and operational statuses.
- The Bus Tracking Component processes and retrieves real-time bus location data.
- The Notifications Manager Component provides notifications about bus availability, schedule changes, and delays. It connects to the user interface component to display these notifications.
- The Schedule Manager Component manages and updates the bus schedules and operational modes, with changes input by bus drivers.

3.4.6 State Machine Diagram

The state machine diagram in Figure 3.34 visualizes the dynamic behaviour of the bus tracking system from a user's perspective (Visual Paradigm, 2024). It shows transitions between different states based on user actions and system events.

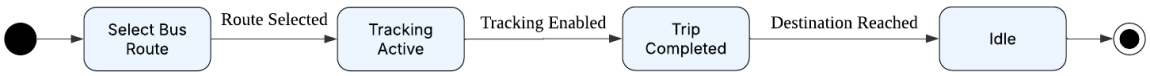


Figure 3.34: State Machine Diagram

In the diagram in Figure, the initial state represents when the system is ready for user interaction. The user starts tracking by selecting a bus route and viewing real-time tracking. Once the bus reaches the pickup point, the system updates the display to show that the bus has arrived at the destination. The Idle State occurs when the user ends the session, and there is no active interaction. The Final State represents the completion of user interaction when the user exits the app.

3.5 Database Design

1. Users Table

Table 3.15: Users Table

Field Name	Data Type	Size	Description	Example
UserID	STRING	28	Unique ID for each user from firebase authentication	VGXOJC0BmgY3Hc GULomMUJOeEWy2
Email	STRING	50	User's email address	12345@siswa.unimas.my (student) or john@gmail.com(bus driver)
Password	VARCHAR	50	Encrypted user password	********

2. Bus Drivers Table

Table 3.16: Bus Drivers Table

Field Name	Data Type	Size	Description	Example
UserID	STRING	28	Unique ID for each user from firebase authentication	VGXOJC0BmgY3Hc GULomMUJOeEWy2
BusId	STRING	10	Unique ID for each bus driver	BS01 (UNIMAS Shuttle Bus) PB01 (PanBorneo Bus)
BusType	STRING	20	Type of bus: PanBorneo or UNIMAS Shuttle	PanBorneo
Email	STRING	35	Bus driver's email address	john@gmail.com
FullName	STRING	25	Bus driver full name	John Doe
PlateNumber	STRING	8	Bus plate number	QCR 9217 (Bus A)

3. Bus Driver Notifications Table

Table 3.17: Bus Driver Notifications Table

Field Name	Data Type	Size	Description	Example
BusNotificationsId	STRING	20	Unique ID for each Notification	37R7Dj2bay4w28JBLFp
BusId	STRING	11	Unique ID for each bus driver	PB01
Message	STRING	100	Message about updated bus status or bus schedule changes or start & end of operational time.	PB01 bus status was updated Bus PB01 is now on duty. The journey has just started.
Status	STRING	10	Indicator icon to define status of bus driver operational time	Active
Timestamp	TIMESTAMP	30	Date and Time	Date : 14 Jun 2025 Time: 09:38:04 PM

4. Bus Schedules Table

Table 3.18: Bus Schedules Table

Field Name	Data Type	Size	Description	Example
BusType	STRING	20	Type of bus: PanBorneo or UNIMAS Shuttle	PanBorneo
ScheduleImageUrl	STRING	58	Image URL that save in ImgBB	https://i.ibb.co/ TMSFMxzk/IMG-20250621-

				WA0025.jpg
UploadedAt	TIMESTAMP	20	Date and Time	Date : 14 Jun 2025 Time: 09:38:04 PM

5. Bus Status Table

Table 3.19: Bus Status Table

Field Name	Data Type	Size	Description	Example
UserID	STRING	28	Unique ID for each user from firebase authentication	VGXOJC0BmgY3Hc GULomMUJOeEWy2
BusId	STRING	11	Unique ID for each bus driver	PB01
BusType	STRING	20	Type of bus: PanBorneo or UNIMAS Shuttle	PanBorneo
PlateNumber	STRING	8	Bus plate number	QCR 9217
StartTime	TIMESTAMP	30	Start time of the bus status	1 June 2025 at 14:00:00 UTC+8
EndTime	TIMESTAMP	30	End time of the bus status	10 June 2025 at 14:00:00 UTC+8
StartDate	TIMESTAMP	30	Start date of the bus status	1 June 2025 at 14:00:00 UTC+8
EndDate	TIMESTAMP	30	End date of the bus status	10 June 2025 at 14:00:00 UTC+8
Availability	STRING	20	On Break, Available or Not Available	Not Available

6. Class Schedules Table

Table 3.20: Class Schedules Table

Field Name	Data Type	Size	Description	Example

UserID	STRING	28	Unique ID for each user from firebase authentication	VGXOJC0BmgY3HcGULomMUJOeEWy2
Days (Mon, Tue, Wed, Thu, Fri)	Array	3	Mon, Tue, Wed, Thu, Fri	Mon
Subject	STRING	10	Course Name	Malay
StartTime	STRING	10	Start time of the class	11:00 AM
EndTime	STRING	10	End time of the clas	01:00 PM
Pickup	STRING	20	Nearest pickup point to the class location	BHEP

7. Driver Location Table

Table 3.21: Driver Location Table

Field Name	Data Type	Size	Description	Example
BusId	STRING	11	Unique ID for each bus driver	PB01
BusType	STRING	20	Type of bus: PanBorneo or UNIMAS Shuttle	PanBorneo
Latitude	NUMBER	18	Date when the status change occurred.	1.4722270537272948
Longitude	NUMBER	18	Time when the bus becomes available or starts the break	110.42881402813717

8. Operational Table

Table 3.22: Operational Table

Field Name	Data Type	Size	Description	Example

BusId	STRING	11	Unique ID for each bus driver	PB01
BusType	STRING	20	Type of bus: PanBorneo or UNIMAS Shuttle	PanBorneo
PlateNumber	STRING	8	Bus plate number	QCR 9217 (Bus A)
TripStartTime	TIMESTAMP	30	Start time of shift	10 June 2025 at 14:00:00 UTC+8
TripEndTime	TIMESTAMP	30	End time of shift	10 June 2025 at 14:00:00 UTC+8
CreatedAt	TIMESTAMP	30	Day and time the trip created	10 June 2025 at 14:00:00 UTC+8

9. Student Notifications Table

Table 3.23: Student Notifications Table

Field Name	Data Type	Size	Description	Example
Student NotificationId	STRING	20	Unique ID for each notification	37R7Dj2bay4w28JBLFp
Message	STRING	25	Message about updated bus status or bus schedule changes or start & end of operational time	PanBorneo Bus (PB01) is now active and on the move. PanBorneo Bus (PB01) has ended its service for today.
Timestamp	TIMESTAMP	30	Date and Time	Date : 14 Jun 2025 Time: 09:38:04 PM

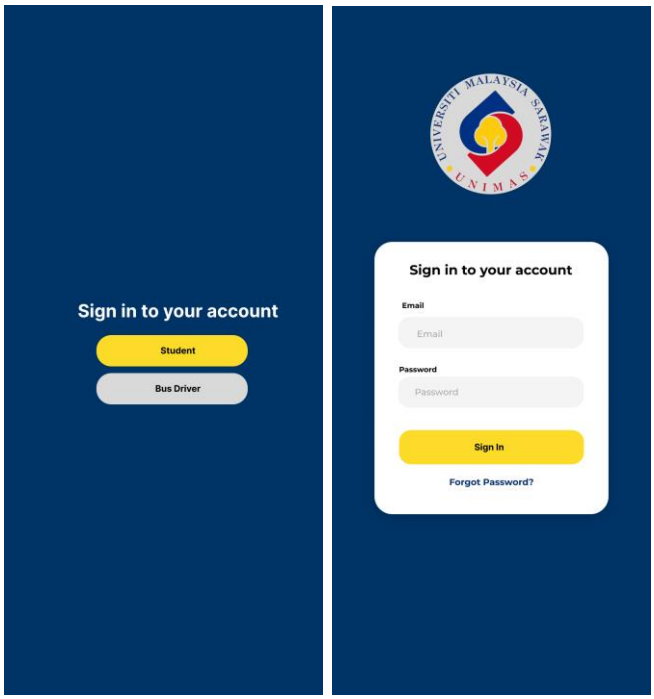
10. Students Table

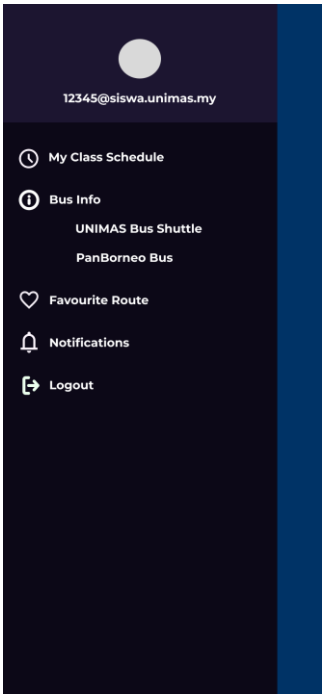
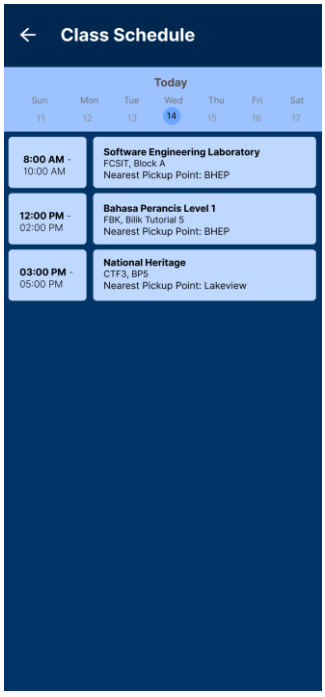
Table 3.24: Students Table

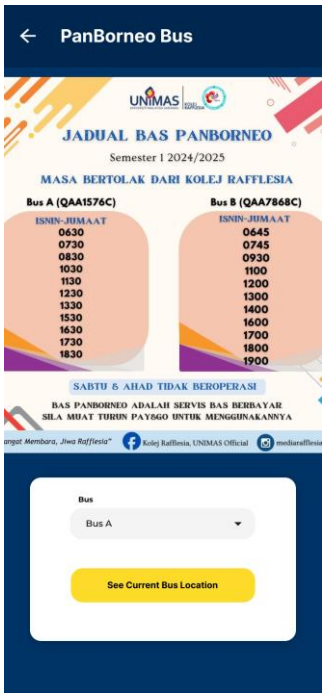
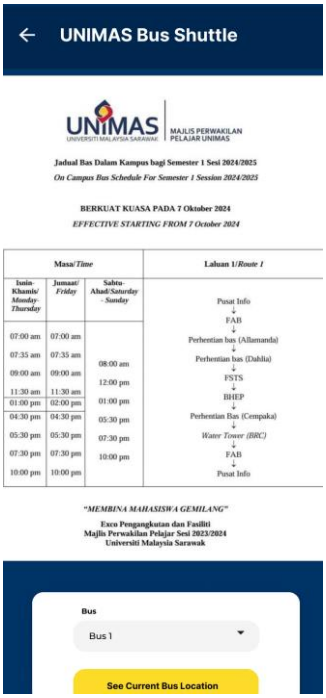
Field Name	Data Type	Size	Description	Example
UserID	STRING	28	Unique ID for each user from firebase authentication	VGXOJC0BmgY3Hc GULomMUJOeEWy2
Email	STRING	35	Student's email address	12345@siswa.unimas.my
FullName	STRING	25	Student full name	John Doe

3.6 User Interface Design Bus Tracker UNIMAS Mobile Application

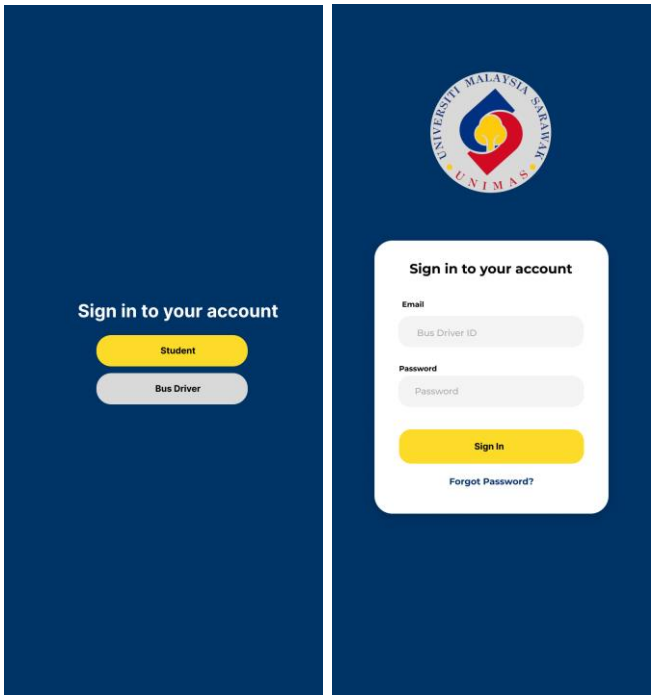
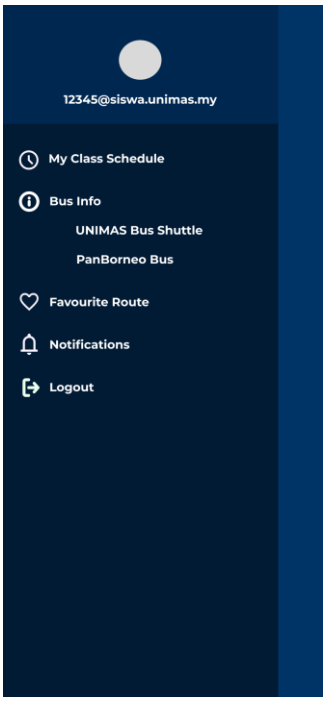
3.6.1 Student Page Interface Design

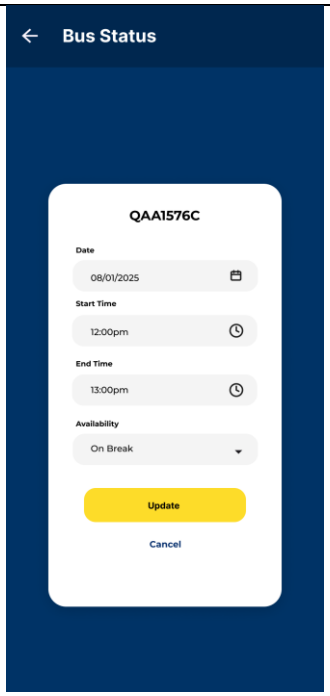
Page Interface	Page Description
	The first page is the starting page where users can sign in to their account by selecting their user type: student or bus driver. The second page is the sign-in page. Students need to enter their email and password to log into the app.
	The next page is the main navigation page, where users can access the map, select a pickup point, and set a destination. Users need to input their pickup and destination points. Additionally, they can select a time or use filter options to refine their search. Based on the inputs, users can view the recommended buses or all available buses.

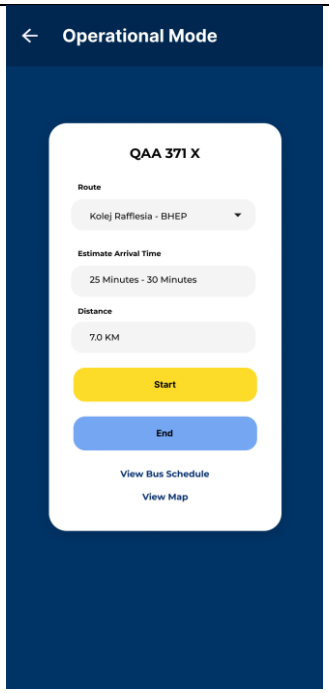
	The student interface has a navigation bar with five sections: My Class Schedule, Bus Info, Favourite Route, Notifications, and Log Out.
	The student can input their class schedule in the "My Class Schedule" section. By selecting the time and date of the class, the student needs to enter the class name and the corresponding class location.

 	The Bus Info section provides details on two bus services: PanBorneo Bus and UNIMAS Bus Shuttle. In this section, users can view the bus schedule and check the current location of their preferred bus by selecting it from a dropdown menu and click the 'See Current Bus Location' button.
	The Notifications page displays alerts and reminders for users, such as updates about bus availability, bus information, delays or schedule changes.

3.6.2 Bus Driver Page Interface Design

Page Interface	Page Description
	The first page is the starting page where users can sign in to their account by selecting their user type: student or bus driver. The second page is the sign-in page. Bus drivers need to enter their Bus Driver ID and password to log into the app.
	The bus driver interface has a navigation bar with five sections: Bus Schedule, Bus Status, Operational Mode, Notifications, and Log Out.
	The Bus Schedule page allows the bus driver to update the bus schedule by clicking the edit button and

	uploading an updated image of the schedule.
	The Bus Status page requires the bus driver to update the operational status of the bus. The driver needs to select the date, start and end time, and availability status, which can be set to On Break, Available, or Not Available.
	The Operational Mode page is where the bus driver updates the operation details. The driver selects the route they will be driving, and the estimated arrival time and distance for the route will automatically be displayed based on the selected route.

	After selecting the route, the driver must click the 'Start' button to indicate that they have started driving and click the 'End' button upon arriving at the destination. Additionally, the driver can view the bus schedule or map by clicking the respective options below the 'End' button.
	The Notifications page displays alerts and reminders for the bus driver, such as updates or alerts about traffic jams.

3.7 Summary

This chapter highlights the requirements for the Bus Tracker app to achieve the objectives stated in Chapter 1, by designing the user interface and system architecture. Additionally, this chapter explains the methodology that utilizes an agile approach for the app. The chosen methodology facilitates flexible and adaptive development. Furthermore, the system architecture and user interface of the app are described with figures for better understanding.

CHAPTER 4: IMPLEMENTATION

4.1 Introduction

This chapter will cover the implementation of Bus Tracker Mobile Application. The app will be named UniBus for simplicity. While the Chapter 3 focused on requirement analysis and design the app, the implementation phase led to some differences between the initial design and the final product. These differences arose due to various technical constraints, challenges, and improvements made during the development process. The chapter will detail the technologies that involved in the development, including the programming languages, frameworks, and development tools. Additionally, it will explain the file structure and code snippets. The challenges faced during the development process will also be discussed. This implementation phase is essential for validating the design by turning theoretical concepts into a practical and functioning application.

4.2 Technologies Used

4.2.1 Programming Languages and Frameworks

For the development of this UniBus App, Dart was chosen as the programming language because it is designed to work well with Flutter. Dart offers advantages, such as Dart support for asynchronous programming using async and await which allows for efficient handling of tasks such as fetching and updating real-time data from GPS, ensuring that users receive accurate and timely bus tracking information.

The framework for this project is Flutter due to its features that align with the UniBus app's requirements. Flutter has capabilities that, when combined with Firebase, make it a strong choice for handling real-time updates in the UniBus app. Flutter also provides an efficient and highly customize way to create visually interactive UIs. This is important for creating a user-friendly and offers engaging experience, especially when dealing with map interactions and dynamic data.

4.2.2 Development Tools

In this project, Android Studio was chosen as the Integrated Development Environment (IDE) for developing the UniBus app. Android Studio is specifically designed to support Android development and provides complete tools for coding, debugging, and testing. It offers support for the Dart programming language and the Flutter framework used in this project, making it a preferred choice for building UniBus app.

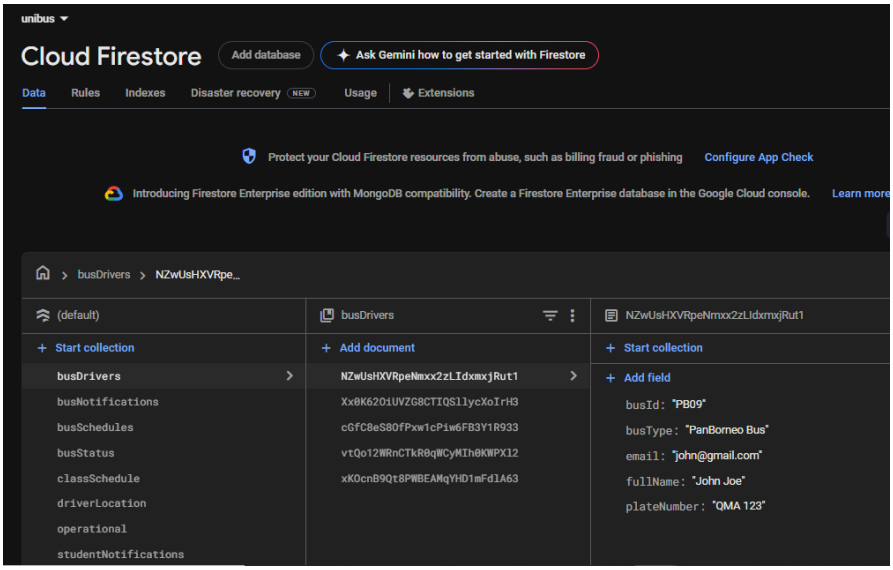


Figure 4.35: Firebase

For backend services, Firebase was selected to handle the app's real-time database and user authentication. The easy integration between Firebase with Flutter ensures efficient handling of backend connectivity and scalability. This is important for the UniBus app, which requires real-time updates, such as bus locations, bus status and bus schedules, and needs to ensure secure user authentication for both students and bus drivers. Based on Figure 4.35, it shows the Firestore that is used for storing and managing various types of data within the UniBus app. In addition to Firestore, Firebase Authentication is used to manage user login and authentication, ensuring that only authorized students and bus drivers can access the app.

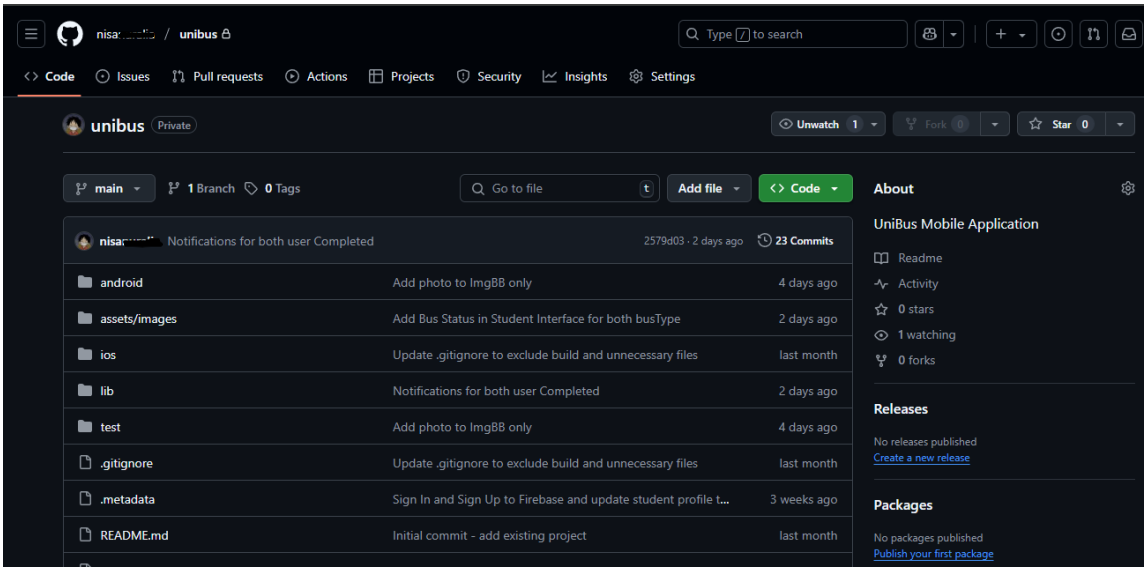


Figure 4.36: GitHub

For version control, in this project GitHub was used to manage and track changes to the UniBus app. GitHub is a widely used platform for software development, which offering powerful version control capabilities with Git. Figure 4.36 show overview of GitHub used for UniBus app. GitHub allows for efficient tracking of every change made to the UniBus project over time. This is very helpful especially to revert to previous versions of the codes when debugging and identifying any issues. The features like commit messages and history in GitHub makes it's easy to understand what was changed, when, and why. Besides, storing code in Github provides better backup and security because it safely stored in the cloud. Even if something happened to the hardware, the code remains accessible from the cloud.

4.2.3 Other Tools and Libraries

In the UniBus app, ImgBB is used as an image hosting service specifically to handle bus schedule images uploaded by bus drivers. Flutter apps cannot directly store images in Firestore, only URLs or metadata can be stored. UniBus integrates ImgBB's REST API to upload image files and obtain the public image URL. Then this URL is saved to Firestore under the respective busSchedules collection, making it accessible for both the bus driver and student to view the bus schedule. The Figure 4.37 shows the overview of ImgBB hosting service.

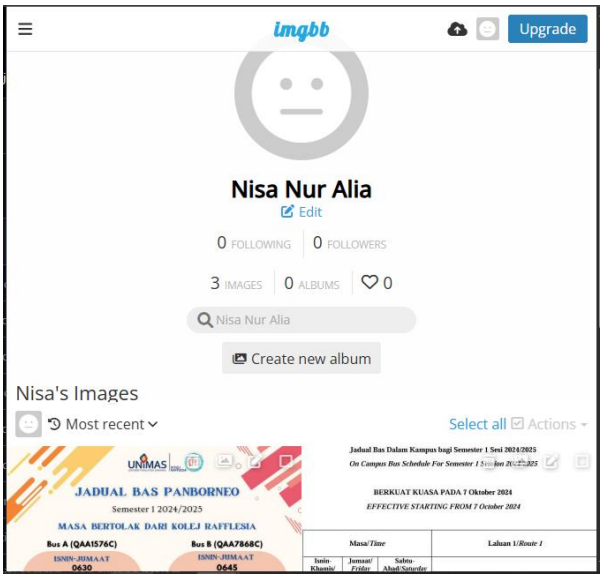


Figure 4.37: ImgBB

For location tracking and map visualization, the Google Maps API was used to provide interactive maps and location services within UniBus app. This integration allows users to view the map, and track bus locations and view markers. The Directions API is used to display the route on the map and provide data for calculating the Estimated Time of Arrival (ETA) and estimated walking time. Additionally, this project uses various third-party packages to enhance functionality and development efficiency, such as Firebase packages, map, location functionalities, and utility packages. Based on Figure 4.38, the dependencies listed in the pubspec.yaml file are shown.

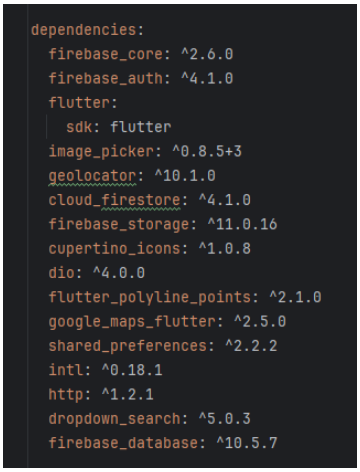


Figure 4.38: Dependencies in pubspec.yaml

Firebase-related packages:

- firebase_core: Initializes Firebase for Flutter app.
- firebase_auth: Provides user authentication services.
- cloud_firestore: Cloud NoSQL database for storing app data.
- firebase_database: Real-time database services.

Mapping and Location packages:

- google_maps_flutter: For integrating Google Maps in UniBus app.
- geolocator: For getting device GPS location and handling location permissions.
- flutter_polyline_points: Helps draw routes on the Google Map.

Other utility packages:

- image_picker: For selecting images from the gallery.
- shared_preferences: For storing simple key-value data locally.
- dio: For making HTTP network requests.
- http: For making HTTP network requests.
- dropdown_search: Provides a UI widget for searchable dropdown lists.
- intl: For internationalization and date/time formatting.
- cupertino_icons: Icon set for iOS style.

4.3 Implementation of Complete Mobile Application

4.3.1 Sign Up Page

All users must sign up for an account before using the app. During the sign-up process, users are required to select their role, either Student or Bus Driver. In the sign-up page the student needs to provide their email and password, while for bus drivers they need to provide their email, password, bus ID and bus type, which can be either UNIMAS Shuttle Bus or PanBorneo Bus. The email and password entered by users are processed through Firebase Authentication to verify their identity and generate a unique user ID (UID).

Once the sign-up data is successfully submitted, the data is stored in Firebase Firestore. Students' details are saved in the students collection, and bus drivers' details are stored in the busDrivers collection. The separate collections for different users ensure that their data is organized and easily accessible based on their role.

4.3.2 Sign In Page

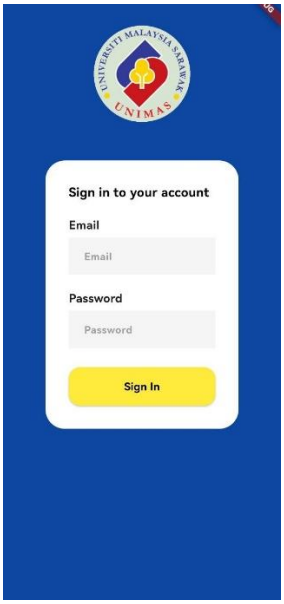


Figure 4.39: Sign in Page

After successfully signing up, users can sign in to their account via the Sign In Page as shown in Figure 4.39. Firebase Authentication checks the entered email and password against the records in the Firebase database to verify the user's credentials. If the email or password is incorrect, the user will receive a pop-up message indicating that the login attempt was unsuccessful due to invalid input.

4.3.3 Student Interface

4.3.3.1 Student Homepage

The map on the homepage as shown in Figure 4.40 displays real-time bus locations, available bus stops, and routes, represented by a orange color polyline. The Google Maps API is used to visualize the bus routes and provide accurate tracking of both the student's and bus's locations. Markers are placed on the map to represent important locations such as the student's location is indicated by a dark blue marker, bus stops are marked in green marker, and the bus driver's location is shown with a red marker. The orange marker indicates the bus stop was selected as pickup and drop point.

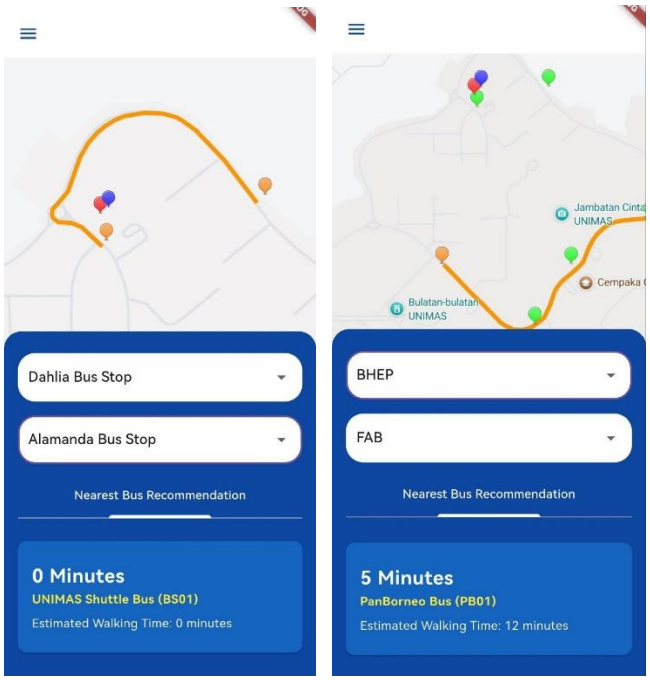


Figure 4.40: Student Homepage

The bus stop locations are predefined in the code, while the bus's real-time location is fetched from Firestore under the driverLocation collection. Using the Geolocator package, the student's current location and the bus's location are continuously updated on the map. The app updates the bus locations whenever the bus moves by at least 5 meters, providing accurate real-time tracking for the student.

Figure show two examples when the pickup point and drop point was selected, the nearest bus recommendation with estimate arrival time of the bus to the selected pickup point,

and the estimated walking time from the current location of the student to the selected pickup point is shown in this page.

The app also includes a dropdown menu for students to select their pick-up and drop-off points from a list of predefined bus stop locations. After both points are selected, the app fetches the route between them using the Google Directions API and displays the path on the map as a polyline. Additional information is also provided such as the estimated time of arrival (ETA), bus type (UNIMAS Shuttle Bus or PanBorneo Bus), bus number, and the estimated walking time from the student's current location to the pick-up point.

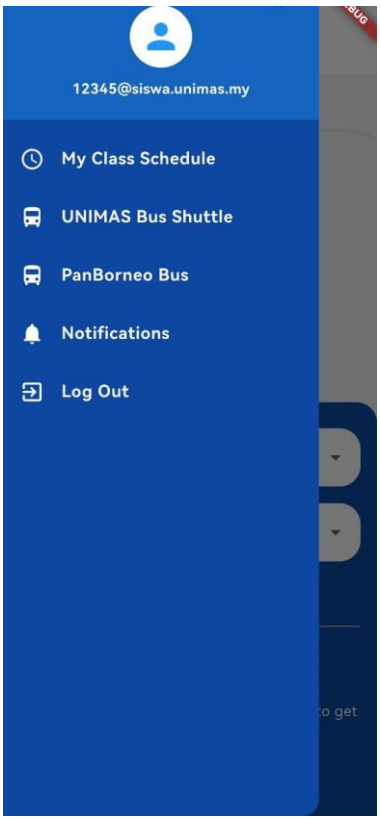


Figure 4.41: Sidebar

The sidebar navigation on the homepage, shown in Figure 4.41, allows easy access to various tabs such as Class Schedule, UNIMAS Bus Shuttle, PanBorneo Bus, and Notifications. The student's email is displayed at the top of the sidebar is used to navigate to profile page.

4.3.3.2 Student Profile Page

Figure 4.42: Student Profile

In the sidebar page, student can navigate to their profile by tapping on the profile picture, or their email address. The student can update their full name, birthday, and gender in this page as shown in the Figure 4.42.

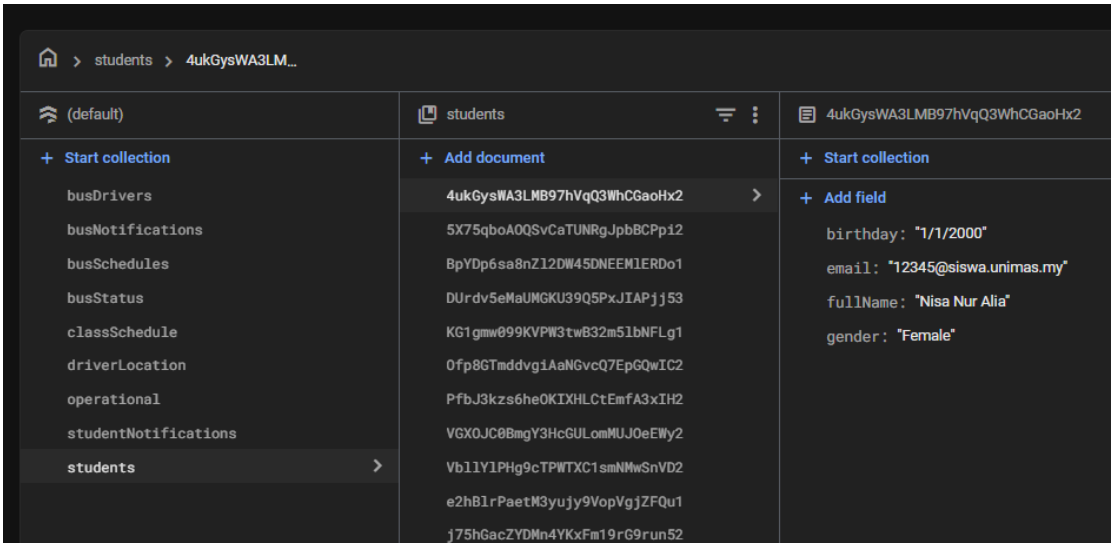


Figure 4.43: Students Collection

The Figure 4.43 show the updated profile data is then stored in Firestore under the students collection.

4.3.3.3 Class Schedule Page

The My Class Schedule page allows students to manage and view their class schedules. It provides functionalities for adding new classes, editing existing ones, and deleting classes. To add a new class, students need to click the '+' button, set the class time, and fill in the class name and location. Based on the class location, the app will automatically suggest the nearest pickup point for the student after class.

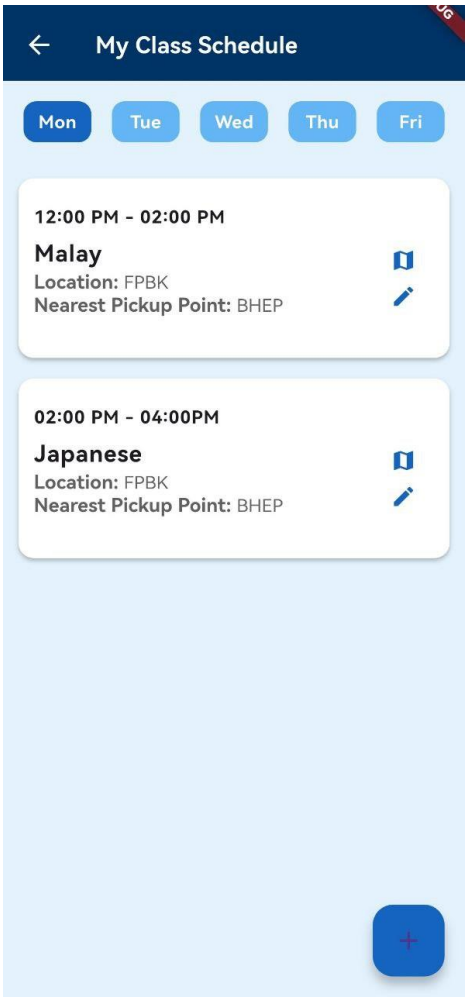


Figure 4.44: Class Schedule

Figure 4.44 illustrates the My Class Schedule page, where students can view class details such as the subject, location, start time, end time, and pick-up point. For each class, students can click the Map button, which will redirect them to the homepage. Based on the personalized pickup point for that class, the pickup point is automatically filled in the dropdown menu, allowing students to view the nearest bus recommendation without manually selecting the pickup point for saving time.

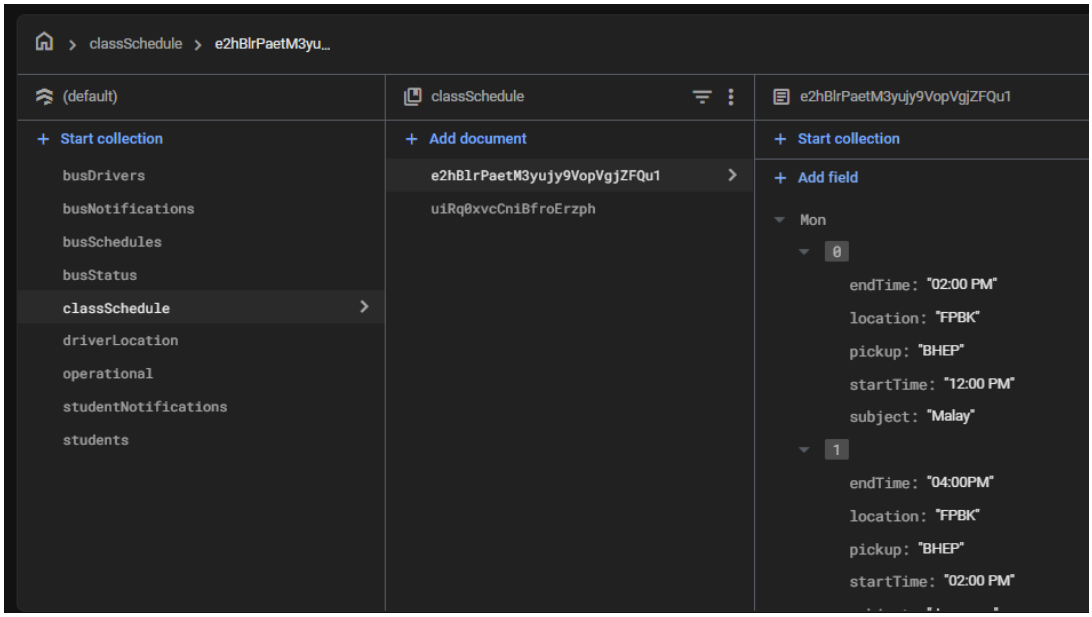


Figure 4.45: ClassSchedule Collection

Figure 4.45 show all class schedule data is stored in Firestore under the classSchedule collection.

4.3.3.4 Bus Services Information Page

The UNIMAS Shuttle Bus Page and PanBorneo Bus Page are designed to allow students to view bus related information. The bus schedule is displayed as an image, and dynamically fetched from a URL provided by the Firestore database.

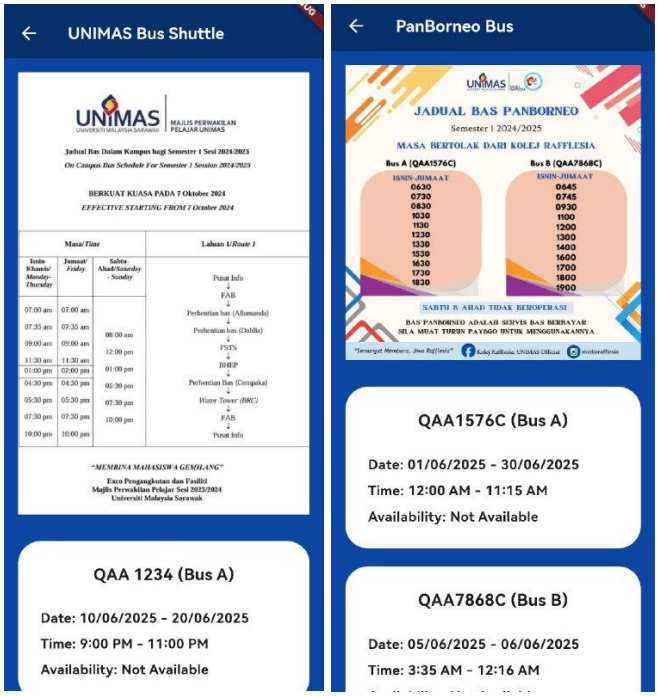


Figure 4.46: UNIMAS Shuttle Bus and PanBorneo Bus Page

Figure 4.46 shows below the bus schedule image, the bus status is displayed, including details such as the bus's plate number, bus number, the date, time, and the availability status of the bus. The availability status can be available, not available, or on break. Figure 4.47 show the information is retrieved from the busStatus collection in Firebase Firestore and presented to the student in real-time.

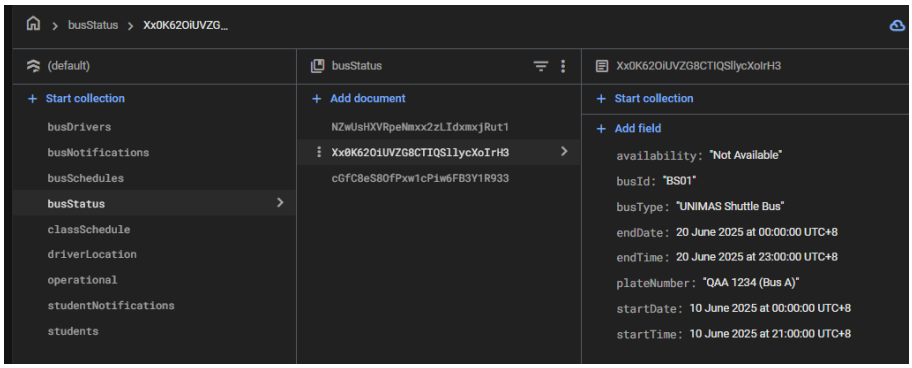


Figure 4.47: BusStatus Collection

These pages ensure that students have up-to-date information about the bus's schedule and status to provide them with essential data for planning their travel.

4.3.3.5 Notification Page

Figure 4.48 shows Student Notifications Page is designed to display important updates to the students. These notifications include bus-related information such as newly uploaded schedules, changes in bus status, and updates on the bus driver's operational hours. The goal of this page is to keep students informed in real-time about any important changes or updates to the bus service.

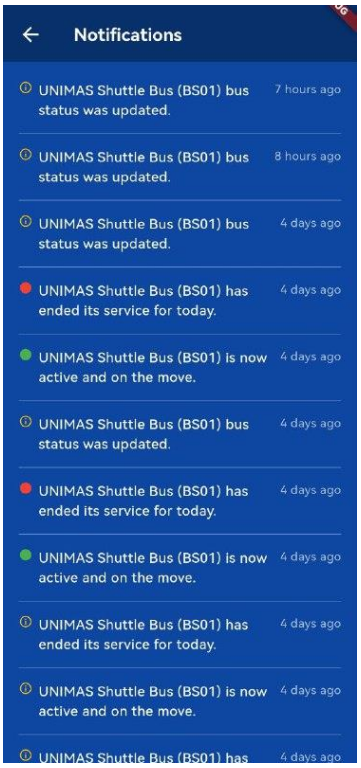


Figure 4.48: Student Notification Page

The notification system uses different indicators, the information icon is used to alerts students about changes in the bus status and bus schedules, the green icon signifies that the bus is active and on the move, while red icon informs students that the bus is inactive and has ended its service for the day. The notification regarding the end of service is important, to ensure that students are aware when the bus service has concluded. This helps students avoid waiting for buses after service hours and encourages them to seek alternative transportation like e-hailing. The notifications data later will be store in studentNotifications collection as shown in Figure 4.49.

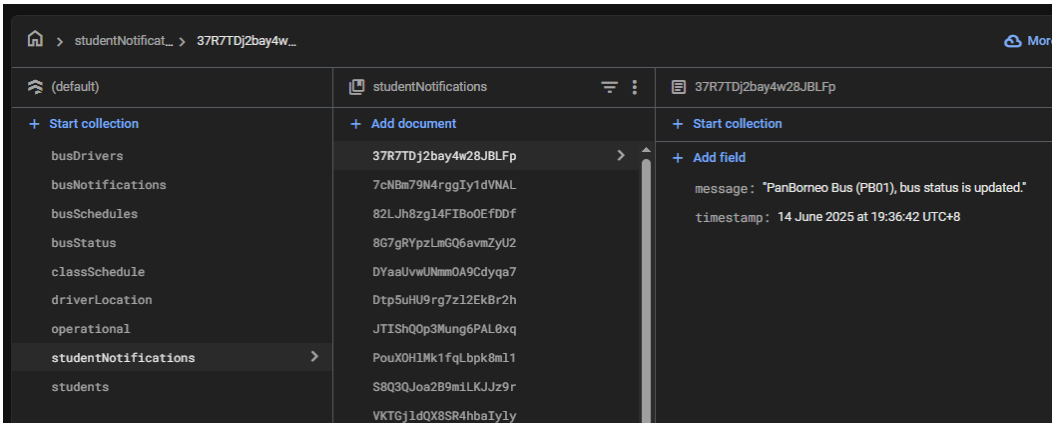


Figure 4.49: StudentNotifications Colletion

4.3.4 Bus Driver Interface

4.3.4.1 Bus Driver Homepage

The map on the homepage displays real-time bus driver locations, available bus stops, and routes. The Google Maps API is used to visualize the bus routes and accurately track the bus driver's location. Markers are placed on the map to represent important locations, with green markers indicating bus stops and a red marker showing the bus driver's location. The bus driver's location is continuously updated on the map using the Geolocator package. Each time the bus moves by at least 5 meters, the bus location is also stored in Firestore under the driverLocation collection. The updated bus location is important to ensure accurate and real-time tracking for all users. When the bus driver starts their journey, the map displays the bus's route to the next bus stop by using the Google Directions API, and also provides the estimated time of arrival (ETA) to the next stop.

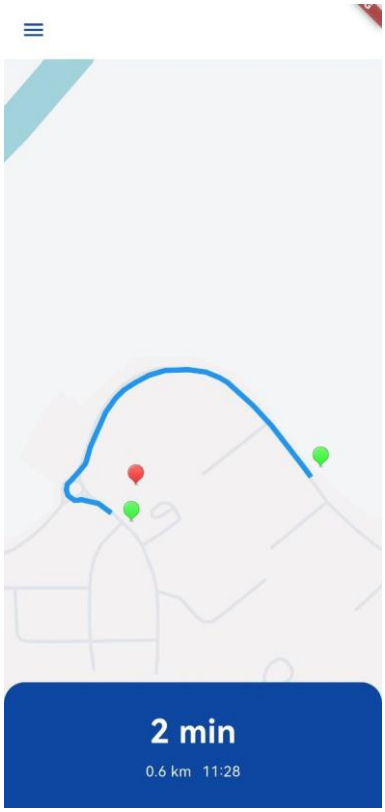


Figure 4.50: Bus Driver Homepage

Figure 4.50 shows the sidebar on the homepage, it provides easy access to various tabs, including the Bus Schedule, Bus Status, Operational Mode, and Notifications tabs. The student's email is displayed at the top of the sidebar for navigation to the profile page.

4.3.4.2 Bus Driver Profile Page

In the sidebar, the bus driver can navigate to their profile by tapping on their profile picture or email address. On the profile page, the bus driver can update their full name and bus plate number. However, the email address, bus ID, and bus type cannot be changed.

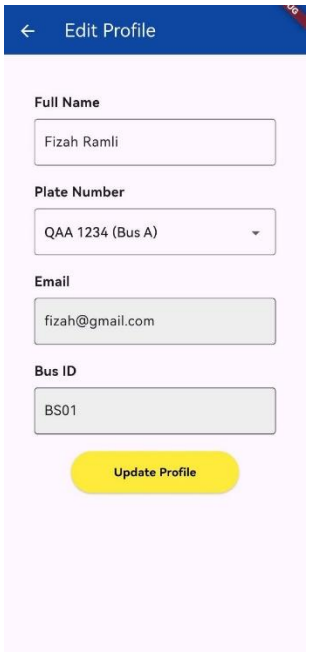


Figure 4.51: Bus Driver Profile Page

The updated profile data is then stored in Firestore under the busDrivers collection which can be review in Figure 4.52.

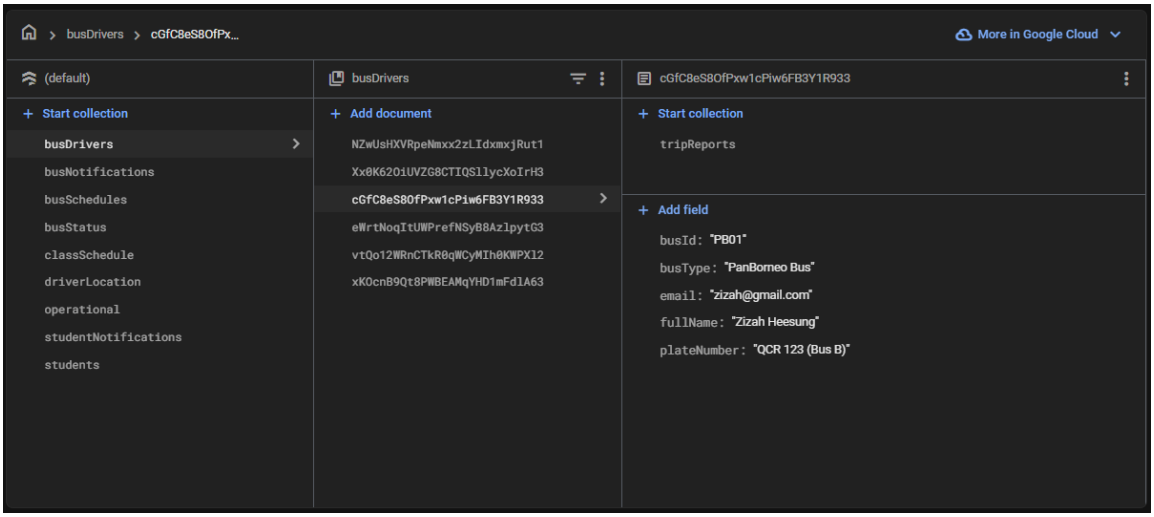


Figure 4.52: BusDrivers Collection

4.3.4.3 Bus Schedule Page

The Bus Schedule Page is for the bus driver to update the bus schedule by uploading a new bus schedule image. The schedule image is uploaded to Firebase Storage, and the URL is saved in Firestore. This allows both the bus driver and students to always have access to the latest bus schedule. Figure 4.53 show the Bus Schedule page which is the page where bus driver can edit bus schedule by tapping the “Edit Schedule" button. The app uses the Image Picker package to allow the bus driver to pick an image from their device's gallery. After the image is selected, the app shows a confirmation dialog asking the bus driver to confirm the upload. The selected image is shown in the dialog, so the driver can verify the schedule before uploading it. Once confirmed, the app uploads the image to Firebase Storage. After the upload, the image URL is retrieved and stored in Firestore under the busSchedules collection. Then, updated schedule image is displayed to the bus driver on Bus Schedule page.

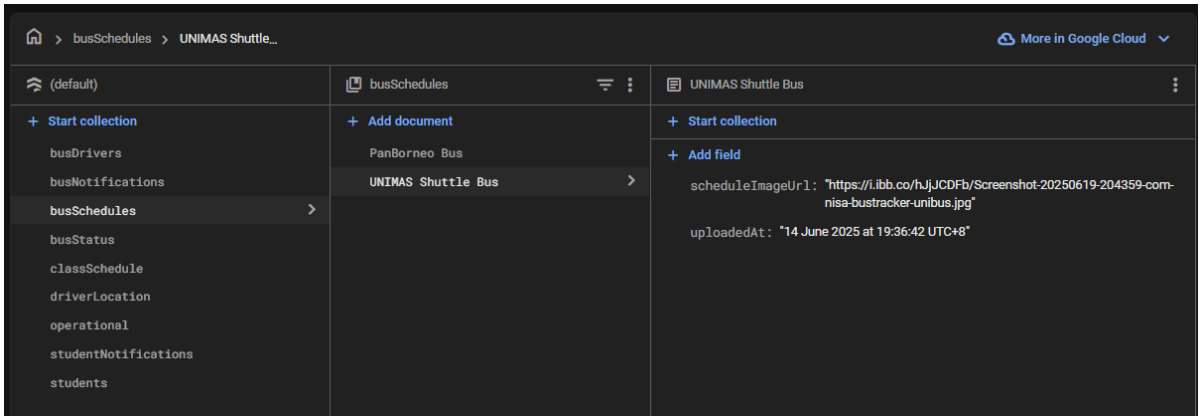


Figure 4.53: BusSchedules Collection

4.3.4.4 Bus Status Page

The Bus Status Page is for the bus driver to update the status of the bus. This feature is important as students often miss important announcements regarding bus availability, especially during holidays and weekends. By updating the bus status, the bus driver ensures that students have access to the most up-to-date information on their page.

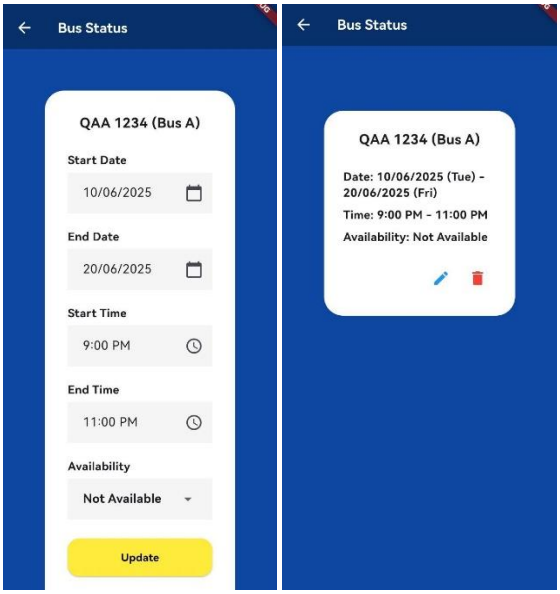


Figure 4.54: Bus Status Page

In Figure 4.54 the form, the bus driver needs to select the start date, end date, start time, and end time for the effective status period. Additionally, the bus driver need to select the availability of the bus, either on break, available, or not available. Once the bus driver clicks the update button, a summary report of the bus status is displayed. This allows the bus driver to review and, if necessary, edit or delete the bus status if there are any changes.

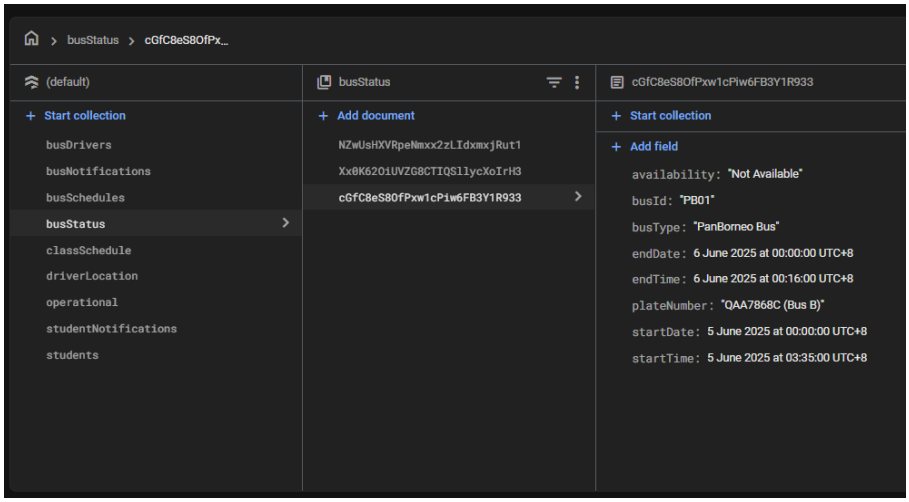


Figure 4.55: BusStatus Collection

All data is stored in Firestore under the busStatus collection which can be review in Figure 4.55.

4.3.4.5 Operational Mode Page

The Operational Mode Page in Figure 4.56 is for the bus driver to indicate the start and end of their shift. By tapping the Start Shift button, the bus driver begins their shift for the day, and by tapping the End Shift button, they indicate the end of their shift.

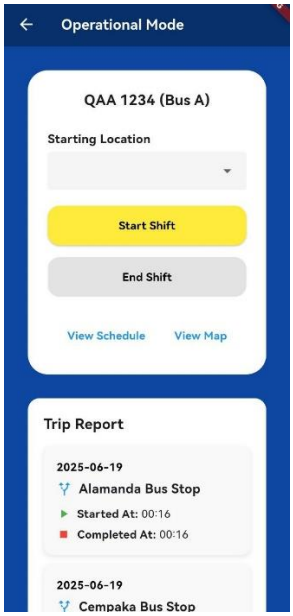


Figure 4.56: Operational Mode Page

Once the shift is completed, a trip report is generated and stored, allowing the bus driver to view their working hours at any time. This page is important because, when the bus driver starts or ends their shift, students receive notifications informing them that the bus is either on the move or has completed its service for the day.

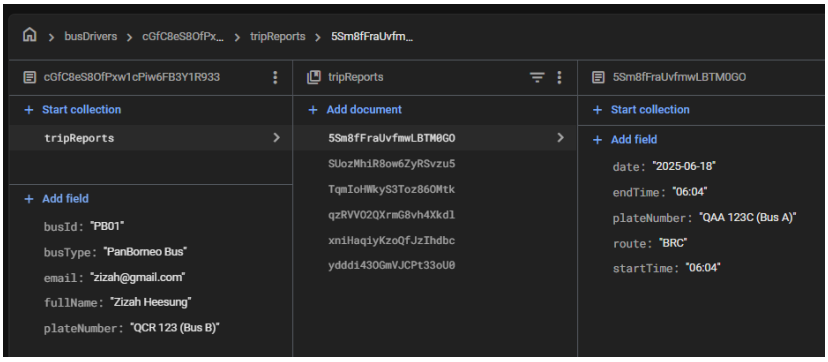


Figure 4.57: TripReports Collection

All data is stored in Firestore under the tripReports collection which can be review in Figure 4.57.

4.3.4.6 Bus Driver Notification Page

The Bus Driver Notifications Page is designed to display updates made by the bus driver. These notifications include bus-related information such as newly uploaded schedules, changes in bus status, and updates on the bus driver's operational hours. The goal of this page is to keep the bus driver informed about the changes they have made.

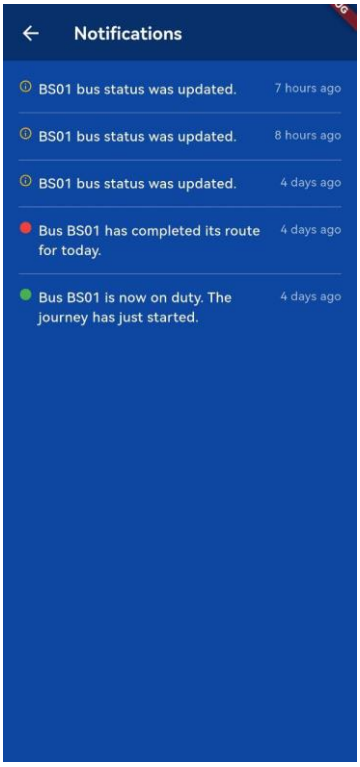


Figure 4.58: Bus Driver Notification Page

Figure 4.58 shows that the notification system uses different indicators such as information icon alerts the driver about changes, they made in the bus status and bus schedules, while the green icon indicates that the bus driver is now on duty and has started their shift and red icon indicates that the bus driver is off duty and has completed their service for the day.

Later all the bus notifications will be store in busNotifications collection as shown I Figure 4.59.

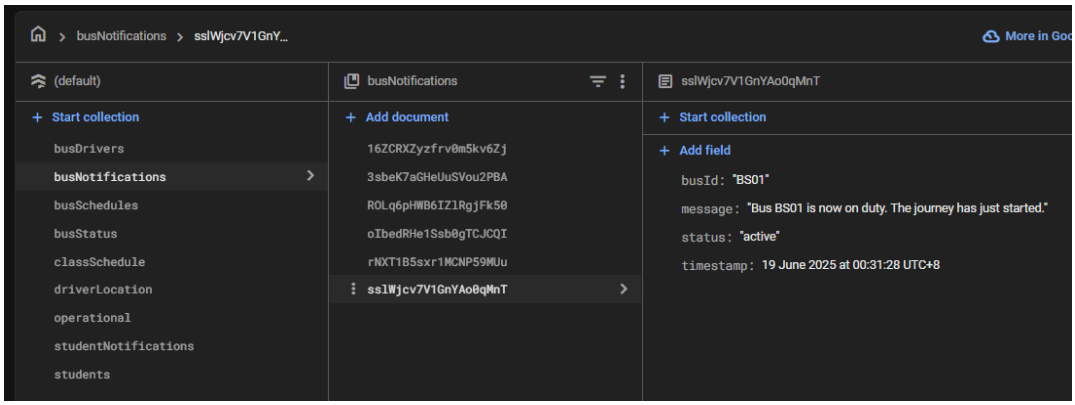


Figure 4.59: BusNotifications Collection

4.4 File Structure

4.4.1 lib Directory

The file structure of the UniBus project follows a standard Flutter project structure. This organization allows for better management, scalability, and ease of development.

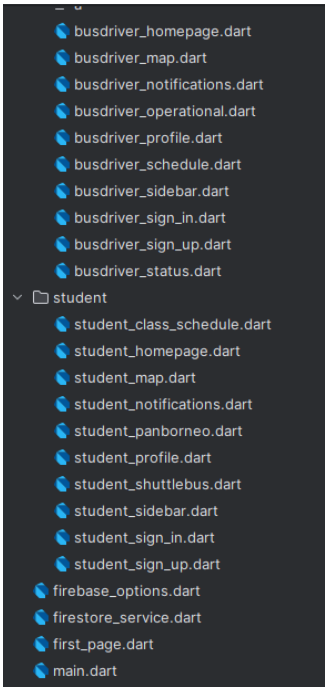


Figure 4.60: lib Directory

Based on Figure 4.60, the lib folder is organized into 2 subfolders, busdriver and student folders, which separate the code for each user role in the UniBus app. The busdriver folder contains all the logic and UI components for the bus driver, while the student folder contains the code related to the student's functionalities. Along with these folders, there are 3 files in the main lib folder that are not placed in any subfolder. firebase_option.dart contains the Firebase configuration to integrate Firebase services like authentication, storage, and database for the UniBus app. main.dart is the entry point of the application, where the app is initialized and where the routes, themes, and other global settings are defined. The third file is first_page.dart, which handles the first page that users see when launching the app, typically the introduction to the app.

4.4.2 assets Directory

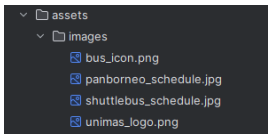


Figure 4.61: assets Directory

The assets folder in the project contains non-code resources required for the UniBus app. Within the assets folder, Figure 4.61 shows there is an images subfolder, which holds the image files for the UniBus app. This includes 3 images that are used in the app, such as for images for displaying bus schedules and showing the UNIMAS logo on the sign-in page.

4.4.3 test Directory

Figure 4.62: test Directory

The test directory in the project contains test files to ensure the functionality and performance of the UniBus app. Based on Figure 4.62, the only file in this directory is widget_test.dart. This file is used to perform widget tests on the Flutter app. The widget_test.dart file contains a basic Flutter widget test. Widget tests are used to verify that the UI components of the app behave as expected. In this file, a simple test is performed on the UniBus app's counter functionality. The testWidgets function is used to simulate user interactions and verify the results.

4.4.4 pubspec.yaml

Figure 4.63 is the pubspec.yaml file which is a configuration file in a Flutter project that defines essential project settings, including project metadata, environment, dependencies, development dependencies, Flutter configurations, and assets. The project metadata specifies the project name, the app's version, and the build number. The environment of this project uses version 3.7.2, which means the UniBus app supports any Dart SDK version equal to or greater than version 3.7.2. The dependencies section includes the Flutter framework, libraries, tools, and other third-party packages needed for the UniBus app's core features and user interface. The development dependencies packages are needed for testing and linting. flutter_test is for writing, running unit and widget tests, while flutter_lints is for maintaining coding standards and consistency. The Flutter configuration enables the use of material design components in the UniBus app and all the images in the assets folder are specified in this file under the assets section.

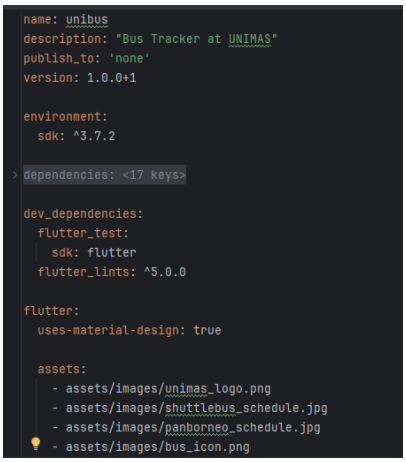


Figure 4.63: pubspec.yaml File

4.5 Code Snippets

4.5.1 Displaying the Map with Bus Locations

The Figure 4.64 shows the part initializes the map and sets the camera position based on the student's current location and the selected pickup and drop-off points. It also handles adding bus stop markers and updating the map with markers for bus drivers and routes.

```
child: GoogleMap(
  initialCameraPosition: _getCameraPosition(), // Ensure the camera is correctly positioned
  onMapCreated: (GoogleMapController controller) {
    _googleMapController = controller;
    _controller.complete(controller);
    _addBusStopMarkers(); // Add the bus stop markers to the map initially
  },
  markers: _markers, // All markers (bus driver, student, bus stops) will be displayed here
  polylines: _polylines, // Add polyline to map if needed
  myLocationButtonEnabled: true,
  zoomControlsEnabled: false,
),  // GoogleMap
```

Figure 4.64: Displaying the Map with Bus Locations

4.5.2 Updating Bus Driver's Location in Firestore

The code in Figure 4.65 listens to location updates for the bus driver and updates it to the Firestore database every 5 seconds with the driver's location in latitude and longitude.

The location is stored under the driverLocation collection in Firestore, which linked to the specific busId and busType.

```
void _updateBusDriverLocation(Position position) {
  if (position.latitude == null || position.longitude == null) {
    print("Error: Latitude or Longitude is null.");
    return;
  }

  // Cancel previous timer if it exists
  if (_debounceTimer != null) {
    _debounceTimer!.cancel();
  }

  // Set a new timer to update Firestore after 5 seconds
  _debounceTimer = Timer(const Duration(seconds: 5), () async {
    LatLng currentLocation = LatLng(position.latitude, position.longitude);

    // Check if latitude and longitude are valid numbers
    if (currentLocation.latitude == null || currentLocation.longitude == null) {
      print("Error: Invalid Latitude or Longitude.");
      return;
    }

    try {
      // Update the driverLocation collection with busId and busType
      await FirebaseFirestore.instance.collection('driverLocation').doc(widget.busId).set(
        'latitude': currentLocation.latitude,
        'longitude': currentLocation.longitude,
        'busId': widget.busId,
```

Figure 4.65: Updating Bus Driver's Location in Firestore

4.5.3 Fetching Real-Time Bus Locations from Firestore

The code in Figure 4.66 demonstrates how the app listens for real-time updates from Firestore to fetch the current location of all buses. It updates the map by removing old bus markers and adding new ones based on the latest data from Firestore.

```
// Fetch the location of all buses from Firestore
void _fetchAllBusLocations() {
  FirebaseFirestore.instance
      .collection('driverLocation')
      .snapshots() // Listen to real-time updates for all buses
      .listen((QuerySnapshot snapshot) {
    setState(() {
      // Remove markers for all bus drivers
      _markers.removeWhere((marker) => marker.markerId.value.startsWith('bus_driver'));

      // Loop through each bus document in the driverLocation collection
      for (var doc in snapshot.docs) {
        final data = doc.data() as Map<String, dynamic>;
        final driverLocation = LatLng(data['latitude'], data['longitude']);
        final busId = doc.id;

        // Add a new marker for each bus driver
        _markers.add(Marker(
          markerId: MarkerId(busId), // Unique marker ID for each bus driver
          position: driverLocation, // Update the bus driver's location
          infoWindow: InfoWindow(title: 'Bus Driver $busId'), // Title for info window
          icon: BitmapDescriptor.defaultMarkerWithHue(BitmapDescriptor.hueRed), // Red color bus driver mar
        ));  // Marker
      }

      _addBusStopMarkers();
```

Figure 4.66: Fetching Real-Time Bus Locations from Firestore

4.5.4 Starting Bus Location Updates

The code in Figure 4.67 shows that if permission for device location is granted, the app starts listening for location updates using Geolocator.getPositionStream(). The locationSettings include high location accuracy to ensure that the location updates are precise. The distance filter is set to 5 meters, meaning the app only triggers a location update if the bus driver moves 5 meters. This reduces unnecessary updates and ensures efficient data fetching.

```
if (permission == LocationPermission.whileInUse || permission == LocationPermission.always) {
  _positionStream = Geolocator.getPositionStream(
    locationSettings: const LocationSettings(
      accuracy: LocationAccuracy.high,
      distanceFilter: 5, // Only trigger an update if the device moves 5 meters
    ),
  ).listen((Position position) {
    _updateBusDriverLocation(position);
  });
}
```

Figure 4.67: Starting Bus Location Updates

4.5.5 Route Calculation Using Google Directions API

This part of the code Figure 4.68 fetches the route between two locations which is pickup and drop-off point from the Google Directions API. It returns the encoded polyline points that can be decoded and displayed on the map as a route.

```
// Fetch route from Google Directions API
Future<String> fetchRoute(LatLng origin, LatLng destination) async {
  final String url = 'https://maps.googleapis.com/maps/api/directions/json'
      '?origin=${origin.latitude},${origin.longitude}'
      '&destination=${destination.latitude},${destination.longitude}'
      '&key=$apiKey';

  final response = await http.get(Uri.parse(url));

  if (response.statusCode == 200) {
    final Map<String, dynamic> data = json.decode(response.body);
    if (data['status'] == 'OK') {
      return data['routes'][0]['overview_polyline']['points']; // Return polyline points
    } else {
      throw Exception('Failed to load directions');
    }
  } else {
    throw Exception('Failed to load directions');
  }
}
```

Figure 4.68: Route Calculation Using Google Directions API

4.5.6 Calculating ETA

The Figure 4.69 shows the part of code to calculates the ETA. The function calculateETA calculates the estimated time of arrival by fetching directions data from the Google Directions API between the bus driver's location and the student's pickup point location. It calculates the ETA based on the distance between the two points and the average speed of the bus. Then ETA is returned and displayed in minutes.

```dart
// Calculate ETA using total distance divided by average speed
Future<int> calculateETA(LatLng driverLocation, LatLng pickupLocation) async {
  final String url = 'https://maps.googleapis.com/maps/api/directions/json'
      '?origin=${driverLocation.latitude},${driverLocation.longitude}'
      '&destination=${pickupLocation.latitude},${pickupLocation.longitude}'
      '&mode=driving&key=$apiKey';  // Use mode=driving for ETA

  final response = await http.get(Uri.parse(url));

  if (response.statusCode == 200) {
    final Map<String, dynamic> data = json.decode(response.body);
    if (data['status'] == 'OK') {
      int distanceInMeters = data['routes'][0]['legs'][0]['distance']['value'];

      // Average bus speed
      double averageSpeedInMPS = 7.78;  // 28 km/h = 7.78 m/s

      // Calculate ETA in seconds
      int etaInSeconds = (distanceInMeters / averageSpeedInMPS).round();

      // Return ETA in minutes
      return (etaInSeconds / 60).round();
    } else {
      throw Exception('Failed to load directions');
    }
  } else {
    throw Exception('Failed to load directions');
```

Figure 4.69: Calculating ETA

4.5.7 Calculating Estimate Walking Time

The code in Figure 4.70 shows in function calculateWalkingTime calculates the estimate walking time from the student's current location to the pickup point using the Google Directions API. It uses the Google Directions API in walking mode and estimates the walking time based on the distance between the two points and the average walking speed. Then the estimated walking time is returned and displayed in minutes.

```
Future<int> calculateWalkingTime(LatLng studentLocation, LatLng pickupLocation) async {
  final String url = 'https://maps.googleapis.com/maps/api/directions/json'
      '?origin=${studentLocation.latitude},${studentLocation.longitude}'
      '&destination=${pickupLocation.latitude},${pickupLocation.longitude}'
      '&mode=walking&key=$apiKey';  // Use mode=walking for walking time

  final response = await http.get(Uri.parse(url));

  if (response.statusCode == 200) {
    final Map<String, dynamic> data = json.decode(response.body);
    if (data['status'] == 'OK') {
      // Extract the distance in meters from the API response
      int distanceInMeters = data['routes'][0]['legs'][0]['distance']['value'];

      // Average walking speed
      double averageWalkingSpeedInMPS = 1.4;  // 1.4 m/s = approx. 5 km/h

      // Calculate walking time in seconds
      int walkingTimeInSeconds = (distanceInMeters / averageWalkingSpeedInMPS).round();

      // Return walking time in minutes
      return (walkingTimeInSeconds / 60).round();
    } else {
      throw Exception('Failed to load walking time');
    }
  } else {
    throw Exception('Failed to load walking time');
```

Figure 4.70: Calculating Estimate Walking Time

4.5.8 Storing Student Class Schedule

The code in Figure 4.71 shows how the class schedule is stored in Firestore, including merging new data with existing data, which is important for keeping the schedule up-to-date without losing previous data.

```
Future<void> storeClassSchedule(Map<String, dynamic> scheduleData) async {
  try {
    // Get the current user's UID (Firebase Auth)
    String uid = FirebaseAuth.instance.currentUser!.uid;

    // Access the classSchedule collection
    DocumentReference scheduleRef = _db.collection('classSchedule').doc(uid);

    // Fetch the existing schedule document from Firestore
    DocumentSnapshot scheduleSnapshot = await scheduleRef.get();

    if (scheduleSnapshot.exists) {
      // Get the existing data
      Map<String, dynamic> existingSchedule = scheduleSnapshot.data() as Map<String, dynamic>;

      // Merge the new schedule data with the existing one
      scheduleData.forEach((day, newClasses) {
        if (existingSchedule.containsKey(day)) {
          // If the day already exists, append the new classes to that day's class list
          List existingClasses = existingSchedule[day];

          // Check for duplicates: add the new class only if it's not already in the list
          for (var newClass in newClasses) {
            bool isDuplicate = existingClasses.any((classItem) =>
            classItem['subject'] == newClass['subject'] &&
                classItem['startTime'] == newClass['startTime']);
            if (!isDuplicate) {
              existingClasses.add(newClass); // Only add if not already in the list
```

Figure 4.71: Storing Student Class Schedule

4.5.9 Fetching and Displaying Class Schedule

The Figure 4.72 shows the function fetches the class schedule from Firestore and maps the data to a list of classes for each day. It is important to show how the app retrieves the schedule data to keep the list of class updated.

```dart
// Retrieve the student's class schedule from Firestore
Future<Map<int, List<Map<String, String>>>> getClassSchedule() async {
  try {
    String uid = FirebaseAuth.instance.currentUser!.uid;
    DocumentReference scheduleRef = _db.collection('classSchedule').doc(uid);
    DocumentSnapshot scheduleSnapshot = await scheduleRef.get();

    if (scheduleSnapshot.exists) {
      final scheduleData = scheduleSnapshot.data() as Map<String, dynamic>;

      final Map<int, List<Map<String, String>>> schedule = {};
      scheduleData.forEach((key, value) {
        final dayIndex = days.indexOf(key);
        if (dayIndex != -1) {
          final classList = (value as List).map((item) => Map<String, String>.from(item)).toList();
          schedule[dayIndex] = classList;
        }
      });

      return schedule;
    } else {
      print("No schedule found for the user.");
      return {};
    }
  } catch (e) {
    print("Error loading class schedule: $e");
    return {};
  }
```

Figure 4.72: Fetching and Displaying Class Schedule

4.5.10 Saving Bus Status and Notifications

The function in Figure 4.73 saves the bus status details such as plate number, bus ID, availability, start time and end dates time to Firestore.

```
Future<void> _saveBusStatus() async {
  try {
    final String uid = FirebaseAuth.instance.currentUser!.uid;
    await FirebaseFirestore.instance.collection('busStatus').doc(uid).set({
      'plateNumber': plateNumber,
      'busId': widget.busId,
      'busType': widget.busType,
      'startDate': startDate,
      'endDate': endDate,
      'startTime': Timestamp.fromDate(DateTime(
          startDate!.year, startDate!.month, startDate!.day, startTime!.hour, startTime!.minute)),  //
      'endTime': Timestamp.fromDate(DateTime(
          endDate!.year, endDate!.month, endDate!.day, endTime!.hour, endTime!.minute)),  // DateTime,
      'availability': availability,
    });
```

Figure 4.73: Saving Bus Status and Notifications

After saving the bus status, in Figure 4.74 shows the code that sends notifications to both bus drivers and students via the busNotifications and studentNotifications in Firestore collections. This ensures that both drivers and students are notified of the status update.

```
final String id = widget.busId.isNotEmpty ? widget.busId : 'Unknown';
final String type = widget.busType.isNotEmpty ? widget.busType : 'Bus';
// Notify bus drivers
await FirebaseFirestore.instance.collection('busNotifications').add({
  'busId': id,
  'message': '$id bus status was updated.',
  'timestamp': Timestamp.now(),
  'status': 'info',
});

//  Notify students
await FirebaseFirestore.instance.collection('studentNotifications').add({
  'busId': id,
  'message': '$type ($id) bus status was updated.',
  'timestamp': Timestamp.now(),
  'status': 'info',
});
```

Figure 4.74: Send Notification About Bus Status

4.5.11 Starting the Bus Shift and Sending Notifications

The code in Figure 4.75 handles the starting of the bus shift. It checks if a route is selected, updates the start time, and sets the bus's status to active.

```
Future<void> _startRoute() async {
  if (startLocations == null || startLocations!.isEmpty) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Please select a route first.')),
    );
    return;
  }

  final now = DateTime.now();
  startTimeString = '${now.hour.toString().padLeft(2, '0')}:${now.minute.toString().padLeft(2, '0')}';

  if (mounted) {
    setState(() {
      hasStarted = true;
      hasEnded = false;
    });
  }
```

Figure 4.75: Starting the Bus Shift and Sending Notifications

When the bus status is active, the code in Figure 4.76 shows notifications sends to both bus drivers and students about the bus starting its journey. This functionality is important for ensuring that the bus is marked as "on duty" and that the students are notified.

```
final String id = widget.busId.isNotEmpty ? widget.busId : 'Unknown';
final String type = widget.busType.isNotEmpty ? widget.busType : 'Bus';

await FirebaseFirestore.instance.collection('busNotifications').add({
  'busId': id,
  'message': 'Bus $id is now on duty. The journey has just started.',
  'timestamp': FieldValue.serverTimestamp(),
  'status': 'active',
});

await FirebaseFirestore.instance.collection('studentNotifications').add({
  'busId': id,
  'message': '$type ($id) is now active and on the move.',
  'timestamp': FieldValue.serverTimestamp(),
  'status': 'active',
});

ScaffoldMessenger.of(context).showSnackBar(
  const SnackBar(content: Text('Shift started.')),
);
```

Figure 4.76: Send Notifications About Start Shift

4.5.12 Ending the Bus Shift and Saving the Report

Figure 4.77 show the part handles the ending of the bus shift. It calculates the end time and saves the trip report details to Firestore,

```
Future<void> _endRoute() async {
  final now = DateTime.now();
  endTimeString = '${now.hour.toString().padLeft(2, '0')}:${now.minute.toString().padLeft(2, '0')}';

  final report = {
    'date': '${now.year}-${now.month.toString().padLeft(2, '0')}-${now.day.toString().padLeft(2, '0')}',
    'startTime': startTimeString ?? 'Unknown',
    'endTime': endTimeString ?? 'Unknown',
    'route': startLocations ?? 'Unknown',
    'plateNumber': plateNumber ?? 'Unknown',
  };

  await FirebaseFirestore.instance
      .collection('busDrivers')
      .doc(uid)
      .collection('tripReports')
      .add(report);
```

Figure 4.77: Ending the Bus Shift and Saving the Report

When the bus status is inactive, the code in Figure 4.78 shows the notifications sends to both bus drivers and students about the bus completing its route. The trip report includes key information like the start times, end times, route, and plate number, which are important for tracking the bus driver's work hours and route details.

```
  final String id = widget.busId.isNotEmpty ? widget.busId : 'Unknown';
  final String type = widget.busType.isNotEmpty ? widget.busType : 'Bus';

  await FirebaseFirestore.instance.collection('busNotifications').add({
    'busId': id,
    'message': 'Bus $id has completed its route for today.',
    'timestamp': FieldValue.serverTimestamp(),
    'status': 'inactive',
  });

  await FirebaseFirestore.instance.collection('studentNotifications').add({
    'busId': id,
    'message': '$type ($id) has ended its service for today.',
    'timestamp': FieldValue.serverTimestamp(),
    'status': 'inactive',
  });

  setState(() {
    tripReports.insert(0, report);
    hasStarted = false;
    hasEnded = false;
    startTimeString = null;
    endTimeString = null;
    startLocations = null;
  });
```

Figure 4.78: Send Notification About End Shift

4.7 Summary

This chapter 4 highlights the implementation of the UniBus mobile application, outlining the technologies, tools, and frameworks used, including Dart, Flutter, Firebase, and Google Maps API. It described the development of system modules for both students and bus drivers, emphasizing real-time tracking, schedule management, and notifications. The chapter also addressed the flow of execution, explaining user interactions and backend processes for core features. It also discussed the implementation challenges such as billing constraints, IDE and dependency issues, and third-party package integration.

CHAPTER 5: TESTING

5.1 Introduction

This chapter will cover the testing phase of the UniBus mobile application. It includes both Functional Testing and Usability Testing to ensure the application performs as expected and offers an intuitive user experience. Functional Testing aims to validate that key features such as user registration, bus location tracking, ETA calculations, and notifications work correctly under various conditions. Usability Testing is designed to assess how users interact with the app, gathering feedback on its ease of use, navigation, and overall satisfaction. The goal is to identify any usability issues and improve the user experience by addressing any challenges encountered. This chapter provides the test cases, methodology, and results for both functional and usability aspects, ensuring that the UniBus app meets both technical and user expectations.

5.2 Functional Testing

Functional testing ensures that the application works as expected, verifying all the key features like sign-up, login, bus location tracking, ETA and estimate walking time calculations, and notifications. Test cases are designed to simulate different user actions and scenarios to check if the UniBus app performs its functions correctly. This testing will confirm the UniBus app's reliability and functionality under various conditions, to ensure all components are integrated and working as intended.

Table 5.25: Test Case Sign Up & Login

Module Name: Sign Up & Login **Test Case Description:** Test the sign-up and login functionality for both students and bus drivers **Testing Objective:** To ensure that users can sign up and log in successfully with the provided credentials. **Remarks:** Use valid email and password combinations for both types of users.								
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**	**Notes**
TC-001	Sign-up and Login	1. Open app. 2. Enter email, password 3. Click sign-up	Email: 12345@siswa.unimas.my / fizah@gmail.com Password: 123456	User successfully signs up and logs in to the app.	User successfully signed up and logged in.	Pass	High	
TC-002	Invalid login credentials	1. Open app. 2. Enter invalid email/password. 3. Click login.	Email: invalid email Password: wrongpassword	Error message appears indicating invalid login credentials.	Error message displayed: Invalid email/password.	Pass	High	Error message correct

Table 5.26: Test Case Bus Location Tracking

Module Name: Bus Location Tracking **Test Case Description:** Test real-time tracking of bus location on the map **Testing Objective:** To ensure that the real-time bus location is accurately displayed and updated on the map. **Remarks:** Bus location updates must be fetched from Firestore based on real-time data.								
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**	**Notes**
TC-003	Display real-time bus location	1. Open app. 2. View map. 3. Monitor bus movement in real-time.	Bus ID: BS01	Bus location is shown and updated accurately.	Bus location updated in real-time.	Pass	High	Bus location updates every 5 meters
TC-004	Bus location accuracy	1. Open app. 2. Compare map location with actual bus position.	Bus ID: BS01	Bus location on map matches real-world position.	Bus location on map matches real-world.	Pass	High	

Table 5.27: Test Case ETA and Estimate Walking Time Calculations

Module Name: ETA and Estimate Walking Time Calculations

Test Case Description: Test the accuracy of estimated time of arrival (ETA) and estimated walking time calculations

Testing Objective: To ensure that ETA and estimated walking time calculations are accurate based on bus location and student position.

Remarks: -

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Severity	Notes
TC-005	Test ETA calculation accuracy	1. Select pick-up and drop-off points. 2. Check ETA. 3. Confirm if ETA is accurate.	Pick-up: Dahlia Bus Stop, Drop-off: Alamanda Bus Stop	ETA displayed should match expected bus arrival time.	ETA calculation matched the expected time.	Pass	High	
TC-006	Test Estimate Walking time calculation accuracy	1. Select pick-up point. 2. Check estimated walking time calculation. 3. Confirm walking time.	Pick-up: Dahlia Bus Stop, Student Location: Alamanda Bus Stop	Walking time calculated should match real walking time.	Estimate Walking time calculation matched real walking time.	Pass	High	

Table 5.28: Test Case Notifications

Module Name: Notifications **Test Case Description:** Test bus schedule and status notifications **Testing Objective:** To ensure users receive notifications about bus availability, schedule changes, and updates. **Remarks:** Simulate changes to bus availability and schedule								
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**	**Notes**
TC-007	Test notification on bus availability	1. Bus Driver update the bus schedule/ update bus status/ update operational time 2. Student receive a notification.	View Notification Page	Notification should appear with bus availability info.	Notification displayed with bus type, plate number and update message.	Pass	High	

Table 5.29: Test Case Pick-up & Drop-off Point Selection

Module Name: Pick-up & Drop-off Point Selection

Test Case Description: Test the functionality of selecting pick-up and drop-off points from a dropdown menu.

Testing Objective: To ensure that users can select pick-up and drop-off points efficiently and without errors.

Remarks: The list of locations should be easily accessible in the drop-down menu.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Severity	Notes
TC-008	Test pick-up and drop-off point selection	1. Open the app. 2. Select pick-up and drop-off points from the list. 3. Confirm selection.	Pick-up: BHEP, Drop-off: Lakeview	Selection is completed without errors.	Points selected successfully without issues and show the nearest bus recommendation.	Pass	High	

Table 5.30: Test Case Bus Status Updates

Module Name: Bus Status Updates

Test Case Description: Test real-time bus status updates

Testing Objective: To ensure that bus status updates are reflected in real-time in the app.

Remarks: Simulate a change in bus status and confirm if it's reflected in the app.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Severity	Notes
TC-009	Test real-time bus status update	1. Change bus status in Firestore. 2. Check if the status is updated in the app.	Bus ID: BS01, Status: On Break	Bus status should be updated and displayed in the app.	Bus status updated and displayed correctly.	Pass	High	

5.3 Usability Testing

The usability testing questionnaire in aims to gather feedback from users about their experience with the UniBus app's key features and functionalities. The responses help assess the app's effectiveness, ease of use, and overall performance. Below is an explanation of each question in the usability testing questionnaire

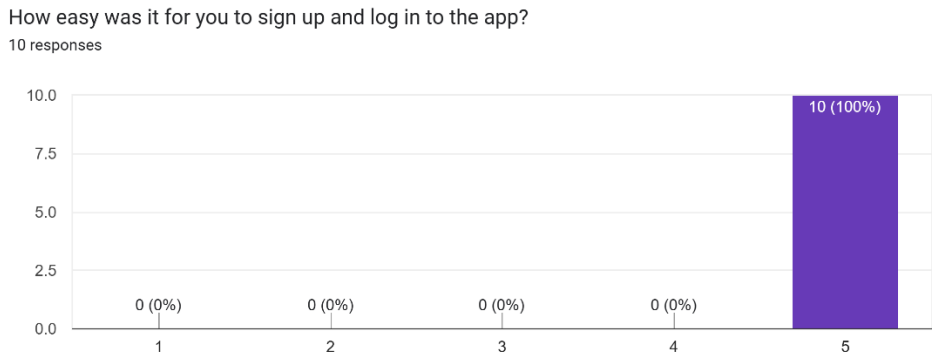


Figure 5.79: Questionnaire about Login and Sign Up Page

In Figure 5.79 shows all participants rated the sign-up and login process as very easy. This suggests that the registration and authentication flow of the app is intuitive and straightforward. Users did not encounter any significant issues during this phase, which is crucial for ensuring a smooth start when using the app.

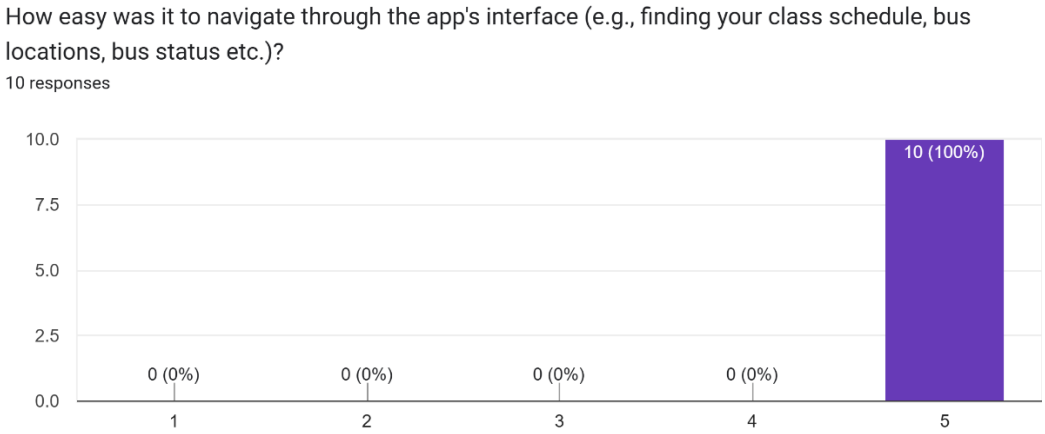


Figure 5.80: Questionnaire about Navigation in the App

In Figure 5.80 shows all participants found the app's navigation to be very intuitive and easy to use. This indicates that the app's interface is well-designed, and users are able to locate the important features, such as class schedules, bus locations, and bus status updates without any confusion.

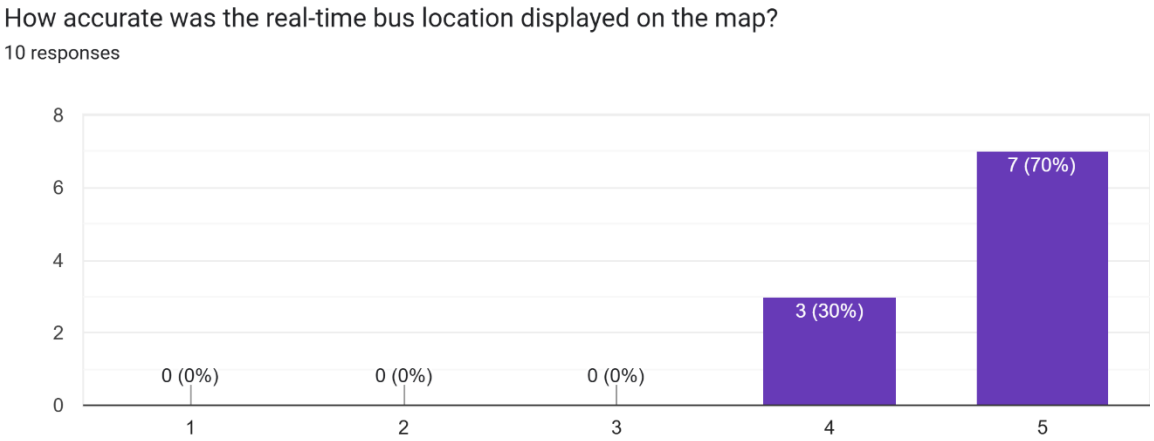


Figure 5.81: Questionnaire About Real Time Bus Location

The Figure 5.81 shows While many users 7 out of 10 users found the real-time bus location to be accurate, 3 participants reported some differences, possibly due to minor delays or inaccuracies in the GPS tracking. The overall feedback suggests that real-time tracking is functioning well, though there may be occasional issues that need addressing, such as updating bus positions more frequently or improving GPS precision.

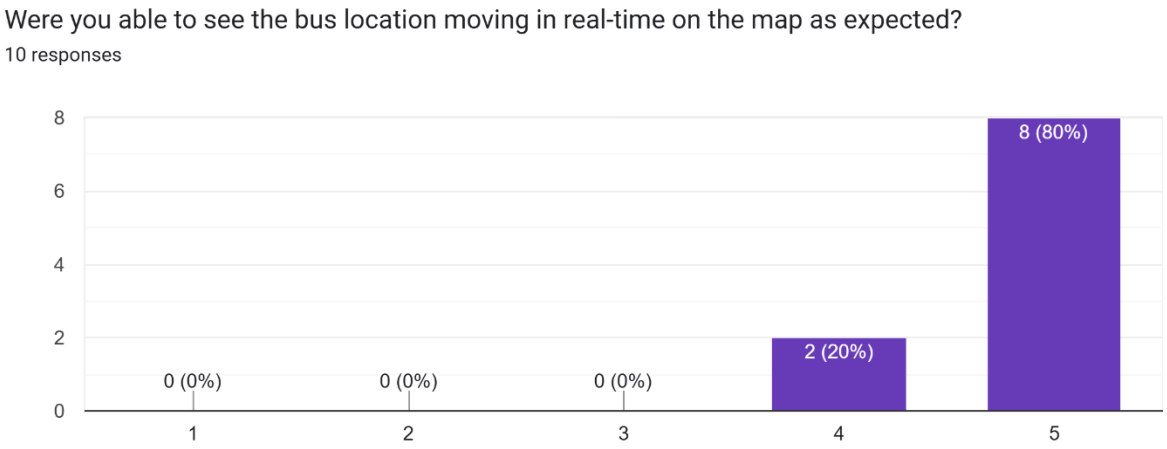


Figure 5.82: Questionnaire About the Real Time Bus Movement

The Figure 5.82 shows 8 out of 10 users 8 out of 10 reported seeing the bus location move in real-time as expected. This suggests that the real-time tracking feature works effectively for many users. However, 2 participants rated it 4, which may indicate occasional issues with map updates, such as minor delays in bus location movement or glitches in the display.

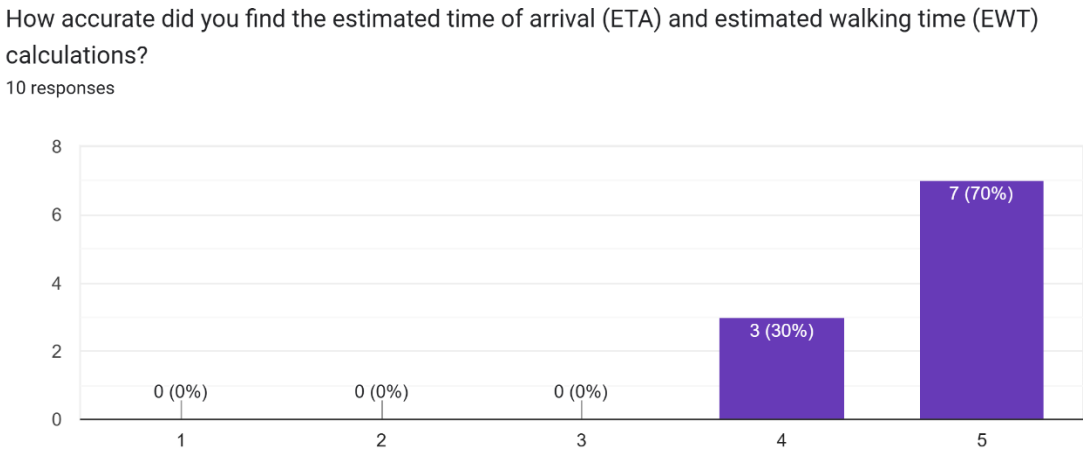


Figure 5.83: Questionnaire About Calculation od ETA and EWT

The Figure 5.83 shows seven participants rated the ETA and EWT calculations as very accurate, while 3 found them less accurate. This could be due to variations in real-time data, such as traffic conditions or walking speed estimations, which can affect the accuracy of the

predictions. Users seem generally satisfied with these features, but there is room for improvement in making the calculations more precise.

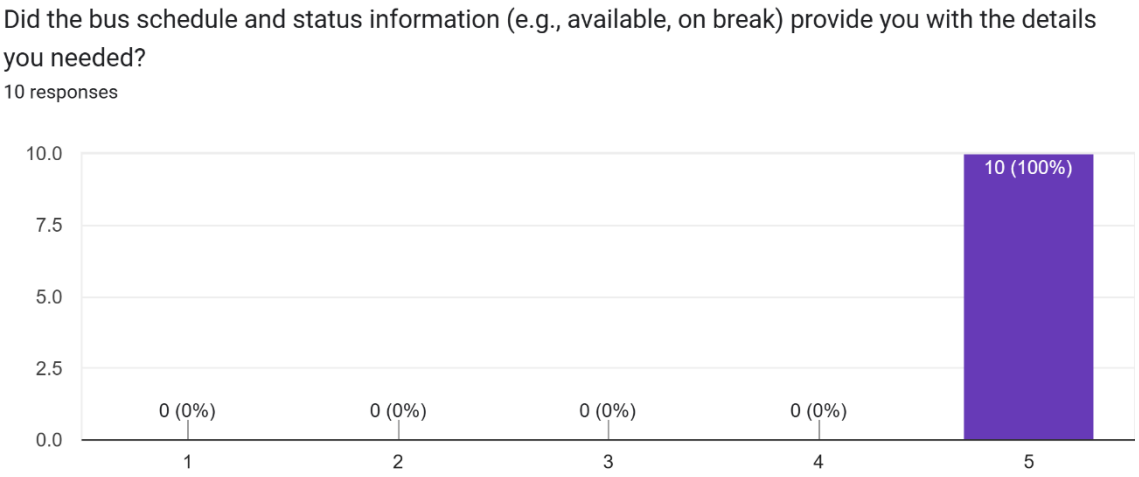


Figure 5.84: Questionnaire About Bus Information

In Figure 5.84 shows all participants rated the bus schedule and status information as highly useful. This indicates that the bus status updates are clear and provide the necessary information for users to plan their trips effectively. The real-time updates regarding bus availability and break periods are considered essential for a smooth user experience.

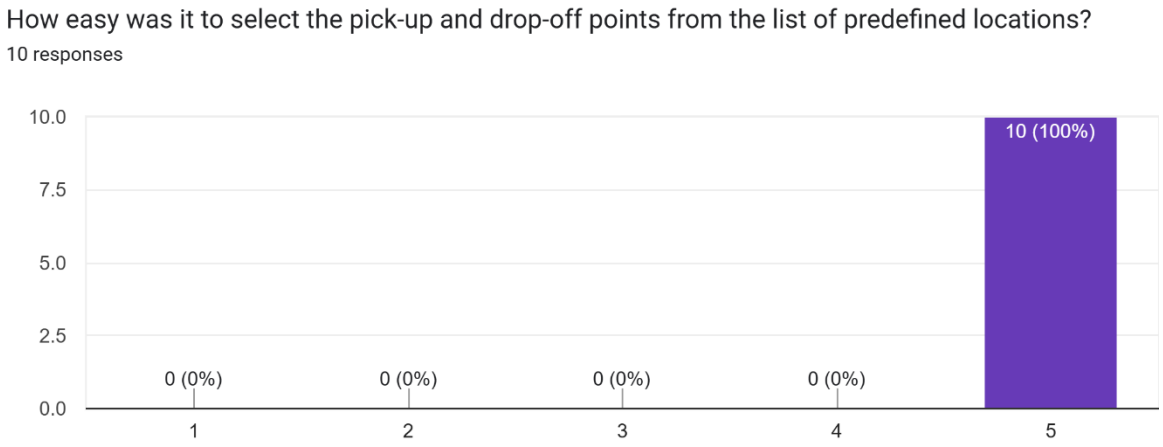


Figure 5.85: Questionnaire About the Bus Stop Selection

Figure 5.85 shows the ease of selecting pick-up and drop-off points was highly rated by all participants. This shows that the predefined location feature is functioning well, making it simple for users to select their travel points without confusion. A seamless user experience in this area contributes to the overall positive feedback on the app's usability.

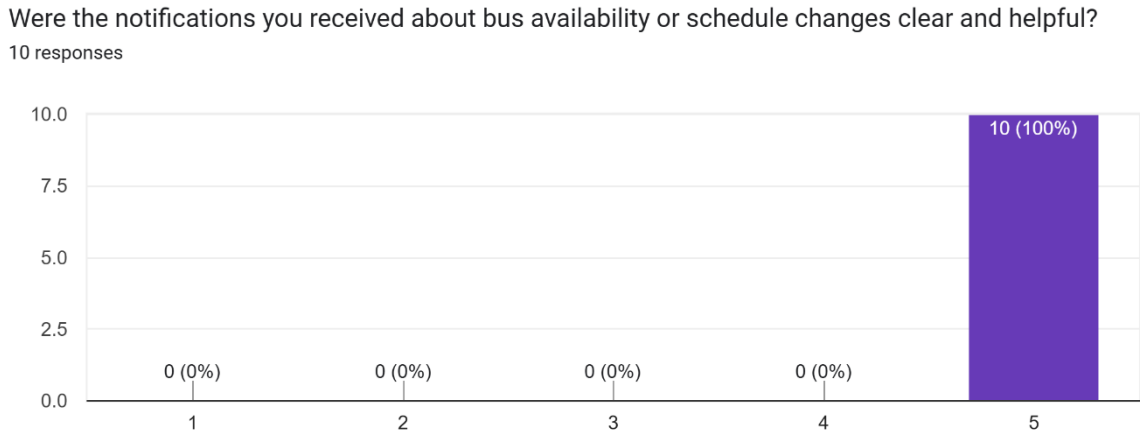


Figure 5.86: Questionnaire About Notifications

In Figure 5.86 shows all participants found the notifications to be clear and helpful. This suggests that the app's notification system is functioning effectively, providing timely and

relevant information regarding bus status, schedule changes, and other important updates. Clear notifications are crucial for keeping users informed in real time.

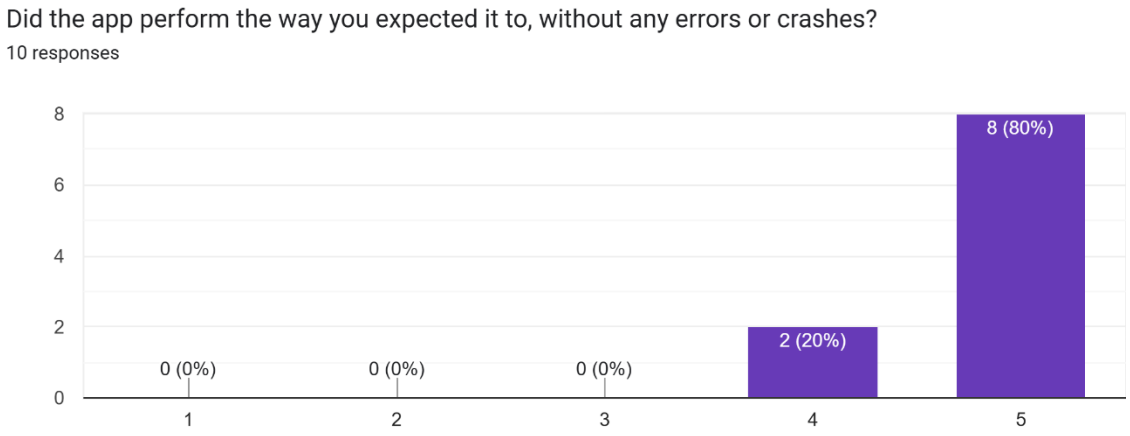


Figure 5.87: Questionnaire About Error and Crashes

The Figure 5.87 shows 8 out of 10 participants reported that the app performed as expected, without any errors or crashes. This indicates that the app is generally stable and reliable for most users. However, 2 participants gave a rating of 4, suggesting that while the app performed well, they may have experienced minor issues, such as occasional slowdowns, minor bugs, or temporary glitches. These issues might need further investigation to ensure the app runs smoothly for all users consistently.

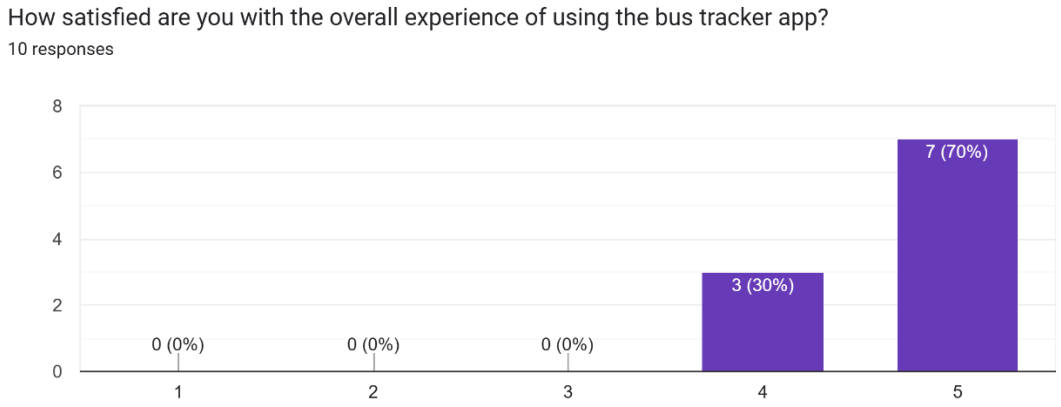


Figure 5.88: Questionnaire About User Experience

Figure 5.88 shows that 7 out of 10 participants were very satisfied with their overall experience using the bus tracker app. However, 3 participants gave a rating of 4, indicating that while they were generally satisfied, there might have been small areas of improvement they felt could enhance their experience. These areas could include minor bugs or features they would like to see improved, such as the accuracy of the ETA or the responsiveness of certain features. Overall, the app's functionality and performance were well-received by most users.

5.4 Summary

Chapter 5 covered the testing process of the UniBus mobile application, focusing on Functional Testing and Usability Testing. The tests confirmed that the app's essential features perform as expected under various conditions. The chapter concludes by analyzing the test results, which demonstrated that the app meets the required functionality and provides a satisfactory user experience.

CHAPTER 6: CONCLUSION AND FUTURE WORKS

6.1 Introduction

This chapter will cover the study and highlights the key findings from the development and testing of the UniBus mobile application. This chapter revisits the primary objectives set out at the beginning of the project and evaluates how well they have been achieved. It also discusses the limitations and challenges of the project and proposes directions for future improvements.

6.2 Objectives and Achievement

Table 6 31: Objectives and Achievement

Objectives	Achievement
To analyze existing bus tracking systems to develop a mobile application that provides real-time tracking of buses and estimates time arrival (ETA).	The app successfully implemented a real-time bus tracking system that provides accurate estimates of time arrival (ETA). This was achieved by utilizing the Google Maps API for real-time location tracking and the Firebase platform to update bus locations.
To design and implement a mobile application that offers nearest bus recommendations and optimal pickup points based on students' class schedules.	The application was designed to recommend the nearest buses based on students' class schedules. By integrating the Google Directions API, the app calculates the ETA and EWT to the pickup point.
To evaluate the usability and functionality of the application in checking bus availability, particularly during late hours.	Extensive usability testing confirmed that the app is functional, and user-friendly. The app successfully meets the goal of providing real-time bus availability and tracking, with positive feedback from users about its efficiency in late hours.

6.3 Project Limitations & Challenges

6.3.1 Billing and Service Quotas

During development, integrating Firebase and Google Cloud services are required setting up billing accounts by adding a card. Services such as Firestore, Cloud Functions, and Cloud Storage are billed based on the amount of data read, written, stored, or transferred.

For Google Cloud Platform, new users receive a $300 (RM1,290.90) credit for the first 90 days, which was a short time for development and testing for this project. With this free trial account, the project was able to integrate the Google Maps API and many other APIs in the Google Cloud services. Due to the testing and debugging during development, the initial free trial for Google Cloud ran out much sooner than expected, within just one month. This was because the credits were quickly consumed while using various services like the Directions API. An alternative approach was explored by creating a new Google account and using a different bank card to extend the services. This approach allowed continued progress on the project while managing the costs associated with the plan upgrade.

Firebase Authentication offers a free tier, allowing up to 10,000 verifications per month for email and password sign-ins. Charges will apply if the number of sign-ins exceeds this limit, based on usage. For instance, $0.01 (RM0.04) per verification for email and password logins. However, for this project, it is unlikely to exceed 10,000 verifications per month, so this is not a major concern. Still, it is important for me to monitor user sign-ins to ensure it stays within the free limits.

Firebase Storage is one of the services offered by Firebase, with certain storage limits in place. Although I had not yet utilized Firebase Storage, the platform prompted to upgrade to a paid plan to access the service. Despite this, I encountered an issue where I was still unable to use Firebase Storage. As a solution, I explored alternative image hosting services and chose ‘imgbb’, which allowed me to continue with my project without incurring the additional costs with upgrading Firebase Storage service. I chose imgbb because it provided a simple and efficient way to host images with good storage limits, and the integration was straightforward, and easy for project development.

6.3.2 IDE and Dependency Issues

Initially, Visual Studio Code (VS Code) was used as the IDE for developing the UniBus app. However, during the early development process, the persistent issues arose related to Gradle and Kotlin dependencies, which caused delays in the implementation of project. After countless attempts to resolve the issues by downgrading and upgrading dependencies and adjusting the Gradle configuration, the problems remained unresolved. This issue is very common encountered by developers that working with Flutter and Kotlin, and it required countless trial-and-error sessions.

To address the situation, the IDE was changed to Android Studio. Android Studio was chosen because it provides a more stable and integrated development environment for Flutter and Kotlin compared to Visual Studio Code. Android Studio offers out-of-the-box support for Flutter and Kotlin, which reduces the compatibility issues that often arise from manually configuring dependencies and external plugins in Visual Studio Code. Additionally, Visual Studio Code requires extra setups and installations for dependencies such as Gradle and Kotlin, whereas Android Studio comes with pre-configured support for these tools. This pre-configured support in Android Studio helps manage the updates for Gradle and Kotlin, minimizing errors and minimize the extensive troubleshooting related to missing or misconfigured dependencies.

6.3.3 Integrating Third-Party Packages

Integrating various third-party packages, such as google_maps_flutter, geolocator, and firebase_auth, led to a significant challenge during the development of the UniBus app. Each of these packages provides important functionality for map visualization, bus location tracking (google_maps_flutter and geolocator), and user authentication (firebase_auth).

The challenge arose from dependency management between the packages in puspec.yaml file, as each package has different version requirements and updates. This led to conflicts when trying to integrate the third-party packages into the app. Updating one package sometimes caused compatibility issues with others. To address these challenges, extensive testing was performed to ensure that all third-party packages worked together across all device platforms and functions.

As with version management, attention was needed to ensure the documentation of each package was followed, and compatible versions were selected. Updated packages were

frequently monitored to ensure that new releases didn’t lead to any further conflicts. When conflicts arose, manual fixes were applied by adjusting the compatible versions or the previous stable version in the pubspec.yaml file and reviewing the release notes of each package.

6.4 Future Works

The following areas can be explored for future enhancements and development:

1. Integration with Hardware GPS:
   When the budget allows, integrating the system with a hardware-based GPS tracking solution could improve the accuracy and reliability of real-time tracking, particularly for bus drivers.

2. Expansion to Other Locations:
   The app could be expanded to support more campuses and cities, incorporating more bus routes and improving its utility beyond the university.

3. AI-Powered Predictions:
   Incorporating machine learning algorithms to predict bus delays based on historical data and traffic conditions could further improve the app's accuracy in providing ETA and EWT.

4. User Interface Enhancements:
   Further usability testing and feedback can be used to refine the user interface to ensure that it is even more intuitive and efficient.

6.5 Conclusions

In conclusion, the UniBus mobile application successfully achieved its goals and objectives. Despite some limitations due to budget and technical challenges, the app provides a valuable tool for students to manage their transportation needs. With future enhancements, such as the integration of hardware GPS and AI-based predictions, the app can evolve into a more comprehensive and reliable solution for real-time bus tracking and management.

6.6 References

Abdullah, M. (2023). *Malaysia Bus Live GPS Tracker.* Google Play Store. https://play.google.com/store/apps/details?id=com.bus.live.myrapid_kl_bus_live&hl=en

Creately. (2022). *Sequence Diagram Tutorial – Complete Guide with Examples.* https://creately.com/guides/sequence-diagram-tutorial/

Geeksforgeek. (2025). *Class Diagram | Unified Modeling Language (UML).* https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/

Hildenbrand, J. (2020). How does GPS work on my phone? https://www.androidcentral.com/how-does-gps-work-my-phone

Lonescu, D. (2021). *How Inadequate Transit Harms College Students.* https://www.planetizen.com/news/2021/12/115586-how-inadequate-transit-harms-college-students

Lowry Solutions. (2024). *How GPS Tracking is Reshaping Transportation and Logistics: 8 Benefits.* https://lowrysolutions.com/blog/benefits-of-gps-tracking-reshaping-transportation-and-logistics/

Mara Labs Inc. (2024). *Real-time Tracking.* https://locus.sh/resources/glossary/real-time-tracking/#:~:text=What%20is%20Real%2Dtime%20Tracking%3F,known%20as%20Real%2Dtime%20Tracking.

Moovit. (2020). *Shaping the future of urban mobility with MaaS.* https://static-main.moovit.com/wp-content/uploads/2020/04/Moovit-Solutions-Booklet-EN_Interactive.pdf

Mural. (2024). A guide to the Agile development lifecycle. https://www.mural.co/blog/agile-development-lifecycle

Nishadha. (2025). *UML Diagram Types Guide: Learn About All Types of UML Diagrams with Examples.* https://creately.com/blog/diagrams/uml-diagram-types-examples/

Prasarana Malaysia Berhad. (December 2020). *PULSE by Prasarana Malaysia Berhad.* MyRapid. https://myrapid.com.my/pulse-by-prasarana/

Public Transport Victoria. (2024). *Bus tracking system and real time bus data.* https://www.ptv.vic.gov.au/footer/about-ptv/improvements-and-projects/bus-and-coach/bus-tracking-system-and-real-time-bus-data/#:~:text=A%20bus%20tracking%20system%20is,in%20'real%2Dtime'.

Sree, N., Mamatha, T., Sreekanth, B., & Mohammed, N. (June 2021). *Integrated College Bus Tracking System*. https://www.researchgate.net/publication/353112571_Integrated_College_Bus_Tracking_System

Visual Paradigm. (2024). *What is State Machine Diagram?* https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

Wilson, T. V. (n.d.). How GPS phones work. HowStuffWorks. Retrieved January 8, 2025. https://electronics.howstuffworks.com/gps-phone.htm