UTAR TRANSPORTATION TRACKER APPLICATION

By

Foo Jia Syuen

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

FEBRUARY 2025

COPYRIGHT STATEMENT

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

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr. Goh Hock Guan, for his support throughout my final year project. Dr. Goh's valuable advice has played important roles in assisting me in the completion of this project.

ABSTRACT

The UTAR Transportation Tracker is a comprehensive system designed to enhance the commuting experience for students who travel to and from Universiti Tunku Abdul Rahman (UTAR) Kampar campus. Commuting in the accommodation area around UTAR has present challenges to the students due to unpredictable bus schedules and limited access to real-time information of the buses, which causes inefficiencies and inconvenience. To address these challenges, the UTAR Transportation Tracker system is developed to meet the transportation needs of the UTAR community. The system integrates real-time bus tracking and weather forecasting functionalities to provide users with accurate information and timely alerts. By employing GPS technology, the system offers real-time tracking of UTAR buses, giving students the ability to monitor the exact location and estimated arrival time of buses to reach the designated bus stops. This function allows students to schedule their journey more efficiently and improves accountability and transparency in UTAR bus operations. In addition, the integration of weather forecasting feature gives users access to the most recent weather information and alerts for adverse weather conditions, enabling students to plan their travels ahead of time. The UTAR Transportation Tracker project demonstrates significant advancement in utilizing technology to enhance the transportation system and improve commuting experience for students and staff who travel to and from UTAR.

Area of Study: Mobile Application Development

Keywords: Transportation Tracker, Google Maps API, User-friendly Application, Realtime data, Firebase

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

GPS	Global Positioning System
UTAR	University Tunku Abdul Rahman
MRT	Mass Rapid Transit
LRT	Light Rail Transit
API	Application Programming Interface

CHAPTER 1: PROJECT BACKGROUND

With the acceleration of modernization and the prominence of traffic issues, people have demanded for the ease and effectiveness of travel. The advent of transportation tracking applications in recent years has completely changed how people travel across urban areas by providing up-to-date information on available transit routes and road conditions. With the ability to navigate busy cityscapes with efficiency, simplicity and convenience, these applications have grown to be essential tools for commuters. The evolution of transportation industry is closely intertwined with the creation and broad use of the Global Positioning System (GPS), which has evolved tremendously since it was introduced by the United States Department for military purposes [1]. GPS tracking is a technology that utilises satellite signals to locate real-time location of a vehicle or device [2]. GPS tracking works through a network of satellites orbiting the Earth, which send signals to GPS receiver positioned on the ground. These receivers can be integrated into various devices like mobile phones, tablets, or dedicated GPS tracking units. By analysing signals from multiple satellites, the GPS receiver determines its precise location through triangulation.

With the introduction of mobile phones and the incorporation of GPS receivers into smartphones, the use of GPS technology in civilian applications accelerated in the late 20th and early 21st centuries [3]. The integration has introduced the development of location-based services and navigation application, which enable users to access real-time location data, route planning, and turn-by-turn navigation ability on their mobile devices. These early transportation tracking apps offered insights into travel durations, traffic conditions, and transit options, giving users an unprecedented level of convenience and efficiency when navigate around the urban areas. Numerous smartphones now equipped with high-precision GPS capabilities, enabling enhanced accuracy in location tracking and navigation.

In this project, the UTAR transportation tracker application is designed to improve the efficiency, accessibility, and safety of the transportation system for UTAR community. Through the use of technology, this initiative seeks to address the transportation challenges experienced by students and staffs who travel to and from UTAR compound, and to enhance the commuting experience.

1.1 Problem Statement and Motivation

Students utilizing UTAR bus transportation have encountered several problems:

1.1.1 Lack of Real-time Information [Major]

The lack of real-time bus information is the major problem faced by UTAR students. In the absence of timely information regarding the bus movements, students might not be aware of the potential incidents, resulting in lost time and missed opportunities. For example, unpredictable or delayed bus arrivals can cause students to miss important classes, or school events, which would negatively affect their academic performance and university experience. In addition, the unpredictability of bus schedules and delays can contribute to increased stress and anxiety among students. Students' mental well-being could be negatively impacted by unwanted stress due to consistently concern about the missing buses or running late for events. UTAR students are frequently in the dark about when the next bus will arrive at the designated bus stops since they do not have access to real-time updates on bus locations and estimated arrival times. Because of this uncertainty, students find it challenging to plan their journeys effectively, which leads to extended waiting times. This wasted time not only results in students' frustration and inconvenience but also leads to productivity loss, as they are not able to utilize their time wisely yet waiting for buses that may be delayed or cancelled.

1.1.2 Inefficient communication platforms

Due to limited communication platforms, UTAR students may experience difficulties in accessing information regarding transportation services, schedules, and changes. Ineffective communication makes it difficult for students to stay informed about the schedule updates and make appropriate travel plans, which results in frustration and dissatisfaction with the transportation services. The bus schedule was primarily communicated via Gmail or Instagram posts, yet many students did not regularly check their inboxes or may have overlooked important announcements. As a result, they may have missed crucial information regarding changes to the bus schedule. Furthermore, the transportation services cannot guarantee the bus could consistently arrive at the designated bus stop for every session.

1.1.3 Weather-related Disruption

Adverse weather conditions and climate change poses challenges for students relying on bus transportation to travel to and from university campus. Inclement weather such as heavy rain can lead to delays, changes of bus schedule, or cancellations, which will make it difficult for students to estimate the arrival time and plan their journey flexibly. Students will experience inconvenience if they fail to plan and prepare for weather-related disruption during their journey.

The motivation behind this project is to develop a transportation tracker application that could help UTAR students to address the challenges of utilizing UTAR transportation services to travel to and from university campus, as well as to improve the commuting experience while improving the transportation system. This application aims to offer commuters real-time transportation tracking so they can more effectively plan their journey and minimize the disruptions caused by factors like unanticipated schedules, inclement weather, etc.

1.2 Project Objectives

1.2.1 Develop a user-friendly mobile application to track real-time location of UTAR buses

The primary objective if this project is to develop a user-friendly mobile application that enables UTAR students to access the real-time information of university buses. The application will precisely display the routes and movements of UTAR buses by utilizing GPS technology and integrating with Google Maps API. This can help the students to plan their trips more efficiently and save waiting times at bus stops. In addition, the application also offers push notifications to notify students of upcoming bus arrivals and timely updates on bus schedules and delays. The development of a user-friendly mobile application for real-time bus tracking is essential to improve the commuting experience for UTAR students and staff travelling to and from campus, while promoting accessibility, convenience, and efficiency utilizing campus transportation.

1.2.2 Integrate weather API to provide users with real-time weather updates and alerts

Another objective of this project is to integrate a weather API into the transportation tracker application to enable UTAR students to access real-time weather updates and alerts. The application will precisely deliver the up-to-date information on current weather conditions. By doing so, users will be able to access the weather information directly in the application, enabling them to plan and make better decisions during journeys.

1.3 Project Scope

The scope of this project includes developing a user-friendly transportation tracker mobile application that aims to address the challenges faced by UTAR students in utilizing UTAR transportation services and improve the overall commuting experience. The project will involve studying and implementing GPS technology and Google Map API to enable real-time tracking of UTAR buses, offering students access to up-to-date bus locations, schedules, and estimated arrival times. In addition, the study and implementation of weather API, which later will be integrated into the application will also be carried out to enable UTAR students to access the real-time weather updates and alerts. On the other hand, this project will also involve exploring Flutter, a cross-platform framework used to develop the mobile application which includes designing a user-friendly user interface of the application. Furthermore, the project scope also includes testing, debugging, and refining the application to come out with a product that satisfies users' needs.

1.4 Main Contributions from the Project

The primary contribution of this UTAR Transportation Tracker application is the development of a user-friendly mobile application that offers UTAR students access to real-time information about the bus location, routes, and delays. The transportation tracker integrates GPS technology to track bus movements in real time, giving users the ability to monitor the precise location and predict the bus arrival time. In addition, by integrating weather APIs, users can monitor the weather forecasts and receive weather alerts, which helps them to plan their travels more effectively and remain aware of any potential weather disruptions. Furthermore, the Transportation Tracker application could generate useful data for transportation management and decision making. The transportation tracker could offer important insights into the performance and efficiency of UTAR's transportation services by collecting and analyzing the data on bus movements, passenger demand, and travel patterns.

1.5 Report Organization

The details of this research are shown in the following chapters. In Chapter 2, some similar existing transportation tracker related applications are reviewed. Then, preliminary studies are carried out ranging from the methods, software, and hardware used, as well as the UI designs are presented in Chapter 3. Later, Chapter 4 details the system design, covering the system architecture, functional modules, system flows, and GUI layouts. Next, Chapter 5 documents the implementation process, from hardware/software setup to operational workflows with screenshots. Chapter 6 evaluates the system's performance through testing results, challenges, and objective fulfillment. Finally, Chapter 7 conclude the project's achievements and proposes future recommendations.

CHAPTER 2: LITERATURE REVIEW

2.1 Review of Technologies

2.1.1 Hardware platform

Most of the transportation tracker applications such as Moovit, Transit, and more, primarily targeting mobile phones such as Android and iOS. The convenient and mobility of mobile phones allow users to access the application easily at any places.

2.1.2 Firmware/OS

The UTAR transportation tracker application is developed to support multiple operating systems, primarily targeting Android and iOS. These operating systems are widely used on mobile devices, providing a large user base for the application.

2.1.3 Database

- **Firebase**

One of the most popular databases reviewed is Firebase which primarily serve as a potential database solution for mobile and website application development due to its scalability, real-time data synchronization, and ease of use. Firebase offers comprehensive documentation and cross-platform SDKs for application development to assist users in creating and releasing apps for Android, iOS, website, Unity, Flutter, and C++ [4].

- Strength:
  - Real-time database:
    Firebase's real-time database feature allows for seamless data synchronization across devices, enabling instant updated for every connected device [5].
  - Firebase authentication:
    Additionally, Firebase provides strong security measures for app such as authentication and access control, ensuring the integrity and confidentiality of application data through Firebase authentication and security rules feature.
- Limitation:

- Limited query capabilities:
  The querying capability of Firebase is not as extensive as those of more conventional databases like MySQL. Complex data manipulation and queries require more effort.

- **MySQL**

MySQL is another popular relational database management system that stores data in tables with rows and columns and establishes relationships between tables using keys.

- Strength:
  - Powerful querying capabilities:
    MySQL offers powerful querying capabilities using Structured Query Language (SQL), enabling users to build complex queries, join tables, etc.
- Limitation:
  - Limited functionality:
    MySQL may have limited functionality in certain areas as compared to modern database. For example, MySQL requires users to build their own data synchronization mechanisms manually.

2.1.4 Programming Language

- **Dart language (Flutter)**
  Dart language serves as a foundation of Flutter, a client-optimized language that help developers to build applications and websites efficiently and effectively.
- Strength:
  - Widget-based development:
    Flutter has provided many pre-designed widgets and customizable UI components, which allow users to build user interface more easily and effectively.
  - Hot reload:
    Flutter has hot reloaded feature which allows developers to view their code changes immediately reflected in the running emulator without

having to restart the emulator. This has helps to speeds up the development process.

- Platform independence:

  Flutter allows developers to deploy their code across multiple platforms including Android, iOS, and websites.

- Limitation:
  - Hard code:

    Unlike Android Studio, developers need to build the user interface by code, without having the ability to sketch the user interface directly.

- **Java programming language (Android Studio)**

  Java programming language is an object-oriented programming language that supports Android Studio to build mobile applications.

- Strength:
  - XML layout:

    The XML layout feature in Android Studio allows developers to design the UI directly on the layout without needing to hard code.

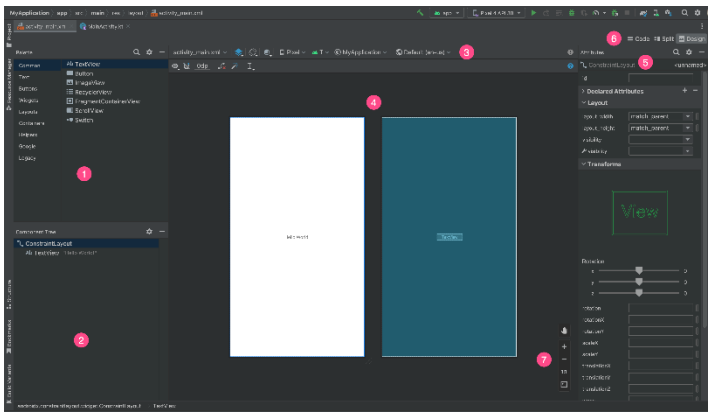


Figure 1 Android Studio XML Layouts

- Limitation:
  - Platform-specific:

    Android Studio focuses on Android application development and does not support cross-platform development.

2.1.5 Summary of the Technologies Review

Database		
	Firebase	**NoSQL**
Real-time database	✅	❌
Authentication	✅	❌
Complex query capabilities	❌	✅
Programming language		
	Dart (Flutter)	**Java (Android Studio)**
Hot reload	✅	❌
Cross-platform development	✅	❌
XML layout	❌	✅

Table 1 Summary of Technologies Review

2.2 Review of Existing Application

2.2.1 Moovit: Bus and Rail Timetables

Figure 2 Moovit application logo

Moovit is a popular mobile application that provides users with real-time information about bus and train schedules in cities around the world [4]. The application aims to simplify public transportation for commuters by providing precise and current information on transit routes, departure and arrival times, and service interruptions. This function can help users to plan their trips more efficiently and save waiting times at bus stops or train stations. Moovit offers different modes of transportation, ranging from buses, trains, MRT & LRT, monorail, cable car, and ferry. Users can choose any lines of the transit type that is available in their cities [5]. In addition, the live direction feature in Moovit provides users the fastest and most convenient instructions to their stations [6]. Furthermore, Moovit helps users to stay aware about any disruptions to their travel plans by notifying them of service disruptions, such as delays, traffic jams, changes of routes, etc. This function enables users to change their travel plans when necessary.

Strength

Moovit offers users access to wide range of transit lines including buses, trains, MRT & LRT, monorail, cable car, and ferry that is available in the application. Users can select the line directly and Moovit apps will show the routes to get to the line.

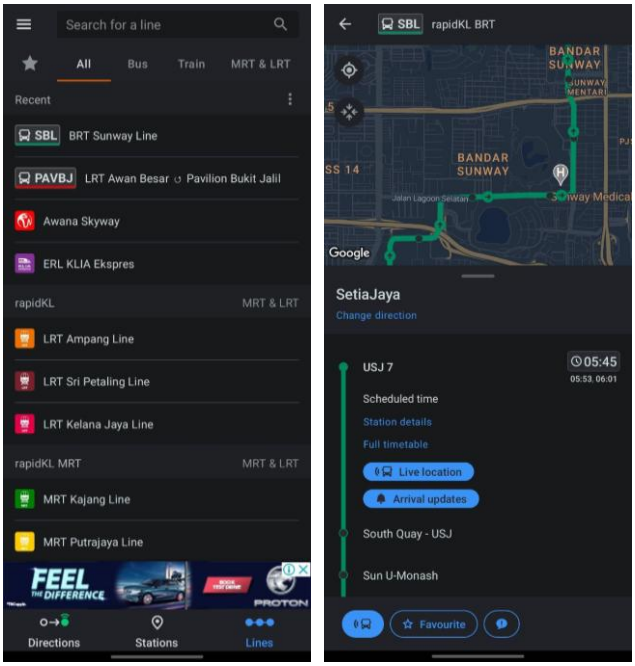


Figure 3 Transit lines

In addition, Moovit's travel filters feature enables users to specify desired transit types to travel from place to place, making them more flexible and convenience when planning their trips [7].

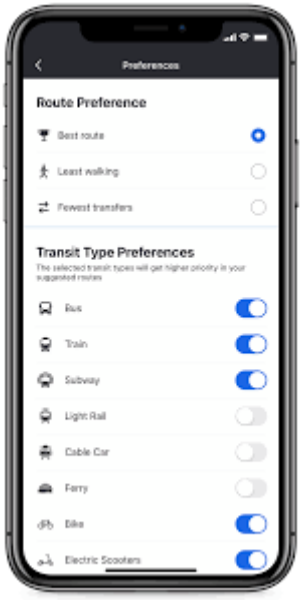


Figure 4 Moovit travel filter feature

Furthermore, Moovit apps also offered offline transit map, which is greatly beneficial for users especially in places with limited internet connectivity [8].

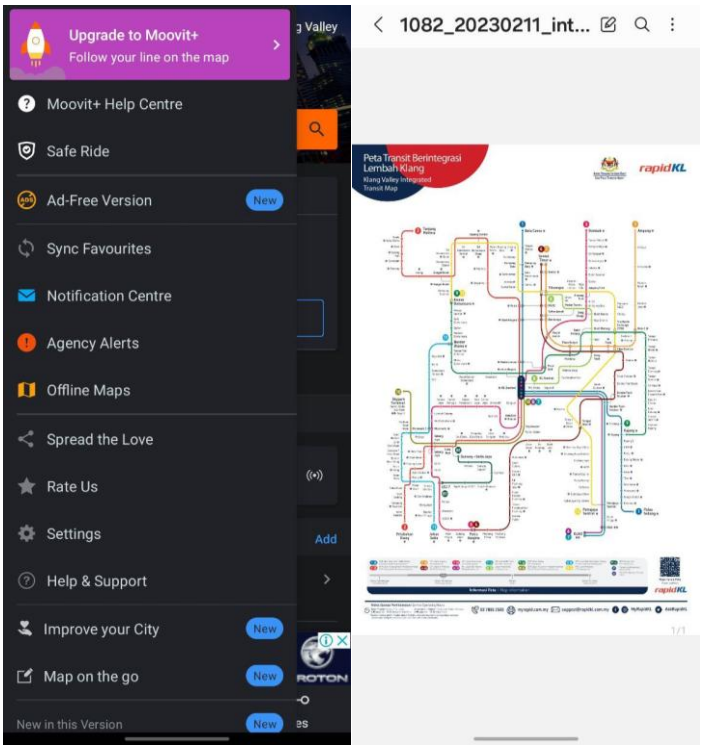


Figure 5 Moovit offline transit map

Limitation

One of the limitations found in Moovit application is the lack of pricing information for each transit route. Although the app is excellent at offering detailed real-time transit data and options for planning routes, users may find it difficult to make decisions based on cost considerations due to the lack of transparency regarding fare prices. Users may encounter difficulties to compare the affordability of various transportation options or to precisely estimate the total cost of their trip without access to pricing information. The limitation may cause users to become frustrated or annoyed, especially for those who depend on public transportation and want to efficiently plan their trip expenses.

Recommendation

Moovit application could collaborate with transit agencies to provide real-time fare information for each transit route. As a result, users would be able to get precise

pricing information within the app and make informed decisions to choose their preferred routes.

2.2.2 Transit: Live Bus and Tube Times

Figure 6 Transit application logo

Transit provides a wide range of features to improve the way users interact with public transit systems. Transit allows users to view transit lines available around them, users can find options for their travel efficiently. The real-time waves animation feature provides intuitive visual on the estimated arrival time cues showing that the transportation is expected to reach in, making users more efficient when choosing routes [9]. By tapping on a transit line, users can access more information and track the real-time location of their ride on the map. In addition, Transit also offers disruption alerts, keeping users stay informed of any service interruptions or delays. Furthermore, Transit also offers a few route possibilities incorporate several forms of transportation, enabling users to compare the options and choose the one they prefer when comes to route planning. Additionally, the step-by-step navigation feature in Transit allows users to reach their destinations with ease and confidence [9].

Strength

The integration of ride-hailing services, Uber in Transit application allows user to hail rides directly within the app in case there is no better options of other transportation modes [10].

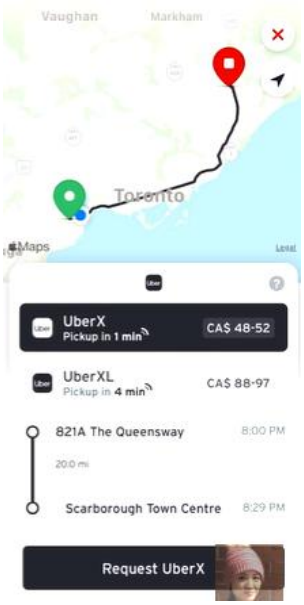


Figure 7 Transit in-app ride-hailing [10]

On the other hand, the inclusion of bike shares and scooters feature even broadens user's transportation choices, while promoting an eco-friendly form of transportation. Users can locate the bikeshare stations that is available around them in the map [11]. Users can access the bikeshare system and purchase bikeshare pass directly in the Transit application.

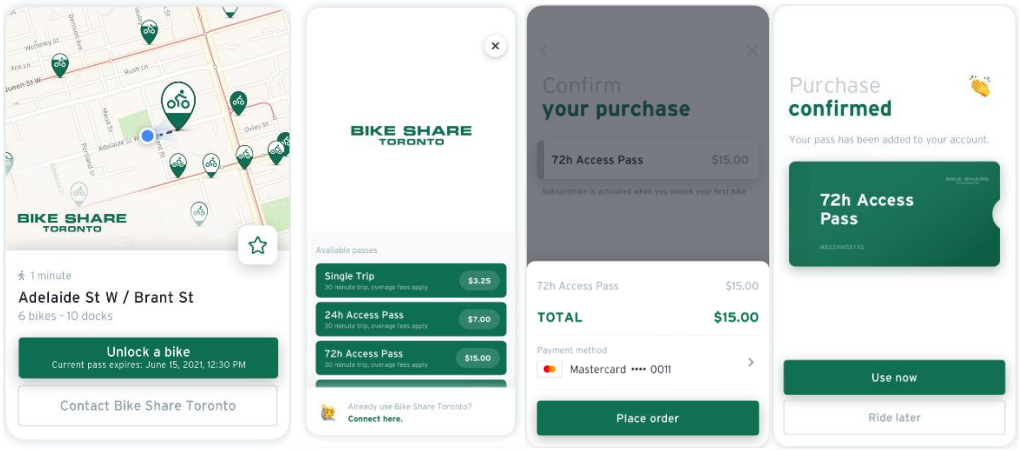


Figure 8 Transit bikeshares feature [11]

In addition, Transit apps also feature crowdsourcing updates which require passengers to give feedback on the transportation crowdedness, providing users with

valuable insights to plan their trips more effectively. Other than crowdsourcing updates, the application also ask custom questions such as the wheelchair accessibility, timeliness, cleanliness, etc [12]. The feedback will be gathered and analyse to provide valuable information for other users.

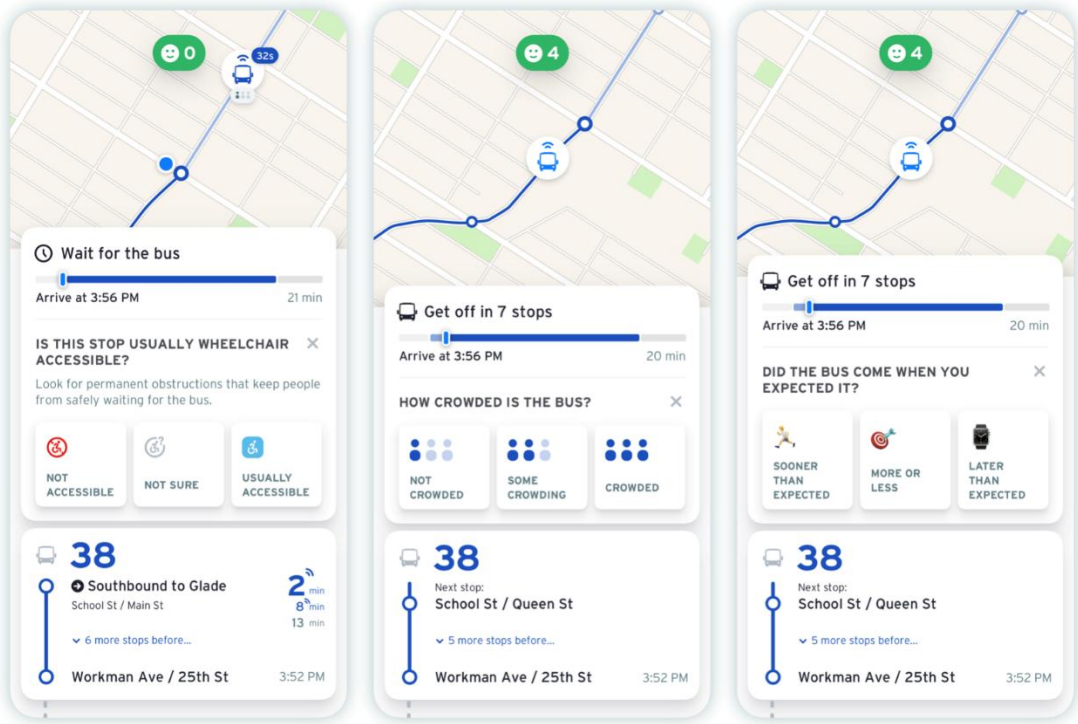


Figure 9 Transit crowdsourcing and feedback feature [12]

Limitation

One limitation of the Transit application is lack of transportation filters. Users are not able to filter travel options based on specific preferences or criteria. For instance, users cannot filter transit routes based on mode of transportation such as bus or train, or accessibility features such as step-free routes. In a result, users may find it more difficult to tailor their trip preferences or meet particular travel needs when using Transit as compared to other applications that offer the filtering features.

Recommendation

Transit application could introduce the transportation filtering feature that allow user to specify their preferred mode of transportation such as bus or train, and accessibility features such as step-free routes. This would facilitate users in filtering

their transit options and choose the routes that meet their travel requirements and preferences.

2.2.3 Citymapper

Figure 10 Citymapper application logo

Citymapper application is designed with variety of functionality aims to enhance user engagement with public transit systems. Users can choose their preferred location/city and access to a variety of city-specific transport options to navigate around the cities. Citymapper's route planning features effectively maps out possible routes from beginning to end while estimating travel times for each mode of transportation. Moreover, users benefit from real-time access to transportation information, make them to be updated about the service updates and interruptions along their chosen route. On the other hand, Citymapper also provides timely notification to ensure users never miss their transits. The application also provides flexibility in terms of transportation scheduling, making it simple for users to modify their travel schedules in the event that they miss their first transit option. With customisable filters that accommodate a range of preferences and accessibility requirements such as step-free routes and minimising walking distance, Citymapper gives users the flexibility to customize their commutes to meet their unique needs [13].

Strength

One of the strengths found in this application is the inclusion of pricing information for different transit routes, which gives users transparency and clarity regarding the cost if their trips [14]. With this information, users can compare the affordability of various transportation options and precisely estimate the total cost of their trips.

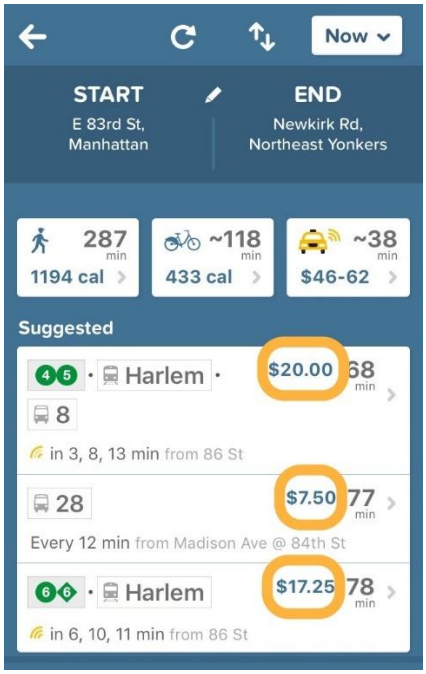


Figure 11 Citymapper pricing information

In addition, Citymapper features payment integration, allowing users to add credit and debit cards to make quick and convenient in-app purchases [15].

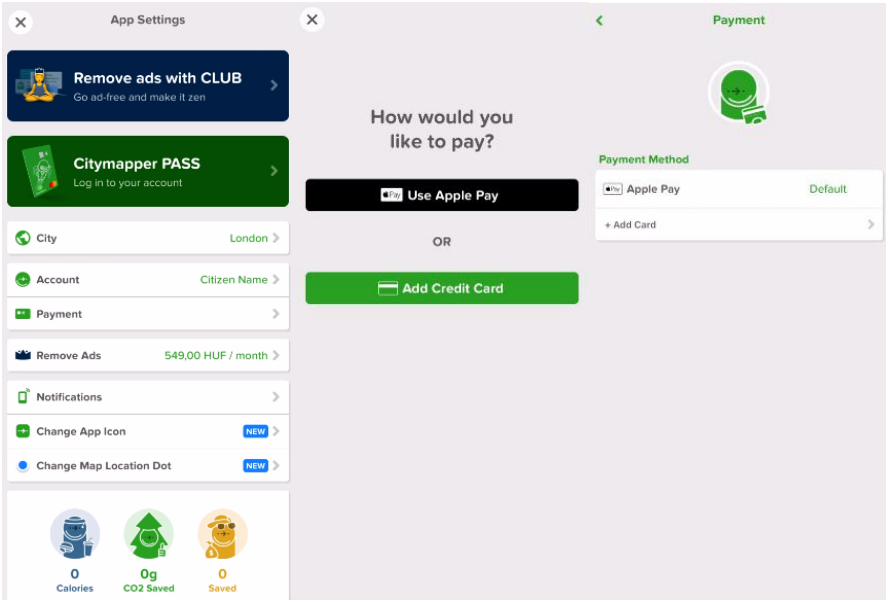


Figure 12 Citymapper payment integration feature [15]

The offline support feature is another significant strength in Citymapper application. It enables users to save routes in advance and access them without the present of internet connection. All the departures information about the chosen route will be preserved offline [16]. This feature guarantees dependability and accessibility in places where network coverage is unstable.

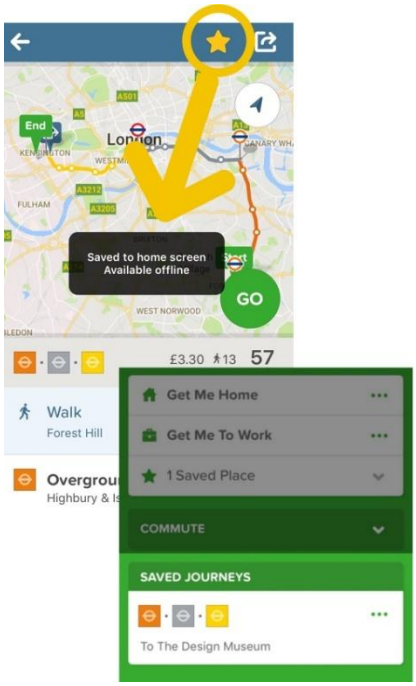


Figure 13 Citymapper offline support [16]

Limitation

In overall, Citymapper covered most of the features as compared to other application. The only limitation in this application is that Citymapper does not cover most of the small cities, it only focuses on big cities.

Recommendation

Citymapper could expand its coverage to include more cities. By doing this, the application would become a more complete transportation option for users in various geographical areas by broaden the user base.

2.2.4 Grab

Figure 14 Grab application logo

Grab is initially developed as a taxi-hailing application, now Grab offers more features ranging from ride-hailing services, food, grocery, and package delivery services, and backing services. Grab uses GPS technology to give drivers and passengers real-time information, to facilitate efficient matching and navigation. Passenger's location is automatically determined using GPS technology when they request a ride in the Grab application. Based on the passenger's real-time location, the app matches them with the closest available driver, minimizing wait times. After matching successfully, passengers can monitor the real-time location of their assigned vehicle and the approximate arrival time. On the other hand, drivers can use the GPS navigation in the Grab app to locate passenger's location. The application offers real-time navigation instructions to help them choose the fastest route to their destination.

Strength

Grab uses algorithms and GPS data to implement dynamic pricing feature. This feature adjusts the fares dynamically in response to variables including traffic congestion, adverse weather condition, and demand patterns, ensuring that drivers are fairly compensated [17].

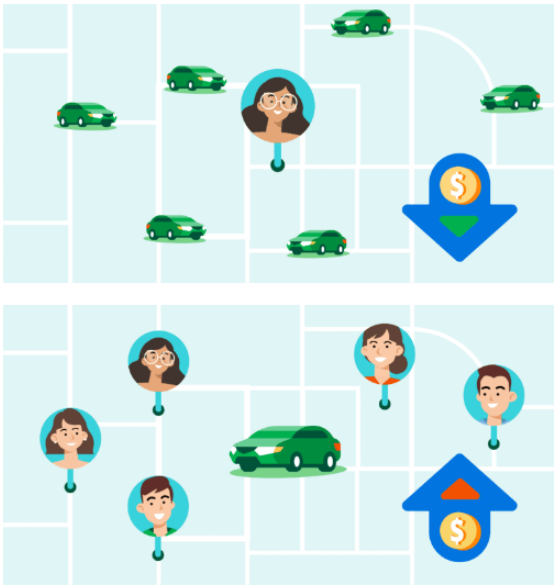


Figure 15 Grab dynamic pricing [17]

The payment system integration in Grab application enables users to make payments more easily and conveniently within the app. Users can link their preferred payment methods, such as credit or debit cards, to their Grab accounts. By doing so, users can complete purchases quickly within the Grab app with just few taps, whether they are ordering food delivery or booking a ride [18]. In addition, Grab gives users flexibility and choice by offering them a variety of payment methods, like in-app credits, digital wallets, etc.

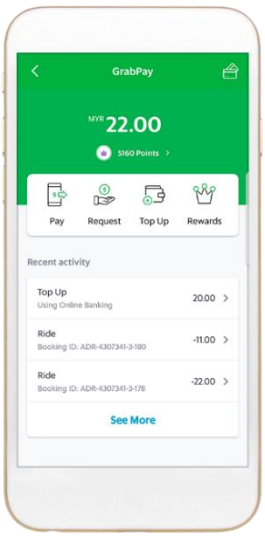


Figure 16 Grab payment system [18]

Limitation

Grab may have limited customization options for users to personalize their experience or preferences. For example, users may have limited control over factors such as vehicle type, driver preferences, route preferences.

Recommendation

Grab could provide a specific preferences menu in the application so that users can quickly configure and modify customization choices. The menu could include settings for vehicle preferences, route preferences, driver preferences, which users can customize choices to meet their needs.

2.2.5 Google Maps

Figure 17 Google Maps application logo

Google Maps is a popular mapping service that offers users with comprehensive geographic information, location-based services, navigation, etc. In the application, users can search for specific places, and receive detailed information, including contact details, photos, and reviews. In addition, Google Maps offers step-by-step navigation instructions for driving, bicycling, walking, and public transportation, assisting users get to their destination efficiently. The application also offers real-time traffic information, alternate routes, and estimated arrival times, to improve navigation accuracy and dependability. Users also can use Google Maps for route planning. By integrating real-time transit data, Google Maps enables users to view updated schedule, service alert, transit fares and arrival times for public transit.

Strength

Google Maps offers users with more information about the places including the address, website, contact number, business hours, and more, which improve their exploration and decision-making. By compiling user-generated evaluations and ratings, Google Maps provides information about the rating and reputation of restaurants, attractions, business, etc. This extensive amount of information enhances users' entire navigation and exploration experience, enabling them to make informed decisions about places to visit, dine, or shop.

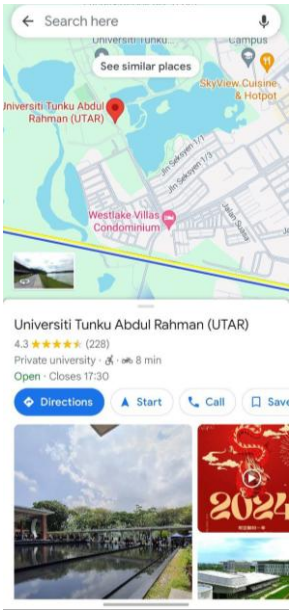


Figure 18 Google Maps place details

Limitation

One limitation of Google Maps is the lack of integration of ride-hailing services such as Grab. Users cannot directly access hail rides service with computed information such as the estimated time to complete the route and fares in Google Maps app, result in inconvenience for people who often use ride-hailing services.

Recommendation

Google Maps could collaborate with ride-hailing services such as Grab and integrate their services directly into the app. This would provide smooth ride-hailing services and computed information access, such as estimated time to complete the route and fares, within Google Maps for users.

2.2.6 Summary of the Existing Application

	Moovit	Transit	Citymapper	Google Maps	Grab
Real-time information	✓	✓	✓	✓	✓
Scheduling	✓	✓	✓	✓	✗
Disruption tracking	✓	✓	✓	✓	✓
Get off notification	✓	✓	✓	✓	✗
Route planning	✓	✓	✓	✓	✗
Step-to-step navigation	✓	✓	✓	✓	✓
Transportation filter	✓	✗	✓	✓	✗
Offline transit map	✓	✗	✗	✓	✗
Offline support	✗	✓	✓	✗	✗
Hail ride integration	✓	✓	✓	✗	✓
Bikeshares and scooter	✓	✓	✓	✗	✗
Crowdsourcing feature	✓	✓	✓	✓	✗
Price stated for routes	✗	✗	✓	✓	✓
Payment integration	✗	✓	✓	✗	✓

Table 2 Comparison of existing application

2.3 Concluding remark

In conclusion, the review of the existing transportation tracker related applications, including Moovit, Transit, Citymapper, Grab, and Google Maps applications, have demonstrated the variety of features and functionalities implemented for users to travel around places. These platforms have offer useful services ranging from real-time transportation information on routes, route planning, scheduling, and more, which have contributed significantly to effective and efficient travel experience for commuters. While each platform shows strengths in particular areas, like route accuracy, interface design, or integration with other services, there remains room for improvement when it comes to addressing challenges such as data accuracy, coverage in remote areas, accessibility for users with different needs, etc. On the other hand, I found that these platforms have not implemented push notifications for forecast weather feature that is one of the key features I will implement in my application to push notifications to alert users if it is going to rain in the next hour so that users can plan their trip earlier wisely.

CHAPTER 3: SYSTEM METHODOLOGY

3.1 System Development Models

3.1.1 Waterfall Methodology

Waterfall methodology is a sequential development process, where project proceed from one phase to another phase, and one phase must be completed before the next phase starts.

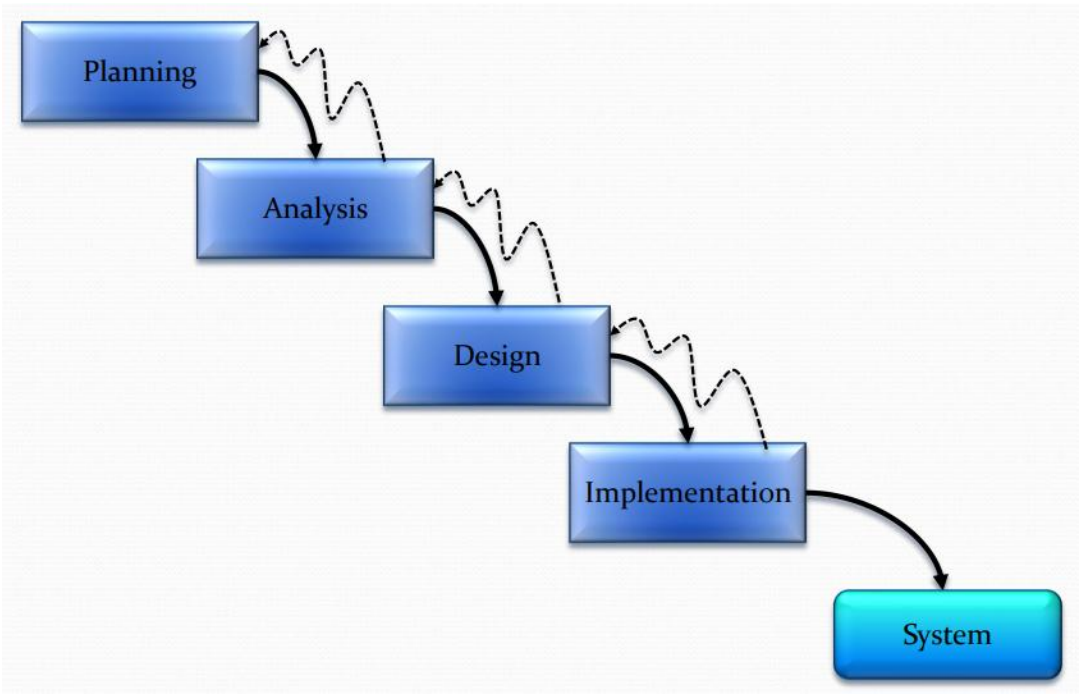


Figure 19 Waterfall Methodology

- **Strengths**
  - Easy to manage:

    The sequential development pattern of waterfall methodology offers a clear and well-defined structure for the project development, making it easy to manage and understand. Project progress can be easily tracked since every phase has its own set of deliverables and milestones.

  - Less changes to the project requirements:

    The waterfall methodology seeks to reduce uncertainty and guarantee that the development team has a clear path forward by specifying system requirements early on before programming start. Modification to the

requirements is discouraged once the requirements have been defined and approved.

- **Limitations**
  - Rigid and inflexible:

    Once the project is in progress, it may be challenging to incorporate modifications or updates due to the sequential development pattern of waterfall model. Any modifications to the requirements may require more efforts to rework and could disrupt the entire process.

3.1.2 Parallel Development Methodology

Parallel development methodology is an approach where different parts of the project are developed concurrently. This may entail breaking the project up into manageable tasks or modules that maybe worked on separately.

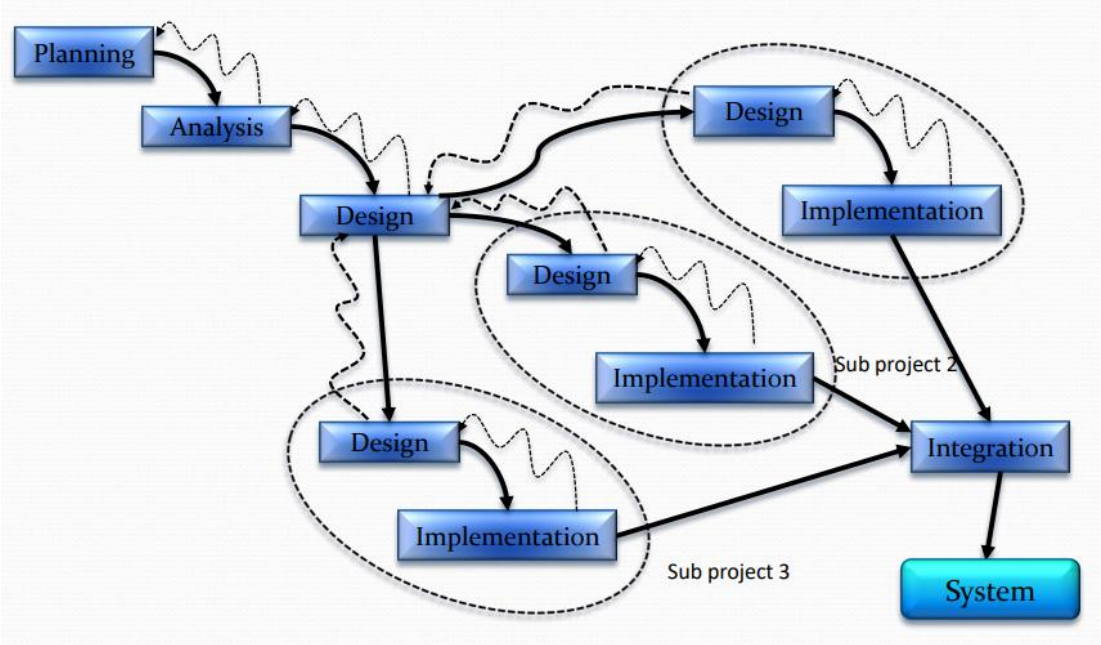


Figure 20 Parallel Development

- **Strengths**
  - Reduced schedule time:

    Parallel development can significantly reduce overall development time since different parts of the project are developed at the same time.
  - Flexibility and adaptability:

The development pattern of parallel development allows flexibility and adaptability to changing circumstances, as each module of the project can update their requirements before the system integration.

- **Limitations**
  - Risk of integration issue

    The integration or combination of different modules that were developed independently could be complicated and prone to errors. This can result in unwanted bugs and performance issues that require additional effort and time to resolve.

3.1.3 Phased Development

Phased development is a methodology in which entire system is break into series of versions that are developed sequentially and has own set of deliverables and objectives. The second version starts after the completion of first version.

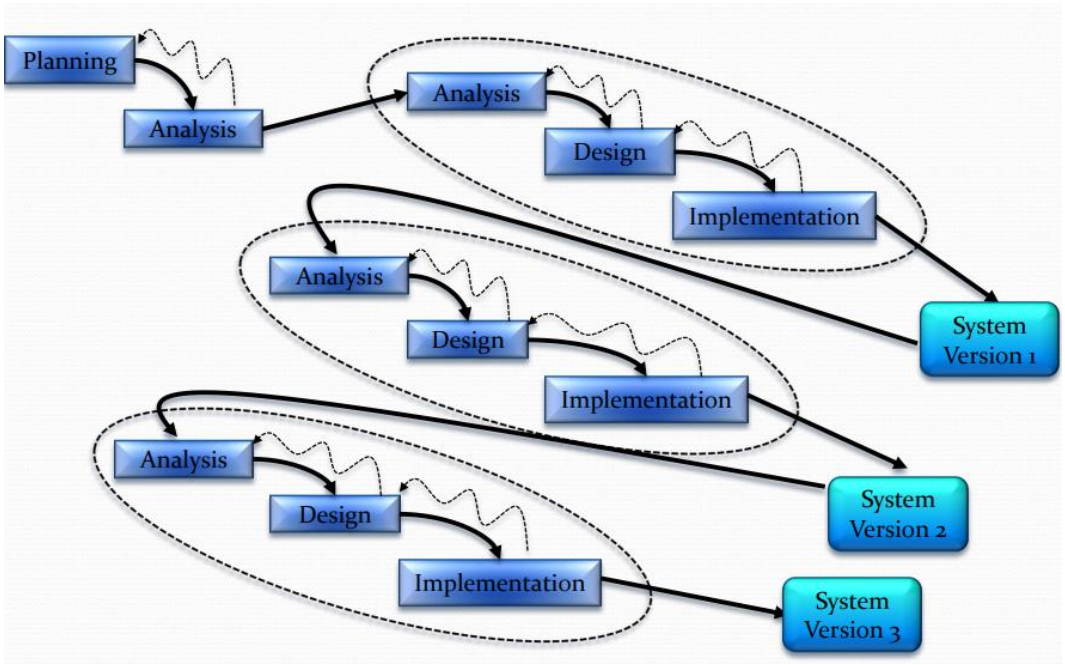


Figure 21 Phased Development

- **Strengths**
  - Early feedback:

    Phased development can produce the first version of the system faster and collect early feedback from users. This allows adjustment and

modification to requirements based on the feedback received, resulting in a delivery that could meet the project expectations better.

- **Limitations**
  - Limited adaptability:

    Phased development follows a sequential development pattern, where each phase must be completed before moving on to the next phase. It may be challenging to accommodate changes during the development of the version.

3.1.4 Prototype-based Methodology

Prototype-based methodology is being used in this project which focus on iterative development and frequent prototyping to improve the system in response to user feedback and changing requirements. The analysis, design, and implementation phases are carried out concurrently in a prototype-based methodology, and all three phases are repeated in a cycle until the system is finalized.

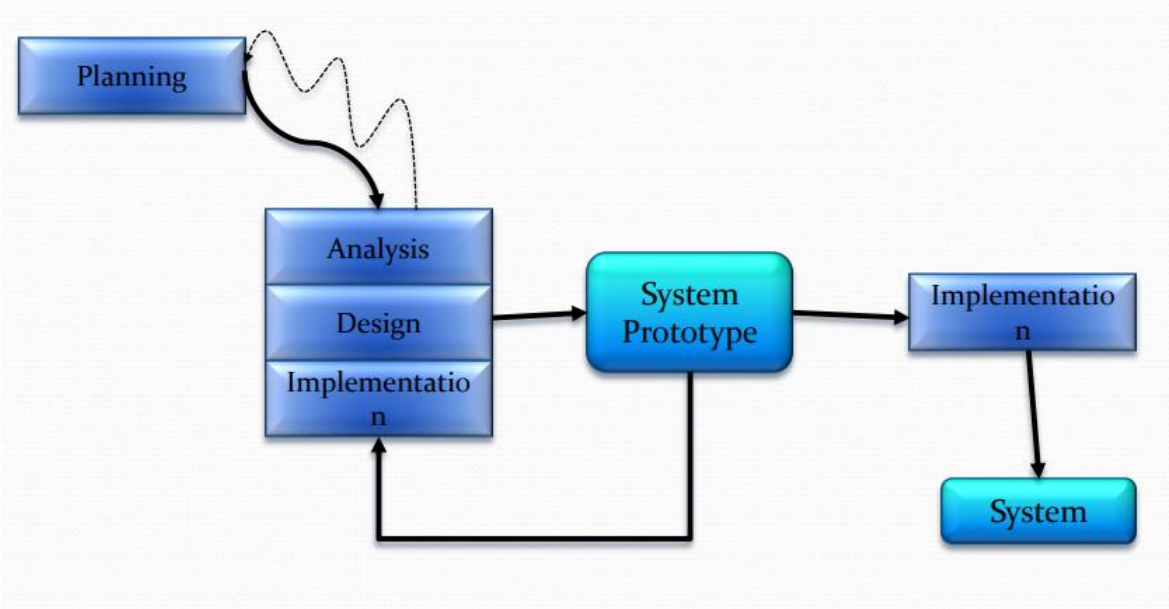


Figure 22 Prototype-based methodology

- **Strengths**
  - Early validation of requirements:

    By developing prototypes early in the development process, prototype-based methodology enables early validation of requirements and design

decisions. This could help to minimize the time and money wasted by identifying potential issues earlier.

- **Limitations**
    - Quality concerns:

      Rapid prototyping may result in a system that is less stable, and poor performance. Striking a balance between the requirements for quick iteration is crucial to making sure that the prototypes are dependable and representative of the final system.

Phases in methodology:

- **Planning phase:**

In the planning phase, initial research on the requirements, defining project goals and project scope was conducted. On the other hand, market research and feasibility studies are also carried out to evaluate the feasibility and possible effects of the transportation tracker system.

- **Analysis phase:**

In the analysis phase, review of existing applications is conducted to gain more understanding on the similar system. This involves examining several aspects including UIs, features, and functionality to identify the strengths and limitations of each application. This review can yield valuable insights that can improve the new application even more to address user needs effectively.

- **Design phase:**

In design phase, the gathered requirements are translated into user interface designs and technical specifications, etc. The selection of suitable technologies, frameworks, and development tools are also conducted in this phase to facilitate the implementation of the transportation tracker system.

- **Implementation phase:**

In implementation phase, the transportation tracker system is built, and test based on the designs and specifications developed in previous phases. Coding, testing, and

system component integration are carried out to produce a working prototype of the transportation tracker application.

3.1.5 Selected Model

The selected model for this project is prototype-based methodology. This reason that this methodology is chosen is because it is more flexible for me to develop my application as I am not confident enough to identify and verify all the requirements and design decisions. The development of prototypes could help me to further identify all the requirements to produce an application that meets the project's objectives and expectations. In addition, prototypes-based methodology helps to mitigate development risk through the early discovery of potential issues. Furthermore, I can further improve the application in terms of features and designs after conducting analysis on the prototype.

3.2 System Requirements

3.2.1 Hardware

- **Laptop**

The laptop is used to implement the coding of the application in both frontend and backend.

Description	Specifications
Model	Asus A512F Vivobook series
Processor	Intel Core i5-10210U
Operating System	Windows 11
Graphic	NVIDIA GeForce MX250 2GB GDDR5 VRAM
Memory	12GB DDR4 RAM
Storage	512GB PCIe® SSD

Table 3 Laptop specification

- **Android Mobile Device**

The android mobile device serves as the platform for showcase the application interface and demonstrate its functionalities.

3.2.2 Software

- **Flutter**

A cross-platform framework that will be used in this project to build the mobile application.

- **Firebase**

A comprehensive platform used to facilitate the backend services.

- **Visual Studio Code**

An integrated development environment serves as the code editor for coding and testing the application.

- **Google Map API**

Google Map API is integrated into the application to provide real-time transportation tracking and mapping functionalities, enable users to monitor bus movements. This can improve user's commuting experience, helping them to plan their journeys wisely.

- Maps SDK for Android
    - Maps for native Android app.
- Directions API
    - Compute route distance and estimated time arrival between two points.

- **OpenWeatherMap API**

OpenWeatherMap API is integrated into the application to retrieve real-time weather data, offer users the functionality to monitor weather forecasts and push alerts to notify users. This can improve user experience and help users to make decisions during journeys.

3.3 Functional Requirement

- Mapping function

This is one of the important features that needs to be implemented in this transportation tracker application, which can indicate the exact location of buses on the

map. Markers which represent the waypoints along the route and polylines which connecting the markers represent the entire route the bus will travel through are displayed on the map to provide comprehensive visual representation for users to understand the route better.

- Select bus route

Users can choose the bus routes based on their preferrable bus stop closest to their location. In this case, UTAR Kampar campus has few bus routes which has already been fixed such as the routes which designed for students stay in Taman Mahsuri Impian or Westlake Homes, etc. Therefore, students are able to choose the route they want and monitor the real-time information of the bus such as the route distance and estimated arrival time to reach the bus stop.

- Monitor real-time information of bus

The main function of the whole application which is crucial that enable users to monitor and access to real-time information of the bus routes. In each bus routes, there are few waypoints that the bus will stop by to pick up and drop up passengers. Users are able to monitor the real-time location of the bus in the map, and access to information such as the route distance and estimated arrival time to reach the bus stop.

- Push notifications for forecast weather conditions

This function will prompt notifications to notify users about the weather conditions, for example, if the weather forecast function read that there will be rain in one hour later, it will push notification to alert users about the weather information so that users can plan their trip efficiently to avoid any weather-related disruptions.

- View bus schedule

Users are allowed to view the bus schedule of all the routes. This will give user early information about the bus time arrival at particular bus stop.

3.4 Project Milestone

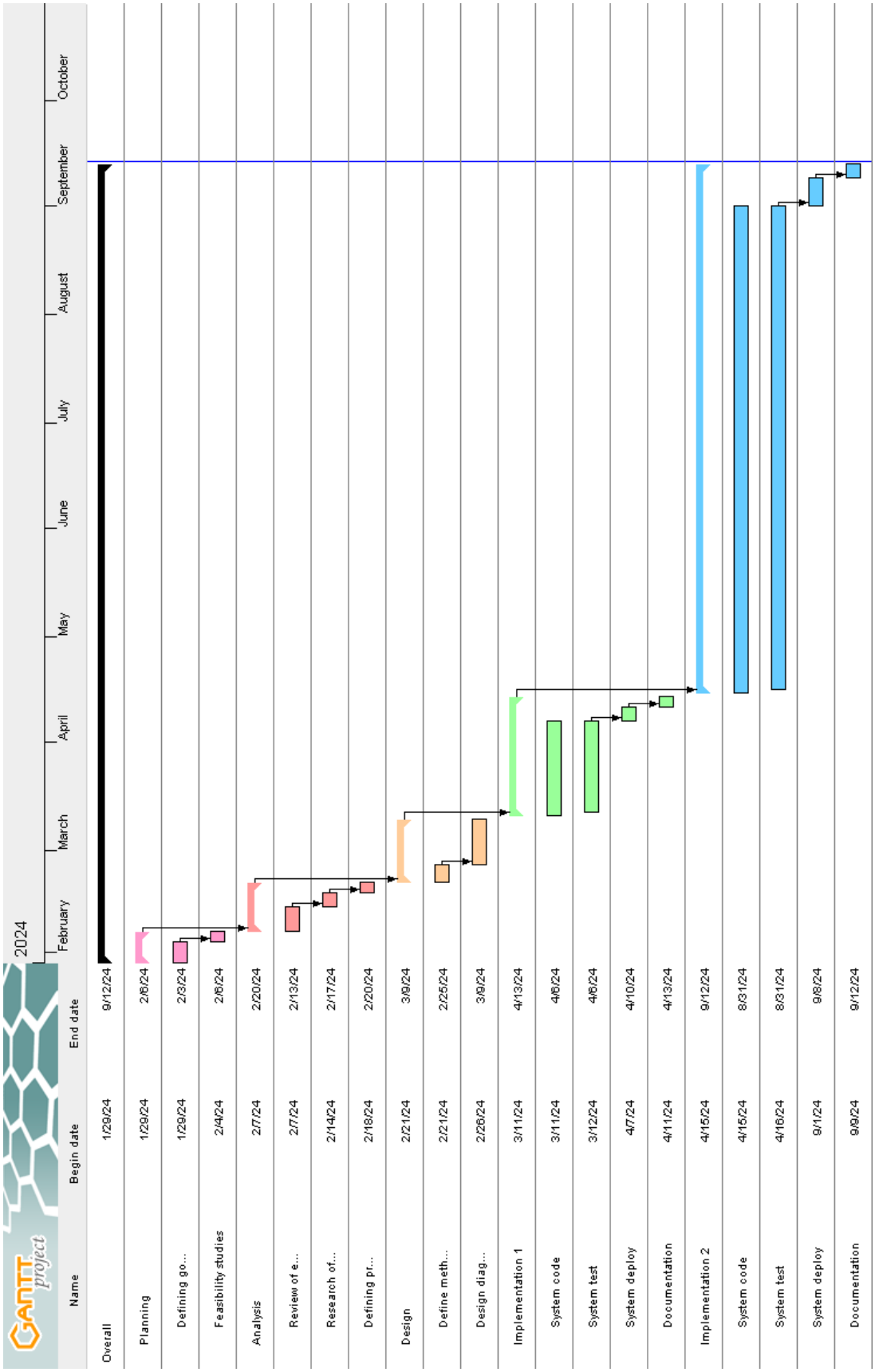


Figure 23 Gantt Chart

3.5 Estimated Cost

In this project, the cost is mainly spent on the usage of Google Map API such as the Maps SDK for Android, Directions API, and more. These APIs offer core functionalities for this project such as the mapping functionality. Billing report is generated upon calling the API requests in the Google Cloud platform to keep track of the costs associated with API requests.

Items	For FYP Development	For Commercialization
• Google Map API	• <RM500	• RM0

Table 4 Estimated Cost

3.6 Concluding Remark

In this chapter, various system methodologies have been explored ranging from waterfall development, parallel development, passed development, and prototyped-based methodology. Through the review and analysis of various system methodologies, prototype-based methodology is chosen to be used in this project, as it allows for rapid iteration, and more flexibility to accommodate changes and requirements. In addition, various software technologies have been outlined to implement the overall system. Flutter will be used as the main framework for this application. Also, the functional requirements have been listed ranging from map function, select bus routes, monitor real-time location of bus, and push notification for forecast weather, has play important role that fulfil the functionalities of this application, as well as to meet the project's objectives. System testing and performances will be carried out to analyse the functionalities, and resolve any potential issues found. Last but not least, an overview of project milestones and estimated costs associated with the project are presented.

CHAPTER 4: SYSTEM DESIGN

4.1 System Architecture

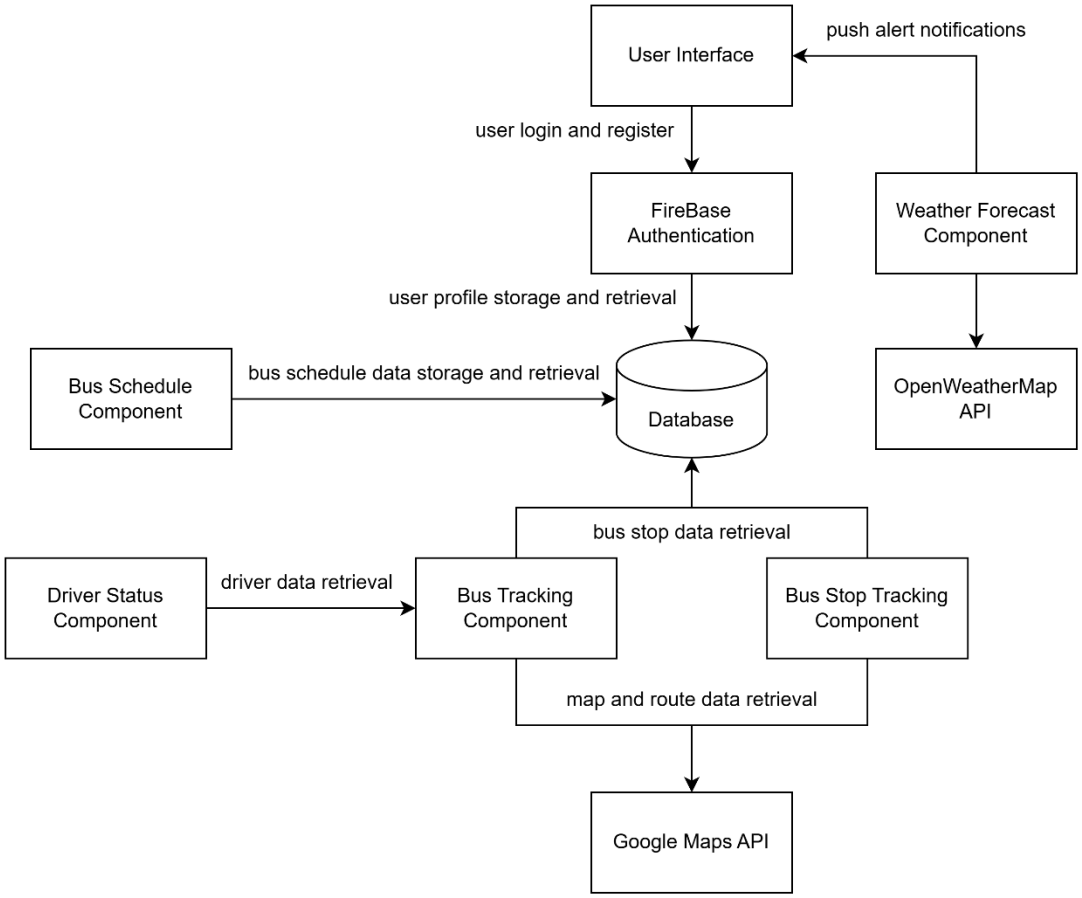


Figure 24 System Architecture Diagram

The system architecture diagram illustrated the various components ranging from user interface, bus tracker component, bus stop tracking component, weather forecast component, bus schedule component, and driver status component, these connections that bring to the whole system. The database acts as a repository to provide backend services, is responsible for handling user identification and authentication, storing bus schedule date and bus tracking. The user authentication component guarantees safe access to the application's functionality through login and register features. The weather forecast component obtains real-time weather data from the external OpenWeatherMap API, and triggers push notifications to notify users about the adverse weather conditions. Furthermore, the transportation tracking component has utilized GPS technology and Google Maps API to track the current location of buses, display

markers and routes on the application map interface, and provide real-time bus information such as the route distance and estimated time arrival. Additionally, the bus stop tracking component allow users to obtain information about the nearest bus stop around and provide the estimated time and distance for them to reach the nearest bus stop. Plus, the bus schedule component allows user to obtain early information about the bus time arrival at particular bus stop. On the other hand, the driver status component is specifically designed for bus driver to obtain their current location utilizing GPS technology once they are assigned and operating on a route.

4.2 Functional Modules in the System

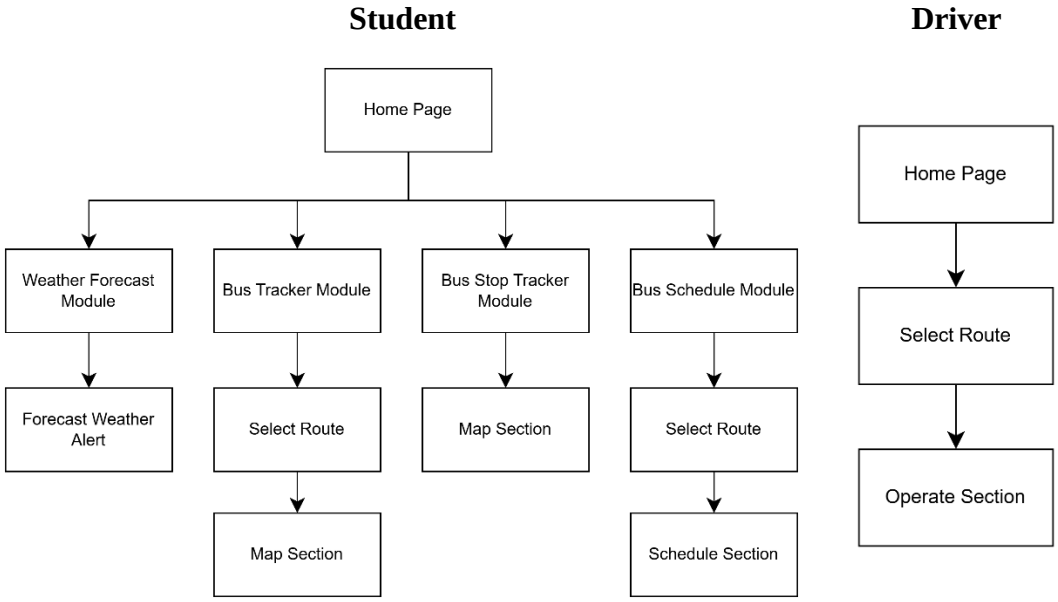


Figure 25 Functional Module Diagram

In the student section, the system mainly consists of four modules ranging from weather forecast module, bus tracker module, bus stop tracker module and bus schedule module. When users (students) are in the home page, users have several options whether they want to access to the information of the weather conditions, real-time bus information, nearest bus stop information or the bus schedule information. In the

weather forecast module, users can gain information about the forecast weather conditions for the day. This module also designed to push notification to alert users about the adverse weather conditions such as raining in the following hour. Whereas in the bus tracker module, users can choose their preferable route to monitor the current location of the corresponding bus and access to the real-time information such as route distance and estimated arrival times in the map section. Additionally, users can also gain access to the information of the nearest bus stop such as the estimated time and distance to the bus stop in bus stop tracker module. Last but not least, users can also view the overall bus schedule which will be updated from time to time in the schedule module. Users can select their preferable route to view the corresponding schedule.

In the driver section, driver will need to select route that they are assigned to and activate it to start the bus tracking feature which also the primary function of this mobile application.

4.3 System Flow

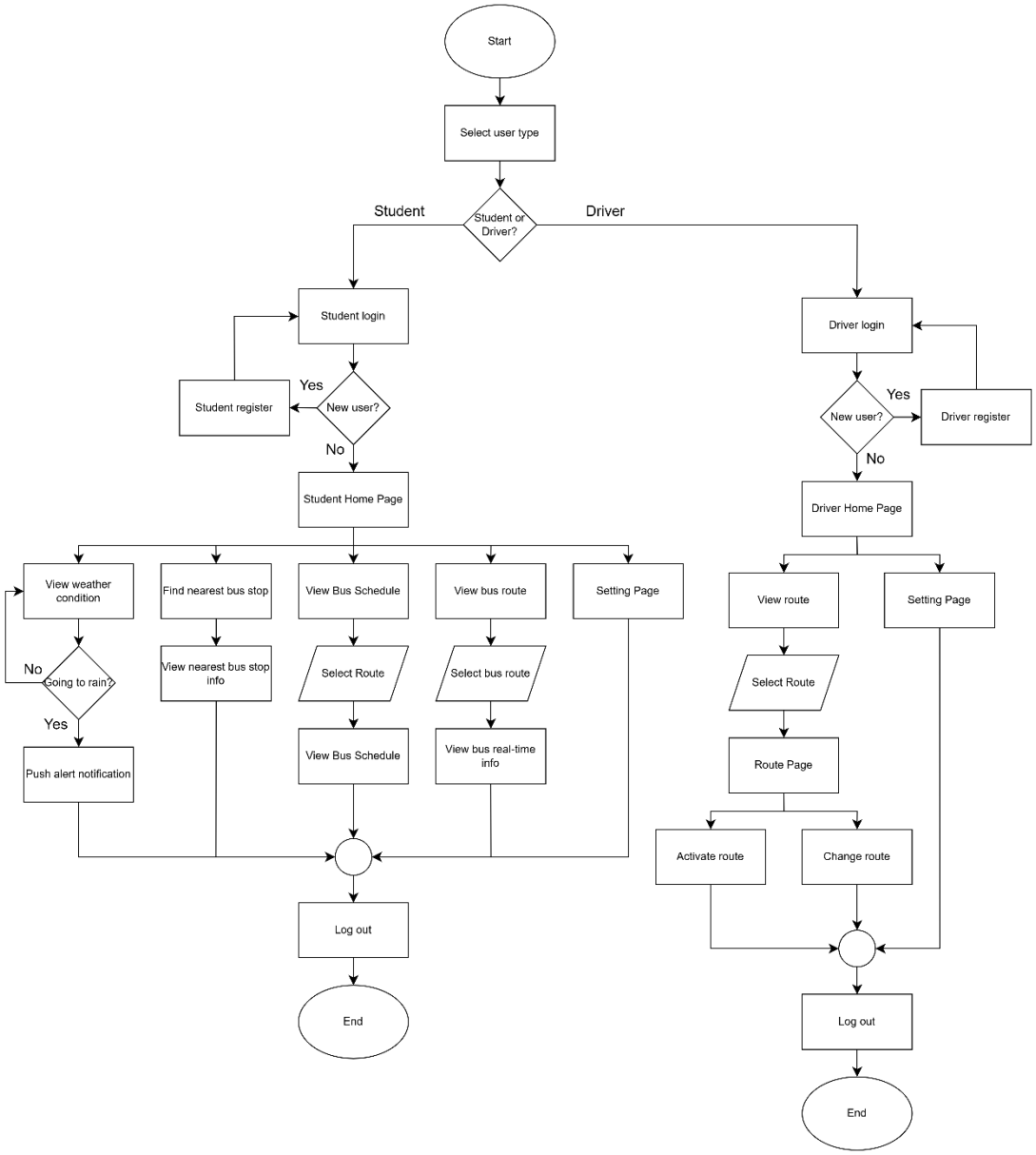


Figure 26 Flowchart

4.4 Use-Case Diagram

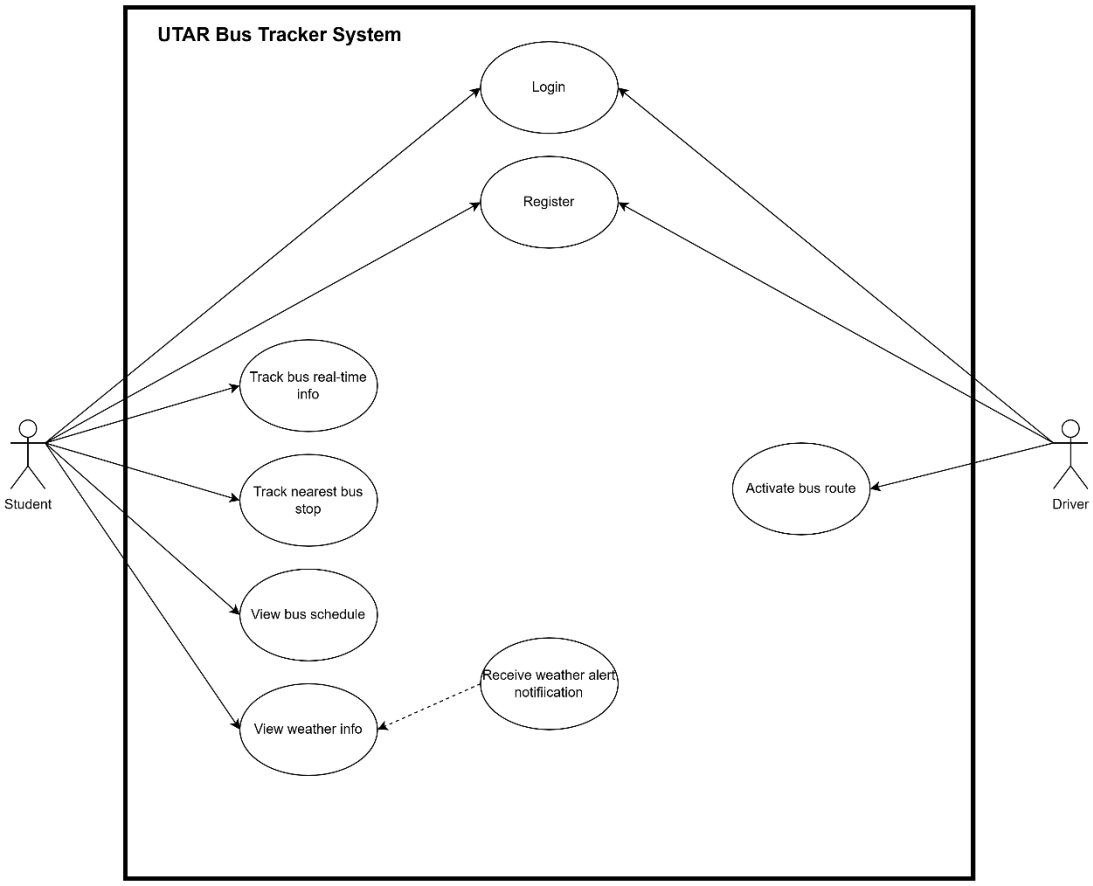


Figure 27 Use-case Diagram

4.5 GUI Design

4.5.1 Student

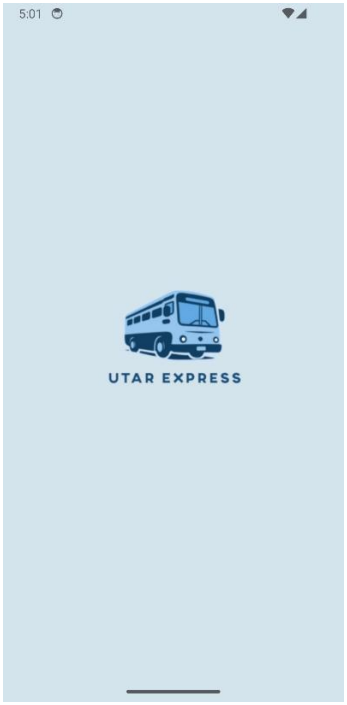


Figure 28 Splash Screen Design

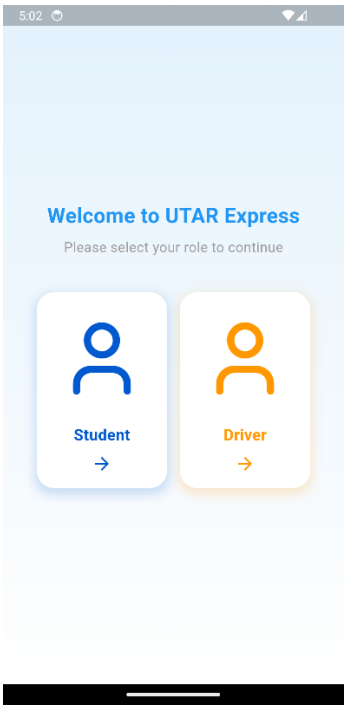


Figure 29 Select User Type Page Design

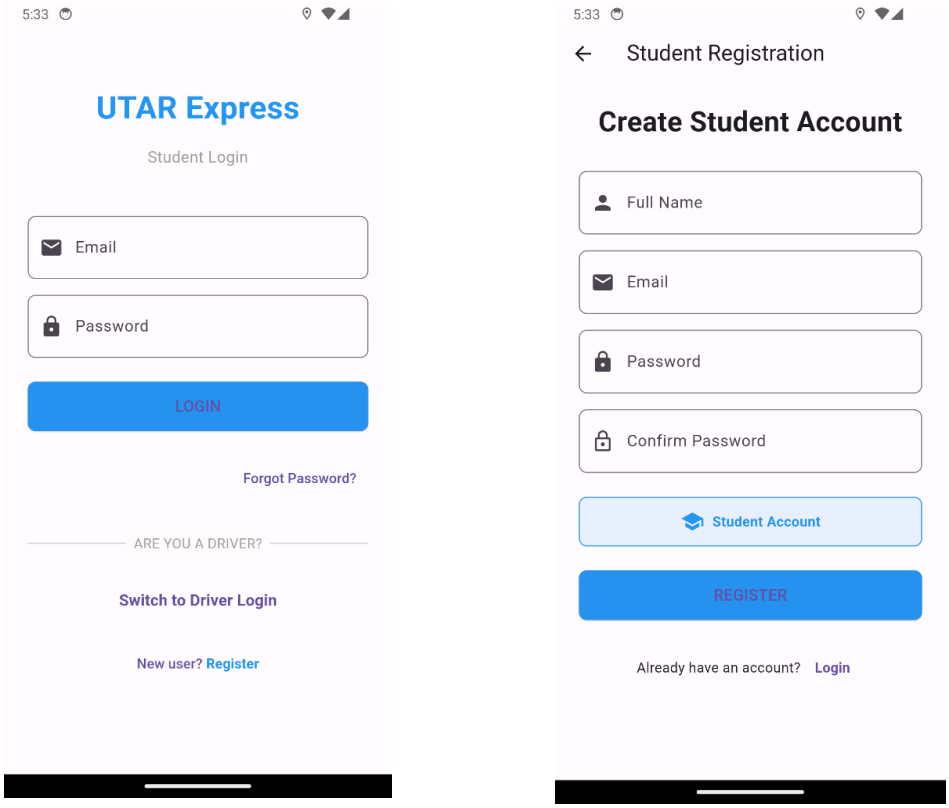


Figure 30 Login and Register Page Design

Figure 31 Home Page Design

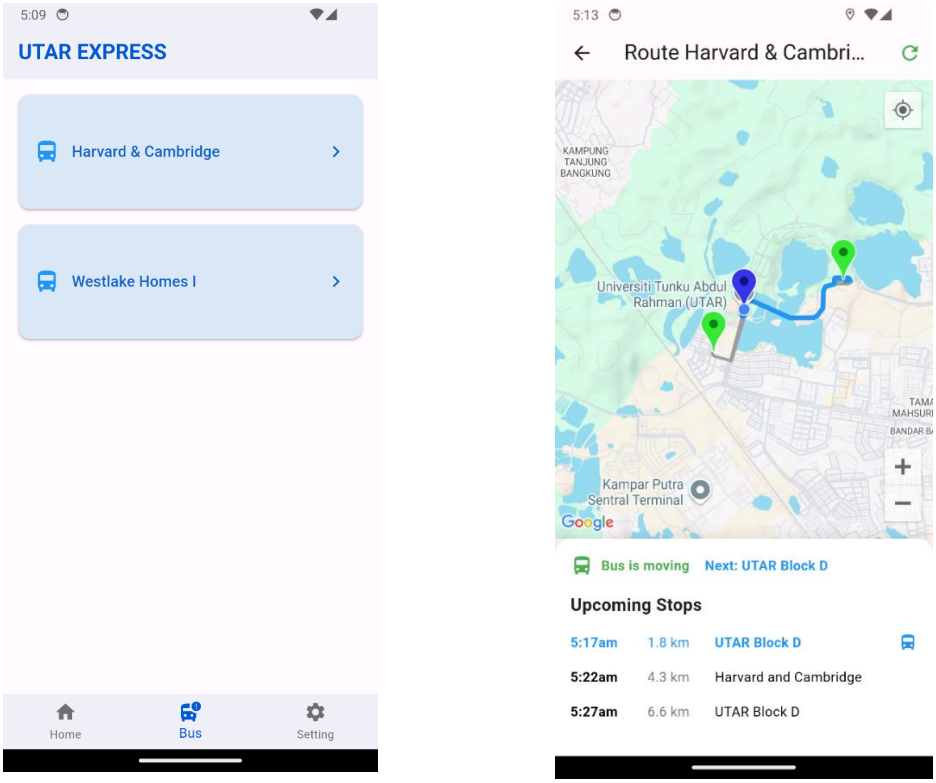


Figure 32 Bus Routes Selection Page and Bus Tracking Page Design

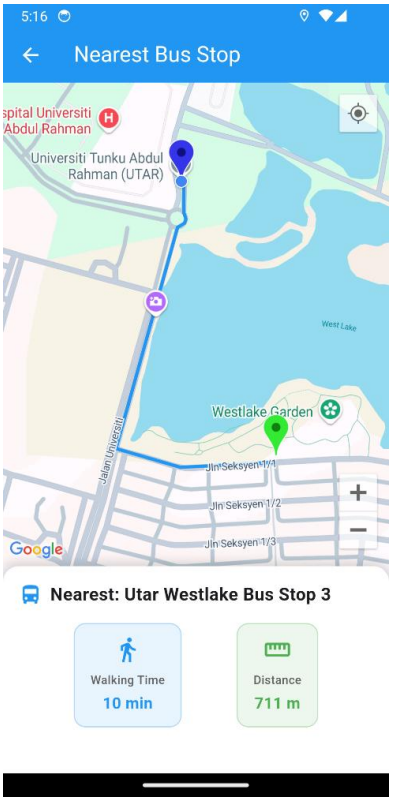


Figure 33 Nearest Bus Stop Page Design

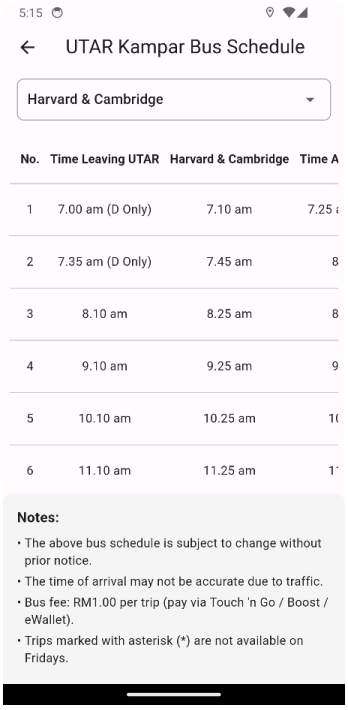


Figure 34 Bus Schedule Page Design

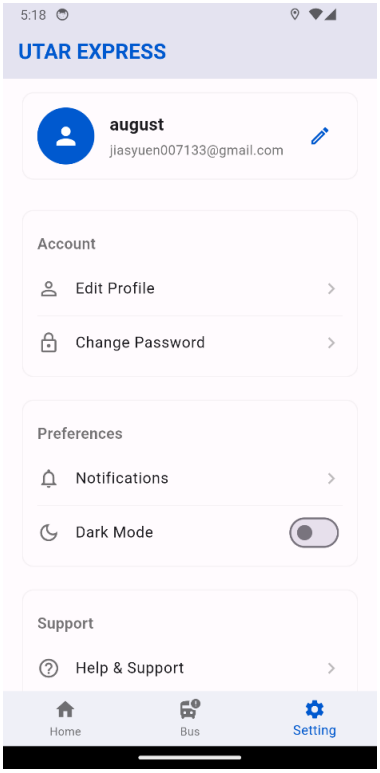


Figure 35 Setting Page Design

4.5.2 Driver

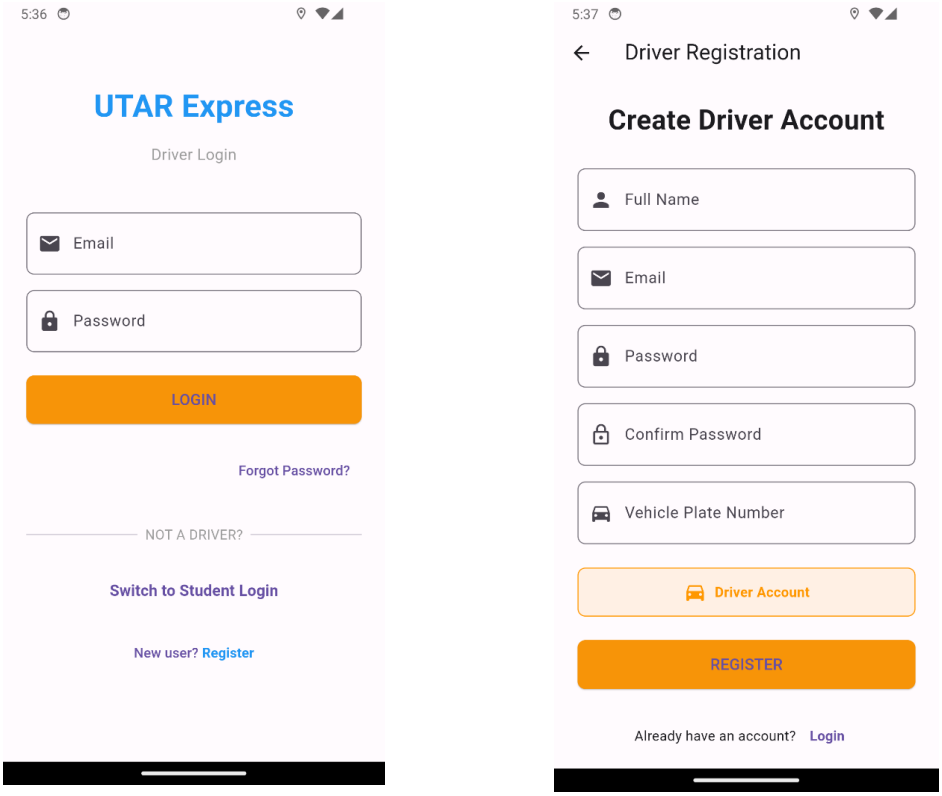


Figure 36 Driver Login and Register Page Design

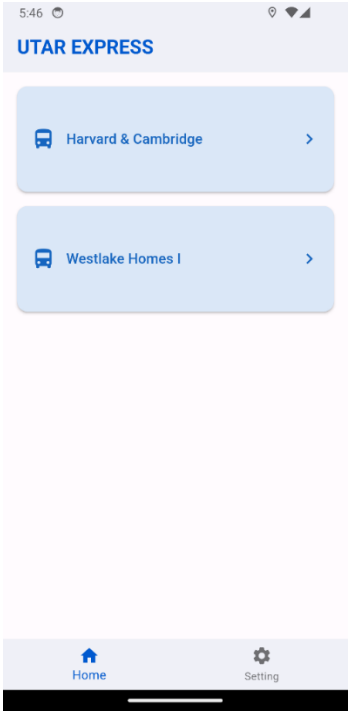


Figure 37 Driver Home Page Design

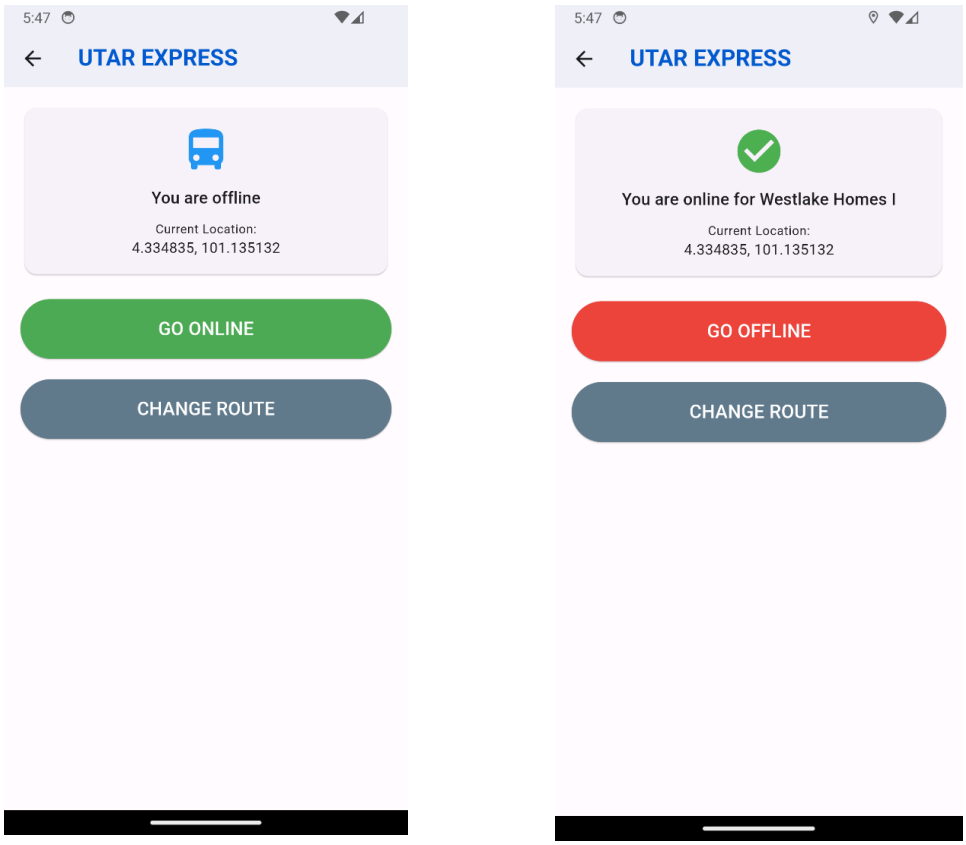


Figure 38 Driver Route Home Page Design

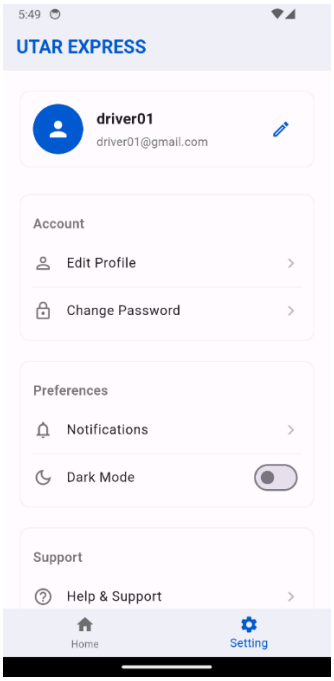


Figure 39 Driver Setting Page Design

4.6 Concluding Remark

In this chapter, we have explored the system design for the UTAR transportation tracker application. The system architecture provides an overview of various components and how they connect with each other to form the entire system. Later, we discussed the functional modules, outlining all the key functionalities that will be implemented in the application to achieve the project's goal. Next, the demonstration of sequence activities and operations between different modules are presented in the system flow. The use-case diagram additionally offered a visual representation of actors, use cases, and interactions inside the system. Finally, the GUI design illustrates the user interface design of the application.

CHAPTER 5 SYSTEM IMPLEMENTATION

5.1 Hardware Setup

To install the application onto our mobile phone, we will first need to open settings in our phone, locate **About Phone** bar. Next, we find **Build Number** bar and tap on it 7 times until a message "You are now a developer" prompted. Later, we navigate back to the setting page, and we will now see the **Developer Options** on the setting panel. Within the Developer Options, we enable **USB debugging**. After that, we use a cable to connect the phone to our PC. Then, a USB debugging prompt will appear on our phone, and we can check the "**Always allow from this computer**" message. Finally, we can run the application on our mobile phone.

5.2 Software Setup

The software that I used to develop the application:

- **Visual Studio Code**
  We can simply just download the software from the official Visual Studio Code website.

- **Flutter SDK**
  We downloaded the Flutter SDK from the official Flutter website. The SDK was extracted into directory "C:\src\flutter", we copy the binary path and add into the system's path environment variable to enable Flutter commands in the terminal. Later, we can check if it configured properly in our laptop by running command "flutter doctor". After completing all the setup, we can simply just create a new project by running command "flutter create project_name".

- **Android Studio**
  First, we downloaded Android Studio from the official Android Studio website. Next, we opened the AVD Manager in Android Studio to create a virtual emulator. Later we integrated it into the visual studio code to run the application.

5.3 Setting and Configuration

5.3.1 Firebase

To utilize Firebase features in our project, we will need to create a project in the firebase console and register our flutter app within the firebase project. First, we download the Firebase CLI binary from the official Firebase website, so we can run firebase command in our terminal. After running the Firebase CLI, we will need to login to our google account. Next, we run the command ‘dart pub global activate flutterfire_cli’ to install FlutterFire CLI tool on our system, which helps to automate Firebase configuration for Flutter projects. After that, we go to the folder where the flutter project locates and run the command ‘flutterfire configure’ to create Firebase project. In our flutter project, we will now see the firebase_options.dart file after configured successfully.

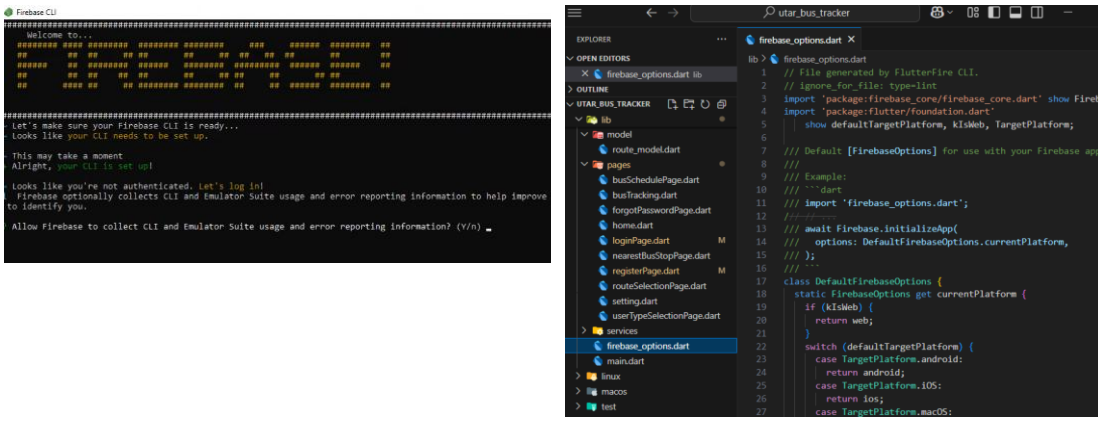


Figure 40 Firebase Setup

- **Firebase Authentication**

  Continuing from the above configuration, we can now see our Firebase project in the Firebase console. To use the Firebase authentication feature, we will need to activate the desired sign-in method. In our project, we will use the Email/Password as the application sign-in method.

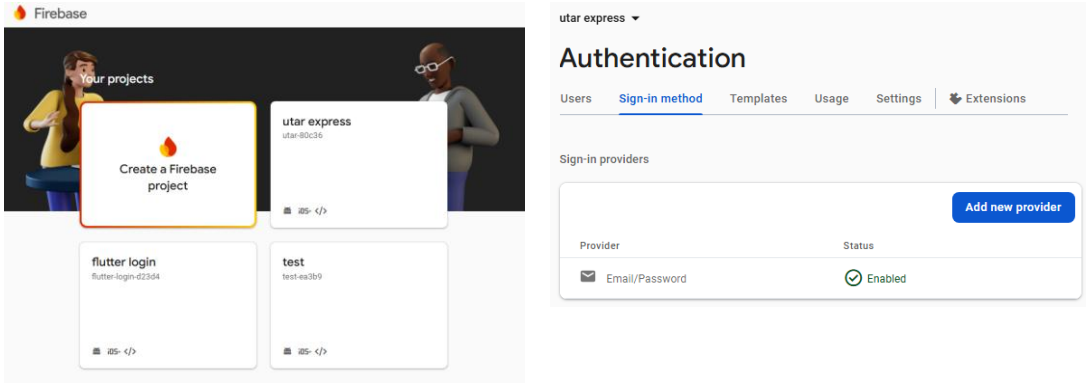


Figure 41 Firebase Authentication Setup

In our flutter project, we will need to add firebase_core and cloud_firestore in the dependencies in pubspec.yaml.

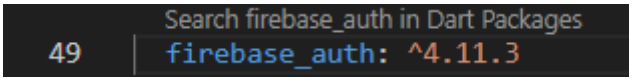


Figure 42 Required Dependencies for Firebase Authentication in Flutter

After activating the sign-in method, we can then test the function by registering some new users and viewing the information on the users' table.

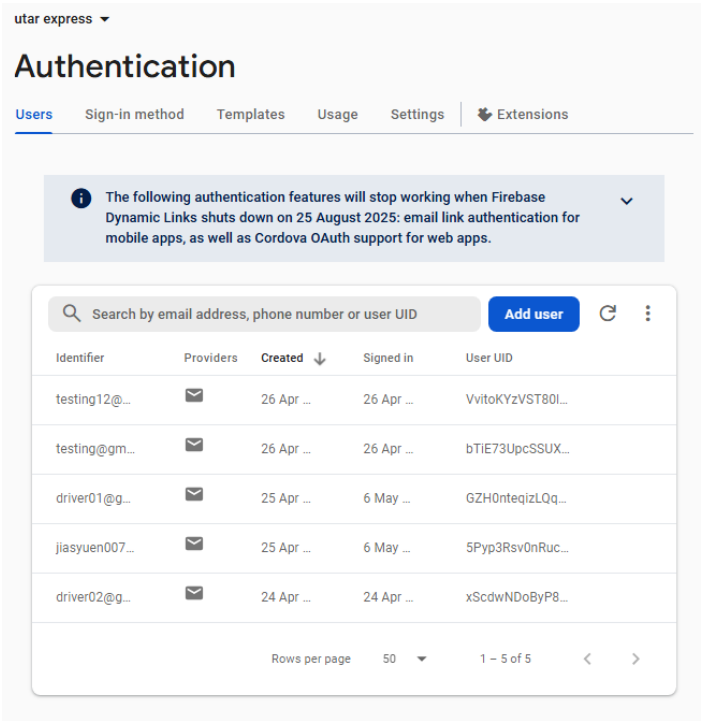


Figure 43 Firebase Authentication Users Table

- **Firebase Firestore**

  To store additional data in the database, we will need to create a Firestore database in the Firebase Console. When creating the database, we will need to set the name and location and configure the secure rules. We have chosen asia-southeast1 as our database location. In the secure rule tab, we selected 'start in test mode' for development purpose.

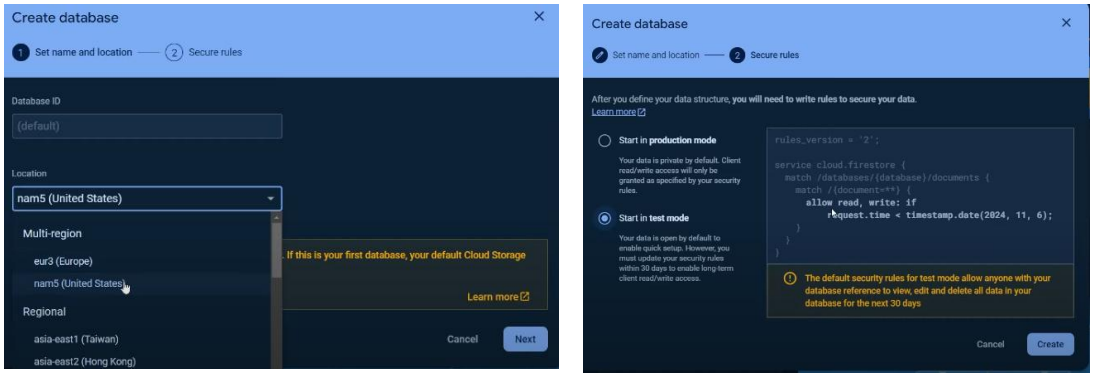


Figure 44 Firebase Firestore Setup

In our flutter project, we will need to add firebase_core and cloud_firestore in the dependencies in pubspec.yaml.

```
firebase_core: ^2.32.0
Search cloud_firestore in Dart Packages
cloud_firestore: ^4.8.3
```

Figure 45 Required Dependencies for Firestore in Flutter

After configured properly, we can implement CRUD operation code to play with the data in Firestore. And we can view the data in the data table.

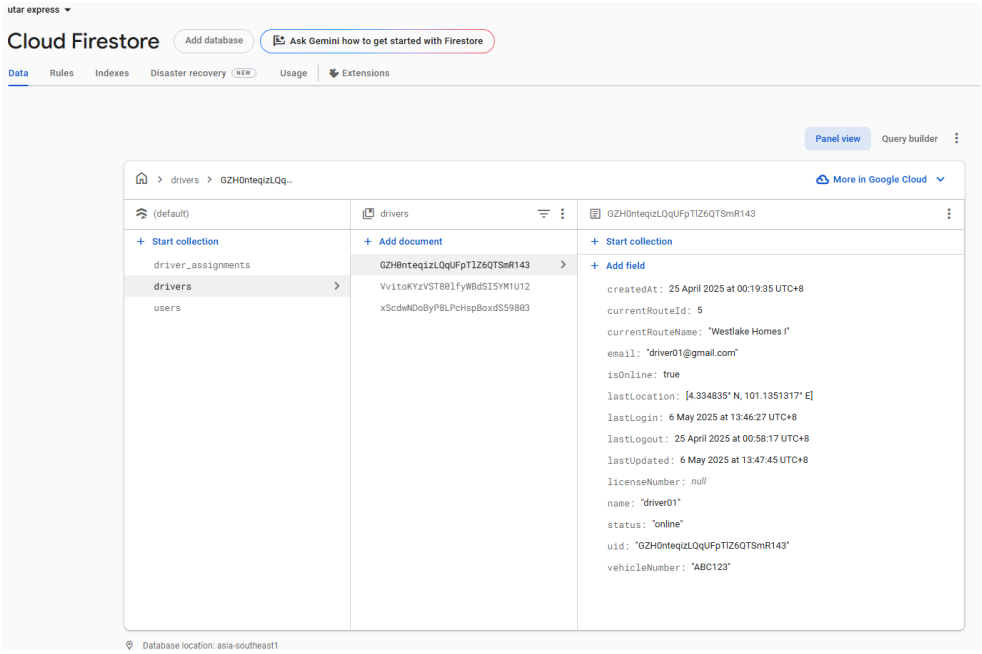


Figure 46 Cloud Firestore Data Table

- **Firebase Real-time Database**

  To play with real-time data, we can navigate to Realtime database in Firebase console, create database and selected “start in test mode” for development purpose. In our Flutter project, we need to add firebase_database to the dependencies in pubspec.yaml.

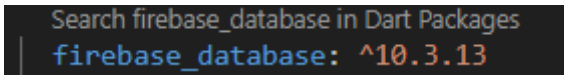


Figure 47 Required Dependencies for Realtime Database in Flutter

After configured properly, we can implement CRUD operation code to play and observe the real time data in the Realtime database data table.

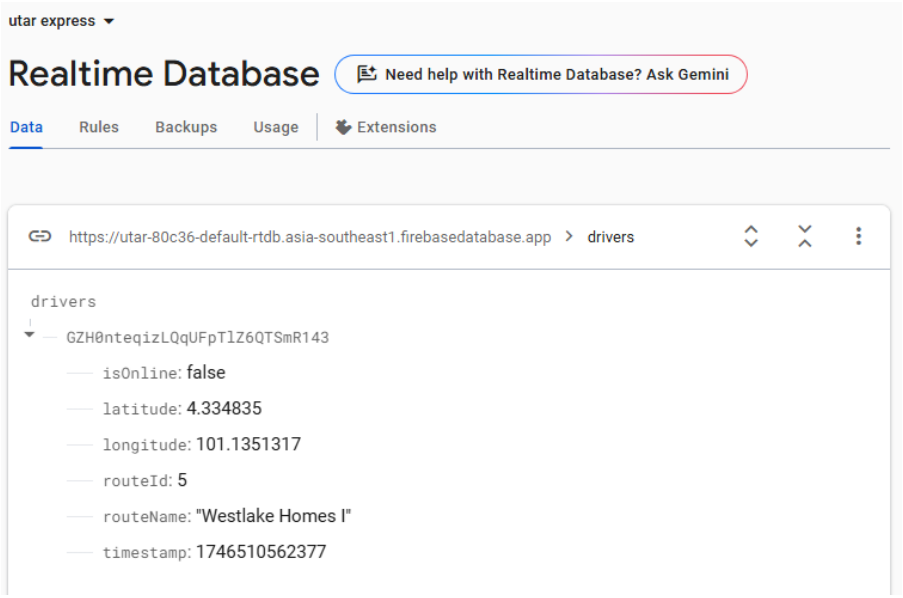


Figure 48 Realtime Database Data Table

5.3.2 Google Maps API

To utilize Google maps API for several functions such as map display, polylines, estimated time arrival, we need to integrate Google Map API into our flutter project. First, we need to add google_maps_flutter, flutter_polyline_points, and https into the dependencies in pubspec.yaml.

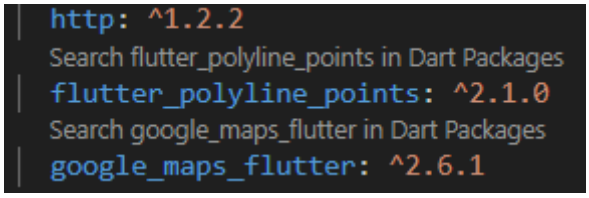


Figure 49 Required Dependencies for Google Maps API

Next, we can create an account in Google Maps Platform, create a project and generate an API key. We also need to enable Maps SDK for android, Maps SDK for iOS, and Direction API in the API library.

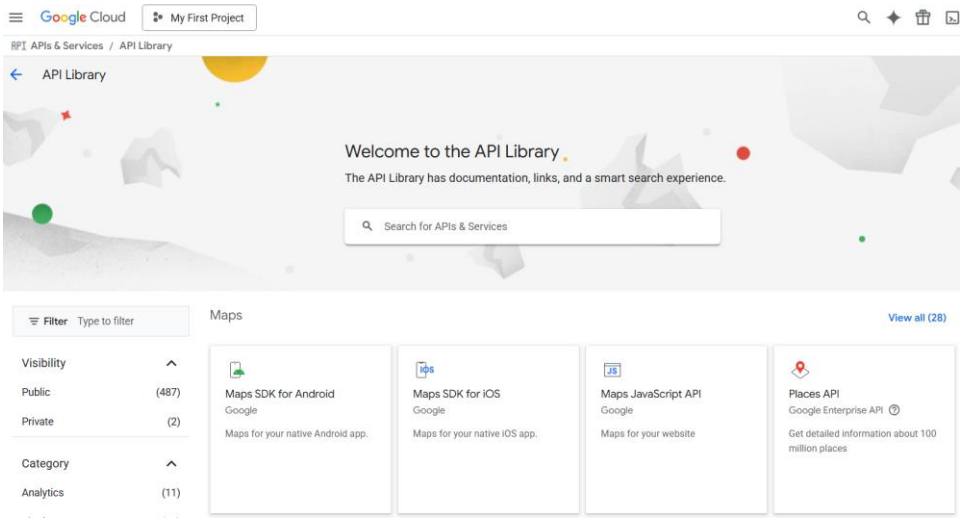


Figure 50 Enable Services in Google Maps API Library

In our Flutter project, we need to add some lines in the AndroidManifest.xml for android and AppDelegate.swift for iOS to integrate the API key.

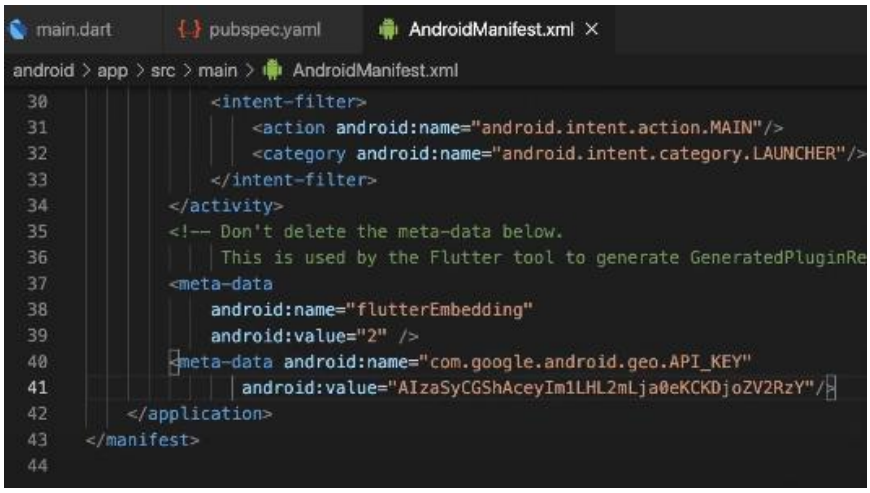
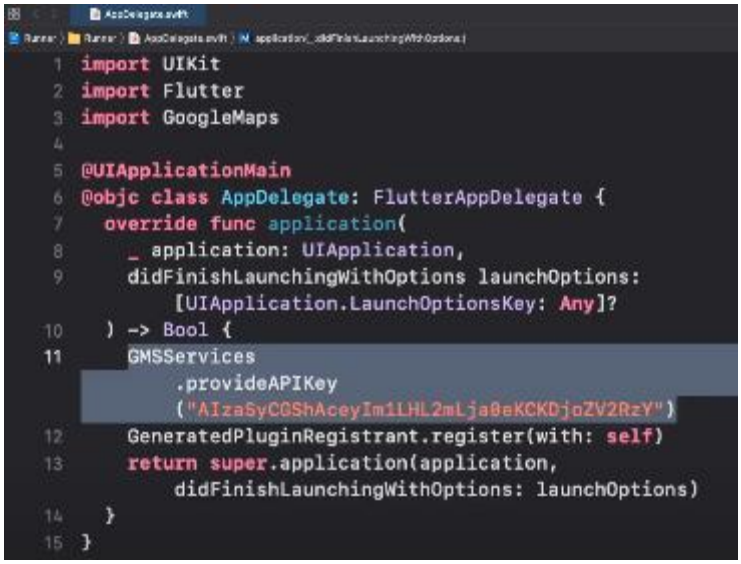


Figure 51 Google Maps API setup in AndroidManifest.xml



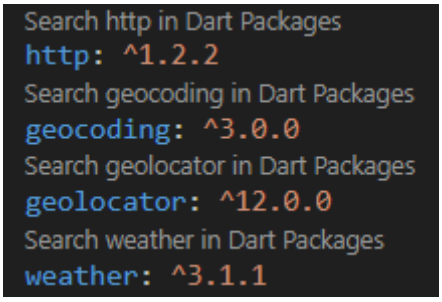
```swift
import UIKit
import Flutter
import GoogleMaps

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
  override func application(
    _ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions:
        [UIApplication.LaunchOptionsKey: Any]?
  ) -> Bool {
    GMSServices
        .provideAPIKey
        ("AIzaSyCGShAceyIm1LHL2mLja0eKCKDjoZV2RzY")
    GeneratedPluginRegistrant.register(with: self)
    return super.application(application,
        didFinishLaunchingWithOptions: launchOptions)
  }
}
```

Figure 52 Google Maps API setup in AppDelegate.swift

5.3.3 OpenWeatherMap

To implement weather forecasting features utilizing OpenWeatherMap API, we will first need to add http, geolocator, geocoding, and weather into the dependencies in pubspec.yaml.



```
Search http in Dart Packages
http: ^1.2.2
Search geocoding in Dart Packages
geocoding: ^3.0.0
Search geolocator in Dart Packages
geolocator: ^12.0.0
Search weather in Dart Packages
weather: ^3.1.1
```

Figure 53 Required Dependencies for OpenWeatherMap API

Next, we created an account on the OpenWeatherMap official website and generated an API key. In our Flutter project, we will need to implement the weather_service.dart file and integrated the API key.

firebase_options.dart | pubspec.yaml | weather_service.dart M

lib > services > weather_service.dart > fetchWeatherForecast

```dart
import 'package:geocoding/geocoding.dart';
import 'package:geolocator/geolocator.dart';
import 'package:weather/weather.dart';
import 'package:logger/logger.dart';


Future<Weather> fetchWeather({required String city}) async {
  String apiKey = 'e94fc3a62dcd9bf3d9e8db6fdf7bac77';
  WeatherFactory wf = WeatherFactory(apiKey);
  Weather weatherOutput = await wf.currentWeatherByCityName(city);

  logger.i('Current Weather for $city:');
  logger.d('Temperature: ${weatherOutput.temperature?.celsius}°C');
  logger.d('Weather Condition: ${weatherOutput.weatherMain}');
  logger.d('Description: ${weatherOutput.weatherDescription}');
  logger.d('Humidity: ${weatherOutput.humidity}%');
  logger.d('Wind Speed: ${weatherOutput.windSpeed} m/s');

  return weatherOutput;
}
```

Figure 54 Weather Service Code Implementation

5.4 System Operation

- **Splash Screen**

Figure 55 Splash Screen

A simple splash screen with application logo.

- **User Type Selection Page**

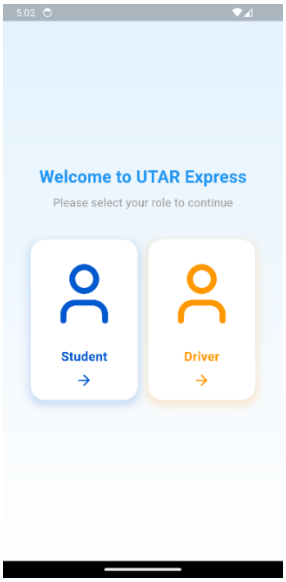


Figure 56 User Type Selection Page

Users are required to select their role whether they are students or drivers, and it will navigate users to the login page respectively.

5.4.1 Student

- **Login Page**

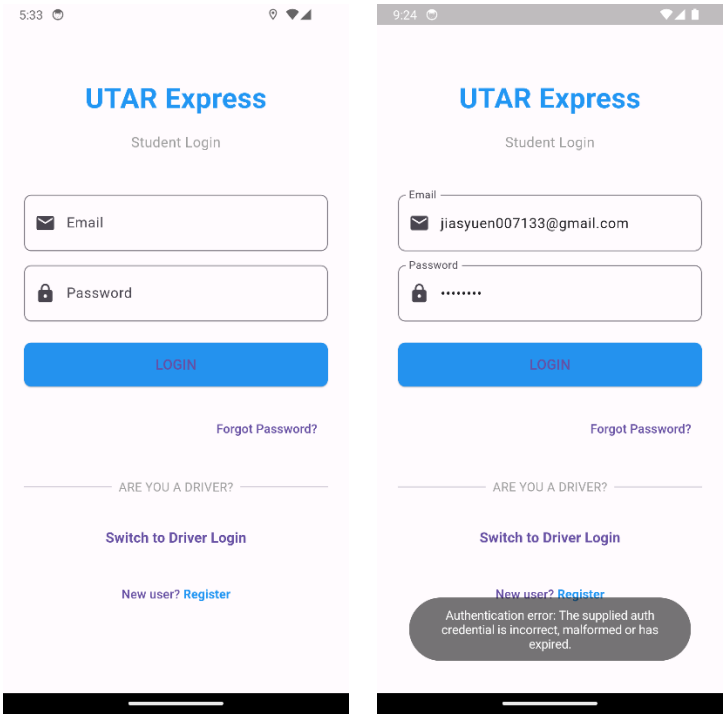


Figure 57 Login Page

The login page allows users to access their accounts by entering their email and password. When users input their email and password, Firebase authentication will check if the email and password match the data in the database. Upon successful authentication, the login page will navigate user to the home page. In contrast, error messages will prompt upon unsuccessful authentication.

- **Register Page**

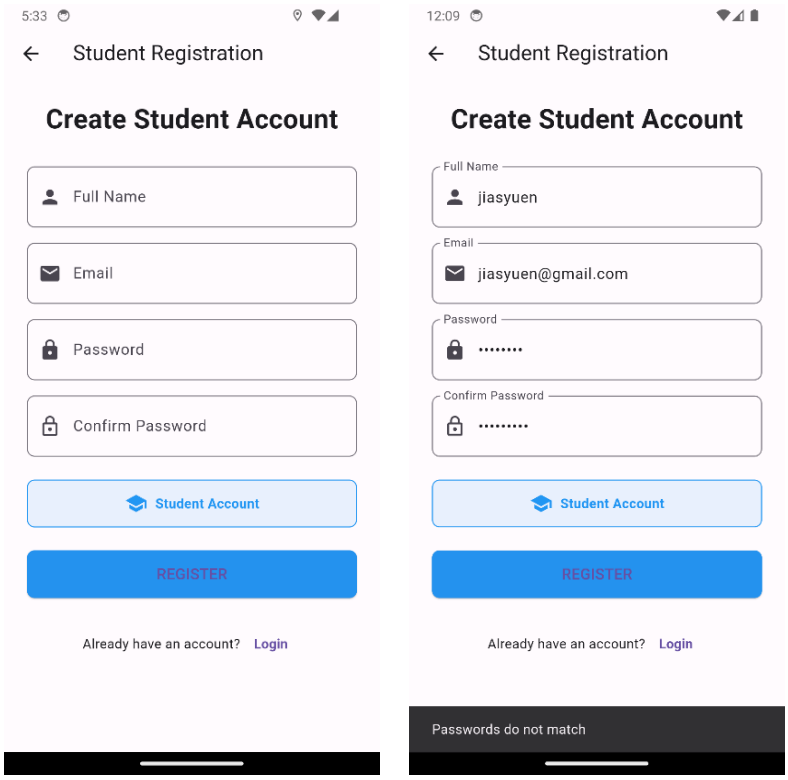


Figure 58 Register Page

The register page allows users to register a new account by taking a few parameters ranging from full name, email, password, and confirm password. Firebase authentication has its own default validation to confirm the registered details are valid such as the email address format validation, and the password and confirm password are match. Upon successful registration, the register page will navigate user to login page. In contrast, error messages will prompt unsuccessful authentication.

- **Home Page**

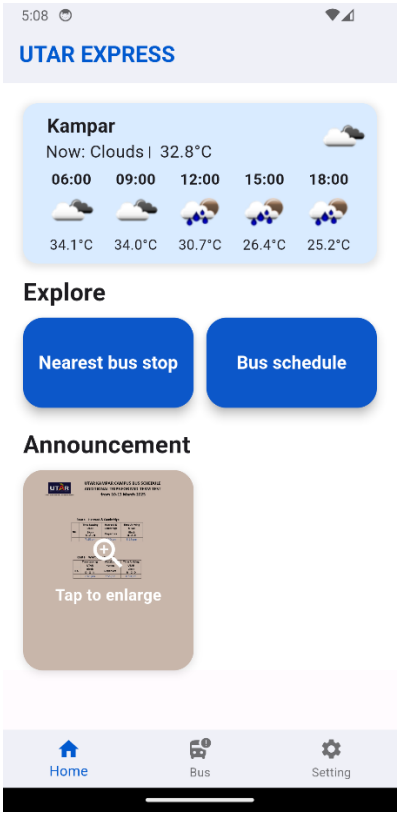


Figure 59 Home Page

Users will navigate to home page after successfully logging in. There are several components on the home page. First is the weather forecasting information where users can acknowledge the current and upcoming weather conditions. Next, there are two buttons in the explore category which are the “nearest bus stop” button, used to track the nearest bus stop around users, and “bus schedule” button, used to view bus schedules of routes available. In addition, there is an announcement category where users can tap to enlarge and view the announcement in picture format. Finally, we have three buttons at the bottom navigation which will navigate to home page, bus tracking page, and setting page respectively.

- **Weather Feature**

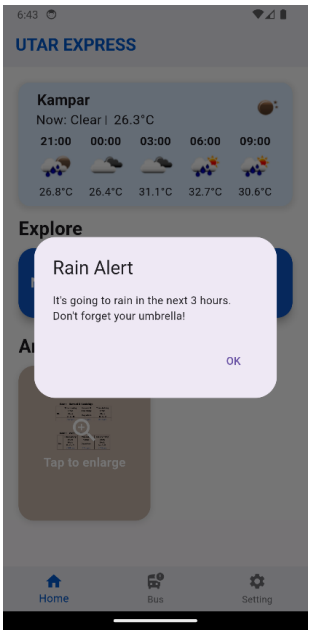


Figure 60 Weather Forecast Alert Prompt

An alert notification about the rain is coming soon will prompt to notify users if the weather forecasting forecasted a raining condition in the next few hours.

- **Bus Tracking**

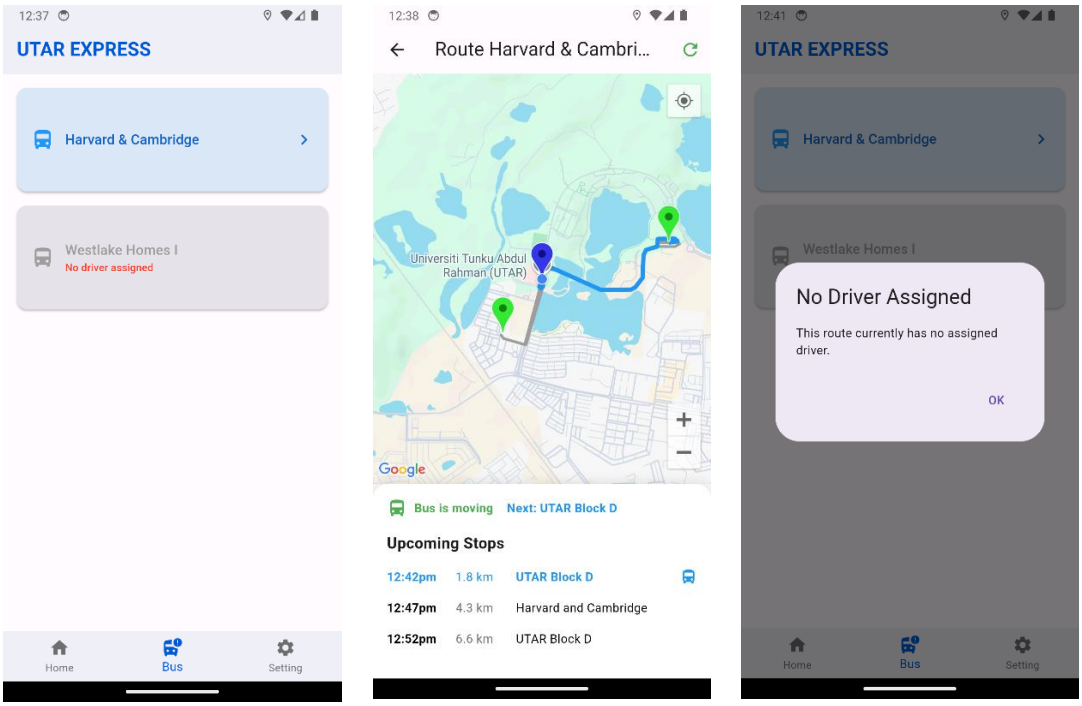


Figure 61 Bus Tracking Page

A bus route selection page will show up when users click on the bus icon at the bottom navigation bar. All the available routes will be displayed here, the route button in blue color indicates that the route is under operation, while the route button in grey color with text “No driver assigned” indicates that the route is offline. Users will be navigated to the bus tracking page when they click on the online route, while a popup dialog box will be prompt when users click on the offline route. In the bus tracking page, users can access real-time bus information such as the estimated time arrival, distance, and the exact location of the bus on the map.

- **Bus Schedule**

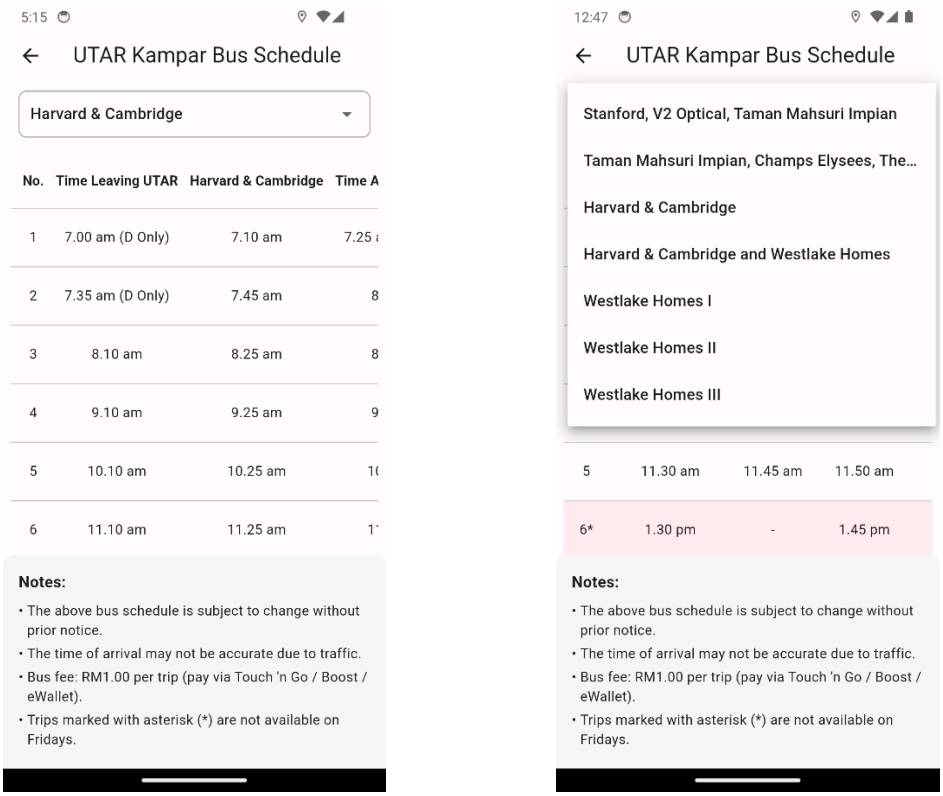


Figure 62 Bus Schedule Page

Users can view bus schedules of all the bus routes and additional information on the bus schedule page.

- **Find the Nearest Bus Stop**

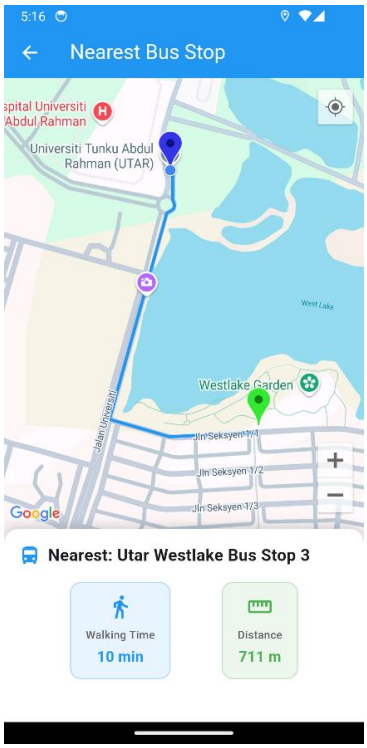


Figure 63 Nearest Bus Stop Page

In the nearest bus stop page, user's location data are used to track the nearest bus stop, the estimated time arrival and distance to the nearest bus stop are estimated utilizing Direction API.

5.4.2 Driver

- **Route Selection Page**

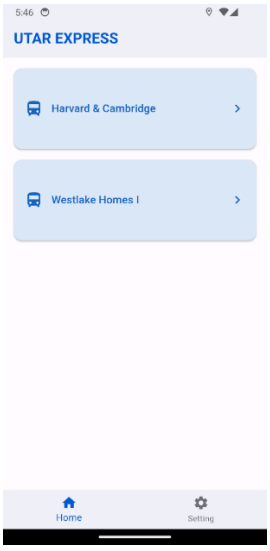


Figure 64 Driver Route Selection Page

After successfully logging in as a driver, users will navigate to the route selection page where all the available route shows. Users can select the route they are going to operate which will navigate to the route home page.

- **Rouge Page**

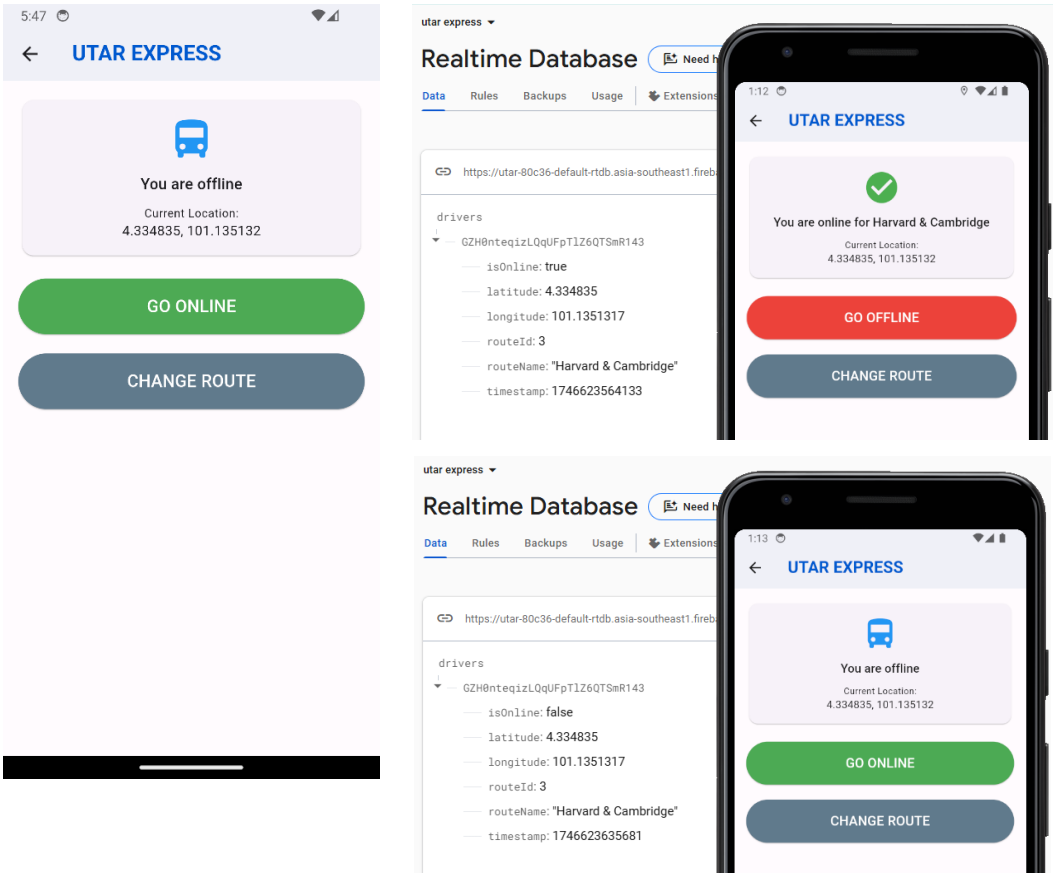


Figure 65 Route Page

In the route home page, there is a simple button toggle “go online” that allows drivers to start operation. Once the button is triggered, the status “isOnline” will be modified and update the route button on the student site route selection page shown above, and the location will be tracked and updated to the Firebase real time database which is later used to display the location of the driver on the student bus tracking page. Drivers can click on the “go offline” button whenever they are off-duty, or “change route” button which navigate them back to the driver route selection page if they are assigned to another route.

5.5 Concluding Remark

This chapter has documented the end-to-end implementation of the whole system, covering the hardware setup, software setup, external API configuration, operational workflows. The development environment was mainly configured using Flutter and Firebase (Authentication, Firestore, and Realtime Database). The steps to integrate software development kit has been demonstrated in this chapter. Finally, the whole system's functionality, workflow, and user interface has been demonstrated with the proof of screenshots.

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

6.1 System Testing and Performance Metrics

There are several major functions that we need to go through testing to make sure the completeness and fully functional of the whole system:

6.1.1 Weather Forecast Alert Notification

- Data retrieval from OpenWeatherMap API
- A raining alert dialog box will popup if raining condition is detected in the weather forecast data

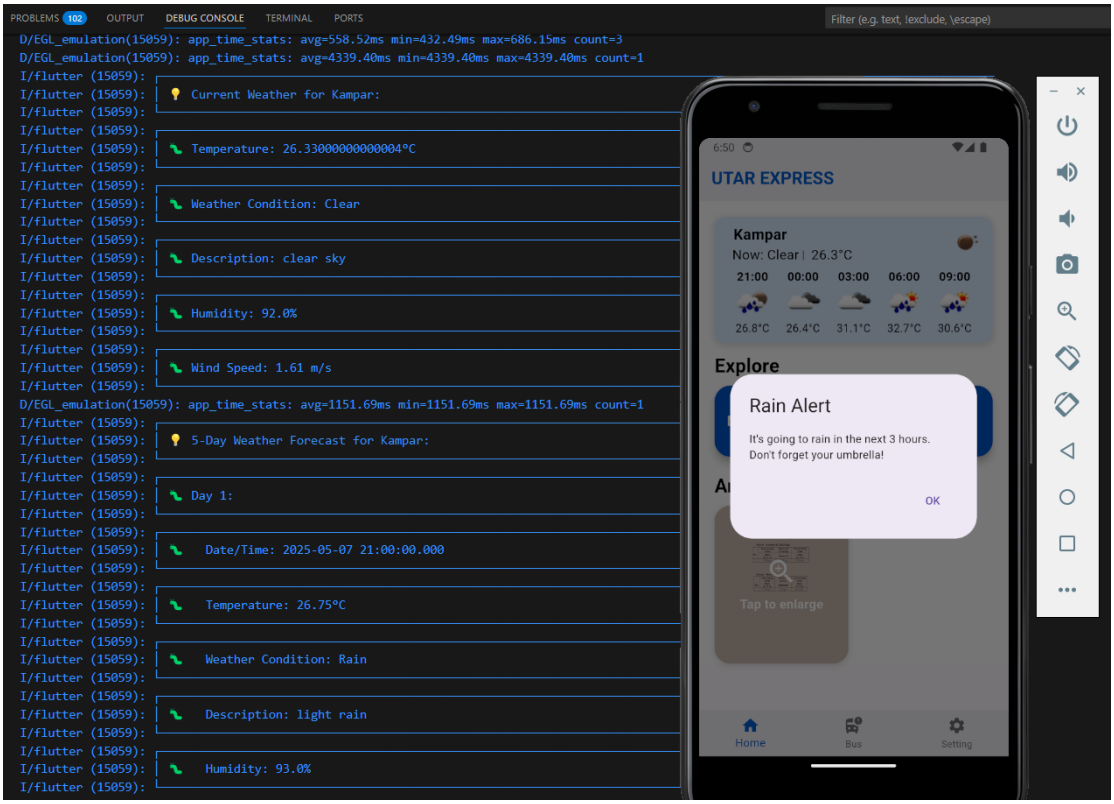


Figure 66 OpenWeatherMap API Data Retrieval

We need to make sure we can make data request call on OpenWeatehrMap through the API key to retrieve the current and forecast weather data. Figure 66 has proven that we can call the weather data of Kampar successfully and a function will be trigger to prompt rain alert dialog box if rain condition is detected in the next coming hours.

6.1.2 Bus Tracking

- Availability of route button in student route selection page

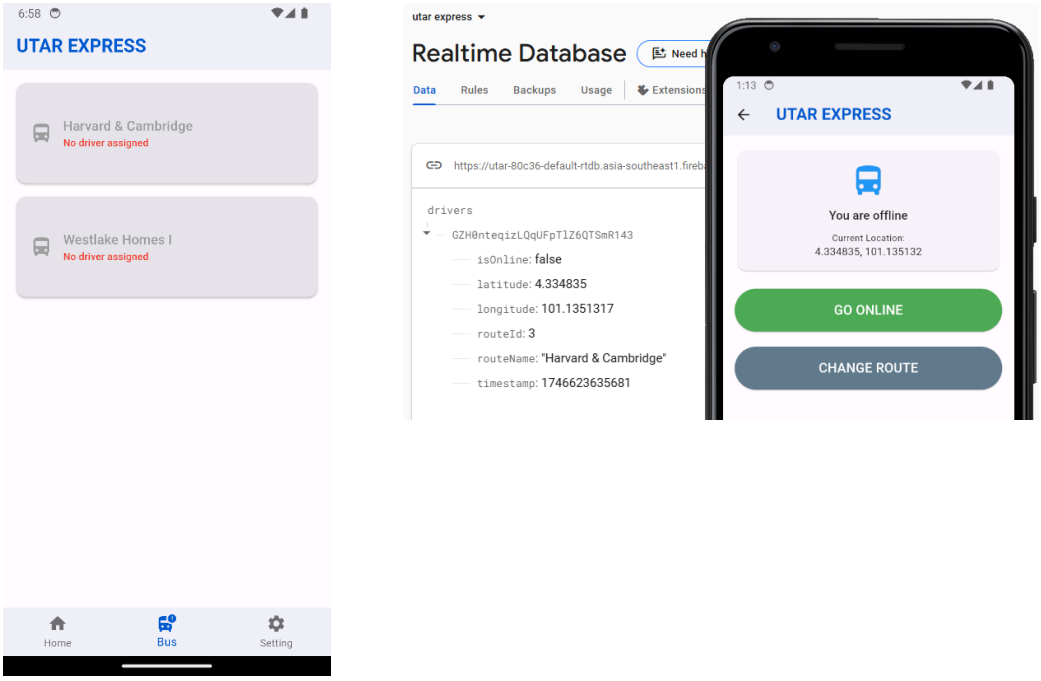


Figure 67 Route Button Disable

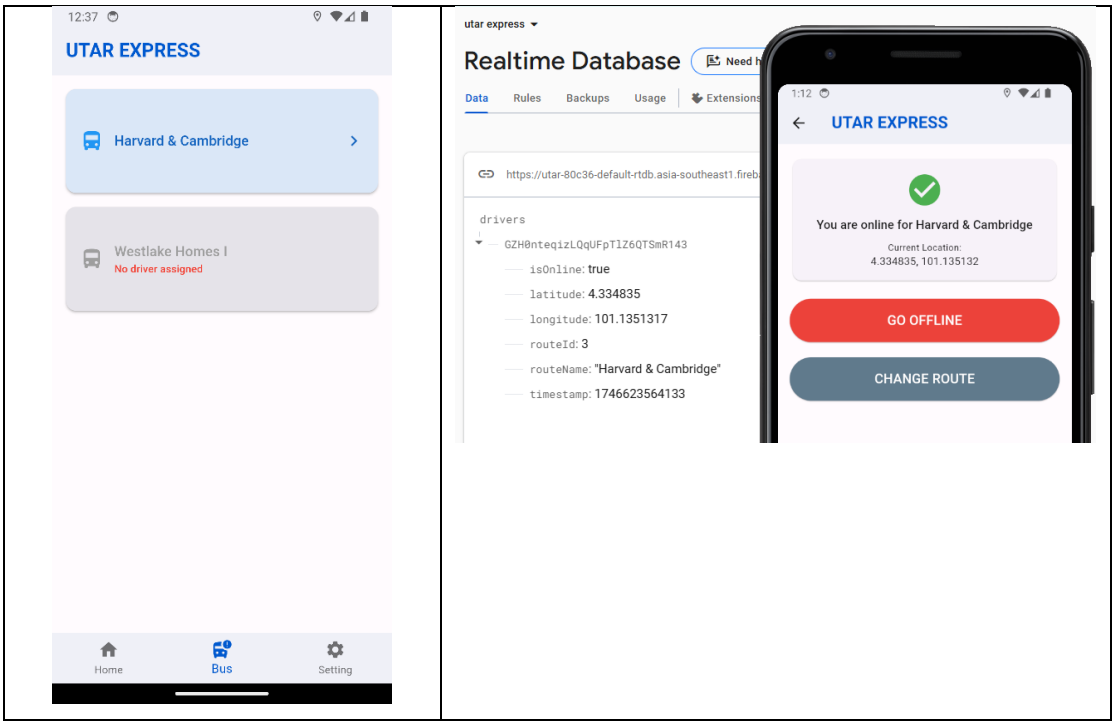


Figure 68 Route Button Enable

We need to make sure the availability of route button in student route selection page based on the status of the driver on the route operation. Given that only if driver has clicked on the “go online” button of the particular route, then only the route button of the particular route in student route selection page is enabled.

- The polyline automates by Direction API between the driver location and all the bus stop
- The data (estimated time arrival and distance) between driver location and all the bus stop

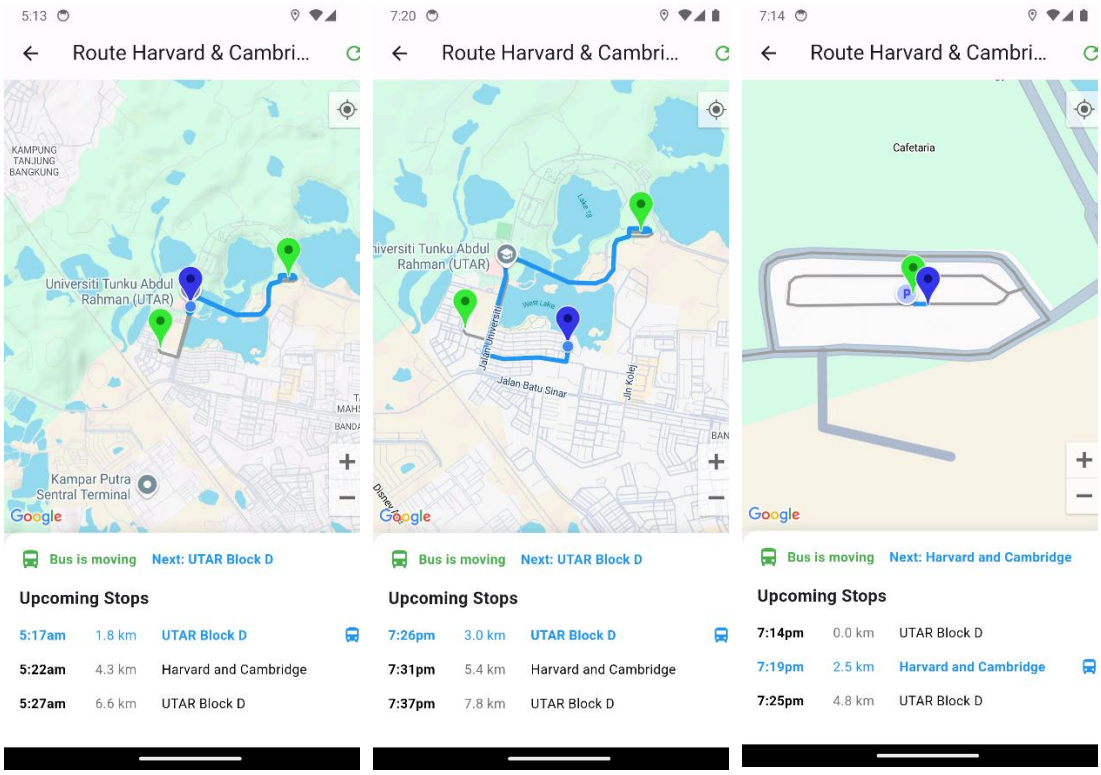


Figure 69 Polylines and Direction API Data from Different Driver Location

From the figure above, we can observe that the polyline and data (estimated time arrival and distance) from the driver location to the bus stop are logically correct and accurate.

- The real time bus location tracking update in Firebase real time database and on student bus tracking page

6.1.3 Track Nearest Bus Stop

- The data (estimated time arrival and distance) between student's location and the nearest bus stop

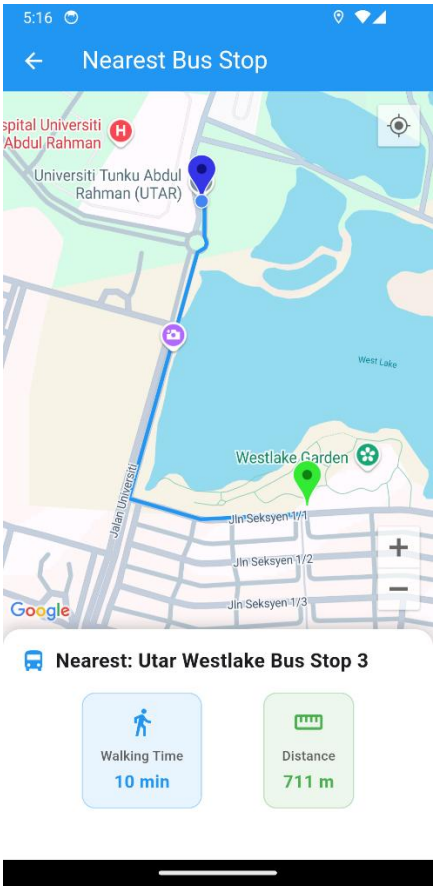


Figure 70 Find Nearest Bus Stop Direction API Data based on Student Location

We can observe that the nearest bus stop can be tracked correctly based on student location including the respective estimated time arrival and distance to the nearest bus stop.

6.2 Testing Setup and Result

Use Case	Test Case	Expected Result	Actual Result	Pass/Fail
Login	Submit empty email or password	Error message prompted	Error message prompted	Pass
	Submit wrong email or password	Error message prompted	Error message prompted	Pass
	Submit correct email and password	Navigate to home page	Navigate to home page	Pass
	Submit student credentials on the driver login page and vice versa	Error message prompted	Error message prompted	Pass
Register	Submit empty input	Error message prompted	Error message prompted	Pass
	Submit short password	Error message prompted	Error message prompted	Pass
	Submit invalid email	Error message prompted	Error message prompted	Pass
	Submit valid credentials	Navigate to login page	Navigate to login page	Pass
Student				
Weather Forecast	Device location is turning off	Prompt dialog box ask user to turn on device location	Prompt dialog box ask user to turn on device location	Pass

	Device location is on	Display current weather and forecast weather info	Display current weather and forecast weather info	Pass
	Rain condition in the next hours	Alert dialog box prompt	Alert dialog box prompt	Pass
Find Nearest Bus Stop	Button clicked	Navigate to nearest bus stop page	Navigate to nearest bus stop page	Pass
	Device location is turning off	Display error message	Display error message	Pass
	Device location is on	Display map section and route data	Display map section and route data	Pass
Bus Schedule	Button click	Navigate to bus schedule page	Navigate to bus schedule page	Pass
	Select available route	Display bus schedule of the route selected	Display bus schedule of the route selected	Pass
Bus Tracking	Button clicked	Navigate to route selection page	Navigate to route selection page	Pass
	Route buttons click	Navigate to bus tracking page and display map section and route data	Navigate to bus tracking page and display map section and route data	Pass

Setting	Button clicked	Navigate to setting page	Navigate to setting page	Pass
	Logout button clicked	Navigate to login page	Navigate to login page	Pass
Driver				
Login	Submit correct credentials	Navigate to route selection page	Navigate to route selection page	Pass
Route Selection	Select route	Navigate to route page	Navigate to route page	Pass
Route Page	Click “go online” button	Display current location data	Display current location data	Pass
		Route button in student route selection page is enable	Route button in student route selection page is enable	Pass
		Data update in Firebase real-time database	Data update in Firebase real-time database	Pass
		Display driver location in student bus tracking page and update the polyline and route data	Display driver location in student bus tracking page and update the polyline and route data	Pass
	Click “go offline” button	Route button in student route selection page is disable	Route button in student route selection page is disable	Pass

		Data update in Firebase real-time database	Data update in Firebase real-time database	Pass
	Click “change route” button	Navigate to route selection page	Navigate to route selection page	Pass
Setting	Logout button clicked	Navigate to login page	Navigate to login page	Pass

Table 5 Testing Setup and Result

6.3 Project Challenges

The challenges faced by this project are mainly the code implementation and bug fixing issue of the bus tracking system between student and driver modules. It was challenging at first implementing the data synchronization between the driver’s mobile device which continuously updates its location from time to time and the student’s interface (bus tracking page) which displays the live location of the bus driver and route data. In addition, the implementation of polyline automation for dynamic route updates, calculating the estimated time arrival and distance between the driver’s current location and the bus stop utilizing Direction API is also bringing challenges in terms of code implementation and managing API usage limits.

6.4 Objectives Evaluation

The major objectives of this application are developing a user-friendly mobile application to track real-time location of UTAR buses and integrate weather API to provide users real-time weather updates and alerts. The bus tracking page allows users to track the driver’s location, which also offers the polyline automation feature, route data (estimated time arrival and distance) utilizing Google Direction API. Students can now plan their trips more efficiently and save waiting times at bus stops. In addition, the weather feature allows users to access current and forecast weather data, which enables them to plan and make better decisions during journeys. A popup dialog box will prompt users to notify if rain is coming in the next hours which ensuring they are aware of this. On the other hand, a new feature is also introduced in this application,

track nearest bus stop feature, which allow users to track the nearest bus stop around based on their current location. This is helpful to users, especially new students that are not familiar with Kampar and UTAR commuting systems. In overall, the core objectives were met, offering users a comprehensive and better commuting experience.

6.5 Concluding Remark

In this chapter, the system's functionality was thoroughly evaluated, ensuring all the core functions operate as intended. The testing phase covered critical aspects ranging from weather forecasting, real-time bus tracking, and nearest bus stop tracking, with all text cases yielding successful result. Despite challenges in implementing real-time data synchronization and API integration, the system has met the core objectives, providing users with bus tracking feature, and weather updates.

CHAPTER 7 CONCLUSION AND RECOMMENDATION

7.1 Conclusion

The UTAR Transportation Tracker Application was successfully developed to address the challenges faced by UTAR students in accessing real-time bus information and weather updates. By leveraging GPS technology, Google Maps API, Firebase, and OpenWeatherMap API, the application offers users the core feature ranging from real-time bus tracking, weather forecasts, and bus stop tracking. These features have significantly improved their commuting experience.

7.2 Recommendations

While the application achieves its main objectives, further improvements could enhance its usability and functionality even more. First, the user interface has room for further enhancements. The application is lack of admin system that is used to configure the setting when necessary, such as announcement update, schedule changes, etc. In the aspect of login feature, we can set validation on the email to only allow students with UTAR email to access the application, since the applications are made for UTAR students. Another idea for improvement is the multi-language support feature as UTAR covers students from different races and countries, making the application more user-friendly to all the users. Additionally, more features can be implemented into the application such as payment system, crowdsourced reporting, forum, and more.

REFERENCES

[1]	I. M. Ops, "How GPS Tracking is Reshaping Transportation and Logistics: 8 Benefits," *Lowry Solutions*, Jan. 16, 2024. https://lowrysolutions.com/blog/benefits-of-gps-tracking-reshaping-transportation-and-logistics/
[2]	"Introduction to GPS tracking: A brief overview of what GPS tracking is and how it works," *Copenhagen Trackers*, Jan. 16, 2023. https://cphtrackers.com/blogs/news/introduction-to-gps-tracking-a-brief-overview-of-what-gps-tracking-is-and-how-it-works
[3]	A. Ali, "GPS Evolution in Smartphones | How GPS Work in Smartphones | Use of GPS in Daily Life | What is GPS?," *www.linkedin.com*, Apr. 03, 2023. https://www.linkedin.com/pulse/gps-evolution-smartphones-how-work-use-daily-life-what-ali-akbar
[4]	Google, "Firebase," Firebase, 2023. https://firebase.google.com/
[5]	"Firebase security checklist," Firebase. https://firebase.google.com/support/guides/security-checklist
[6]	"Moovit: A global transport application | data.europa.eu," *data.europa.eu*, Mar. 01, 2019. https://data.europa.eu/en/news-events/news/moovit-global-transport-application
[7]	"Everything You Need to Know About Any Line," *Moovit*. https://support.moovitapp.com/hc/en-us/articles/211393089-Everything-You-Need-to-Know-About-Any-Line
[8]	"Live Directions & Get Off Alerts," *Moovit*. https://support.moovitapp.com/hc/en-us/articles/211392929-Live-Directions-Get-Off-Alerts

REFERENCES

[9]	"Ver 5.94: Trip options based on your pre-set preferred transit types," *Moovit*. https://support.moovitapp.com/hc/en-us/articles/5773446923154-Ver-5-94-Trip-options-based-on-your-pre-set-preferred-transit-types
[10]	"Download transit maps to use when you are offline," *Moovit*, Oct. 19, 2016. https://updates.moovit.com/download-transit-maps-use-offline/
[11]	"How to Use Transit - Transit Support," *help.transitapp.com*. https://help.transitapp.com/article/93-how-to-use-transit
[12]	"How to use the TRANSIT app for bus streetcar and train schedules," *www.youtube.com*, Nov. 30, 2022. https://www.youtube.com/watch?v=Kv68EfGSIcc&t=520s
[13]	"Purchase a Bikeshare Pass - Transit Support," *help.transitapp.com*. https://help.transitapp.com/article/213-purchase-a-bikeshare-pass
[14]	"What is GO crowdsourcing? - Transit Support," *help.transitapp.com*. https://help.transitapp.com/article/91-what-is-go-crowdsourcing
[15]	"Introducing 'Routing Powers,'" *Citymapper*. https://citymapper.com/news/2296/introducing-routing-powers
[16]	"Compare Your Fare," *Citymapper*. https://citymapper.com/news/931/compare-your-fare
[17]	"How to set the payment method in Citymapper?," *www.youtube.com*. https://www.youtube.com/watch?v=FI34tvhpzto
[18]	"Citymapper works if you're OFFLINE," *Citymapper*. https://citymapper.com/news/1282/offline-support

[19]	Grab, "Grab's dynamic pricing explained | Inside Grab," *Grab SG*, Mar. 08, 2023. https://www.grab.com/sg/inside-grab/stories/surge-dynamic-pricing-explained/
[20]	Grab, "GrabPay – Mobile Wallet Payment Solution | Grab MY," *Grab MY*, 2016. https://www.grab.com/my/pay/

Poster


UTAR TRANSPORTATION TRACKER
Real-time Campus Mobility Solution
Key Features
Live Bus Tracking
Google Maps integration with real-time Firebase updates
Driver location sharing via Realtime Database
Smart Bus Stop Navigation
Nearest bus stop finder with walking directions
Dynamic ETAs and route distance
Bus Schedule
Interactive timetable for all routes
Weather Integration
Realtime weather information