JOMNAIK: PUBLIC TRANSPORT NAVIGATION WITH REAL-TIME TRACKING AND ETA PREDICTION

By

Ng Jia Qin

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 2026

COPYRIGHT STATEMENT

This Final Year Project proposal 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 proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal 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, Prof. Ts. Liew Soung Yue for accepting my proposed title and give me his invaluable advice, guidance and support throughout this project. His insights and feedback have been a constant source of inspiration in helping me overcome challenges and improve quality of my work. A million thanks to you.

Special thanks go to my friends and course mates for their suggestions and comments. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

Malaysia's transportation is becoming more and more developed. Due to the population growth, the number of daily trips made by Malaysian is increasing day by day. In addition to taking a private car and using taxi booking services, people can also use public transportation to reach their destinations. However, the navigation system in Malaysia currently has still immature, especially for public transportation. Therefore, this project aims to develop a navigation system specifically for public transportation in Malaysia. JomNaik is a mobile application where it is developed for Android users. It focuses on displaying the real-time position of public transportation and updated estimated arrival times. This project adopted the incremental development approach that breaking down the system into smaller pieces, allowing the application to be built and delivered in small, functional units over time. The core features of this system are providing real-time bus tracking, allowing passengers to plan their trips with public transportation, and receiving push notifications when the targeted bus is approaching. A segment-based ETA prediction model was created to provide ETA for every live bus. This system also uses GTFS-RT vehicle location data provided by public transportation agencies, such as MyRapid, to always update the location of public transportation. This project uses Flutter as a tool platform to develop applications. All ETA-related data will be stored in PostgreSQL for analysis. The expected outcome is a reliable and user-friendly navigation system that allows users to estimate the time when planning to travel to their destination with public transit in Penang, Malaysia.

Area of Study: Mobile Application Development, Intelligent Transportation System

Keywords: Estimate Time of Arrival, Real-time Tracking, Mobile Application, Navigation System, Flutter, PostgreSQL

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ETA	Estimate Time of Arrival
KTM	Keretapi Tanah Melayu
ETS	Electric Train Service
MRT	Mass Rapid Transit
LRT	Light Rail Transit
DL	Deep Learning
GTFS	General Transit Feed Specification
GTFS-RT	General Transit Feed Specification – Real Time
LSTM	Long Short-Term Memory
GNN	Graph Neural Network
ID	Identification
GPS	Global Positioning System

CHAPTER 1

Project Background

In this chapter, we present the background of our research, outlining the key problems and gaps that this study aims to address. We also provide the motivation of the research. Finally, we highlight our contributions to the field and how they advance the current state of knowledge.

1.1 Background Information

1.1.1 Evolution of Public Transportation in Malaysia

Malaysia had public transportation in pre-colonial period. Back then, without a comprehensive infrastructure, people relied on walking, cycling, or some traditional transportation like horse-drawn carriages, ox-carts and river boats for urban connectivity [1,2]. Trams, the earliest form of public transportation, were introduced to Malaysia by the British colonial government in the 1880s, particularly in Penang area [2]. As times changed, trams were gradually replaced by buses, and the buses have become the primary mode of public transportation today [1,2].

Malaysia’s transportation system plays an important role in the country's economic growth and urban connectivity. As the population of Malaysia increases, the demand for transportation also increases. Therefore, in addition to private cars and buses, Malaysia KTM Berhad has established KTM Komuter service in 1980s [1,2]. Later, to meet the people’s travel plans, Malaysia also introduced more public transportation., such as LRT, ETS, MRT, Monorail and so on, which now serve millions of passengers [3].

Public transport service is now controlled by different agencies, Prasarana Malaysia Berhad, as the main public transportation agency in Malaysia, provides rail and bus services in the Klang Valley, Penang and Kuantan areas [4]. In addition to these areas, each state is served by different public transportation agencies like Bas.My, formerly MyBAS in Ipoh, Seremban and many other places [5], Causeway Link in Johor [6] and so on.

1.1.2 Relationship between Population and Use of Transportations in Malaysia

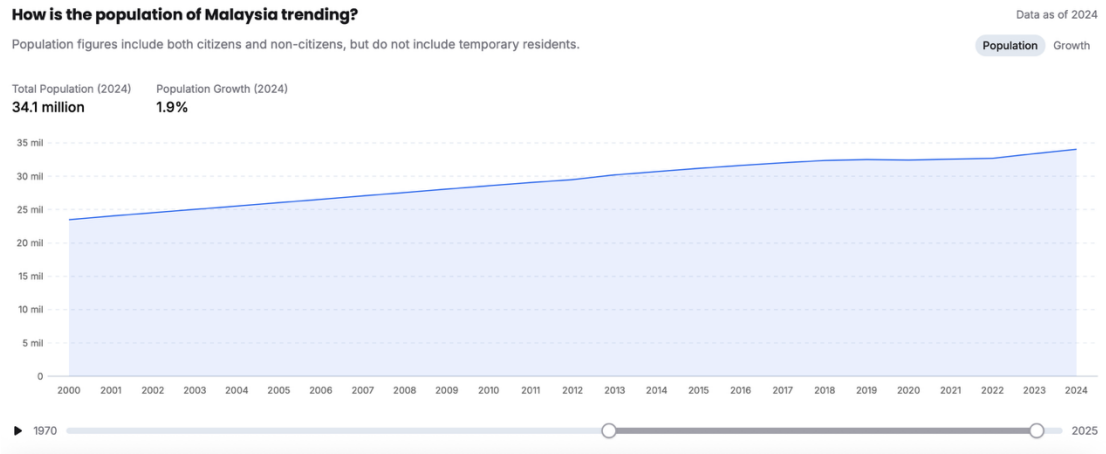


Figure 1.1: Population of Malaysia from 2000 until 2024[7]

Malaysia's cities are developing rapidly. Figure 1.1 shows that the current population of Malaysia has increased from 23.5 million in 2000 to 34.1 million in 2024. Moreover, according to the Department of Statistics (DOSM), the nation will experience steady population growth with 36.49 million people expected by 2030, followed by 41.79 million in 2050 and 42.36 million in 2060[8]. All data from current and expectation population have shown that people's travel needs will increasing from time to time, which will lead to increased use of transportation. This statement is further proven by the analysis from DOSM, which has shown that Malaysians will make an estimated 131 million daily trips in 2030[9]. Therefore, there is a positive correlation between transportation usage and population, which means that it will increase as the population rises, whether it is private transportation or public transportation.

1.1.3 Background of Transportation Navigation System

Navigation is a process that confirm a person or an object location and bring them to their destination [10]. Before any advanced technologies are established, people used physical maps and manually planned routes. Global Positioning System is a technology that uses satellites, receivers, and algorithms to determine the specific location of an object [10], which is mainly used by most of the navigation system. A transport navigation system uses GPS to provide features such as positioning, navigation routes, real-time traffic conditions, and estimated arrival time [10], which is applicable to vehicles, walking, public transportation, etc. For example, when people travel by

private car, they usually use a navigation system to determine the route and check the traffic conditions. Malaysia has a variety of suitable navigation systems, the most well-known of which are Waze, Google Maps, etc. These applications improve the efficiency of the transportation system. As for public transportation, the usefulness of navigation is to provide passengers with the estimated arrival time of vehicles, transfers, schedules, and real-time location, so that passengers can use public transportation conveniently.

1.2 Problem Statement and Motivation

Previously, we stated that the population in Malaysia has growth rapidly, people's travel needs are increasing. Not only that, Malaysia as a tourist destination, the number of foreign tourists visiting Malaysia is increasing [11]. However, according to the problem stated by [12], public transport in Malaysia has a **lack of service problem, for example late arrival and departure times**, cause passengers to spend time waiting. This particularly demonstrate that the poor service provided in Malaysia is a reason for people to abandon use of public transport.

Moreover, the public transportation often lack of coverage, they are not able to provide seamlessly travel, which means that bring passenger from origin to destination is unavailable. According to the research done by [13], where they are reviewing and researching the public transportation in Penang and stated that **poor coverage of public transportation lead to user discontent and cause some impacts on public acceptance on the public transportation**. Through the result obtained, one of the recommendations they mentioned is that provide a real-time information display would enhance public satisfaction, which could mean that until now public transportation in Penang still do not have a real-time arrival provided for the passenger. Therefore, it would cause some inconvenience while passenger tried to plan a journey with public transportation.

Another thing is **the existing navigation system does not cover all public transportation in Malaysia**, means that the navigation system and people cannot plan routes through the existing navigation system. People who want to use public transportation will also feel inconvenienced by public transportation because there is a

lack of seamless connection between bus services and other transportation services such as MRT and LRT, and they do not know how to use them effectively [13].

From the perspective of passengers, **the public transportation system is complex and difficult to use**. When to transfer and when to get off from public transport can only be planned before travelling, without relying on the navigation system**.** This has prevented people from relying on public transportation and reduced the use of public transportation. In Malaysia, most navigation systems focus on private transportation, and only a few navigation systems include guidance for the public transportation system. However, **the features of these systems are somewhat scattered.** Some applications plan routes for passengers using public transportation, while others provide passengers with public vehicle information. Passengers need to download multiple systems in order to obtain information about public transportation when they are travelling.

The thesis aims to integrate public transportation navigation, unifying the features of public transportation navigation systems into a single system. In addition, this project will also provide an accurate estimated time of arrival and display the real-time location of public buses, so that passengers can have a rough prediction of when the bus will arrive and be prepared not to miss the bus. We hope that in this way, we can eliminate the confusion of passengers on how to plan public transportation routes and stimulate more citizens and travellers to use public transportation in order to reduce urban traffic congestion.

1.3 Objective

The project aims to develop a smart, useful navigation system that offers real-time tracking and ETA predictions for Malaysia Penang users. Below are some sub-objectives of this project:

1) To collect and integrate data
    - Collect data from internal sources and external sources.
    - Utilize data provided by public transport agencies, Malaysia open-source data and traffic data to provide navigation guidance and collect all data to build a historical dataset.

2) To build an ETA prediction model
    - Use a real-time data, historical or default value as fallback value to provide ETA.
    - Collect travel time for each segment to provide real-time ETA, store the daily travel time as historical data for future use as fallback value.

3) To develop a mobile application
    - Visualize bus movements and use animation to display the bus running.
    - Check if the current location of the bus is close to the selected stop and send notifications to remind user.
    - Create a graphic user interface to display all functionalities and let user interact with the system.

1.4 Project Scope

The scopes of the project include develop a mobile application focused on public transportation, providing a guide for Malaysian passengers to use public transportation and take them to their destinations. The system will be targeted in Penang, Malaysia. Users can plan their journey according to the route and find the target bus to track and understand the location of the bus in order to avoid wasting time waiting for the bus. In addition, the system will collect past data as historical data, so all data will be stored in a database. We invented this system with the goal of having more people use public transportation and reduce urban traffic congestion in the future.

The following is a summary of system deliverables:

1) Provide basic route function, guide user to take the public transportation.
2) Provide ETA for every bus to the corresponding stops
3) Provide real-time bus tracking to show users the specific location.
4) Provide push notifications to notify users to prepare for the ride when the bus is about to arrive at the stop.

However, since our system is a preliminary development, it is currently suitable only for users in Penang. Our system focuses more on public transport as we emphasize the

use of public transportation, and our purpose is to encourage people to use public transportation so as to reduce the use of private transportation.

1.5 Impact, Significance, and Contributions

Our invention aims to solve the problems described in Section 1.2 by providing a mobile application that will bring convenience to Malaysian passengers using public transportation and foreign tourists visiting Malaysia. First of all, we will use a third-party service provider to obtain a series of basics requirement for navigation, such as routing and maps. For the bus arrival time, we will develop an ETA model to calculate the estimated time of arrival so that passengers do not need to rely on a predetermined schedule. In addition, we will provide real-time bus tracking with moving coordinates for passengers to understand the specific location of the bus.

With this smart public transport system, we aim to improve our target audience experience by providing bus tracking and ETA prediction, allowing user to get latest information at anywhere, anytime. Furthermore, a seamless connectivity between bus service and other areas of public transport services is provided to users so they can simply enter their destination, and they will be guided starting from their current location, using a combination of public transport. This application can integrate with other public transport services in the future, as its clear separation between modules provides scalability of adding or removing newly added or no longer supported features in the future.

The significance of developing a mobile application on public transport lies in improving the user experience during transit, which potentially could attract more people to switch from private cars to public transport. Furthermore, as the project is able to provide user expectations on having a comprehensive public transport navigation system, we believe that it will bring more than just convenience for the passenger who always uses public transport for traveling, and ultimately contribute to a more sustainable and inclusive environment in Penang, Malaysia.

1.6 Report Organization

As for the rest of the paper, it is organized as follows: Chapter 2 Literature Review which reviews existing applications, Chapter 3 System Methodology which will be the system methodology of our developed system. Chapter 4 System Design will be the system design that mainly describes how the system works. Chapter 5 System Implementation will be our implementation work on the system. Chapter 6 System Evaluation and Discussion will be the chapter that evaluates how the system performs. Chapter 7 Conclusion and Recommendation will be the last chapter that discusses the overall work done and the future work for the implemented system.

CHAPTER 2

Literature Review

In this chapter, we will review the existing literature and research in detail, focusing on navigation system and its basic functionalities. We will discuss the features and characteristics of existing navigation systems and explore the limitations that not covered by its scope of features. Finally, we will provide further improvements to overcome these limitations.

2.1 Previous Works on Existing Application

2.1.1 Google Maps

Google Maps is a web and mobile application mapping service launched by Google Inc., providing detailed information about geographic areas and locations around the world. Google Maps offers an interactive user interface that allows users to interact with the map to search for places of what they are looking for and provides the suitable routes and real-time traffic updates when people are using navigation service. Other than that, ETA predictions and reporting emergencies such as traffic jams and road closures are also within their included services. Street map is their key features that allow users to explore the world as if. Moreover, Google Maps covers route navigation for driving, walking, cycling, public transport and two-wheeled vehicles, providing convenience for different user groups. Thus, it can be said to be a relatively comprehensive navigation system [14].

We studied what techniques Google Maps uses to implement basic navigation functions. First, Google Maps uses graph data structures, which is Dijkstra algorithm and A* algorithm to calculate the shortest distance from the source to the destination. Dijkstra algorithm is a greedy algorithm that visits the node with the smallest distance to the source and updates the distances of adjacent nodes until the target node is reached. The A* algorithm combines the information used by Dijkstra algorithm with greedy best-first search, which is an informed search algorithm. Google Maps combines these two algorithms to provide the best route and uses Machine Learning to improve accuracy and efficiency in order to provide users with the latest information [15].

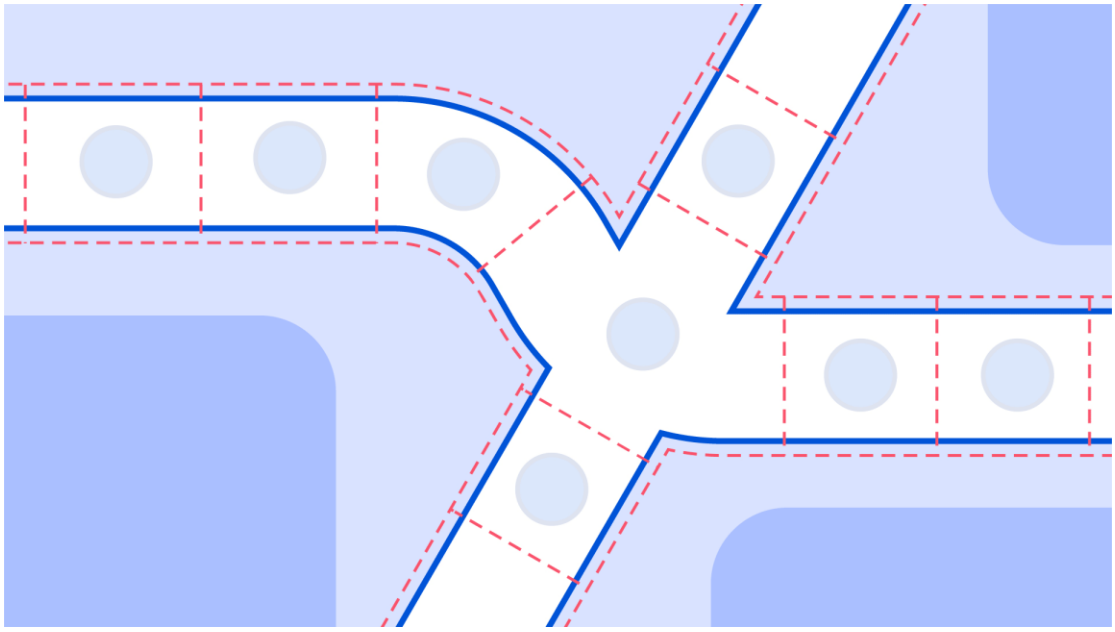


Figure 2.1 Road Partitioning for Graph Neural Network (GNN) [17]

The next is regarding the calculation method of ETA predictions. First of all, ETA predictions require the distance from the source to the destination, and then consider other factors such as vehicle speed. Therefore, obtaining the shortest route is the main basis for ETA prediction. In the early days, Google used the A* algorithm to calculate the shortest route [16]. This algorithm does not take into account real-time traffic conditions, any traffic emergencies such as traffic accidents, road closures, and traffic police detection will affect the arrival time. Thus, nowadays Google uses more advanced techniques, that is, graph neural networks (GNN). It is an artificial neural network that processes graph data. Refer to figure 2.1, nodes represent road segments and edges represent the connections between road segments. Google considers many factors that will affect ETA prediction, like roads, road conditions, historical data, static conditions such as speed limits, and these all factors are used as parameters for ETA prediction. ETA prediction is based on a series of predictable connected road segments. The ETA of each road segment is predicted when people set off, and one road segment will continue to update the next road segment [17].

2.1.2 Waze

Waze is a user-driven navigation application that provides navigation, ETA, traffic conditions etc. focusing on private car. Waze encourages “Drivers help drivers” that is through users reporting accidents, police detection and other road conditions, and share this information to other users to make other users be vigilant and avoid risks. Compared with other existing navigation systems, its outstanding feature is encouraging user participation and cooperates with various transportation departments, enterprises and other institutions to save every driver’s time on the road [18].

We have studied the technology used by Waze. Existing paper does not clearly indicate which algorithm used by Waze for routing. Some discussion say that their technology is confidential, while others state that they use the A* algorithm to calculate [20]. Therefore, it is inferred that they may their own proprietary technology for routing. Some discussion in Waze Community Wazeopedia indicate that Waze constructs routes from one node to another on map, using traffic data along the way to calculate the fastest point to the destination and the travel distance [19]. They use historical and real-time data to quickly respond to or predict traffic conditions in advance [20] and combine traffic data recorded when users use Waze to provide routes.

For ETA prediction, Waze uses a DL model called “Smartsum”, which is based on long short-term memory (LSTM) to calculate and predict ETA. It receives a set of multiple route segments as input, estimates the expected transit time for each road segment and add them up. The model considers many indicators, currently using traffic condition data, spatial information (physical characteristics of the road), time information (daytime, night-time, weekends), and plans to use other information in the future to provide more accurate ETA predictions. The reason why Waze improves its ETA prediction model is not only to provide more accurate ETA, but also to reduce the update of real-time ETA, which is greatly improving the user experience [21]. Each update means that the previous ETA is inaccurate. According to Royal Sasson’s [21] results, Smartsum reduced ETA refresh by 50%.

2.1.3 MyRapid

MyRapid is a public transportation system owned by Prasarana Malaysia, its services covers Kuala Lumpur, Penang, and Kuantan. It provides web application and also mobile application to let the user check real-time bus information.

2.1.3.1 Bus Kiosk

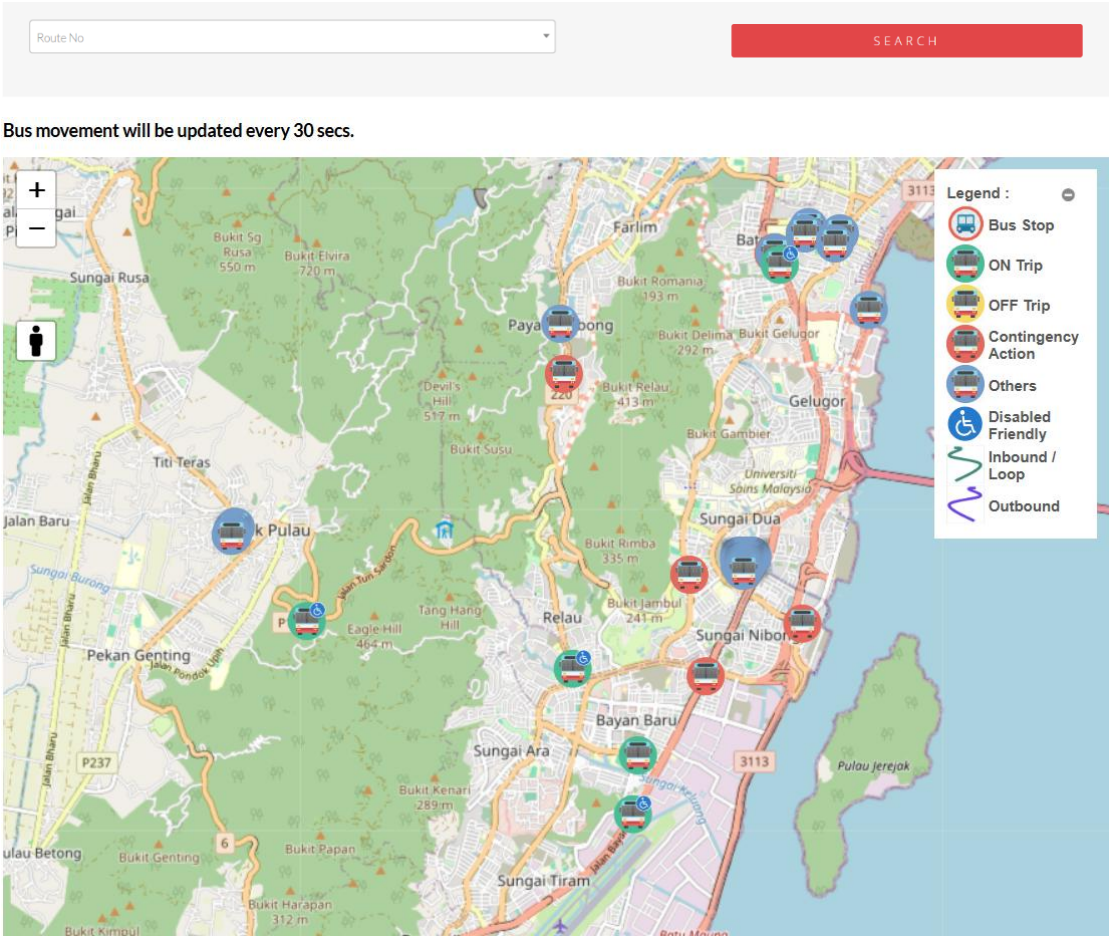


Figure 2.2 Rapidbus Bus Kiosk - Penang Area

Bus kiosk is a web application which aims to update the running status of each bus in real-time every 30 seconds, allowing users to track the bus movement and plan their trips. Other than that, it also provides bus icons, such as buses on trip or off trip, buses that provide disabled friendly services. Users can also search for bus with its route ID, find out the specific location of the buses that are on trip, and also provide the location of each stop that the bus will pass through [22].

2.1.3.2 MyRapid Pulse

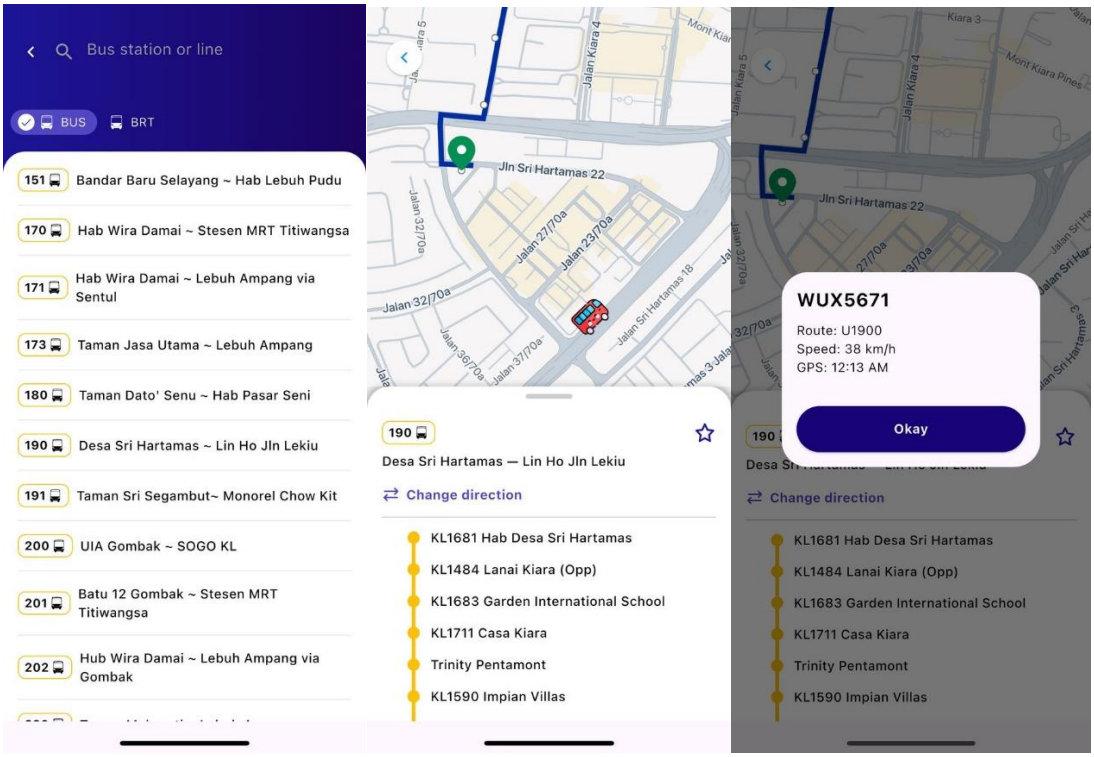


Figure 2.3 MyRapid Pulse User Interface, search bus route id(left), display bus current location(middle), bus information(right)

A mobile application provided for MyRapid services. It allows users to plan routes using the public transportation provided by the company and also provides real-time updates on public transportation. Like the web application, users can check the current location of the bus according to the desired bus route id. Users can also enter their current location and destination to know which bus they need to take.

The navigation service provided by MyRapid only focuses on public transportation especially their own services such as buses, MRT, LRT, and provides arrival time for each station according to the scheduled timetable[23].

Worth to note that is, this application is only applicable for KL area/Selangor, no bus tracking for Rapid Bus in Penang.

2.1.4 Moovit

Moovit is a city travel planner application and mobility as a service provider, it provides real-time trip planning, navigation of various public transportation and other services in Kuala Lumpur, Penang, Perak Ipoh, Johor Bahru and Kuantan in Malaysia [24].

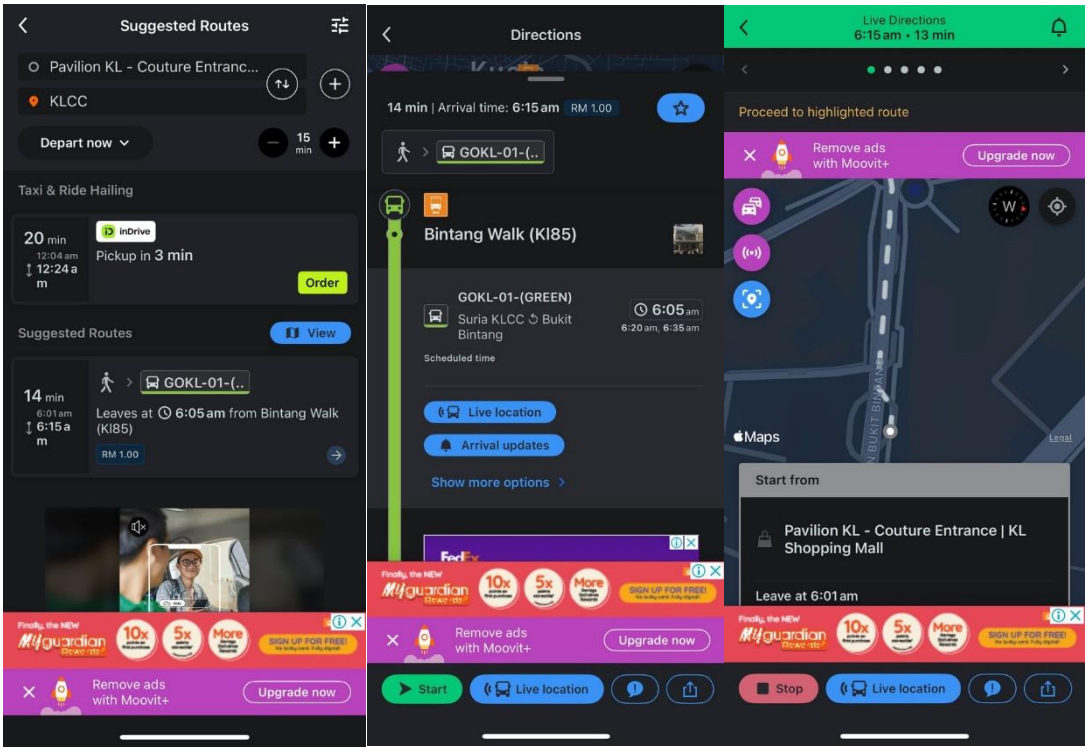


Figure 2.4 Moovit User Interface, route suggestions(left), route direction(middle), route instructions in map(right)

Based on the figures in 2.4, we can see that Moovit's navigation method is to guide users from a specific location to the nearest bus or metro station and take the transportation to their destination. It refers to the scheduled timetable of public transportation, and provides ETA based on the total time it takes the user to get to the station and provides the required fees so that users can have an idea about the cost [25]. Not only that, it uses different colours to represent the accuracy of the ETA predictions, so that users can trust the arrival time they provide, which green indicates accurate ETA, yellow indicates fairly accurate, red indicates may not be accurate and if no live information for the transportation, it display in black colour [26].

2.2 Overall Comparison of Previous Studies

Features	Google	Waze	MyRapid	Moovit	Proposed system
Plan Journey	Yes	Yes	Yes	Yes	Yes
Route	Yes	Yes	Yes	Yes	Yes
ETA Prediction	Yes	Yes; For car only	**No**	Yes	Yes
Bus Current Location Tracking	**No**	**No**	Yes	**No** (for Penang)	Yes
Bus Arrival Time	Yes; Follow Timetable	**No**	Yes; Follow Timetable	Yes; Follow Timetable	Yes, Real time updates
Transport Mode Coverage	All modes	Cars only	Specific to provided service, e.g. RapidPenang	All public transport	Public Transport

Table 2.1 Comparison between the existing system and the proposed system

2.3 Limitation on Previous Study

Features Separation

Based on the features listed in Table 2.1, we found that Waze mainly focuses on private cars, and MyRapid is mainly focused on public transportation, especially its own services. Moovit is mainly based on urban transportation planning. Although the services provided by Google Maps have a large coverage, both private and public transportation are within their service area. But it is worth noting that for Malaysian public transportation, Google Maps does not display bus current location [27]. If all navigation systems provide their own features and can run smoothly, when people plan

to use multiple public transportation to get from one place to another, they must switch and refer to multiple systems in order to arrive at their destination. For example, if a foreign visitor wants to travel from Ipoh to Penang Island, and they have to take the KTM Komuter from Ipoh, Perak to Butterworth, Penang, and then take a ferry to Penang Island, then take the bus to travel. He may need to use Google Maps and the official website of the relevant public transportation agency to check the timetable and specific routes for all public transportation, and then use other application such as MyRapid Pulse to check the specific location of the bus to know the specific location of the current on trip bus to ensure they have not missed the bus or the bus will take some time to arrive at the destination bus stop.

No ETA Prediction on Public Transit Schedule

Among all the navigation systems we reviewed, we found that none of them provide real-time ETA for public transportation trips, they only display arrival times based on schedules. Although public transportation should arrive at every station on time as it is a service provided to the public, the passenger who intends to use public transportation services will arrive at the corresponding station in advance at the specified time to wait for public transportation to arrive. However, plans cannot keep up with changes. Public transportation such as buses may arrive early or late at the current station due to some reasons such as traffic jams, human factors, emergencies and so on, and waiting passengers can only wait without a clue. Therefore, there are too many uncertain factors when waiting for the arrival of public transportation only according to the timetable. Users not only waste a lot of time just waiting for public transportation, but also cannot properly arrange their trips, and thus give up using public transportation.

Data Refresh Every 30 Seconds

The services provided by MyRapid such as bus trips, bus current location, etc. can find and obtain in the GTFS-RT API Endpoint provided by the data.gov.my which is shared by relevant public transportation agencies. According to the API documentation, it is not like Grab or Foodpanda that always update and feedback the exact location of the

rider, it will only be updated every 30 seconds [28]. Therefore, during this 30-second period without feedback, we cannot know the location of the bus, whether it is stuck in place due to traffic congestion or traffic light, or the traffic congestion has been relieved so that the bus can arrive at the station earlier than the original time. The most important of which is that the bus location displayed in the User Interface will keep "teleporting" when it is updated, resulting in a poor user experience. On the other hand, if the vehicle location remains unchanged after 30 seconds, the user cannot know whether it is due to traffic congestion or system internal problems.

2.4 Proposed Solutions

This project aims to propose some possible solutions to the above limitations. First, several feasible methods are proposed to achieve the purpose of providing real-time ETA. Then, a navigation system that covers all basic features for public transportation.

Overall System Solutions

According to Table 2.1, we know that the features of the existing system are scattered. Thus, we would like to propose creating a navigation system that integrates all functions, especially the integration of public transportation. For the features currently provided by the system on buses, such as MyRapid PULSE, which updates the real-time location of the bus every 30 seconds, Google and other systems only provide the arrival time based on the timetable. Integrating these functions will greatly improve the user experience.

In addition, as the static and live bus data that we mainly utilize in this project are obtained from the Open API developed by the Malaysian government, similar to MyRapid PULSE, we will make full use of these API to provide a bus navigation system, on which we calculate the ETA based on the GTFS static data and real-time updated data. Not only that, we will get the live bus location to update the bus coordinates on the map component so that users will see that the bus is not stationary but moving constantly. Passengers can also get notifications so that they can prepare

for the ride based on the approximate location of the bus, which enhances the visualization effect of the system.

Specific Functionality – Road Segmentation for ETA Predictions

According to the documentation and the returned data from the API of GTFS-RT, we found that no column actually returns the next stop where the live bus is heading. However, it is important to understand the next stop of the live bus for ETA predictions. Google uses a GNN architecture for ETA prediction, where they leverage nodes and edges to capture spatio-temporal dependencies. Inspired by this, this project borrows the core idea of **road partitioning** but has optimized the model implementation specifically for the actual needs of the Penang bus system.

CHAPTER 3 System Methodology/Approach

3.1 System Methodology

This application will be developed following the idea of an incremental development approach, which is breaking down the project into smaller and manageable tasks [29]. Each incremental step contains a portion of the system functionalities, which will go through the design, testing, and implementation phases. Since some of the system requirements need to be done based on previous functionalities, it is best to use this approach to handle this software development life cycle, as we need to ensure all the functionalities are functioning properly before moving on to the next step.

Below is a diagram showing each incremental step:

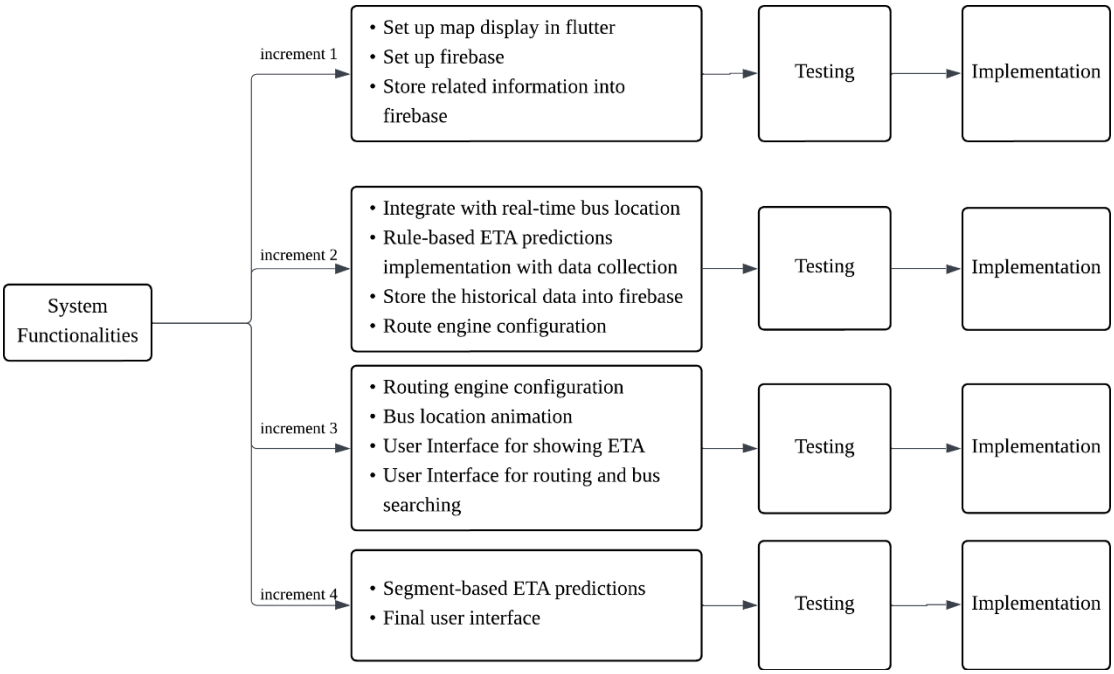


Figure 3.1: Incremental development approach diagram

From the figure 3.1, the system functionalities are divided into 4 incremental steps, where in each step, the first rectangle box indicates the features that need to be designed and developed. Following the testing phase to check how the software performs and lastly the implementation phase to verify everything is working well before going to

the next step. This process will continue to iterate until the final system is produced, and all steps are subject to change if any unexpected issues arise.

3.2 System Design Diagram

3.2.1 Use Case Diagram

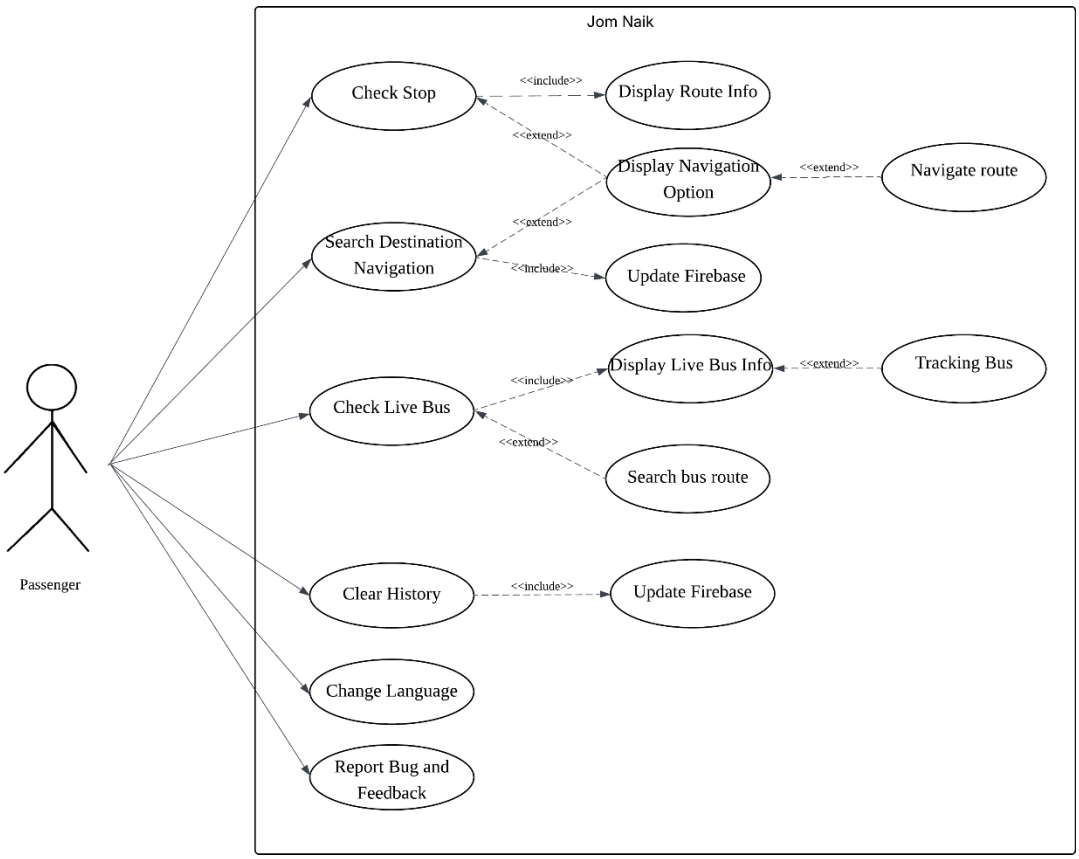


Figure 3.2.1: Use Case Diagram

Figure 3.2.1 shows the use cases the user can perform in the application. The actors are "Passenger" who mainly use public transportation for transit. The use cases are mainly about check stop info, get route and navigation service, search destination, live bus info, and so on.

Firstly, this system provides a route service mainly to guide passengers to nearby bus stops or any location within Penang. Furthermore, the system allows passengers to view information on bus routes stopping at each stop, so that passengers can better plan their trips. Additionally, ETAs are provided for each bus to the next stop and after that stop, clearly showing passengers the bus's current location and the estimated time the bus will arrive at the bus stop. Lastly, the system provides simple interactive features such

as clearing history, changing language, and reporting bugs and feedback to improve the user experience when passengers are using the application.

3.2.2 Use Case Description

In this section, a detailed, step-by-step narrative description of each action was provided in the use case diagram using use case description. The purpose of providing use case descriptions in detail is to show how actors can interact with our system.

<table>
<tr><td>Use Case Name: Check Stop</td><td>ID: 1</td><td>Importance Level: Medium</td></tr>
<tr><td>Primary Actor: Passenger</td><td colspan="2">Use Case Type: Detail, essential</td></tr>
<tr><td colspan="3">Stakeholders and Interests:
Passenger– wants to know what bus routes this stop offers and how to navigate to it</td></tr>
<tr><td colspan="3">Brief Description: The use case describes the steps on how the passenger checks the stop info and gets the guidance for navigation</td></tr>
<tr><td colspan="3">Trigger: Passenger taps a stop marker on the map on the homepage
Type: External</td></tr>
<tr><td colspan="3">Relationships:
Association: Passenger
Include: Display Route Info
Extend: Display Navigation Option
Generalization: Not Applicable</td></tr>
<tr><td colspan="3">Normal Flow of Events:
1. Passenger taps a stop marker displayed on the Google map on the homepage.
2. System redraws the bus stop marker based on bus route and adds polylines for bus trip
3. System displays the bus stop info dialog for the marker tapped by the passenger
4. Passenger interacts with the system.
If the passenger taps on the info dialog,
the S-1: navigation activity is performed.
If the passenger taps on any other blank space of the screen,</td></tr>
</table>

the S-2: clear activity is performed. 5. The passenger will be directed to the respective interface
Sub Flows: S-1: Navigation Activity 1. Execute Display Navigation Option use case S-2: Clear Activity 1. The system removes the selected bus stop markers and polylines and displays all bus stop markers on the Google map
Alternate/Exceptional Flows: Not applicable

Table 3.2.2.1: Use Case Description – Check Stop

Use Case Name: Display Navigation Option	ID:2	Importance Level: High
Primary Actor: Passenger	Use Case Type: Detail, essential	
Stakeholders and Interests: Passenger– wants to get a route suggestion to go to the destination		
Brief Description: The use case describes the steps for how the passenger selects a route before navigation		
Trigger: Passenger selects a destination Type: External		
Relationships: Association: Passenger Include: Not Applicable Extend: Navigate route Generalization: Not Applicable		
Normal Flow of Events: 1. Passenger selects a destination from the geocoding result or chooses a bus stop as a destination.		

2. System receives a destination coordinate and calls the Route Server with the passenger location as the source coordinate to fetch possible bus routes and walk routes. 3. Passenger switches between different route suggestions. 4. System displays a polyline for each selected route suggestion on the map. 5. Passenger chooses the best suited for the trip and executes the Navigation Route use case.
Sub Flows: Not Applicable.
Alternate/Exceptional Flows: 2a. If the passenger's location is outside the service boundary (Penang): 2a1. The Route Server fails to return any route suggestion 2a2. System display error message. 2a3. The use case terminates

Table 3.2.2.2: Use Case Description – Display Navigation Option

Use Case Name: Navigate Route	ID:3	Importance Level: High
Primary Actor: Passenger	Use Case Type: Detail, essential	
Stakeholders and Interests: Passenger– wants to start navigation to the destination		
Brief Description: The use case describes the steps of how the passenger interacts with the navigation function.		
Trigger: Passenger confirms with a selected route suggestion. Type: External		
Relationships: Association: Passenger Include: Not Applicable Extend: Not Applicable Generalization: Not Applicable		
Normal Flow of Events:		

1. Passenger selects a route suggestion and starts departure. 2. System displays the first trip leg and guides passenger to the leg destination. 3. Passenger arrives at the leg destination and proceeds to the next leg. 4. Repeat steps 2 and 3 until passenger arrives at their final destination. 5. Passenger terminates the route navigation.
Sub Flows: Not Applicable.
Alternate/Exceptional Flows: Not Applicable.

Table 3.2.2.3: Use Case Description – Navigate Route

Use Case Name: Search Destination Navigation	ID: 4	Importance Level: High
Primary Actor: Passenger	Use Case Type: Detail, essential	
Stakeholders and Interests: Passenger– wants to go to the destination.		
Brief Description: The use case describes the steps of how the passenger search destination through the system and selects the appropriate destination from the list.		
Trigger: Passenger taps the search button. Type: External		
Relationships: Association: Passenger Include: Update Firebase Extend: Display Navigation Option Generalization: Not Applicable		
Normal Flow of Events: 1. Passenger start searching by typing the destination name. 2. System calls the geocoding service using the passenger input. 3. System filters all return results with only the location in Malaysia. 4. Passenger selects a destination. 5. System records the passenger's current selection as history and store to Firebase and executes the Display Navigation Option use case.		

Sub Flows: Not Applicable.
Alternate/Exceptional Flows: 3a. If the result does not have any destination in Malaysia, the use case terminates.

Table 3.2.2.4: Use Case Description – Search Destination Navigation

Use Case Name: Check Live Bus	ID: 5	Importance Level: High
Primary Actor: Passenger	Use Case Type: Detail, essential	
Stakeholders and Interests: Passenger– wants to view the live bus info.		
Brief Description: The use case describes how the system displays all live buses on the map in the application.		
Trigger: Passenger navigates to the line page. Type: External		
Relationships: Association: Passenger Include: Display Live Bus Info Extend: Search Bus Route Generalization: Not Applicable		
Normal Flow of Events: 1. Passenger navigates to the line page. 2. Passenger selects a bus route provided by the system or searches for a bus route. If the passenger wants to choose a bus route without searching, the S-1: select bus route is executed. If the passenger wants to choose a bus route by searching, the S-2: search specific route is executed. 3. System filters all live buses related to the bus route. 4. System displays all related live buses and draws a polyline that indicates the bus trip. 5. Passenger selects a bus marker on the map.		

6. System fetches ETA info from the cloud instance and displays it to the passenger.
Sub Flows: S-1: Select Bus Route 1. System displays all active bus routes. 2. Passenger selects a bus route. S-2: Search Specific Route 1. Passenger taps the search box. 2. Passenger enters bus route name. 3. System displays bus route based on the route name. 4. Passenger selects the bus route
Alternate/Exceptional Flows: 3a. If the result does not have any destination in Malaysia, the use case terminates. 6a. If the passenger taps any stop marker on the map after selecting a live bus marker, execute the Tracking Bus use case.

Table 3.2.2.5: Use Case Description – Check Live Bus

Use Case Name: Tracking Bus	ID: 6	Importance Level: Medium
Primary Actor: Passenger	Use Case Type: Detail, essential	
Stakeholders and Interests: Passenger– wants to get a notification when a bus is about to arrive at its stop		
Brief Description: The use case describes the steps a passenger can perform to get notified.		
Trigger: Passenger taps an info dialog of the stop marker after selecting a live bus marker. Type: External		
Relationships: Association: Passenger Include: Display notification Extend: Not Applicable		

Generalization: Not Applicable
Normal Flow of Events: 1. Passenger taps a stop info dialog after selecting a live bus marker. 2. System calculates the distance between live bus coordinate and stop coordinate every 10 seconds. 3. Repeat step 2 until the distance is less than 200m. 4. System pushes a notification to the user.
Sub Flows: Not Applicable
Alternate/Exceptional Flows: Not Applicable

Table 3.2.2.6: Use Case Description – Tracking Bus

Use Case Name: Clear History	ID: 7	Importance Level: Medium
Primary Actor: Passenger	Use Case Type: Detail, essential	
Stakeholders and Interests: Passenger– wants to remove their search history.		
Brief Description: The use case describes the steps of how passenger remove their history		
Trigger: Passenger taps clear history selection. Type: External		
Relationships: Association: Passenger Include: Update Firebase Extend: Not Applicable Generalization: Not Applicable		
Normal Flow of Events: 1. Passenger taps the clear history selection in the settings page. 2. System displays a confirmation dialog for the deletion of history: If passenger selects “yes”, the S-1: delete history is performed. If passenger selects “no”, the S-1: cancel deletion of history is performed.		

3. System closes the confirmation dialog.
Sub Flows: S-1: Delete History 1. System sends a delete request to Firebase Firestore using the current authenticated User ID. S-2: Cancel Deletion of History 1. System acknowledges the cancellation and maintains existing data records.
Alternate/Exceptional Flows: Not Applicable

Table 3.2.2.7: Use Case Description – Clear History

Use Case Name: Change Language	ID: 8	Importance Level: Medium
Primary Actor: Passenger	Use Case Type: Detail, essential	
Stakeholders and Interests: Passenger– wants to use the system in different languages		
Brief Description: The use case describes the steps for changing the system display language.		
Trigger: Passenger taps change language selection. Type: External		
Relationships: Association: Passenger Include: Not Applicable Extend: Not Applicable Generalization: Not Applicable		
Normal Flow of Events: 1. Passenger taps the change language selection in the settings page. 2. System displays a window for passenger to choose the languages. 3. Passenger selects language. If passenger chooses “English”, the S-1: change to English is performed. If passenger chooses “Chinese”,		

the S-2: change to Chinese is performed. If passenger chooses “Malay”, the S-3: change to Malay is performed. 4. System closes the window.
Sub Flows: S-1: Change to English 1. System changes the local cache of the preference language to English. 2. System changes the language of all other pages to English. S-2: Change to Chinese 1. System changes the local cache of the preference language to Chinese. 2. System changes the language of all other pages to Chinese. S-3: Change to Malay 1. System changes the local cache of the preference language to Malay. 2. System changes the language of all other pages to Malay.
Alternate/Exceptional Flows: Not Applicable

Table 3.2.2.8: Use Case Description – Change Language

Use Case Name: Report Bug and Feedback	ID: 9	Importance Level: Medium
Primary Actor: Passenger	Use Case Type: Detail, essential	
Stakeholders and Interests: Passenger– wants to report an issue regarding the system		
Brief Description: The use case describes the steps of how user report bug and give feedback for the system.		
Trigger: Passenger taps report bug and feedback selection. Type: External		
Relationships: Association: Passenger Include: Not Applicable		

Extend: Not Applicable Generalization: Not Applicable
Normal Flow of Events: 1. Passenger taps report bug and feedback selection in settings page. 2. System gets confirmation from passenger about leaving the application with a confirmation dialog. If passenger taps “yes”, the S-1: open report form is performed. If passenger taps “no”, the S-2: cancel process is performed. 3. System closes the confirmation dialog.
Sub Flows: S-1: Open Report Form 1. System redirects passenger to Google Form outside the application. 2. Passenger fills in the issue and submits the Google Form. S-2: Cancel Process 1. System cancels the process and remains on the same page.
Alternate/Exceptional Flows: Not Applicable

Table 3.2.2.9: Use Case Description – Report Bug and Feedback

3.2.3 Activity Diagram

This section will only showcase some of the complex core activities, such as how a passenger begins navigation, receives a notification, and obtains ETA information, instead of showing all activities. The main purpose of designing the activity diagram is to show the flow of each core activity, ensuring that the system design will have a clear concept in the future.

3.2.3.1 Navigation Activity Diagram

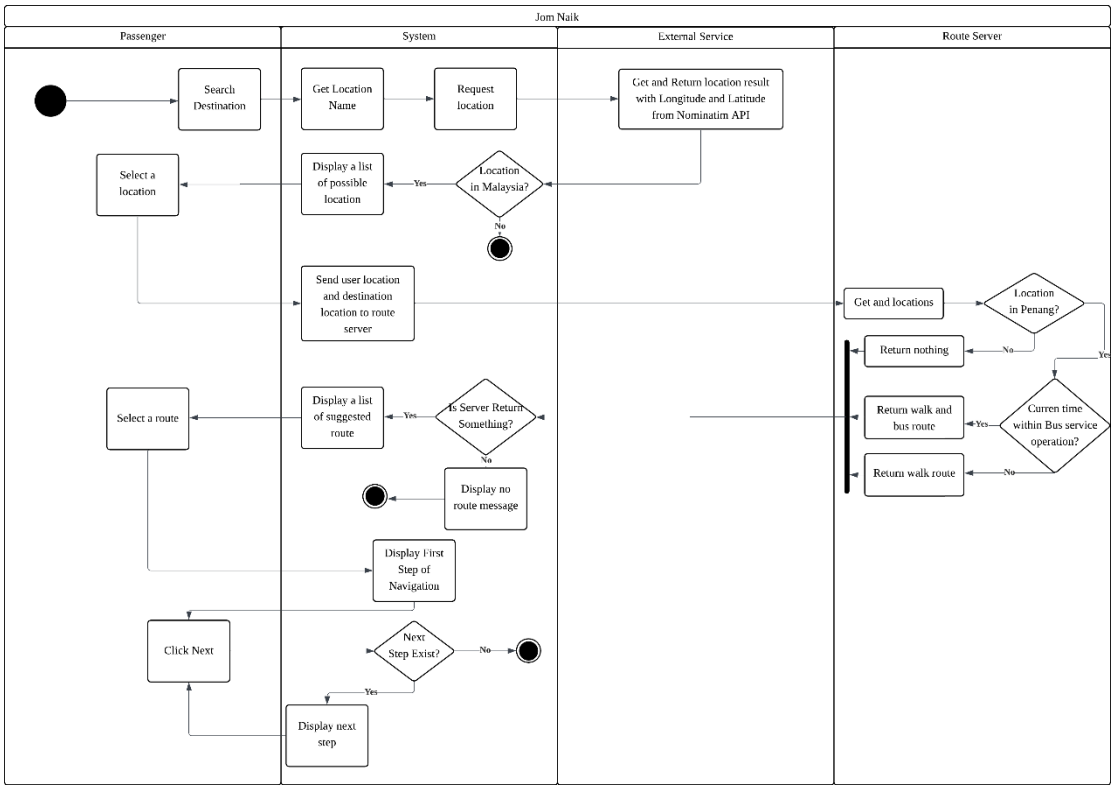


Figure 3.2.3.1: Navigation Activity Diagram

Figure 3.2.3.1 illustrates the activity diagram of system, indicates one of the mainly used on this application. The activity flow between Passenger, System, External Service such as search engine, and route server. The process starts from user search a location and the system will go thru a series of backend logic and guide user to choose a route and lastly navigation service.

3.2.3.2 Push Notification Activity Diagram

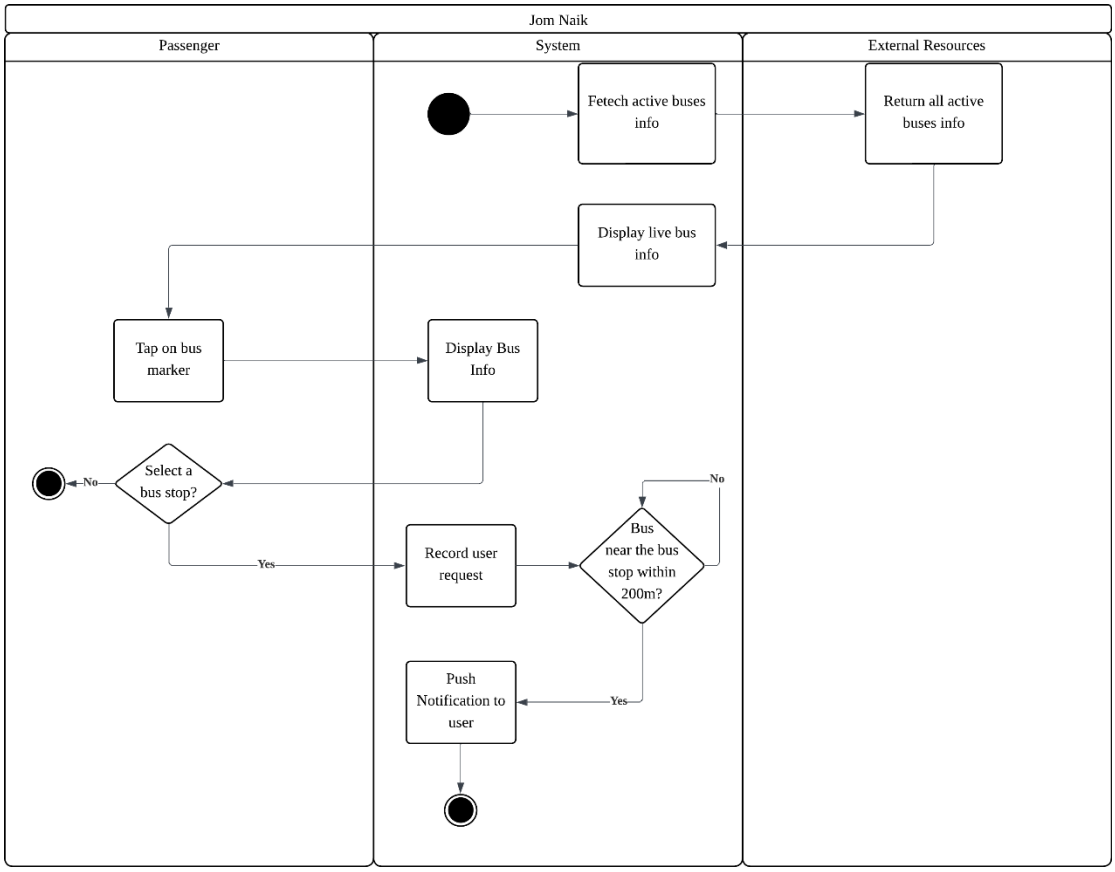


Figure 3.2.3.2: Push Notification Activity Diagram

Figure 3.2.3.2 illustrates the activity on how user can utilize the application to get notified when their targeted bus is coming. The system will always start from fetching the external source like GTFS-RT to display active bus markers on the map, so that the user can check the live bus, choose the bus they want to ride and set a notification when a bus is near to a stop.

3.2.3.3 ETA Activity Diagram

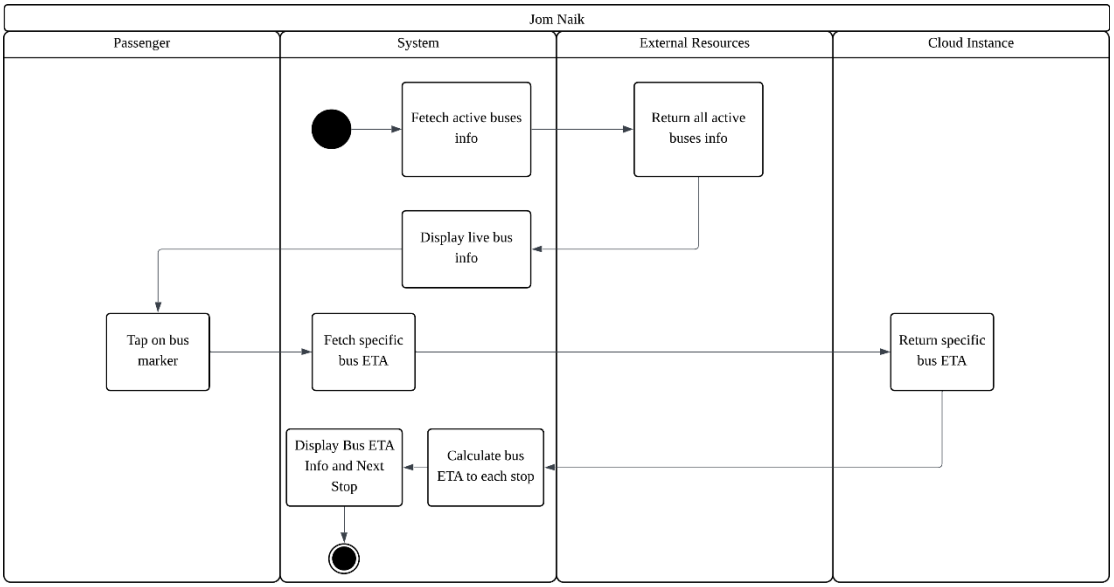


Figure 3.2.3.3: ETA Activity Diagram

Figure 3.2.3.3 shows the activity of how the system displays the ETA prediction to the user. The process is almost the same as the Push Notification Activity Diagram, but with more focus on displaying the bus-related information to the user. The system will fetch live bus information from external sources and display it on the map, so when the user selects a bus route and taps on a bus marker, the system will immediately fetch the ETA info using the bus's unique key from the cloud instance, calculate the ETA for the bus to each consequence stop, and display it to user.

3.3 Sequence Diagram

Sequence diagram is used to show how processes interact over time. In this section, two sequence diagrams are designed to illustrate the core activities in our system. Instead of drawing a sequence diagram for one use case at a time, we draw sequence diagram based on scenario to visually represent the data lifecycle within the system boundary.

Figure 3.3.1 primary illustrate the complete process of route navigation, starting from the Search Destination Navigation use case, and the subsequent use cases that trigger it, such as the Display Navigation Option use case and the Navigation Route use case.

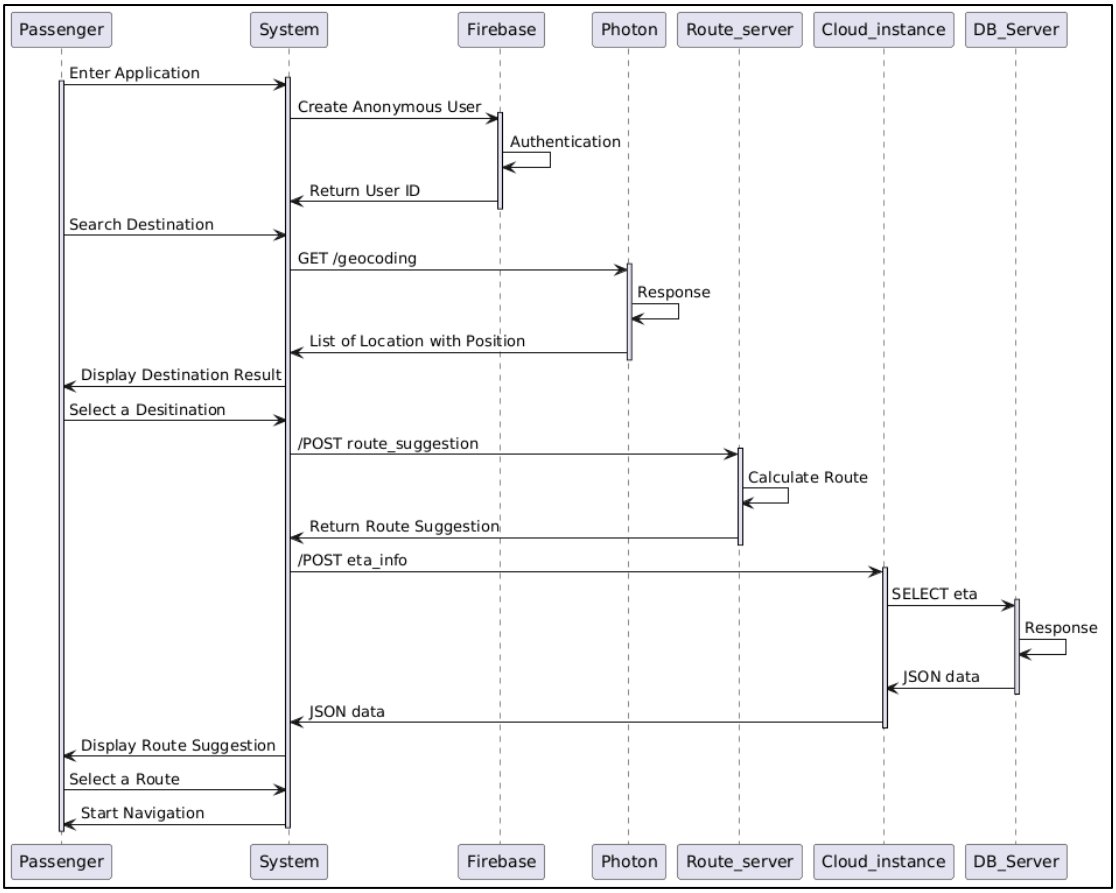


Figure 3.3.1: Sequence Diagram for Route Navigation Process

Figure 3.3.2 illustrates the complete process of live bus checking, starting from the Check Live Bus use case, and the subsequent use cases, such as the Display Live Bus Info use case, the Search Bus Route use case, and the Tracking Bus use case.

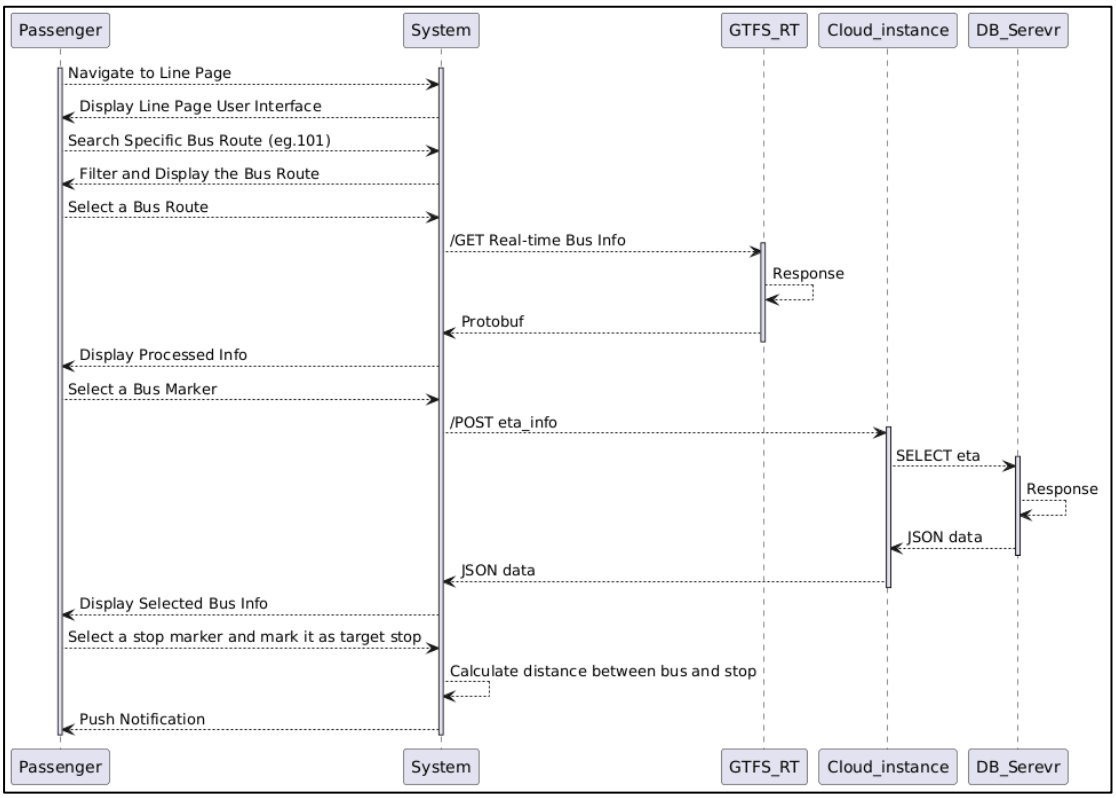
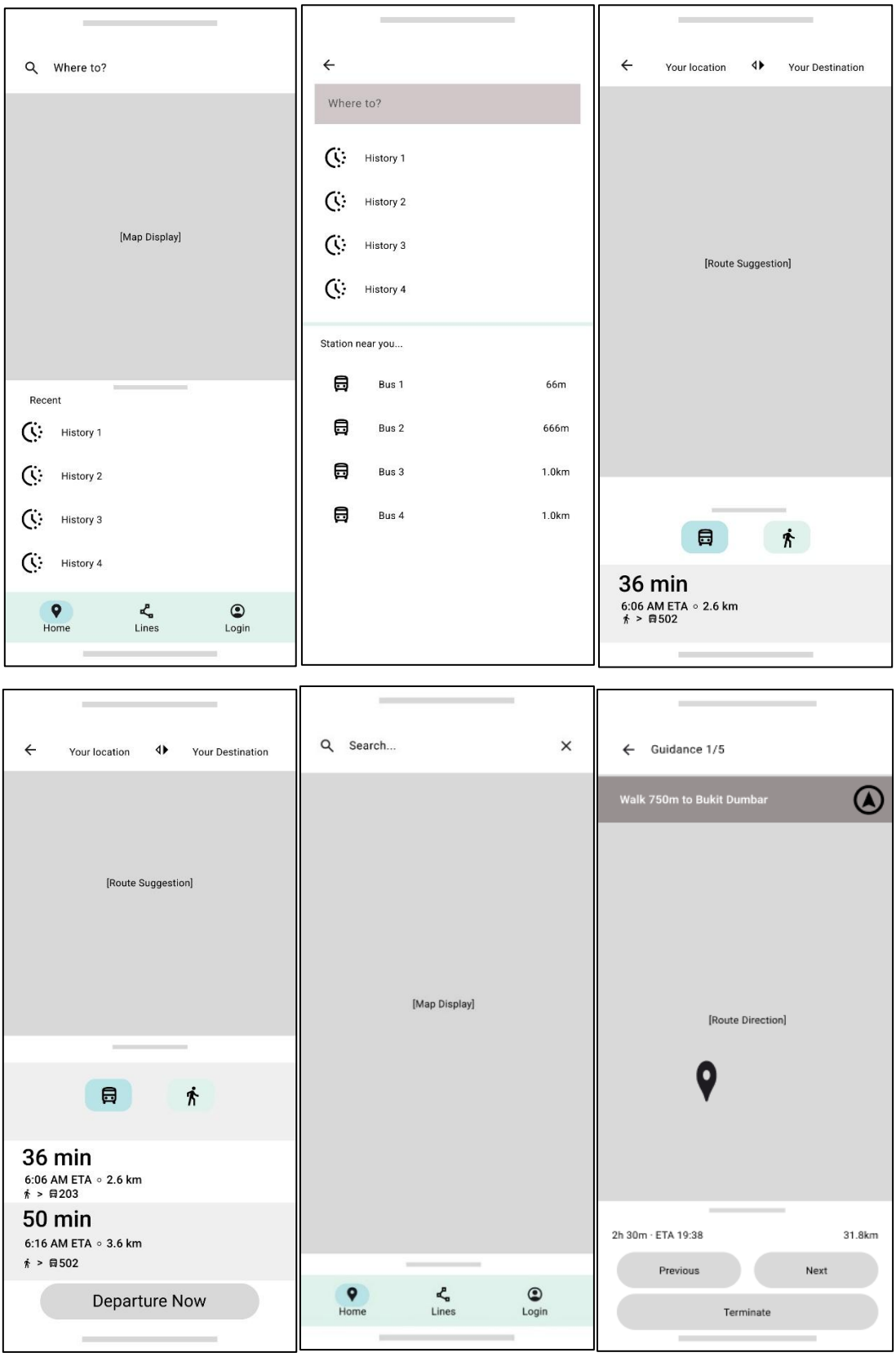
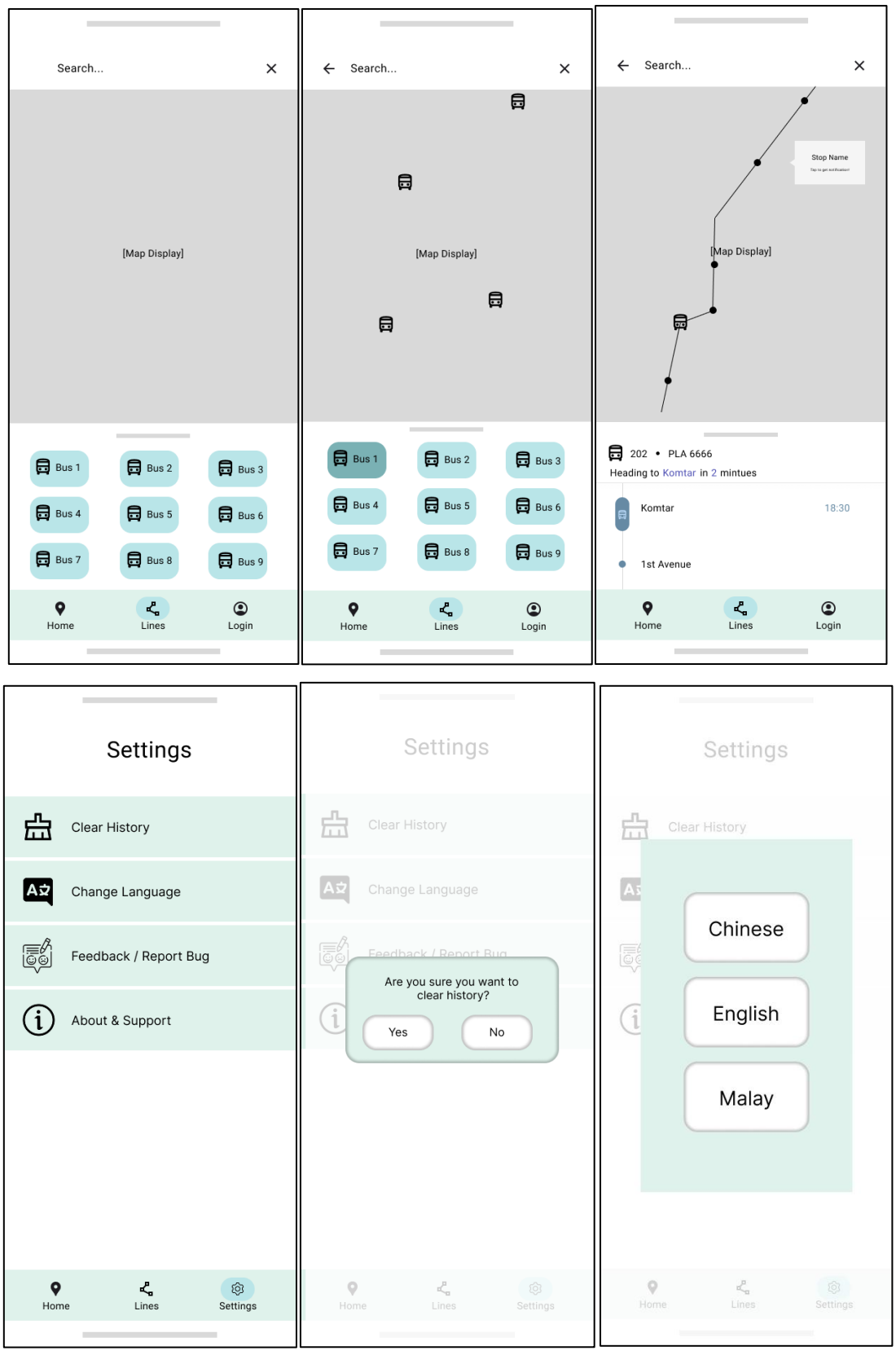


Figure 3.3.2: Sequence Diagram for Live Bus Checking Process

3.4 Wireframe Design




Search...
[Map Display]
Bus 1
Bus 2
Bus 3
Bus 4
Bus 5
Bus 6
Bus 7
Bus 8
Bus 9
Home
Lines
Login
Stop Name
202 • PLA 6666
Heading to Komtar in 2 mintues
Komtar
18:30
1st Avenue
Settings
Clear History
Change Language
Feedback / Report Bug
About & Support
Are you sure you want to clear history?
Yes
No
Chinese
English
Malay

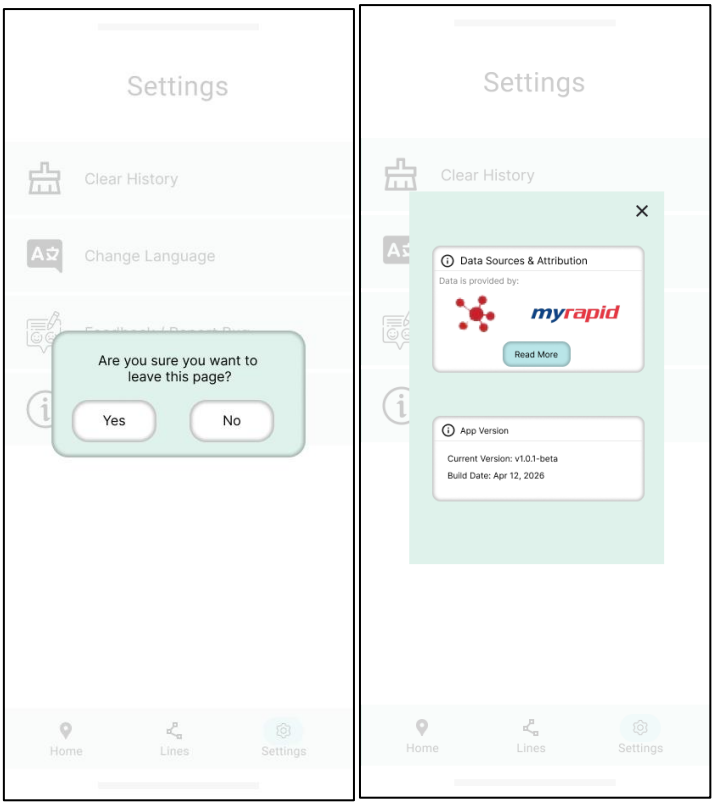


Figure 3.4.1: Wireframe Design

3.5 System Architecture

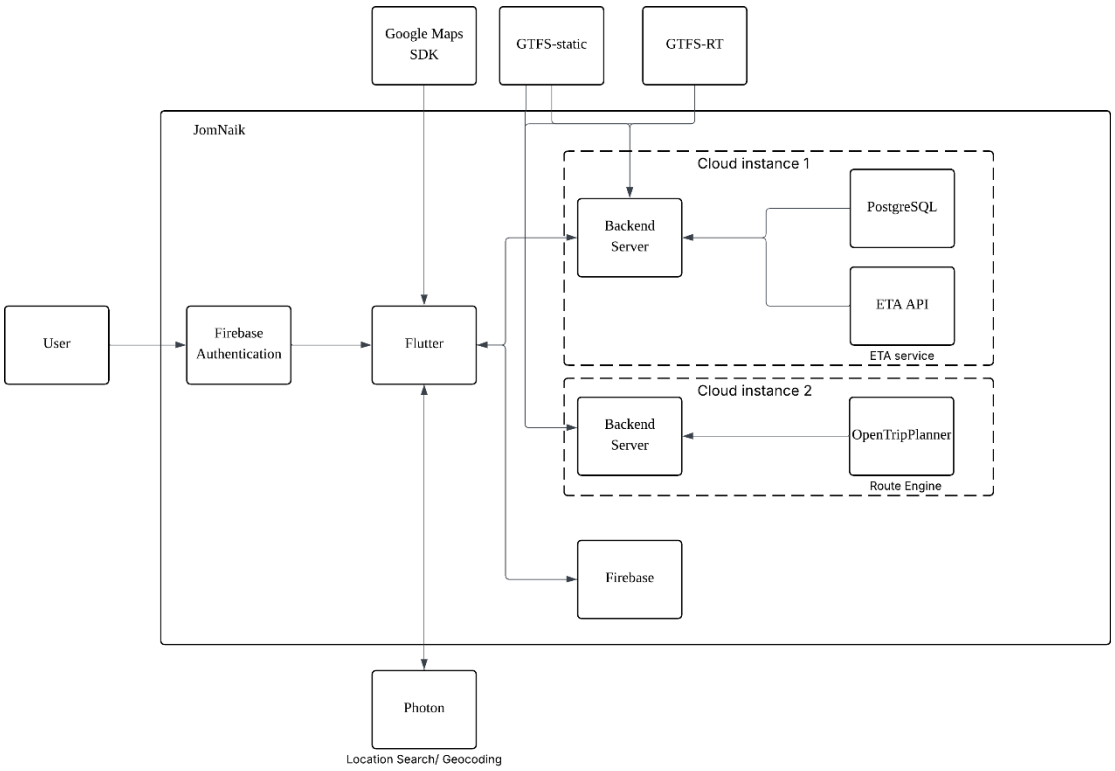


Figure 3.5.1: System Architecture

Figure 3.5.1 illustrates the architecture of this system, which refers to the concept of multi-tiered architecture that has a clear separation of concerns. This concept allows for greater flexibility in application implementation, allowing for independent development of modules without interfering with each other. This simplifies the debugging and management process by allowing other modules to work as usual, even when some component services fail.

3.6 Timeline

The following timelines provide details on the tasks needed to be completed and the duration of each task.

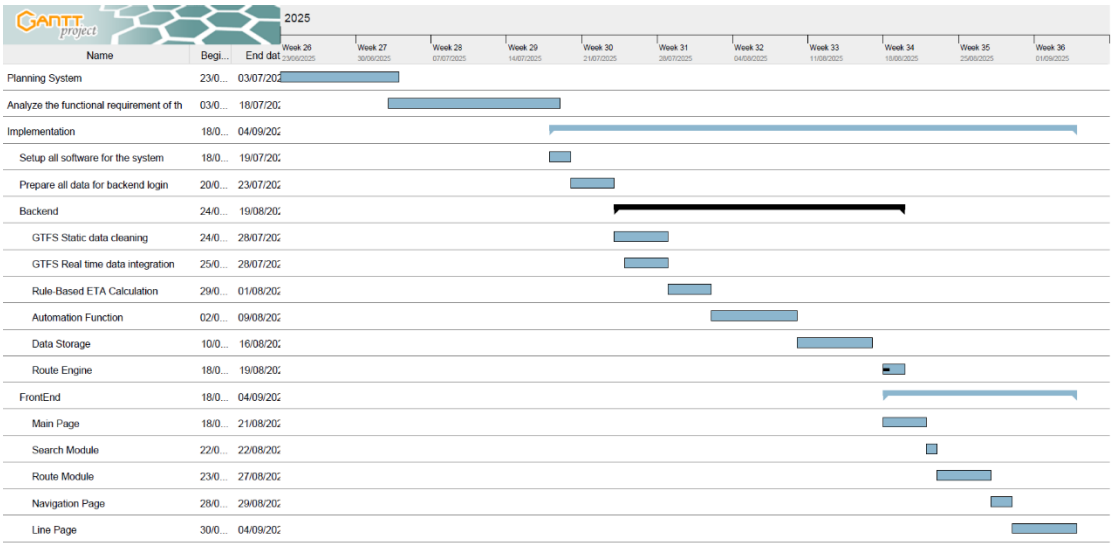


Figure 3.6.1: Project I Timeline

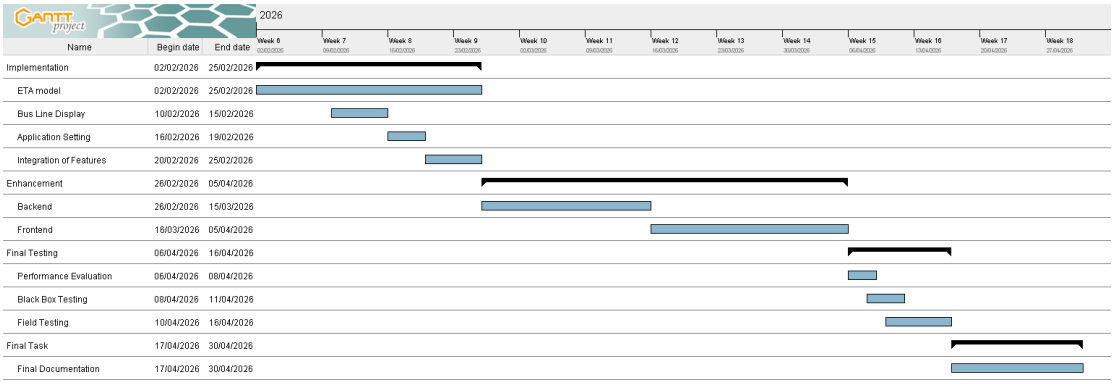


Figure 3.6.2: Project II Timeline

CHAPTER 4 System Design

4.1 System Flowcharts

An overall system workflow is divided into several diagrams to provide a detailed information on how front-end work with back-end logic.

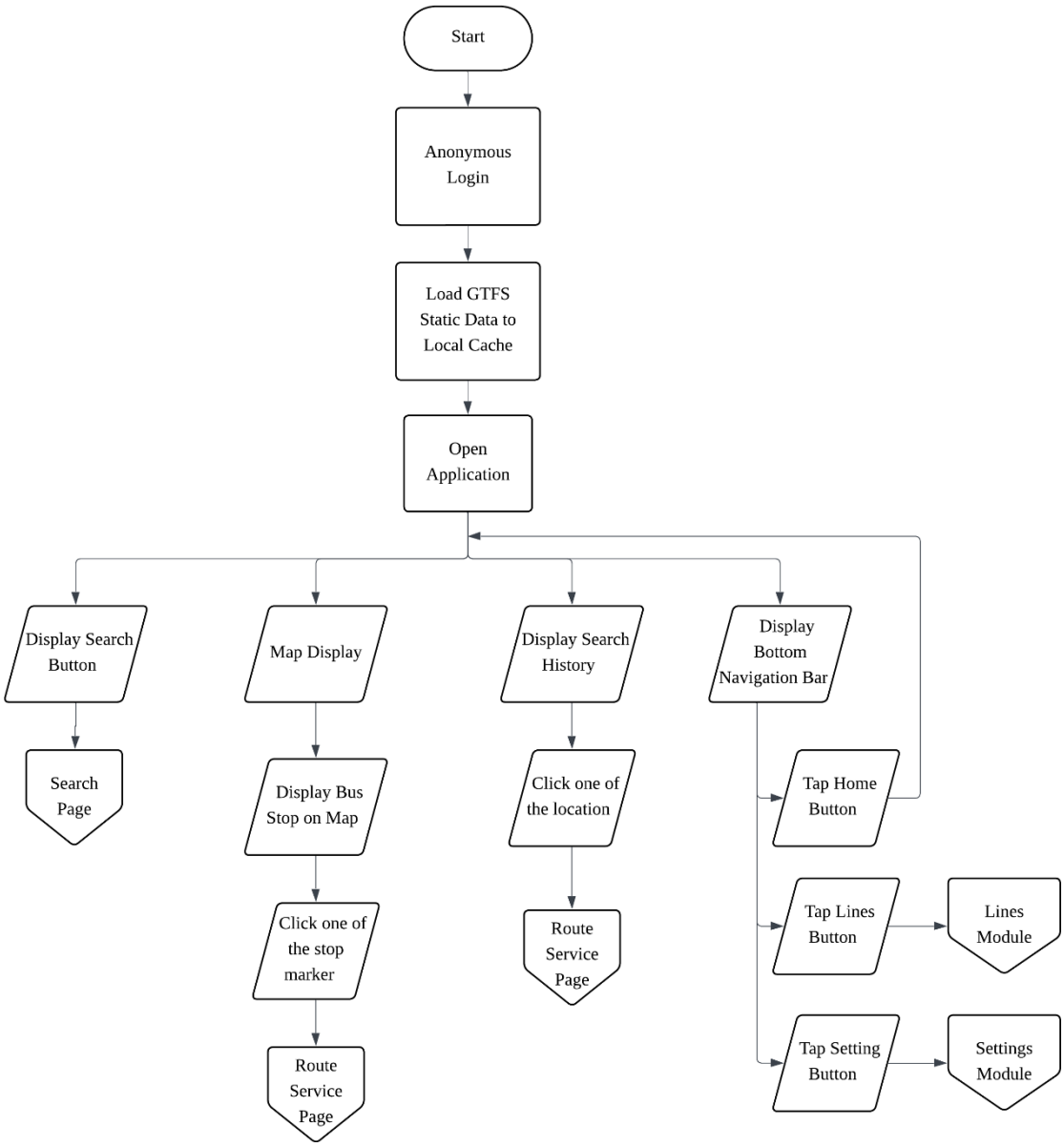


Figure 4.1.1: Main Page

Figure 4.1.1 demonstrates the flow start from user open the application. Once user enter to the main page, the system will automatically check whether the user use with registered or anonymous account but will not force user to login with an account. Then, several functions like map display with stop marker, place search button, history search and navigation bar including home option, bus lines optionand settings option are

displayed. When user click on the marker displayed on map will pop out an information dialog that show the information about the stop. When user click on search button, will be redirected to Search Page. When user click on a history search card will be redirected to Route Module to choose a suggested route and start navigating. When user click on lines button, will be redirected to Lines Page that allow user to look for specific bus. Lastly, when user click on setting button, will be redirected to Settings Page that allow user to change application settings.

4.1.1 Search Page

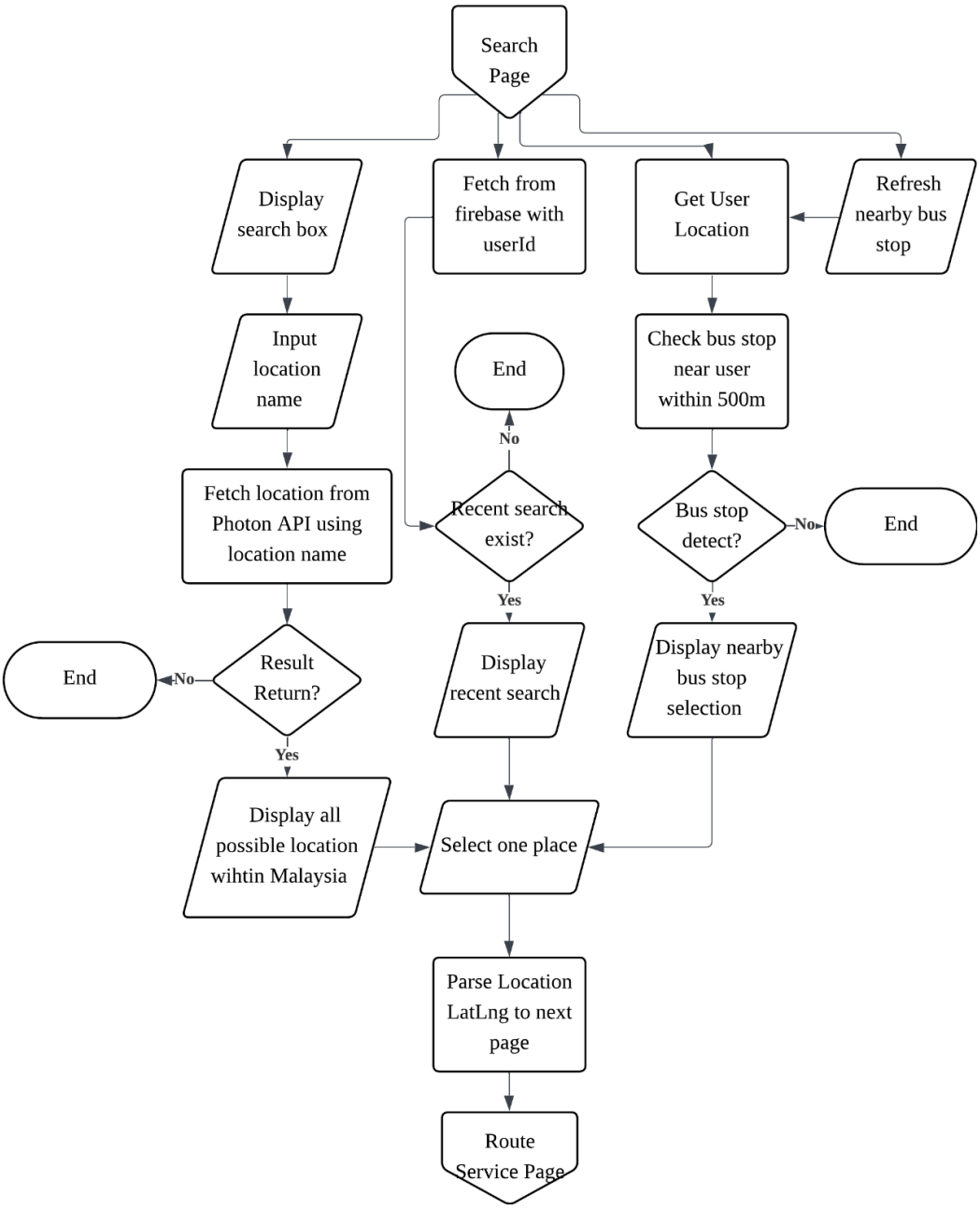


Figure 4.1.2: Search Page

Figure 4.1.2 illustrates the flow of search page. After user click on a search button, they will be redirected to this page. This page allows users to search for their destination, or they can choose a place from recent search results. Additionally, the system will use user current location to find nearby bus stop, guide user to the selected bus stop when user choose to do so.

4.1.2 Route Service

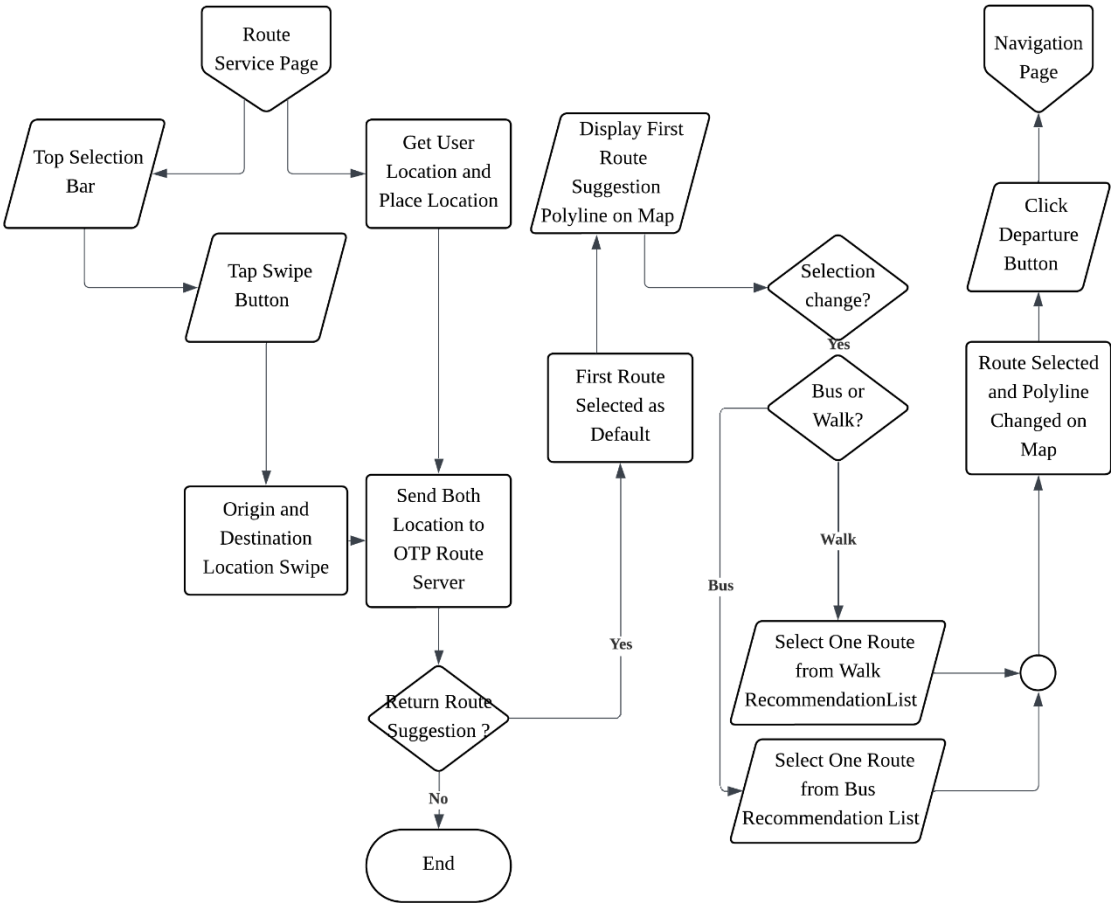


Figure 4.1.3: Route Service

After user selects an option from search page, the system will parse longitude and latitude of the destination to route server. By default, the route server only serves for Penang, Malaysia area, so if user selects a location outside the Penang boundary, the server will not return any route suggestions. This module also allows users to exchange between starting point and ending point for a new location, providing a flexible service for user.

4.1.3 Navigation Module

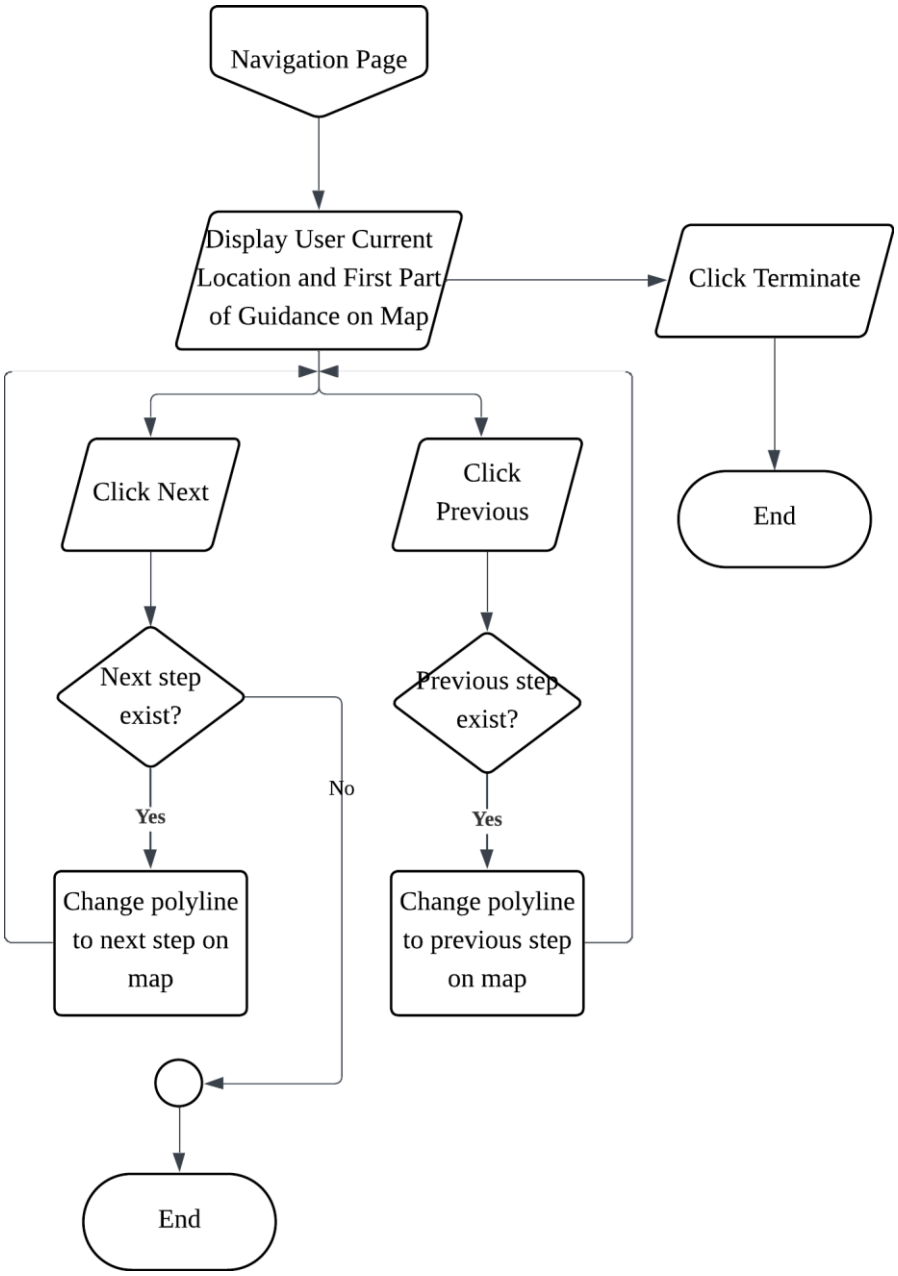


Figure 4.1.4: Navigation Module

When receiving user selection on route, the system will parse the route details to Navigation Module. This Module will divide the route into several steps based on the number of route instructions. The user can click “next” for the next guidance, the system will automatically proceed to the next step. The flow will loop until the last step of guidance, or user selects to terminate the navigation service.

4.1.4 Lines Module

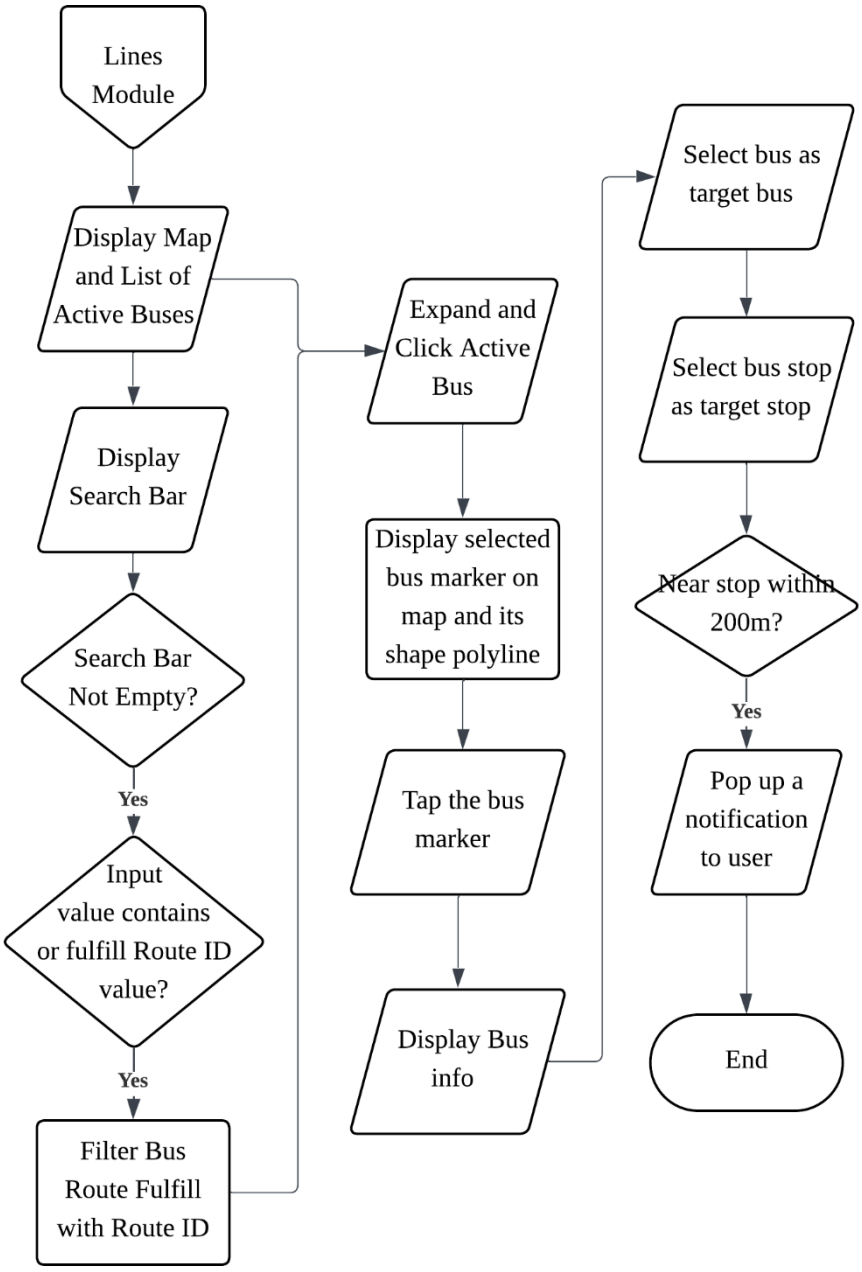


Figure 4.1.5: Lines Module

Figure 4.1.5 is a module that shows all active buses on map. This page allows users to find a specific bus based on the route name. When tapping for a bus route, the system will display the bus marker on map, which allows user to choose the bus as targeted bus. If user wants to get a notification when the selected bus is about to arrive at a stop, they can choose a stop as the target stop and start the process. When the bus is near the stop location, a push notification will pop up to remind user to take the targeted bus.

4.1.4 Settings Module

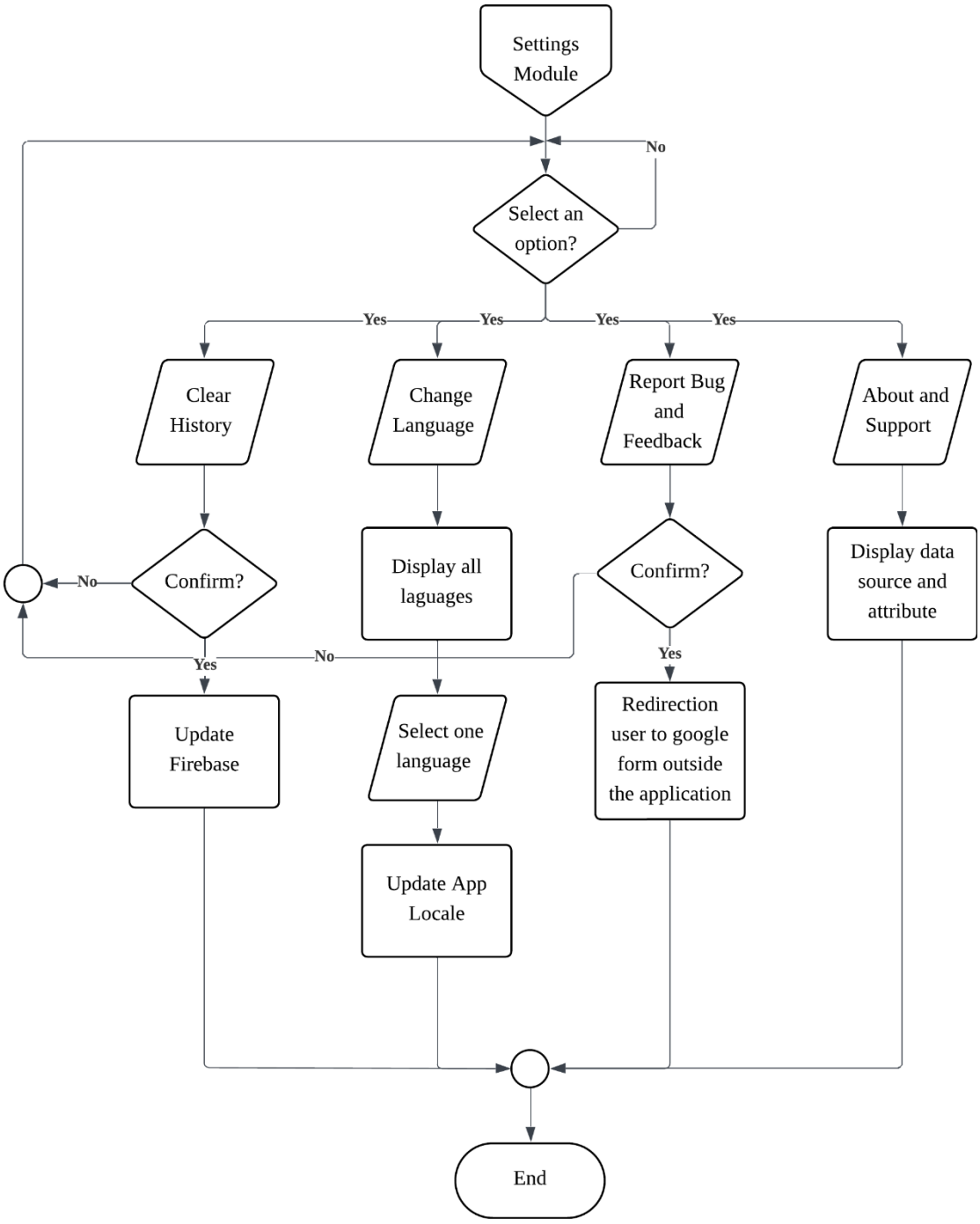


Figure 4.1.6: Settings Module

Figure 4.1.6 is a module that allow passenger interacts with application settings. The main features provided by this module are clear search history, change language (English, Chinese, Malay), allow user to report bugs and provide feedback, and lastly view about and support.

4.2 Database Design

In this project, there are two types of database are used, which are Firebase Firestore and PostgreSQL.

Firebase Firestore is a NoSQL document database for storing user-related data. Currently, it is only used to store search history. The purpose of using Firestore is that it can seamlessly integrate with Firebase Auth, so we can make good use of the obtained user ID to store user info for each passenger.

PostgreSQL is used as an object-relational database management system (ORDBMS) to manage all data related to geospatial, where PostGIS is used as an extension for the PostgreSQL DBMS to interact with geospatial data. PostgreSQL can compensate for the limitations of NoSQL in interacting with the database, especially for data involving geographic location calculations, such as using live bus coordinates to find the nearest segment in database.

4.2.1 Firebase Firestore

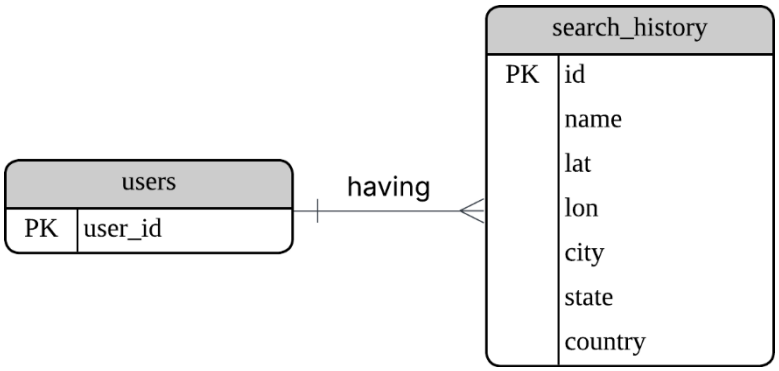


Figure 4.2.1: Entity Relationship Diagram – Firebase Firestore

Figure 4.2.1 shows the entity in our Firestore, each entity representing a collection in our Firestore instance. The diagram mainly reflects the user-related information we store, which consists of “users” as a main collection, and “search_history” as a sub-collection of “users”. Every time the user opens the application, a user ID is stored in our database, and it is reused until the application is deleted. When a user searches for a location name and selects a destination returned from the search engine results, a search history record is created and added to the Firestore.

4.2.2 PostgreSQL

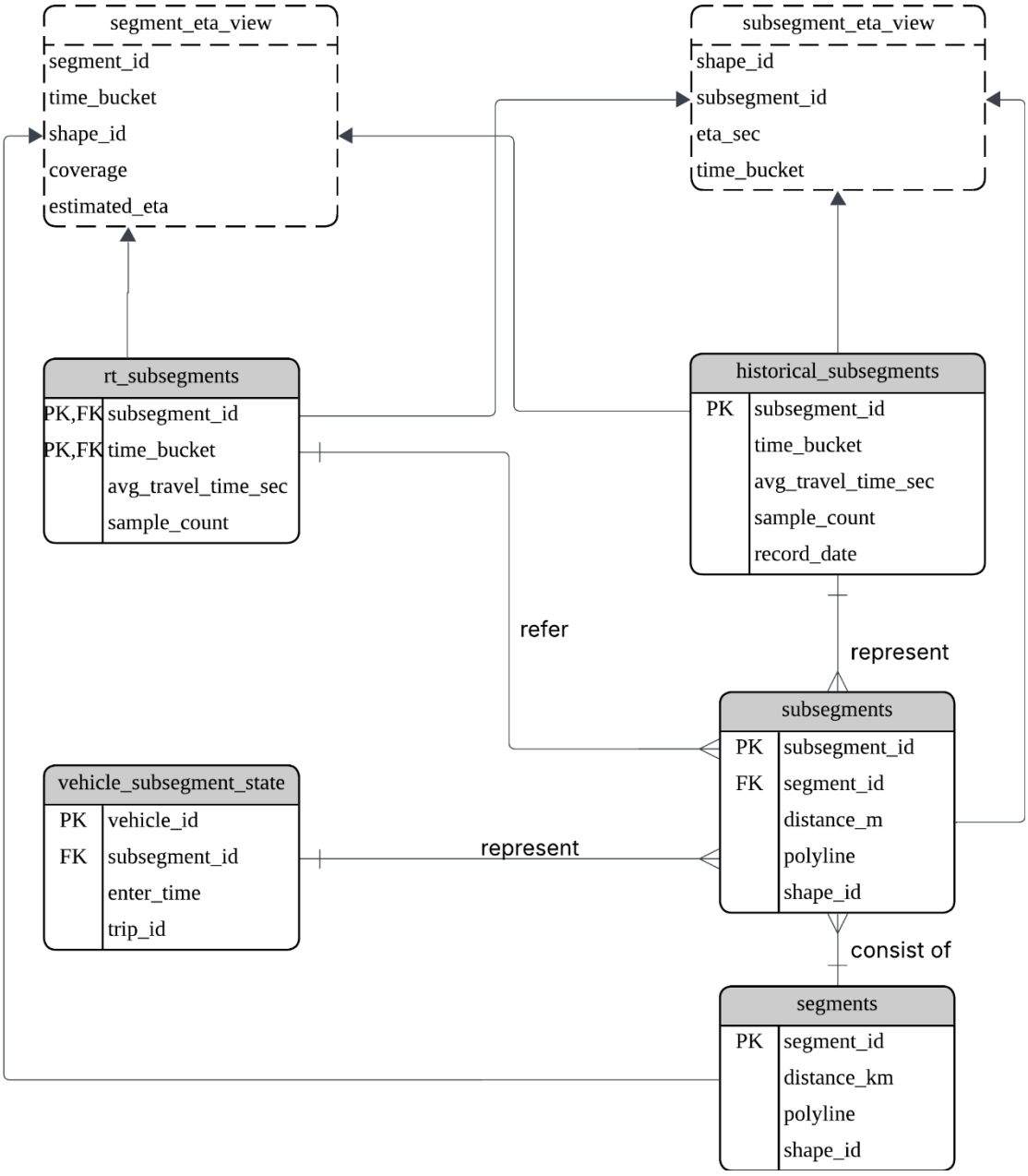


Figure 4.2.2: Entity Relationship Diagram – PostgreSQL

Figure 4.2.2 shows the entity relationship in our PostgreSQL database. The solid line entity represents a table in our database, and the dotted line entity represents a view derived from tables in our database.

"Segments" and "subsegments" tables store the road segmentation data and are used as a lookup table for other tables like "rt_subsegments" and "vehicle_subsegments_state". The "vehicle_subsegment_state" is used to store the current status of each active bus. Once the system gets the first record for a live bus, the first thing it will do is to query

if there is an existing record stored in the table. If it is a new record, it will find the nearest possible subsegment for that live bus and store this info along with the time the system obtained the record. If the system finds there is an existing record for the live bus, it checks whether it is in the same subsegment as the previous record. If it is in the same subsegment, it calculates the travel time using the enter time and current time and stores it in the "rt_subsegments" table. Otherwise, it continues to monitor the status of this live bus.

The "rt_subsegments" table stores the real-time data for each subsegment once there is a record is established in "vehicle_subsegments_state." This table mainly stores the travel time for each subsegment in different time buckets and combines it with historical datasets to provide ETA information. The data stored will be truncated every day to make room for the data of the next day.

The "historical_subsegments" stores the data from "rt_subsegments" before it was truncated, which used to provide average travel time when subsegment coverage is insufficient in the "rt_subsegments" table.

Lastly, two virtual tables are derived from the physical tables. The purpose of creating these views is to store SQL statements in the views so that whenever we need ETA information, we can directly access the view without running lengthy SQL statements.

4.3 Segment-based ETA Prediction

This section will mainly discuss how we plan to provide the Estimated Time of Arrival (ETA). Data exploration revealed that GTFS-RT data does not provide any information regarding the next stop each live bus is heading to, which is crucial in predicting ETA. Hence, to understand which is the next stop for each live bus, we mainly utilize the GPS of the live bus to find the nearest road segment and determine which stop it is heading to.

Additionally, we found that the current speed for each live bus returned by GTFS-RT is mostly 0. Since the speed was consistently 0, calculating ETA based on the rule of speed, distance, and time could not provide a reasonable Estimated Time of Arrival (ETA). Thus, we introduced segment-based ETA prediction in our project to address the implementation challenges of not providing next-stop information and the lack of effective travel speed.

ETA calculation

Segment-based ETA is the sum of the travel time for all segments of the transit trip. Before diving into ETA calculation, we need to define the road segment. According to most researchers' methods, the most common approach is stop-to-stop segmentation. However, this method is more effective for vehicles with a continuous GPS sampling frequency. For our system, which only has a sampling frequency of once every 30 seconds, it is difficult to determine the exact location of the vehicle within a road segment. Therefore, we plan to segment the road based on stop-to-stop segmentation, and then further partition the segment into subsegments every 100 meters.

Below is a high-level concept of segment-based ETA:

1. $T_{segment_i} = \sum_{j=1}^{n} T_{subsegment_{ij}}$
2. $T_{subsegment_j} = T_{enter} - T_{exit}$

Where:

- $T_{segment_i}$: The travel time for a segment *i*
- $T_{subsegment_j}$: The j^{th} subsegment of the segment
- T_{enter}: The time a bus enters the subsegment *i*
- T_{exit}: The time a bus exits the subsegment *i*

Time Bucket

In addition to segmenting road sections, we also considered the impact of traffic conditions on ETA. Therefore, we divided peak hours and off-peak hours based on the congestion level in Georgetown, Penang, Malaysia, provided by TomTom (as shown in Figure 4.3.1).

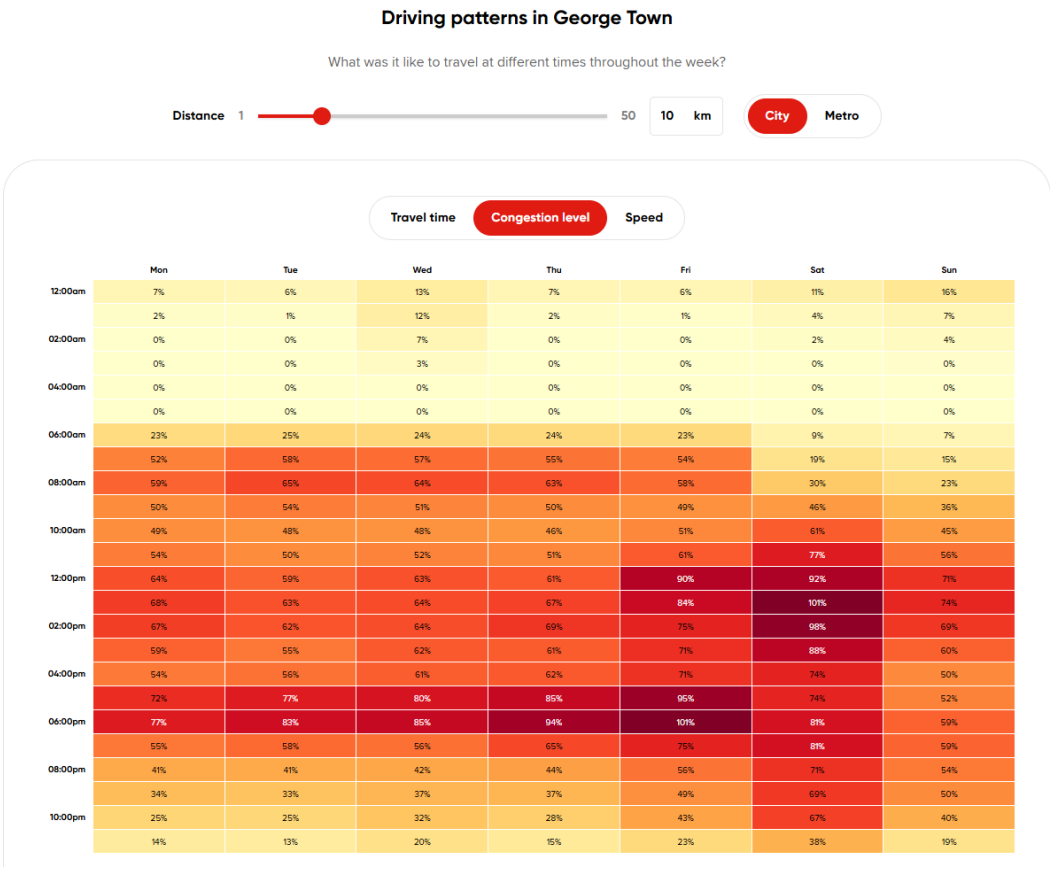


Figure 4.3.1 Congestion Level in Georgetown Penang, Malaysia.[30]

Based on the colour intensity, we can visually observe traffic patterns in Penang. For example, on weekdays (Monday to Friday), the congestion percentage gradually increases after 7 am and only subsides around 7 pm. The traffic pattern on weekends (Saturday and Sunday) differs significantly from weekdays, especially with severe traffic congestion from 12 pm to 2 pm. Therefore, based on this heatmap, we initiate time buckets, differentiating between peak and off-peak hours depending on whether it is a weekday or weekend.

Below is a table showing the time buckets, where the system collects the real-time info every day, starting from 6 am to 11.59 pm:

Time Bucket		
Name / Days	Weekday	Weekend
Morning Peak	7 am to 9 am	9 am to 11 am
Afternoon Peak	11 am to 5 pm	11 am to 4 pm
Evening Peak	5 pm to 7 pm	4 pm to 11 pm
Off-peak	At all other times except for the times mentioned above.	

Table 4.3.1: Time Bucket

Historical Data

The real-time lookup database that collects ETA data will be truncated at the end of the day to make room for the next day's real-time data. Instead of removing the useful data, we plan to move the data collected throughout the day to another archival database as a historical dataset, so that we can refer to historical data when there is insufficient data for a specific road segment for the next day.

Below is the equation that indicates how we combine real-time data with historical data to provide the ETA:

$$\mathrm{ETA}_{\mathrm{segment_{ij}}} = \mathrm{c_{rt}} \cdot \mathrm{T_{rt}}_{ij} + \mathrm{c_{hist}} \cdot \mathrm{T_{hist}}_{ij} + (1 - \mathrm{c_{rt}} - \mathrm{c_{hist}}) \cdot \mathrm{T_{avg}}_{j}$$

Where:

- i: The segment i
- j: The time bucket j
- $ETA_{segment_{ij}}$: The ETA for the segment i is given in the time bucket j
- c_{rt}, c_{hist} : The coverage for real-time and historical data respectively
- $T_{rt_{ij}}$, $T_{hist_{ij}}$: The real-time and historical travel time of the segment i in time bucket j respectively
- $\mathrm{T_{avg_j}}$: The average travel time in time bucket j

CHAPTER 5 System Implementation

In this chapter, we will discuss how the system will be implemented, some of the limitations we discovered, and how these issues can be addressed during the development phase to reduce their impact.

5.1 System Requirement

5.1.1 Hardware

The hardware involved in this project is computer and android mobile device. A computer is used to implement the system such as user interface, animation and all the required process. A smartphone is used to test and deploy the mobile application.

Description	**Specifications**
Model	G7 GE
Processor	12th Gen Intel(R) Core(TM) i5-12500H, 2500 Mhz, 12 Core(s), 16 Logical Processor(s)
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX 3050 Laptop GPU
Memory	16GB RAM
Storage	512GB NVMe SSD

Table 5.1 Specifications of laptop

Description	**Specifications**
Model	SM-N970F/DS (Samsung Galaxy Note 10)
Processor	Exynos 9825 Octa-core (7nm)
Operating System	Android 12, One UI 4
Graphic	Mali-G76 MP12
Memory	8GB RAM
Storage	256GB UFS 3.0

Table 5.2 Specifications of smartphone

5.1.2 Software

There is some software need to be installed and set up in the laptop for developing the system and application. The following requirements are the software requirements for this project.

Development	Software Tools
Mobile Application Implementation	Visual Studio Code
ETA and ETA predictions logic implementation	Python, Oracle Cloud Instance (Ubuntu OS)
Routing engine implementation	Java Runtime Environment, Oracle Cloud Instance (Ubuntu OS)
Android Application Testing	Android Studio Emulator

Table 5.3 Software requirements

5.1.3 Techniques

To implement this project, some Application Programming Interface (API) and Software Development Kit (SDK) are used for different purposes. The following is a description for each usage of the API and SDK.

Name	Description
Google Maps SDK	Map Display
OpenTripPlanner (OTP)	A free routing engine. OTP is a Java-based application. It helps in calculating the route.
GTFS-RT	A free real-time transport API provided by data.gov.my. Used for live bus tracking data.
GTFS	A free real-time transport API provided by data.gov.my. Used for transit stops, route, scheduled timetable.
Flutter	An SDK that builds mobile applications in Android and iOS.
Firebase	An SDK that is used for cloud storage in the mobile app.

Photon	A free geocoding service
PostgreSQL	Store geospatial data (bus stops, road segments) and historical data. Support PostGIS query.
FastAPI	Build a RESTful APIs, providing ETA prediction results to the front end.

Table 5.4 Techniques requirements

5.1.4 Programming Language

There are some programming languages that will be used to implement this project. The following is a description of each usage of the programming language.

Name	**Description**
Dart	A dynamic object-oriented language used to develop mobile application. In this project mainly used by flutter.
Python	A high-level programming language to handle ETA logic calculation.
SQL	A programming language used for complex geospatial queries (PostGIS) and managing relational databases (PostgreSQL)

Table 5.5 Programming Languages

5.2 Setting and Configuration

5.2.1 Software

Before starting to develop the application, there are some software needed to be set up:

1. Visual Studio Code

   To begin development in this IDE, several extensions need to be downloaded in advance:

   a. Flutter SDK
   b. Dart
   c. Gradle for Java
   d. Python

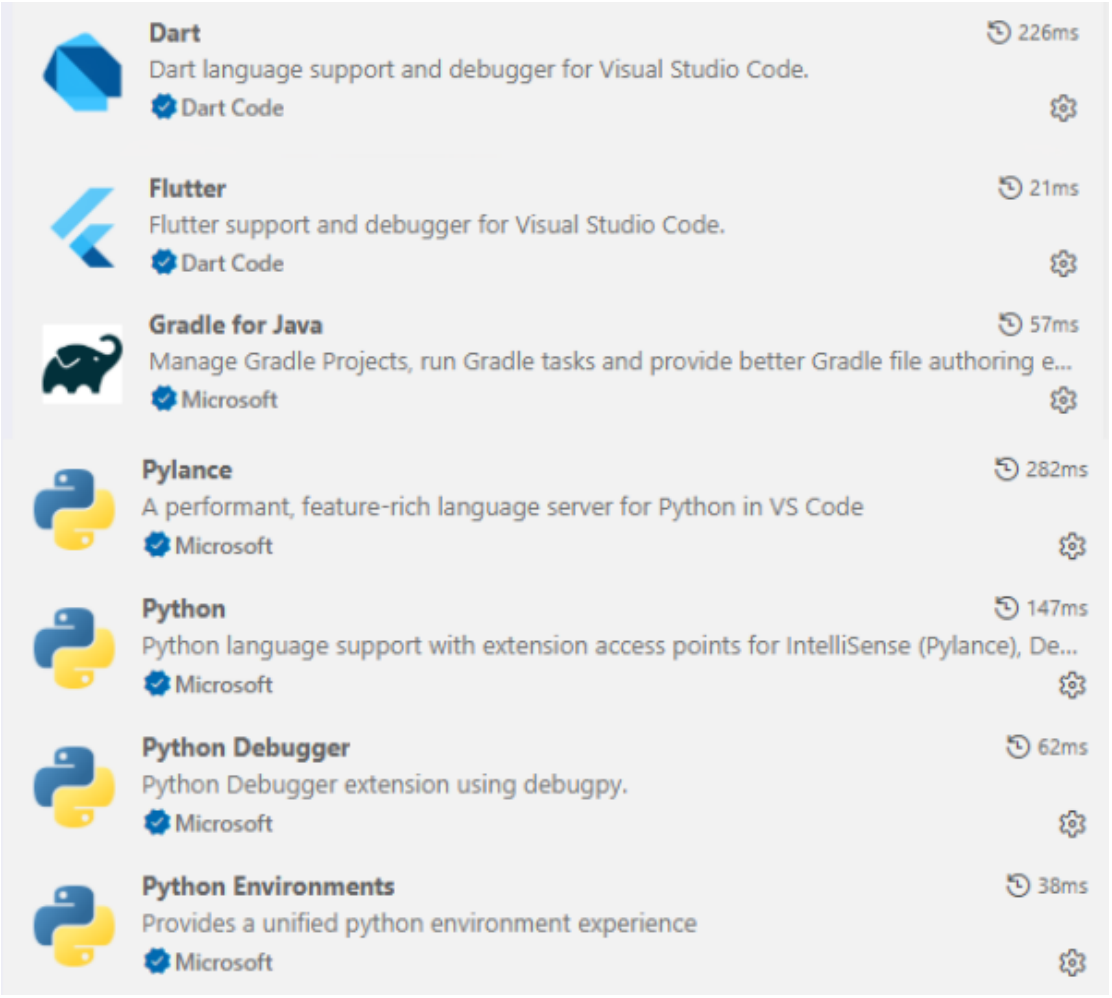


Figure 5.2.1 Extensions in Visual Studio Code

2. Cloud Instances

   Two instances were created, one for hosting the route server and the other for collecting real-time data in PostgreSQL and hosting FastAPI to provide ETA information.

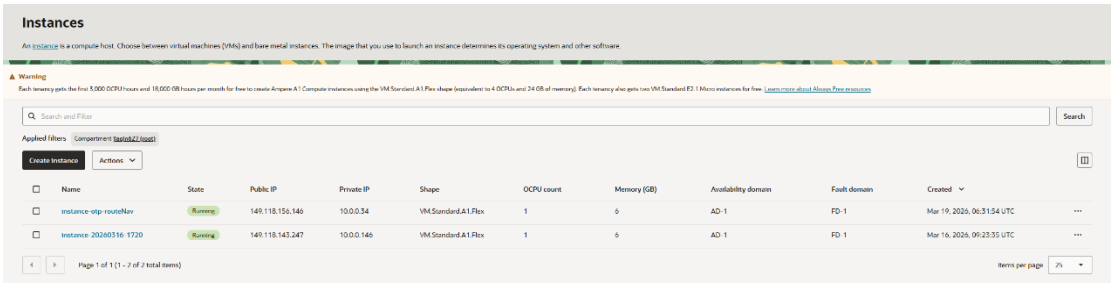


Figure 5.2.2 Oracle Cloud Instance Console

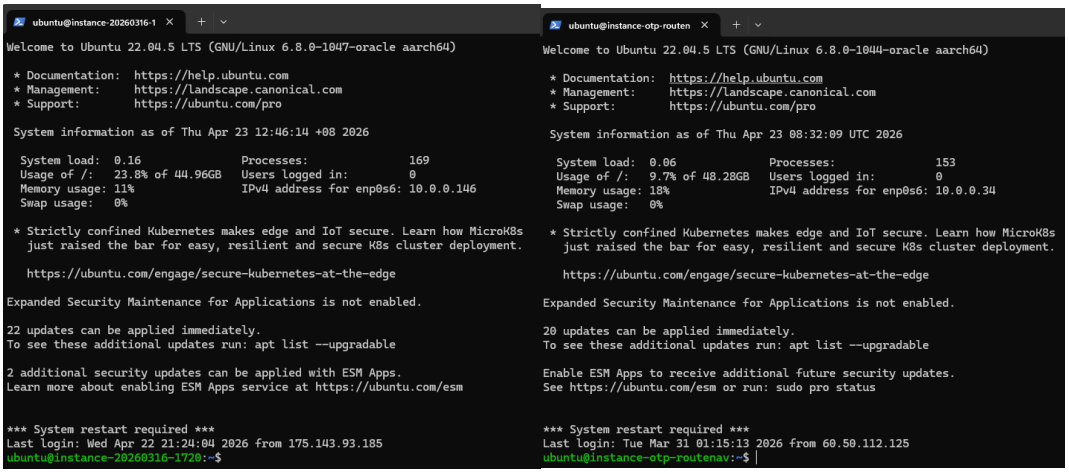


Figure 5.2.3 Terminal of Cloud Instance

5.2.2 External Resource

1. Transport API
   - A GTFS Static API URL and related documentation was obtained from Malaysia's Open Data Portal without account registration
   - A GTFS-RT API URL and related documentation was obtained from Malaysia's Open Data Portal without account registration
2. Google Maps API

   A Google Maps API key was obtained from the Google Cloud Console (GCP) with some enabled services as follows:
   - Maps SDK for Android
3. Search Engine API
   - Photon (Open-Source geocoding with OpenStreetMap data)

4. Route Engine

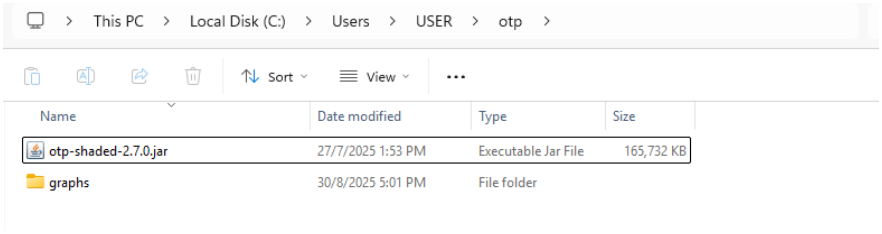


Figure 5.2.4: Files of OTP service

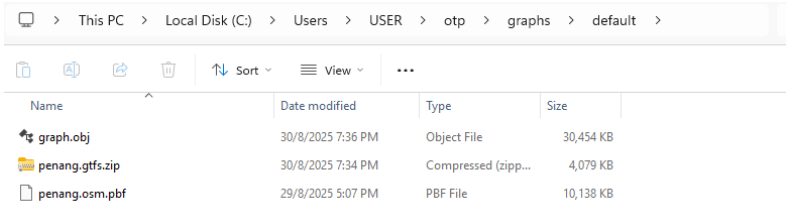


Figure 5.2.5: Files of OTP service

- OpenTripPlanner (v2.7.0), an open-source multi-modal trip planner that utilizes scheduled public transportation to provide route service.
- GTFS for Transit Schedules and Stops, tailored to Penang, Malaysia Area
- OSM PBF data for the same geographic region as the GTFS feed.
- A graph.obj is built after preparing and running the OTP locally

All files were moved to the cloud instance after ensuring the server was functioning as intended. A crontab has been configured to update the route server when the GTFS static data is updated.

```
ubuntu@instance-otp-routenav:~$ ls
otp  refresh.sh  script
ubuntu@instance-otp-routenav:~$ cd otp
ubuntu@instance-otp-routenav:~/otp$ ls
graphs  otp-shaded-2.7.0.jar
ubuntu@instance-otp-routenav:~/otp$ ll
total 165744
drwxrwxr-x 3 ubuntu ubuntu      4096 Mar 19 06:48 ./
drwxr-x--- 7 ubuntu ubuntu      4096 Apr 23 10:32 ../
drwxrwxr-x 3 ubuntu ubuntu      4096 Mar 19 06:49 graphs/
-rw-rw-r-- 1 ubuntu ubuntu 169709201 Mar 19 06:48 otp-shaded-2.7.0.jar
ubuntu@instance-otp-routenav:~/otp$ cd graphs
ubuntu@instance-otp-routenav:~/otp/graphs$ ll
total 12
drwxrwxr-x 3 ubuntu ubuntu 4096 Mar 19 06:49 ./
drwxrwxr-x 3 ubuntu ubuntu 4096 Mar 19 06:48 ../
drwxrwxr-x 2 ubuntu ubuntu 4096 Mar 19 08:03 default/
ubuntu@instance-otp-routenav:~/otp/graphs$ cd default
ubuntu@instance-otp-routenav:~/otp/graphs/default$ ll
total 209584
drwxrwxr-x 2 ubuntu ubuntu      4096 Mar 19 08:03 ./
drwxrwxr-x 3 ubuntu ubuntu      4096 Mar 19 06:49 ../
-rw-rw-r-- 1 ubuntu ubuntu  30994961 Apr 23 10:10 graph.obj
-rw-rw-r-- 1 ubuntu ubuntu 169709201 Mar 19 06:51 otp-shaded-2.7.0.jar
-rw-rw-r-- 1 ubuntu ubuntu   3514085 Apr 23 10:08 penang.gtfs.zip
-rw-rw-r-- 1 ubuntu ubuntu  10381134 Mar 19 06:51 penang.osm.pbf
```

Figure 5.2.6: Files of OTP service in Cloud Server

5.2.3 Data Collection and Processing

Road Segmentation

This road segmentation process is done by using the Python programming language. The data used to process the road segmentation is derived from the GTFS static data.

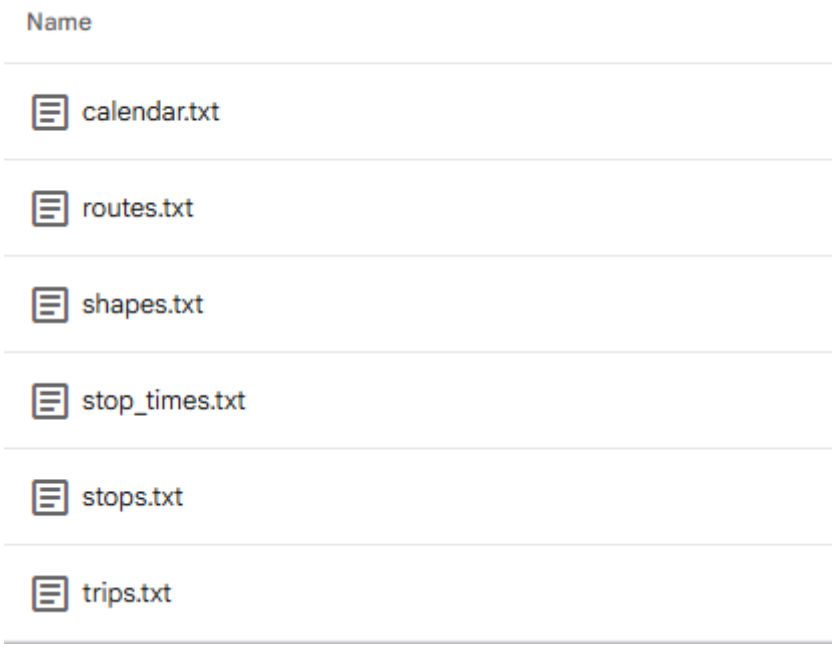


Figure 5.2.7: Files of GTFS Static

We first utilize the data from the GTFS static data, that is “stop_times.txt”, to find all unique stop-to-stop records as a segment. Then, combine with “stop.txt” to get the starting point (from stop) and ending point (to stop) of each segment. Currently, the data contains duplicate segments. A road segment unique key is created using the stop IDs.

Furthermore, we refer to the “trips.txt” to get the “shape id” for each segment. The main purpose is to **get a polyline** for each segment in “shape.txt” later. This is also a unique key to ensure the **obtained segment** when getting the ETA information is correct, as the road segment needs to be distinguished not only by the stop-to-stop information but also by the shape of the route. For instance, the polyline portion of shape A overlaps with shape B, but their routes are different. The segment could be different when finding the road segment without using shape_id.

Before getting the information in “shapes.txt”, we dropped the duplicated segments based on the segment key with the shape ID. Then, we merge the current data with “shapes.txt” to get the polyline. Figure 5.2.8 shows all the road segments with its polyline on the map.

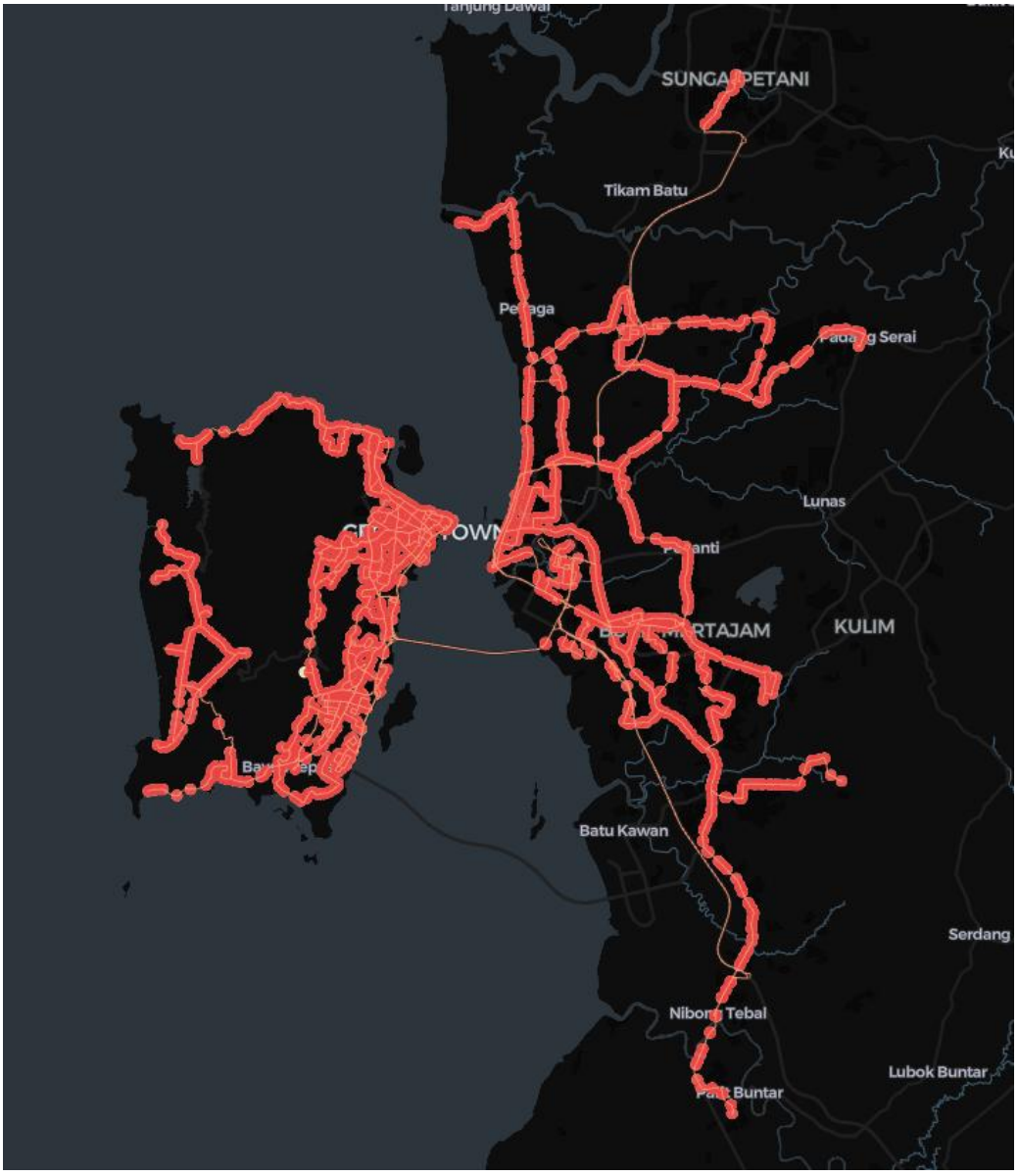


Figure 5.2.8: Stop-to-stop segmentation

The next step is to divide the road segment into several subsegments every 100 meters. This not only helps in finding the bus location, but also shows through the data exploration that the distances of all segments are different. Thus, dividing the road segment allow use to manage the road segmentation more efficiently. We divide the segments into subsegments based on the distance obtained from “shapes.txt”. Figure

5.2.9 shows the subsegment on the map. We can clearly see the dot at short intervals, proving that we have successfully divided the road segment into subsegments.

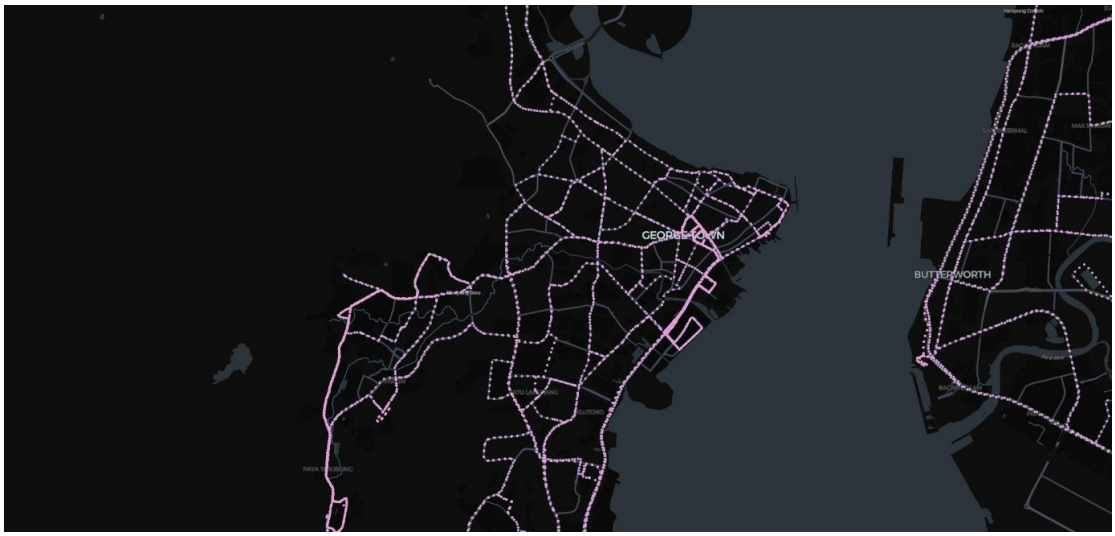


Figure 5.2.9: 100m sub-segmentation

	segment_id	distance_km	shape_id	sub_id	polyline
0	12002114_12000007	0.186782	1000000962	1000000962	[(5.4137981, 100.3416874), (5.413728599999999, 100.3415988), (5.4138595, 100.3414701), (5.4139316, 100.3413735), (5.41391, 100.34129), (5.41376, 100.34106), (5.41364, 100.34085), (5.41372, 100.34077), (5.41413, 100.34047)]
1	12000007_12001916	0.2664429999999993	1000000962	1000000962	[(5.41413, 100.34047), (5.41456, 100.34018), (5.41539, 100.33955), (5.41579, 100.33916), (5.41621, 100.33874)]
2	12001916_12001918	0.2274060000000001	1000000962	1000000962	[(5.41621, 100.33874), (5.41678, 100.33821), (5.417000000000001, 100.33794), (5.41751, 100.3374)]
3	12001918_12001920	0.1803089999999994	1000000962	1000000962	[(5.41751, 100.3374), (5.41767, 100.33698), (5.417999999999999, 100.33684), (5.41827, 100.33638), (5.418390000000005, 100.33617)]
4	12001920_12001922	0.306797	1000000962	1000000962	[(5.418390000000005, 100.33617), (5.41874000000001, 100.33542), (5.41909, 100.33457), (5.41925, 100.33426), (5.41948, 100.33368)]
5	12001922_12001064	0.2248320000000014	1000000962	1000000962	[(5.41948, 100.33368), (5.419890000000005, 100.33285), (5.419919999999999, 100.3326), (5.41984, 100.3324), (5.41945, 100.33214), (5.41909, 100.33185)]
6	12001064_12001061	0.1491599999999996	1000000962	1000000962	[(5.41909, 100.33185), (5.41897, 100.33175), (5.41863, 100.33155), (5.41847, 100.33146), (5.41836, 100.33139), (5.41805, 100.33118)]
7	12001061_12002116	0.6101589999999999	1000000962	1000000962	[(5.41805, 100.33118), (5.41763, 100.3309), (5.41725, 100.33061), (5.4171, 100.33044), (5.416930000000001, 100.3304), (5.41677, 100.33044), (5.41656, 100.33063), (5.416090000000005, 100.331), (5.4155, 100.33155), (5.4145900
8	12002116_12000356	0.7638349999999998	1000000962	1000000962	[(5.41426540000001, 100.330575), (5.4138328, 100.3301619), (5.4138435, 100.3297006), (5.41371, 100.32948), (5.413959999999999, 100.32902), (5.41412, 100.32888), (5.41431, 100.32885), (5.41446, 100.32886), (5.41545, 100.3
9	12000356_12000357	0.4428070000000002	1000000962	1000000962	[(5.417999999999999, 100.32836), (5.4186, 100.32789), (5.41913, 100.32726), (5.41963, 100.32665), (5.420080000000005, 100.32606), (5.4201, 100.32606), (5.42054, 100.32556), (5.4206, 100.32553), (5.42089, 100.32522)]
10	12000357_12000358	0.2150660000000002	1000000962	1000000962	[(5.42089, 100.32522), (5.4212, 100.32492), (5.42135, 100.32475), (5.42146, 100.32457), (5.42162, 100.32436), (5.42175, 100.3241), (5.42226, 100.32328)]

Figure 5.2.10: Example of segment data

	segment_id	subsegment_id	polyline	distance_m	shape_id	from_lat	from_lon	to_lat	to_lon
2	12002114_12000007	1000000962_12002114_12000007_0	[(5.4137981, 100.3416874), (5.413728599999999, 100.3415988), (5.4138595, 100.3414701), (5.4139316, 100.3413	99.9999995	1000000962	5.413798	100.3417	5.4136982	100.340951938788227
3	12002114_12000007	1000000962_12002114_12000007_1	[(5.413698250736149, 100.34095193878827), (5.41364, 100.34085), (5.41372, 100.34077), (5.41413, 100.34047)]	81.96677246237701	1000000962	5.4136982	100.34095	5.41413	100.3405
4	12000007_12001916	1000000962_12000007_12001916_0	[(5.41413, 100.34047), (5.41456, 100.34018), (5.414864286447354, 100.33994903558816)]	100.00000670277115	1000000962	5.41413	100.3405	5.4148642	100.33994903558816
5	12000007_12001916	1000000962_12000007_12001916_1	[(5.414864286447354, 100.33994903558816), (5.41539, 100.33955), (5.415562496650523, 100.33938181576575)]	99.99999573765308	1000000962	5.4148642	100.33994	5.4155624	100.33938181576575
6	12000007_12001916	1000000962_12000007_12001916_2	[(5.415562496650523, 100.33938181576575), (5.41579, 100.33916), (5.416202647112585, 100.33874735288741)]	99.99999776	1000000962	5.4155624	100.33938	5.4162026	100.33874735288741
7	12000007_12001916	1000000962_12000007_12001916_3	[(5.416202647112585, 100.33874735288741), (5.41621, 100.33874)]	1.153688048929743	1000000962	5.4162026	100.33874	5.41621	100.3387
8	12001916_12001918	1000000962_12001916_12001918_0	[(5.41621, 100.33874), (5.41678, 100.33821), (5.416857651076043, 100.33811470095213)]	100.00000096160049	1000000962	5.41621	100.3387	5.4168576	100.33811470095213
9	12001916_12001918	1000000962_12001916_12001918_1	[(5.416857651076043, 100.33811470095213), (5.417000000000001, 100.33794), (5.417464272994446, 100.33744	100.00000053200938	1000000962	5.4168576	100.33811	5.4174642	100.33744841682942
10	12001916_12001918	1000000962_12001916_12001918_2	[(5.417464272994446, 100.33744841682942), (5.41751, 100.3374)]	7.38777377901398	1000000962	5.4174642	100.33744	5.41751	100.3374

Figure 5.2.11: Example of subsegment data

5.2.4 Database Creation

1. Firebase Firestore

   The user-related information is stored in Firebase Firestore for easier access. Figure 5.2.12 shows the “users” collection with user ID as document ID. Each document ID refers to one “search_history” subcollection, where this collection will store the search history of users when using the app (as shown in Figure 5.2.13).

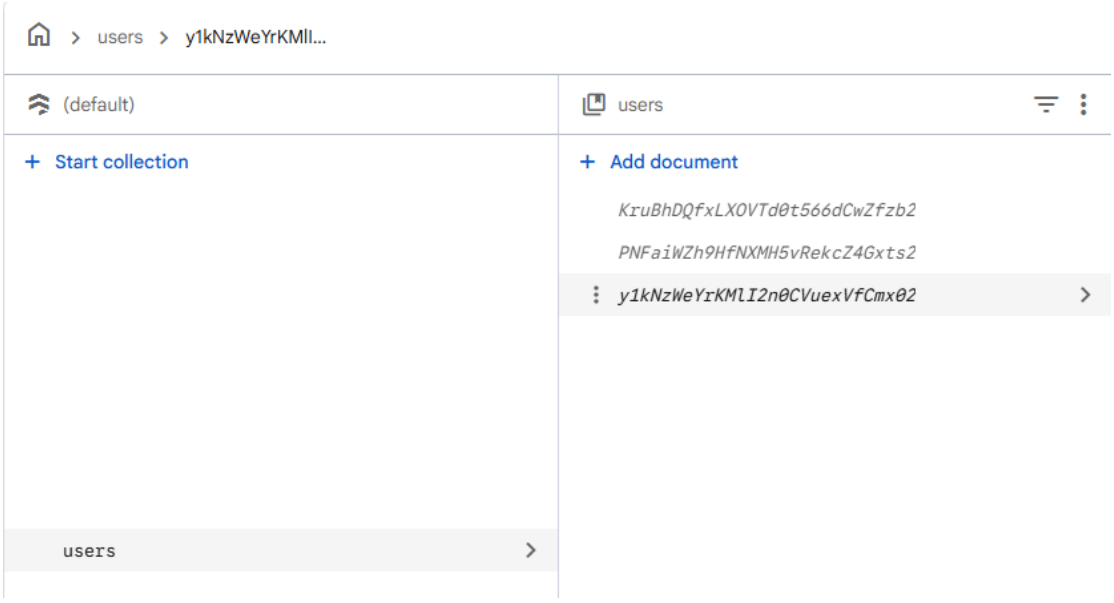


Figure 5.2.12: “Users” Collection in Firestore

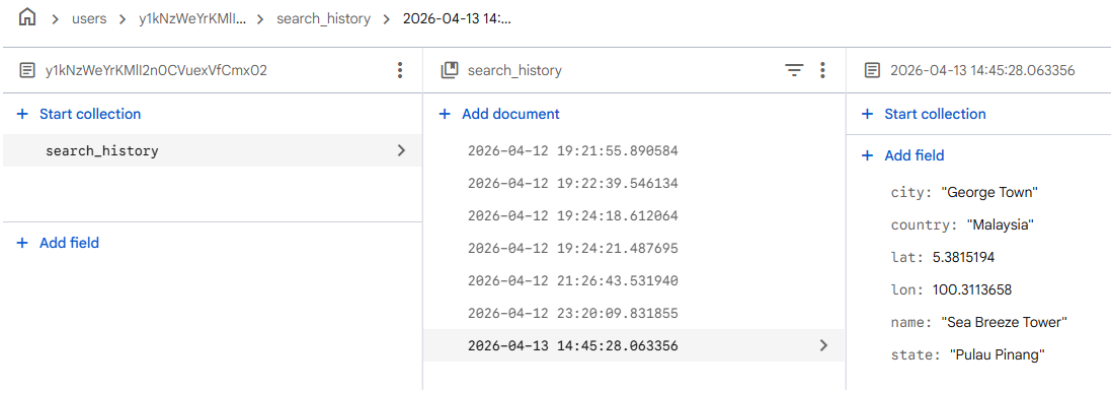


Figure 5.2.13: Example of search history record

2. PostgreSQL

The database is hosted in a cloud instance so that it can communicate with the database via the network.

The road segmentation data and the GTFS data (Static and Real-time) were stored in a PostgreSQL database, with the real-time bus information to provide ETA information for our application. Since the GTFS static data is updated after a period of time, we implement a crontab in the cloud instance to automate the update process so that we can always focus on more important development tasks.

Table

table_schema	table_name	column_name	data_type
public	segments	segment_id	text
public	segments	distance_km	double precision
public	segments	polyline	USER-DEFINED
public	segments	shape_id	text
public	segments	sub_id	text

Figure 5.2.14: Segment Table

table_schema	table_name	column_name	data_type
public	subsegments	subsegment_id	text
public	subsegments	segment_id	text
public	subsegments	distance_m	double precision
public	subsegments	polyline	USER-DEFINED
public	subsegments	shape_id	text

Figure 5.2.15: Subsegment Table

table_schema	table_name	column_name	data_type
public	rt_subsegments	subsegment_id	text
public	rt_subsegments	time_bucket	text
public	rt_subsegments	avg_travel_time_sec	double precision
public	rt_subsegments	avg_speed	double precision
public	rt_subsegments	sample_count	integer

Figure 5.2.16: Rt_subsegments Table

table_schema	table_name	column_name	data_type
public	vehicle_subsegment_state	vehicle_id	text
public	vehicle_subsegment_state	subsegment_id	text
public	vehicle_subsegment_state	enter_time	timestamp without time zone
public	vehicle_subsegment_state	speed	double precision
public	vehicle_subsegment_state	trip_id	text

Figure 5.2.17: Vehicle_subsegment_state Table

table_schema	table_name	column_name	data_type
public	historical_subsegments	subsegment_id	text
public	historical_subsegments	time_bucket	text
public	historical_subsegments	avg_travel_time_sec	double precision
public	historical_subsegments	sample_count	integer
public	historical_subsegments	record_date	date
public	historical_subsegments	avg_speed	double precision

Figure 5.2.18: Historical_subsegments Table

View

A virtual table derived from physical tables. These virtual tables are used to provide total travel time of segments as an ETA prediction. The view displayed here is the result of an SQL statement. For the query definition on how to create it, please refer to Appendix B.

view_name	column_name	data_type
segment_eta_view	segment_id	text
segment_eta_view	time_bucket	text
segment_eta_view	shape_id	text
segment_eta_view	coverage	double precision
segment_eta_view	estimated_eta	double precision

Figure 5.2.19: Segment_eta_view View

view_name	column_name	data_type
subsegment_eta_view	shape_id	text
subsegment_eta_view	subsegment_id	text
subsegment_eta_view	eta_sec	double precision
subsegment_eta_view	time_bucket	text

Figure 5.2.20: Subsegment_eta_view View

5.3 System Operation (with Screenshot)

5.3.1 Loading Screen

Figure 5.3.1.1 Loading Page

Figure 5.3.1.2 Animation after loading

Figure 5.3.1.1 shows the loading page of application. Once user opens the application, the system will begin initializing all required information to access the application. This process including initialize session for anonymous login, fetching GTFS static data from Firebase Storage and store in the cache, initializing the notification service and getting the locale language. After the initialization process, a login animation will be played to indicate that the process has completed.

5.3.2 Homepage

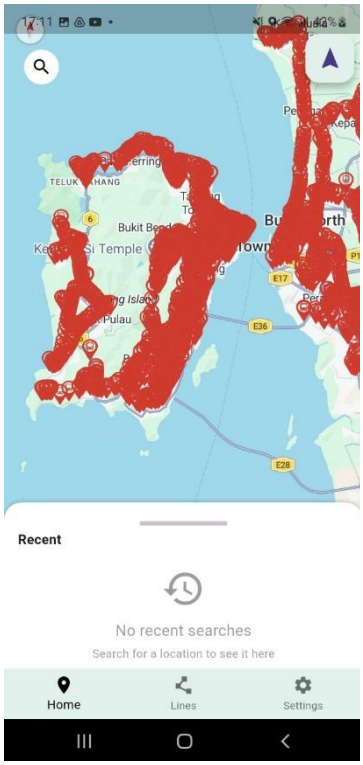


Figure 5.3.2.1 Homepage

Figure 5.3.2.2 Homepage UI after tapping the marker

Figure 5.3.2.3 Homepage UI after tapping the marker

After sucessfully enter the application, the first page that user will see is homepage, as shown in Figure 5.3.2.1. System will display all stop markers on the map, along with user's current location. User can interact by tapping any stop marker on map, and the system will redraw the markers and add polylines depending on the bus route that will stop at the selected bus stop (as Figure 5.3.2.2). When the user taps the marker again, an info dialog will pop out and show all the bus routes that will pass by that stop, and taps the info dialog will start navigation process (as Figure 5.3.2.3).

5.3.3 Search Page

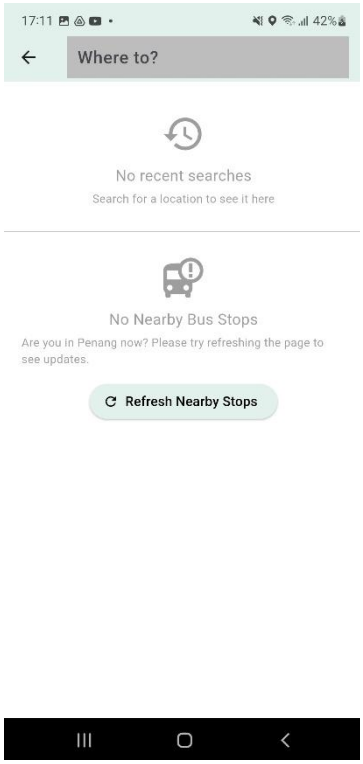


Figure 5.3.3.1 Search Page

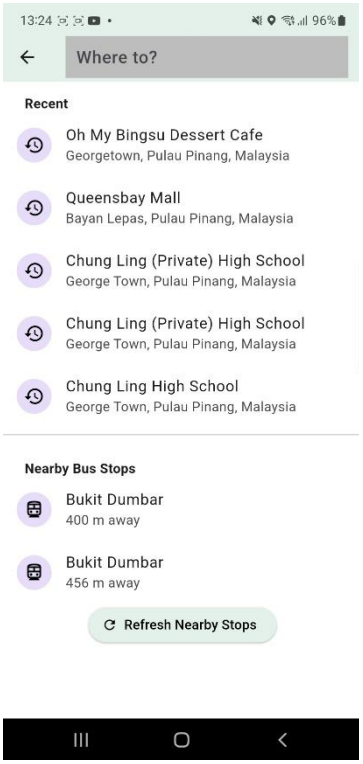


Figure 5.3.3.2 Search Page with history and nearby stop

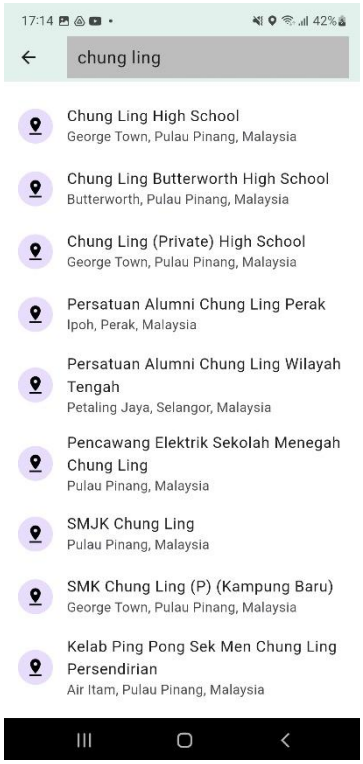


Figure 5.3.3.3 Search Destination

When user taps on the search marker in the top left corner of the homepage, search page will be displayed (as Figure 5.3.3.1). This page allows users to search their destination, view their history (if exist) and view nearby bus stops. The system will check if any bus stops are near the user within 500 meters, otherwise an error message will display to indicate there is no nearby stop, and a button that allows user to refresh, where the system will recalculate the nearby bus stop and display it if exist. Figure 5.3.3.2 is how the search page looks when the user has search histories in Firestore and there are bus stops near the user. Lastly, Figure 5.3.3.3 shows the destination selection returned by the Photon geocoding service when user types in search box.

5.3.4 Route Suggestion

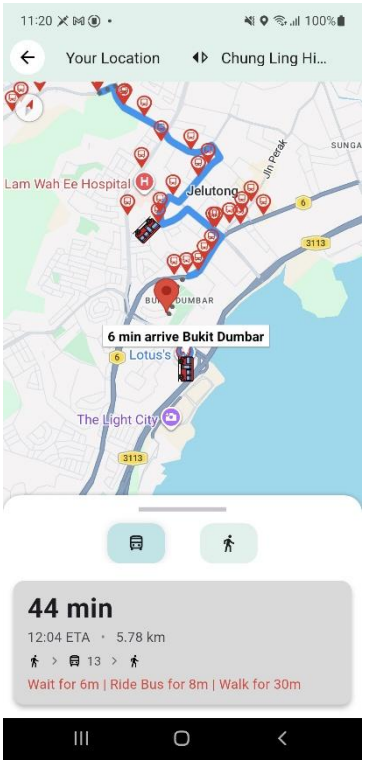


Figure 5.3.4.1 Bus Transit (real-time)

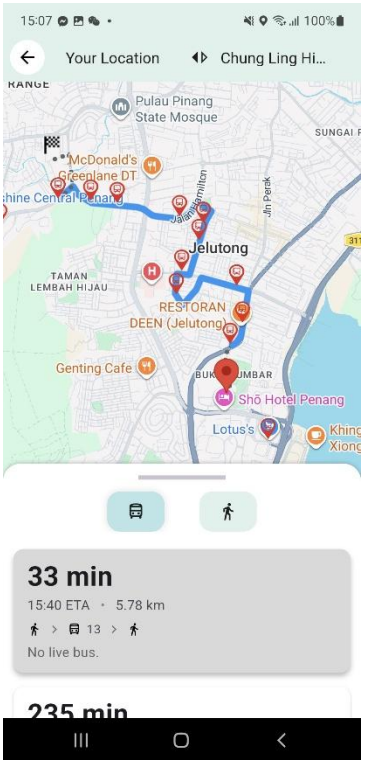


Figure 5.3.4.2 Bus Transit (static)

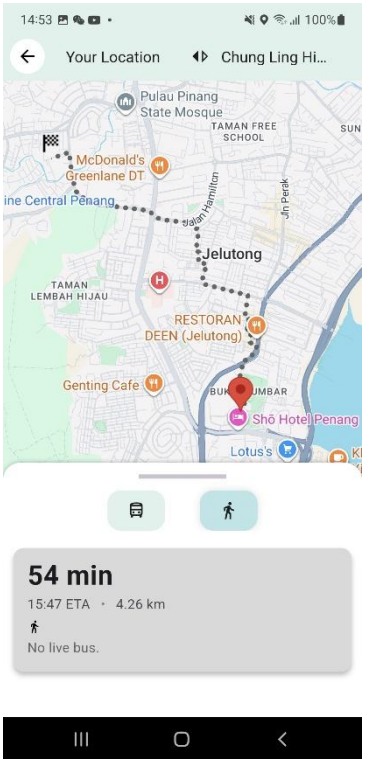


Figure 5.3.4.3 Walk Transit

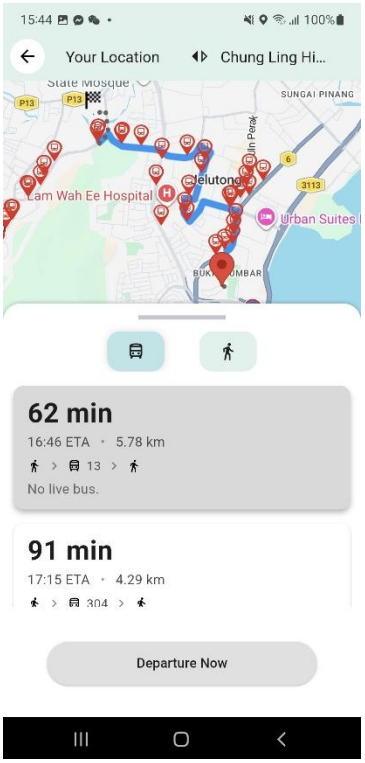


Figure 5.3.4.4 Departure Button

Once user selects an option in search page or selects a history record from homepage or search page, they will be guided to route suggestions page to choose one of the route suggestions returned by the OTP route server (as Figure 5.3.4.1 to Figure 5.3.4.3). There are 2 types of transit, one is bus transit and the other is walk transit. The bus transit will display the live bus marker on the map along with all the stops and polyline for that bus route suggested, but the real-time information will only be displayed when a live bus passes by the stop suggested by the navigation route. Figure 5.3.4.4 shows the departure button, user may pull up the sheet to tap the button and start the navigation.

5.3.5 Navigation

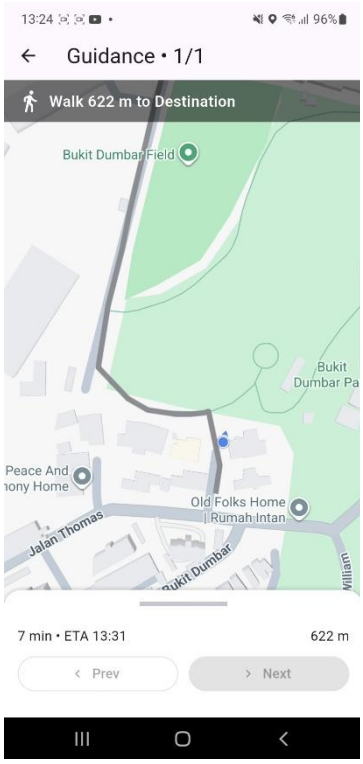


Figure 5.3.5.1 Navigation - Walk

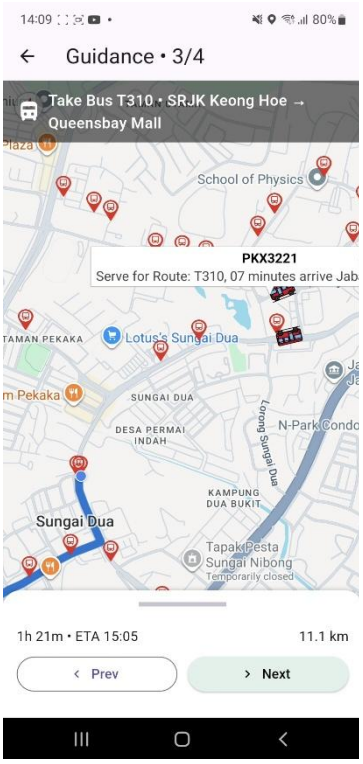


Figure 5.3.5.2 Navigation - Bus

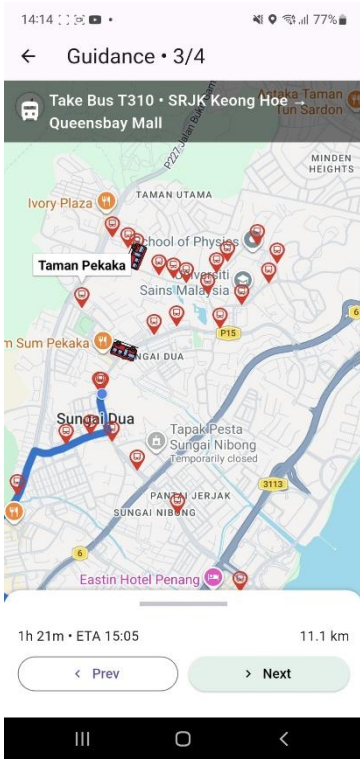


Figure 5.3.5.3 Navigation – Bus

Figure 5.3.5.1 and Figure 5.3.5.2 illustrate the navigation pages in “walk” mode and “bus” mode respectively. There are two buttons allow user to proceed to next step or back to previous step if the navigation route contains multiple trips (as figure 5.3.5.2). Otherwise, the buttons are not available for use (as figure 5.3.5.1). If the current leg is for a bus trip, the system will display all the live buses related to the bus route and its

polyline and stops on the map. The user can interact with the bus marker (as figure 5.3.5.2) or the stop marker (as figure 5.3.5.3) to check the information respectively.

5.3.6 Linepage

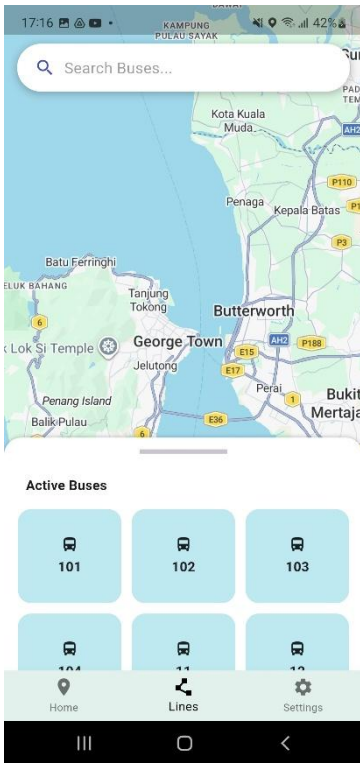


Figure 5.3.6.1 Line Page

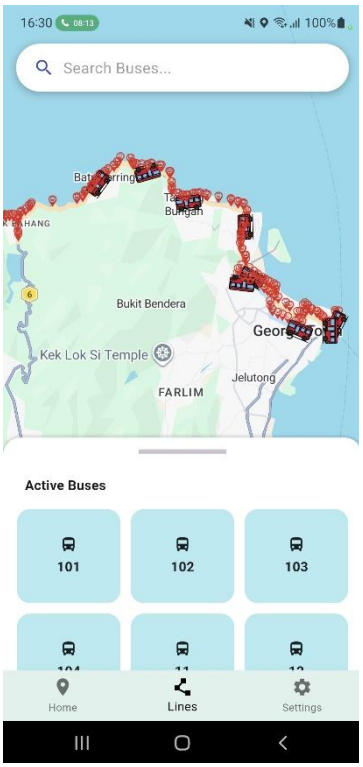


Figure 5.3.6.2 Route 101 Buses

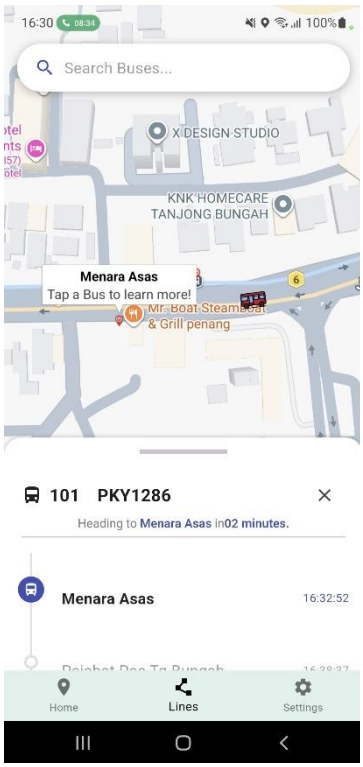


Figure 5.3.6.3 Route 101 Bus with interaction

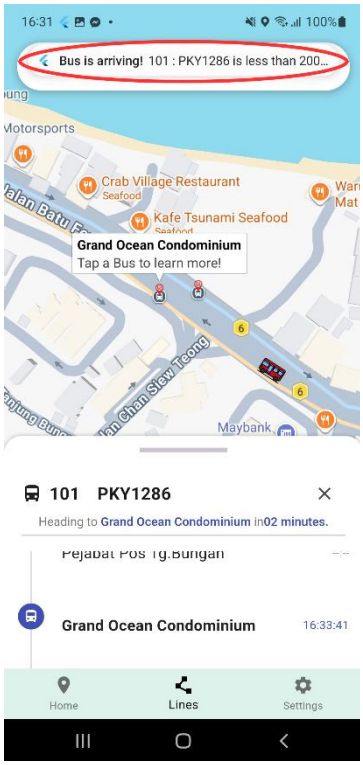


Figure 5.3.6.4 Notification

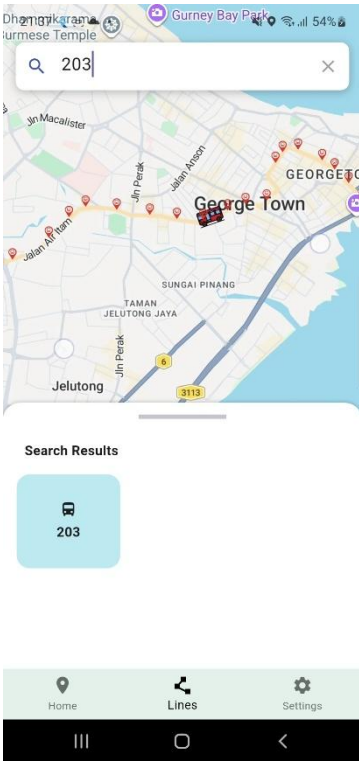


Figure 5.3.6.5 Search Bus Route

This section describes the line page in the application where the user can be redirected from the bottom navigation bar. On this page, the system will automatically fetch all live buses from GTFS-RT and categorize them based on route name (as shown in Figure 5.3.6.1). User can select any bus route to see all running buses on map, and interact with them like checking the next stop, getting the ETA info, and getting a notification when the selected bus is near the selected stop. If the user wants to know specific bus route, they can type the route name in the search box and the system will display the search result on the sheet (as shown in Figure 5.3.6.5).

5.3.7 Settings

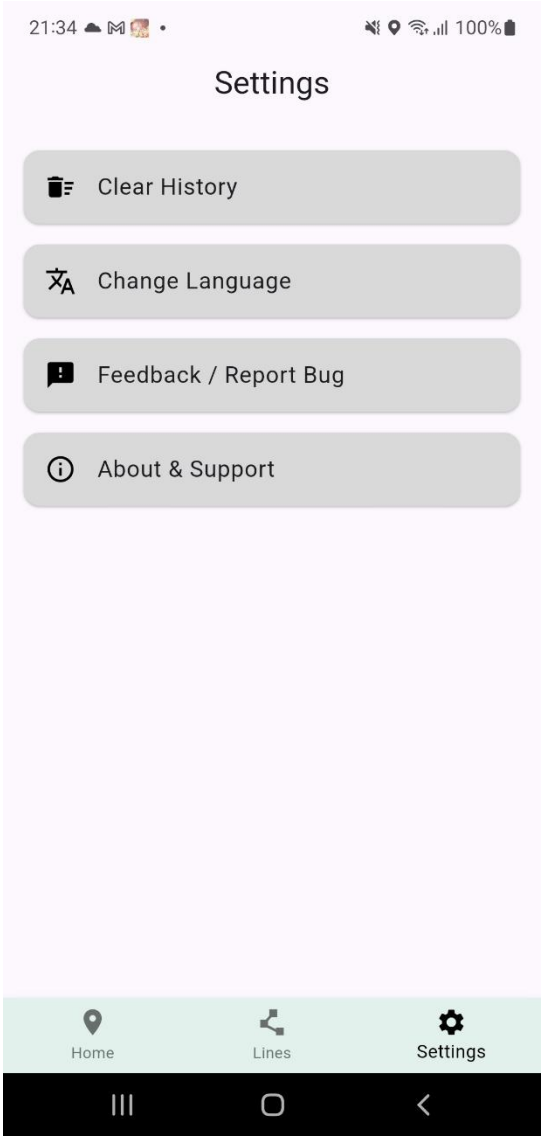


Figure 5.3.7.1 Settings Page

On the setting page, user is presented with 4 selections: clear history, change language, feedback and report bug, and lastly a description about the application.

5.3.7.1 Clear History

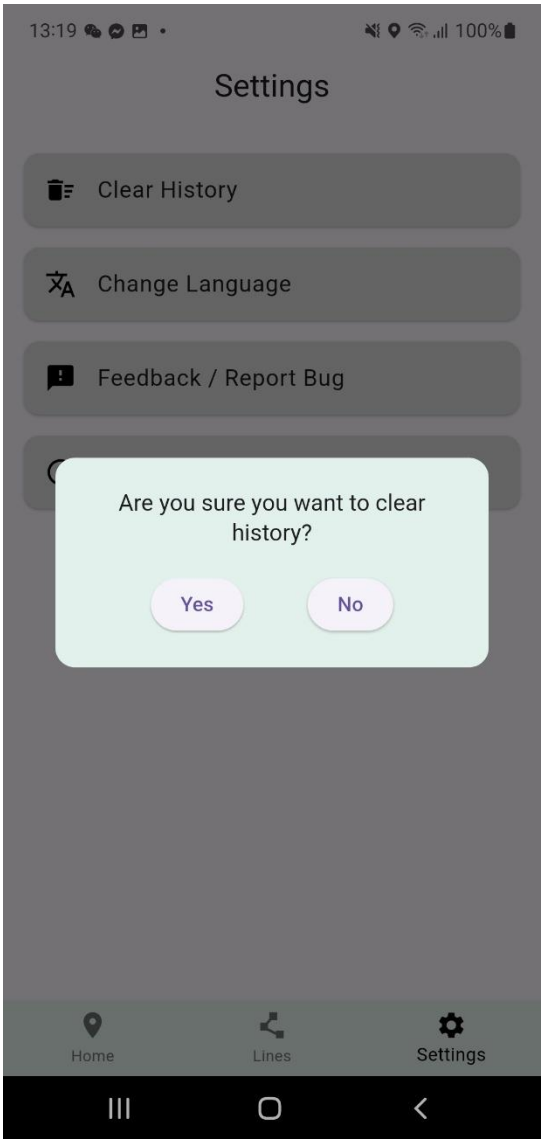


Figure 5.3.7.2 Settings Page – Clear History

When user taps on the first button “clear history”, the system will display a confirmation box to seek user acknowledgement on clearing history. If yes is chosen, the system will send request to Firebase to delete the search history using the user ID. If no is chosen, the box is closed.

5.3.7.2 Change Language

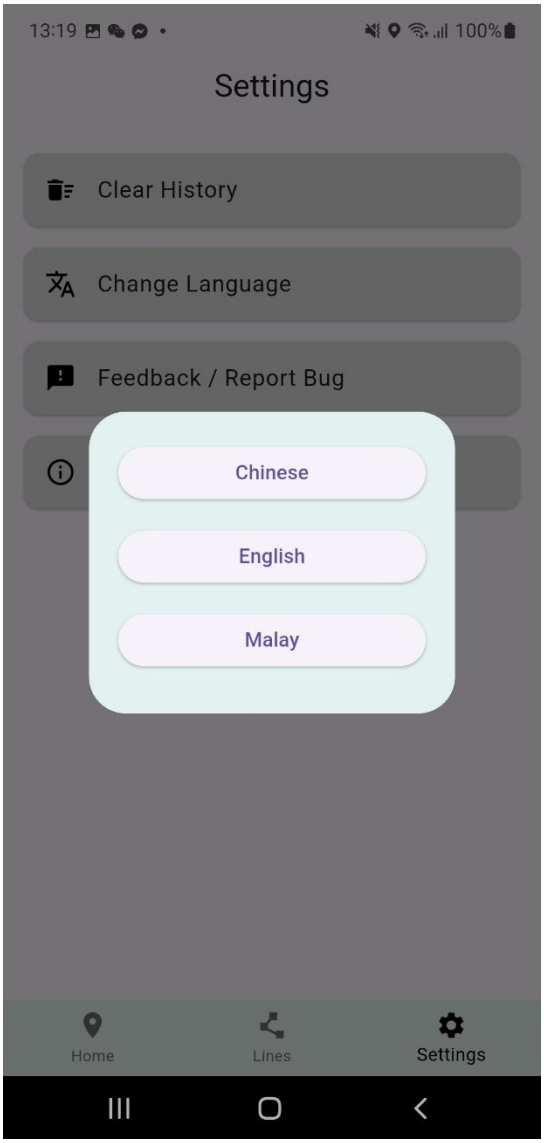


Figure 5.3.7.3 Settings Page – Change Language

When user taps on the second button “change language”, the system will display three language selections and allow user to choose any language provided. Once a language is selected, all texts in all pages will immediately change to the selected language. A detailed system UI with different languages can be found in Appendix C.

5.3.7.3 Feedback and Report Bug

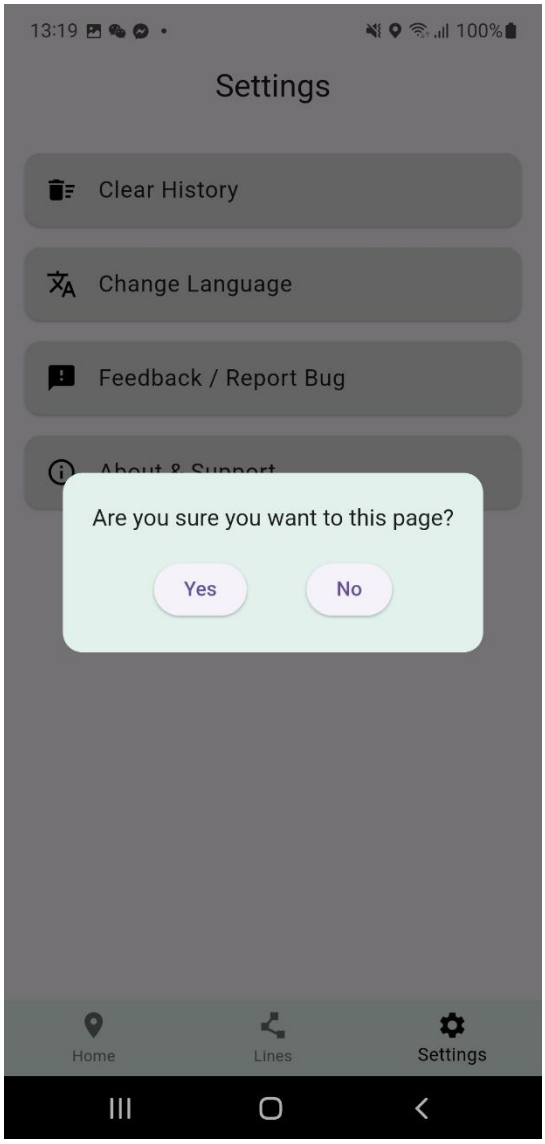


Figure 5.3.7.4 Settings Page – Feedback and Report Bug

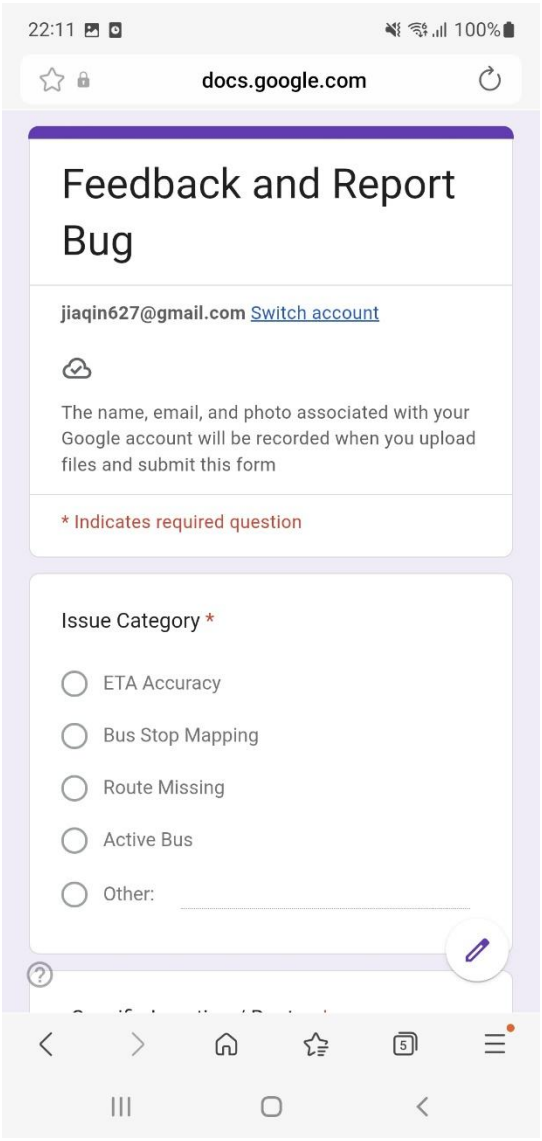


Figure 5.3.7.5 Google Form

This page is presented when the third button “feedback/report bug” is tapped by user. The system will then display a confirmation box to seek user acknowledgement on leaving the application. If the selection is yes, the system will launch a URL and redirect the user to that URL outside the application. A Google Form was presented after user signed in with their Google account. This page allows users to report any bug or provide feedback on any issues found or if there are any improvements to make the application better.

5.3.7.4 About and Support

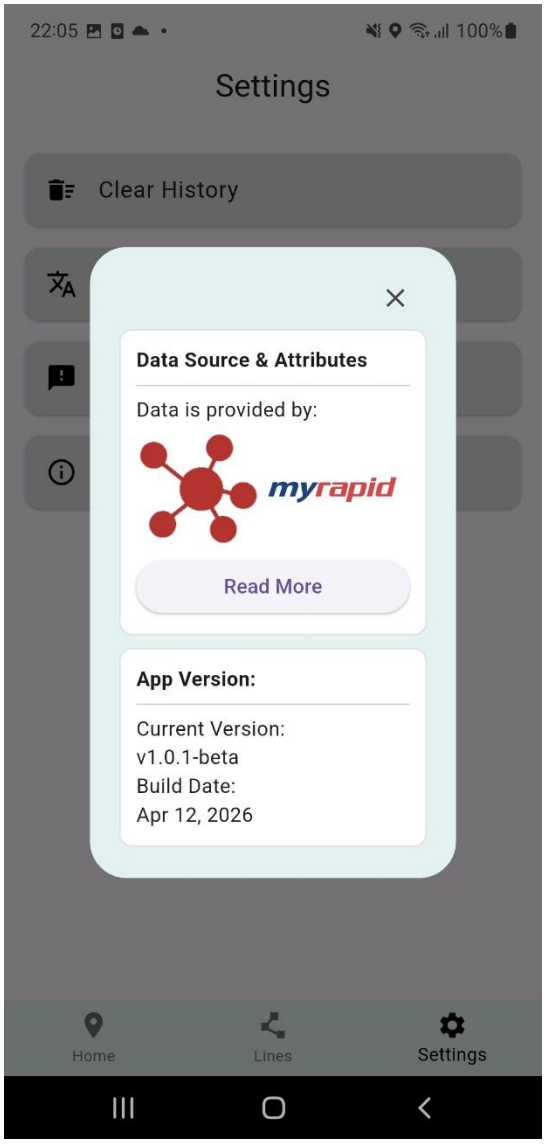


Figure 5.3.7.6 Settings Page – About and Support

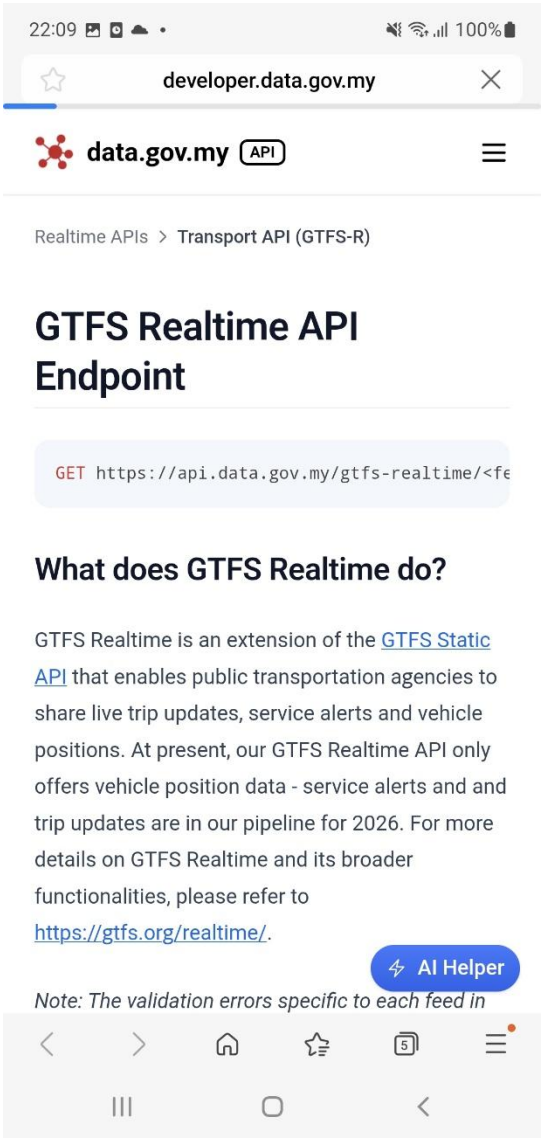


Figure 5.3.7.7 GTFS-RT Description

This is the last selection of the button on the settings page, which mainly covers the source and the version of application. A button “Read More” is provided at the data source and attributes section. If the user wants to know more about the main data utilized in the application, tapping the button will redirect them to the data portal.

5.4 Implementation Issues and Challenges

Upon development of application, several challenges were identified during the implementation phase.

An issue we have discovered is insufficient requests for the GTFS-RT service. This is because GTFS-RT only allows 4 requests per minute, which is insufficient if we call the service individually during system refreshes and call GTFS-RT for more than 2 modules every 30 seconds. Therefore, we create a class that acts as a data repository, which will keep fetching the real-time data and wrap it in a variable. Other features that will need to leverage this resource will call a get function to get the variable. At the end, only one object calls the service and provides it to other classes.

When it comes to collecting the real-time data for ETA prediction, there is an implementation challenge. Since we have to manually find out what the next stop is for each live bus, and we know that the data returned by GTFS-RT is usually with a speed of 0, we are not able to get ETA by using rule-based calculation. Thus, we use the nearest segments to find which road segment the bus is in, in order to know what the next stop of the segment is (destination = to stop). However, there are possible that the system detects the wrong nearest segment and returns a wrong result, especially when the GPS of the bus is unstable, which then affects the result of next stop and ETA. Thus, we added a shape ID to ensure that each live bus has the same shape ID as the nearest road segment detected by the system.

5.5 Concluding Remarks

In conclusion, the system implementation phase shows how the actual implementation was conducted according to the system methodology and system design. The development process began with setting up all necessary external services and databases, followed by implementing the frontend of application, and lastly implementing the ETA logic and integrating it with frontend. As a result, all the services can be integrated and function as intended. By leveraging Flutter for the frontend and several services within the system boundary, including Firebase, Oracle Cloud and PostgreSQL for the persistent backend data ingestion, the core infrastructure required to support real-time transit tracking has been successfully established.

Overall, the system not only provides the real-time ETA and bus tracking, but also allows user to plan their transit trips in Penang with route suggestions. While the system is now operational, its performance and the accuracy of ETA provided have yet to be quantified. These aspects will be analyzed in more detail in the following chapter.

CHAPTER 6 System Evaluation and Discussion

This chapter discusses the “JomNaik” system performance evaluation to ensure all the system features meet the functional requirements and work as expected. Then, we analyse the project challenges faced during testing and the objectives evaluation.

6.1 System Testing and Performance Metrics

In this section, we will define the method and the metrics used to perform testing on our application and functions. The testing will cover several aspects, including system functionality and ETA accuracy.

1. Statistical Evaluation (MAE/RMSE)

   Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are the performance metrics for regression task to measure error magnitude. ETA will be the primary target for testing. The data will be the historical dataset in the PostgreSQL Database table “historical_subsegments”; a specific day will be selected as the main test subject to examine how the predicted statistical results differ from the actual results. This testing will be conducted several times to ensure the consistency of the results.

2. Performance Testing (Response Time)

   This metric will focus on the response latency of 2 services: the ETA API and the Route Service API. Measuring the latency ensures that the process of queries when users interact with the system does not compromise user experience.

3. Functional Testing (Black Box Approach)

   A black box testing will be conducted as the system functional testing to validate if the features perform according to the specified requirements. We will first determine the function that features are expected to perform based on the use case, then compare it with actual performance. The test case will cover all interactions that the user can perform with the application. The result of a test case will be classified as “Pass” if feature performed as expected, or “Fail” if the system fails to perform some feature and causes errors.

4. Field Test (Real-World Monitoring)

   This testing serves as an evaluation method for reviewing how the "JomNaik" system performs in real-world scenarios. The main purpose is to prove that the system can be used in real traffic conditions in Penang, and test if the ETA and route service features the system provides are working properly.

6.2 Testing Setup and Result

6.2.1 Statistical Evaluation

We extract a historical data CSV file from the database table, the total number of data is 2,096,508, and this data will be used to do the evaluation. Several days are selected as the test date to ensure the results are consistent. The train set is data excluding the target date, and the test set consists of the data on target date. Before calculation, we merge all the train set data based on subsegment_id and time bucket, then get the mean of travel time as the predicted estimated time ($\hat{y}$). Then, use the predicted time to compare with the actual time (y) in the test set. Below are the equations of MAE and RMSE:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

		Test Date	Mean Absolute Error (MAE)	Root Mean Squared Error (RMSE)	Biggest Absolute Error
Weekday	1	2026-03-30	6.46 second	14.36 second	284.54 second
	2	2026-04-08	6.57 second	14.86 second	328.60 second
	3	2026-04-17	5.93 second	13.49 second	290.92 second
Week	4	2026-04-04	6.18 second	14.08 second	318.72 second

	5	2026-04-12	6.00 second	13.76 second	325.58 second
	6	2026-04-13	6.22 second	14.20 second	325.35 second

Table 6.2.1.1 Statistical Evaluation

Through the statistical evaluation on ETA, the system achieves a consistently low MAE across all testing dates, regardless of weekdays or weekends. This means that the ETA logic is effective in both peak and off-peak scenarios.

In terms of RMSE, the values are almost around 14 seconds, and all are higher than MAE scores, indicating that the data contains outliers with large values (as the Biggest Absolute Error). This is expected for real-world transportation systems, as we cannot predict reasons that cause buses to stay in the same subsegment for a long time, such as traffic accidents or the driver waiting for passengers at bus stop.

6.2.2 Performance Testing

Postman Runner was used to test all APIs implemented in this project to get the required information to enable the application to function properly. The purpose is to understand how the system backend performs, especially when providing ETA prediction to front-end, and ensure the integration between the application and the services hosted on Cloud Instance can deliver information within an acceptable threshold for real-time usage. The target of the API being tested includes the API for **retrieving single record of travel time** within a segment, the API for **retrieving multiple records of travel time** on a route with multiple segments, and the **route suggestions** service.

Testing Configuration:

- Iterations: 100
- Delay: 10ms

No	API	Method	Duration	Average Response Time
1	/eta?lat={lat}&lng={lng} &shape_id={shape_id}	GET	15s 273ms	37ms
2	/eta?shape_id={shape_id}	GET	4m 31s	2597ms
3	/otp/gtfs/v1	POST	22s790ms	111ms

Table 6.2.2.1 API Testing Result

Table 6.2.2.1 shows the testing results of APIs. All APIs were successfully tested without any server errors or timeouts, which suggests that the services deployed on Cloud Instance have high availability. In detail, the average response time for fetching one record API and the route engine API is surprisingly good, which indicates that the user will get the ETA result and route suggestion immediately after they request it. Regarding the second API, which retrieves all related travel times within a bus route, the performance was expected since the system will get about 70 – 100 records in a single request.

Overall, this testing validates that the backend architecture was able to interact reliably in terms of communication. All APIs are functioning exactly as designed under normal conditions, and none of the packets were dropped.

6.2.3 Functional Testing

In this section, a black box testing will be performed to ensure the whole system operates as intended. Each test case consists of Test Case ID, Description, Test Input, Expected Result, Actual Result, and Status.

Below are the 6 tables used to categorize the test results for the functions of Homepage, Search, Route Service, Navigation, Linepage, and Settings.

Test Case ID	Test Description	Test Input	Expected Result	Actual Result	Status (Pass/ Fail)
H_01	Verify if user can launch the application	Launch the app	An image is shown in the center of screen, and it	An image is shown in the center of screen, and it	Pass

			will become an animation after a period of time. Then, Homepage is shown	becomes an animation after a period of time. Then, Homepage is shown	
H_02	Verify that if only related polylines and stops shown after tapping a stop marker.	Tap a stop marker	One or more polylines are drawn, and only the related stop markers are drawn	One or more polylines are drawn, and only the related stop markers are drawn	Pass
H_03	Verify that if only related info is shown	Tap a stop marker, and tap any marker on the map	An info dialog about the stop pops up.	An info dialog about the stop pops up.	Pass
H_04	Verify that if user can start route to the destination stop	Tap the info dialog	Route Suggestion page shown	Route Suggestion page shown	Pass
H_05	Verify the system is able to clear all filtered polylines and stop markers	Tap a stop marker, and tap other blank space	Polylines are removed and all stop markers are shown	Polylines are removed and all stop markers are shown	Pass
H_06	Verify that the page returns to the user's	Tap the button in the top right corner	The map scrolls to	The map scrolls to	Pass

	location on map		user's current location	user's current location	
H_07	Verify that user can start route to the destination when search history is not empty	Tap one search history on the sheet	Route Suggestion page shown	Route Suggestion page shown	Pass
H_08	Verify that system displays an error message properly when search history is empty	None	The system displays "No recent searches" on the sheet	The system displays "No recent searches" on the sheet	Pass
H_09	Verify that user can be redirected to another page	Tap "Lines" option in bottom navigation bar	Lines Page shown	Lines Page shown	Pass
H_10	Verify that user can be redirected to another page	Tap "Settings" option in bottom navigation bar	Settings Page shown	Settings Page shown	Pass
H_11	Verify that user can be redirected to another page	Tap search button in the top left corner	Search Page shown	Search Page shown	Pass

Table 6.2.3.1 Testing for Homepage Function

Test Case ID	Test Description	Test Input	Expected Result	Actual Result	Status (Pass/ Fail)
SP_01	Verify that system display error message properly when search history is empty	None	The system displays “No recent searches” on the section	The system displays “No recent searches” on the section	Pass
SP_02	Verify that user can start route to the destination when search history is not empty	Select one search history on the section	Route Suggestion page shown	Route Suggestion page shown	Pass
SP_03	Verify that system displays error message properly when nearby bus list is empty	None	The system displays “No nearby bus” on the sheet	The system displays “No nearby bus” on the sheet	Pass
SP_04	Verify that user can start route to the destination when nearby	Select one nearby bus on the section	Route Suggestion page shown	Route Suggestion page shown	Pass

	bus list is not empty				
SP_05	Verify that user able to get result for searching	Type “Queensbay” in the search box	A list of results about “Queensbay” is displayed on the screen	A list of results displayed on the screen	Pass
SP_06	Verify that user can start route to the destination when result list is not empty	Select one destination	Route Suggestion page shown	Route Suggestion page shown	Pass

Table 6.2.3.2 Testing for Search Page Function

Test Case ID	**Test Description**	**Test Input**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
RSP_01	Verify that user can choose walk mode	Tap Walk Mode Button on top bar of the sheet	Route Suggestion cards changed	Route Suggestion cards changed	Pass
RSP_02	Verify that user can choose bus mode	Tap Bus Mode Button on top bar of the sheet	Route Suggestion cards changed	Route Suggestion cards changed	Pass
RSP_03	Verify that user can view different route	Tap other route suggestion card	Related polylines and stop markers being drawn	Related polylines and stop markers being drawn	Pass

	suggestion for bus mode			if live bus exist	
RSP_04	Verify that user can interact with bus marker	Tap bus marker if exist	Bus ETA info displayed	Bus ETA info displayed	Pass
RSP_05	Verify that user can interact with stop marker	Tap stop marker	Stop name displayed	Stop name displayed	Pass
RSP_06	Verify that the source and destination exchange	Tap swipe button on top of the screen	Source and destination exchange and the route suggestion replanned	Source and destination exchange and the route suggestion replanned	Pass
RSP_07	Verify that user can start navigation	Scroll the sheet to button and tap departure button	Navigation Page shown	Navigation Page shown	Pass

Table 6.2.3.3 Testing for Route Service Page Function

Test Case ID	Test Description	Test Input	Expected Result	Actual Result	Status (Pass/ Fail)
NP_01	Verify that user cannot choose to go forward and backward if they only have one trip	Walk Route selected, Tap Next Button	Nothing happens	Nothing happens	Pass
NP_02	Verify that user can choose to go forward and backward if they have multiple trips	Bus Route selected, next trip is bus mode. Tap Next Button	The polyline drawn according to the current trip	The polyline drawn according to the current trip	Pass
NP_03	Verify that user can choose to go forward and backward if they have multiple trips	Bus Route selected, next trip is walk mode. Tap Next Button	The polyline drawn according to the current trip	The polyline drawn according to the current trip	Pass
NP_04	Verify that user can interact with bus marker	Bus Route selected, current trip is bus mode. Tap live bus marker	License plate of bus, bus route name and ETA shown in an info dialog.	License plate of bus, bus route name and ETA shown in an info dialog.	Pass

NP_05	Verify that user can interact with stop marker	Bus Route selected, current trip is bus mode. Tap any stop marker	Stop name shown in an info dialog	Stop name shown in an info dialog	Pass
NP_06	Verify that system able to refresh the live bus status	Bus Route selected, current trip is bus mode. Wait for 1 minute	The location of live bus marker is refreshed, related info changed	The location of live bus marker is refreshed, related info changed	Pass
NP_07	Verify that user can terminate the navigation	Scroll up the sheet, tap terminate button	Homepage shown	Homepage shown	Pass

Table 6.2.3.4 Testing for Navigation Page Function

Test Case ID	**Test Description**	**Test Input**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
LP_01	Verify that user can choose to show any bus route	Tap bus route “101”	Live bus marker displayed on map, camera move to focus on all buses	Live bus marker displayed on map, camera move to focus on all buses	Pass
LP_02	Verify that user can search bus route	Type “20” in the search box	Bus routes “201”, “202”, “203”, “204” are displayed	Bus routes “201”, “202”, “203”, “204”	Pass

			on sheet if related buses are in operation	are displayed on sheet	
LP_03	Verify that user can search bus route	Type “203” in the search box	Bus route “203” is displayed on sheet if related buses are in operation	Bus route “203” is displayed on sheet	Pass
LP_04	Verify that user can choose to show any filtered bus route	Tap bus route “203”	Live bus marker displayed on map, camera move to focus on all buses	Live bus marker displayed on map, camera move to focus on all buses	Pass
LP_05	Verify that user can interact with the bus marker	Tap any bus marker on map	Sheet updated with bus info, bus stop timeline with ETA	Sheet updated with bus info, bus stop timeline with ETA	Pass
LP_06	Verify that system able to refresh the live bus status	Tap any bus marker on map, wait for 1 minute	Sheet updated with bus info, bus stop timeline with ETA, bus marker move to new location on map	Sheet updated with bus info, bus stop timeline with ETA, bus marker move to new location on map	Pass

LP_07	Verify that user can interact with stop marker	Tap any stop marker on the map	An info dialog appear with stop name	An info dialog appear with stop name	Pass
LP_08	Verify that user can get notification when bus is approaching	Tap any bus marker on map, tap stop marker that bus heading to, tap the info dialog	A notificaiton pop out when bus marker location is near to the selected stop marker	A notificaiton pop out when bus marker location is near to the selected stop marker	Pass

Table 6.2.3.5 Testing for Lines Page Function

Test Case ID	**Test Description**	**Test Input**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
SP_01	Verify that user can remove search history	Tap clear history button, choose yes	A bar pops out, indicating history was cleared	A bar pops out, indicating history was cleared	Pass
SP_02	Verify that user can cancel the remove search history process	Tap clear history button, choose no	Confirmation box is closed	Confirmation box is closed	Pass
SP_03	Verify that user can change the system language	Current language English Tap change language	The entire application language has changed to Chinese	The entire application language has changed to Chinese	Pass

		button, choose Chinese			
SP_04	Verify that user can change the system language	Current language Chinese Tap change language button, choose English	The entire application language has changed to English	The entire application language has changed to English	Pass
SP_05	Verify that user can change the system language	Current language English Tap change language button, choose Malay	The entire application language has changed to Malay	The entire application language has changed to Malay	Pass
SP_06	Verify that user can report a bug or give feedback	Tap the feedback / report bug button, choose yes	The system redirects the user to Google Form	The system redirects the user to Google Form	Pass
SP_07	Verify that user can cancel the process of reporting a bug or giving feedback	Tap the feedback / report bug button, choose no	Confirmation box is closed	Confirmation box is closed	Pass

SP_08	Verify that user can check the information about the application	Tap the about and support button	An information box appears with data source & attributes and app version	An information box appears with data source & attributes and app version	Pass
SP_08	Verify that user can learn more about the data source	Tap the about and support button, tap read more button	The system redirects the user to data portal	The system redirects the user to data portal	Pass

Table 6.2.3.6 Testing for Settings Page Function

After a series of careful tests, we can confirm that all features are technically working correctly. The system provides a reliable interface for real-time public transit information retrieval.

6.2.4 Field Test

A field test was conducted to evaluate the system's performance under actual environmental constraints to verify that the application's features were usable.

Testing Setup:

- Starting location: Block 95 Sea Breeze Tower, Penang.
- Destination: Queensbay Mall
- Device: Samsung Galaxy Note 10, connected to Maxis 5G Network
- Trip sequence: Walk -> Bus Route "301" -> Bus Route "T310" -> Walk
- Starting Time: 13 April 2026 13:24
- Estimated Time of Arrival: 15:05

Trip	Route	Start Point	End Point	ETA	Actual Time
Walk	-	Block 95 Sea Breeze Tower, Penang	Bukit Dumbar Stop	13:31	13:35
Bus	301	Bukit Dumbar Stop	SRJK Keong Hoe Stop	14:20 (19min for waiting, 30min for traveling)	14:08 (15min for waiting, 18min for traveling)
Bus	T310	SRJK Keong Hoe Stop	Queensbay Mall Stop	14:53 (12min for waiting, 21min for traveling)	14:27 (6min for waiting, 13min for traveling)
Walk	-	Queensbay Mall Stop	Queensbay Mall	15:05	14: 32

Table 6.2.4.1 Field Test Record

Table 6.2.3.1 shows the field test results using the manual logging method, where the data in the ETA column is obtained from the “JomNaik” system, and the Actual Time column is the time of arrival at the end point. Once the trip navigation starts, the ETA is static throughout the process. For the image prove of field test, please refer to Appendix D.

We look into the second row (bus mode trip), see that the ETA is about 12 minutes slower than the actual time, which is mostly caused by the overestimation of traveling time. The same goes for the next row of trips, where the ETA for travelling time is about 8 minutes slower than the actual travel time.

From this testing, several key points were identified. Firstly, human factors have a significant impact on ETA, especially during their walking trip. Although the walking time data was obtained from a third party, it has a certain margin of error compared to

the actual walking time, which the length of their walking time can indirectly affect how long they will wait for the bus or the different bus they will take. Regarding the overestimated arrival time, our model calculates ETA using both real-time and historical data. As noted in Chapter 6.2.1, the historical data contains about 300 seconds of travel time, which can lead to a calculated ETA that is longer than the actual.

In summary, the system functioned like a normal navigation system, smoothly guiding people from the starting point to the destination. The navigation guidance provided is detailed and clear, and it also displays the location of the live buses as markers on the map during navigation process, allowing users to prepare for boarding the bus. Thus, we can sum up that this system meets the objectives of this project implementation.

6.3 Project Challenges

Several issues were identified in this project. The biggest challenge is handling real-time data. First, during field testing, it was observed that bus drivers typically do not slow down when passing a bus stop, only stopping when someone is about to disembark, or someone raises their hand at the station to take the bus. This traveling speed affects the reliability of the ETA. Additionally, as mentioned earlier, the reliability of ETA is heavily influenced by human factors. A critical challenge in end-to-end navigation is the variability of the “last mile’ leg. The user is a free agent, any change in pace or direction significantly degrades the temporal reliability of the initial ETA. Moreover, GPS drifting can sometimes occur. In such cases, the nearest subsegment returned by the system might be incorrect. Fortunately, this issue only lasted for 30 seconds, and the system can tolerate this short period of error. Lastly, while 30-second refresh rates may seem sufficient overall, they cannot keep up with real-time events during navigation, as the data provides the bus’s location at that specific moment, not the streaming data. In a word, the most important significance of real-time data lies in its timeliness; once it has passed, it loses its meaning.

After that, this system significantly relies on the network connectivity, both external data (GTFS-RT, -Static) and internal data (ETA, Route suggestions) are obtained through API calling. This means that if the signal is weak, the “JomNaik” system will be unusable.

6.4 Objectives Evaluation

- **To collect and integrate data**

  This project managed to collect data from internal sources and external sources and utilized the data in mobile application, ETA service and route service, where the internal sources represent travel time that very crucial for ETA prediction, and external sources represent the data provided by public transport agency (MyRapid), whether static or real-time data. Through the data collection and processing for providing ETA services, a historical dataset was built and used as part of ETA information. Thus, we conclude that this project achieved the objective of data collection and integration.

- **To build an ETA prediction model**

  An ETA model was built in this project, which is based on road segmentation we derive from GTFS static data. This model mainly utilizes the travel time of each segment to calculate the total travel time as ETA info. The model starts the process from early morning to late night each day, and the daily data is archived to an archival table and used as historical data on subsequent days. In case of missing real-time ETA info and historical ETA info, the travel time for that road segment will use default value as a fallback value.

- **To develop a mobile application**

  This project successfully developed a mobile application for public transit navigation system in Penang. All the live buses are represented as marker in this navigation app, and an animation is implemented to visualize bus movements during updates. Moreover, the application is able to push notifications to users when they choose to receive notifications to remind them that a bus is about to arrive. Lastly, we confirm that the user is able to interact with the system, which is further proven by system testing. Therefore, this project achieved the objective of developing a mobile application.

6.5 Concluding Remark

This chapter presents comprehensive testing to confirm that the “JomNaik” public transit navigation system has been successfully implemented and tested according to its objectives. In detail, many testing procedures have been carried out in this chapter, such as statistical evaluation, performance testing for API, black box testing, and even field testing to verify that the system meets the minimum viable product standard.

CHAPTER 7 Conclusion and Recommendation

7.1 Conclusion

In conclusion, the aim is to develop an application for guiding users to use public transportation, providing a live bus tracker and estimated time of arrival (ETA) so that users can have an understanding of when and where the buses will arrive, and plan their trip with bus transit.

Upon completion of this project, a navigator application is delivered to user in Penang with ETA, real-time bus tracking, and navigation features. To achieve this requirement, we mainly utilize open data sources such as GTFS provided by data.gov.my and software like Flutter, Firebase, PostgreSQL and Oracle Cloud Instance。

Based on our implementation work done, we can conclude that this system is implemented according to the system methodology and design described in Chapters 3 and 4, and it does its best to provide ETA predictions to solve the problem of no real-time ETA updates on public transit. The “JomNaik” system acts as a platform that combines all features of a navigation tool to provide a trip planning assistant. Not only that, but no matter where users are, as long as they are connected to the Internet, they can get real-time information about active buses.

Through a series of tests on different perspectives, such as evaluation of data used to calculate ETA, simulating user actions by interacting with the application in controlled lab and real-world situations, we confirmed that all features work as expected, providing comprehensive bus information to the user in real-time updates.

7.2 Recommendation

Several methods can be used to improve the system's functionalities and user experience, bringing it closer to commercial use. Firstly, **offline support** can be introduced in this system. As we know from testing, this system cannot provide service without a network connection, meaning users in areas with poor network coverage will have difficulty receiving real-time information. Adding offline caching can effectively solve this problem. For example, replacing GTFS static data with direct storage in the user's mobile phone's memory may reduce the need for retrieving through the network. Additionally, offline maps can be added, allowing users to know their current location and find the nearest bus stop even without network connections.

In addition, to improve user experience, **user accounts along with social feedback** features can enhance our system functionality. This feature allows users to report the status of running buses or get to know the status of buses in real time. For example, if a user takes a bus and finds the bus overcrowded, they can immediately report the bus as very crowded in the system. This way, other users can check the status while waiting for the bus, avoiding wasted time due to being unable to board.

During field testing, we noticed that not all bus stops had a bus sign, and some stops lacked complete information about which bus routes would pick up passengers at those stops. We believe that users would be confused when faced with discrepancies between the system and reality, unsure if they had arrived at the correct location. Therefore, **providing photos of each stop** may allow the users to understand the location of the bus stop in reality from the system would avoid any unnecessary trouble when taking the bus.

Finally, we can also add **voice prompts** during the turn-by-turn navigation process to remind users when to board and alight. While user is waiting at a bus stop or traveling by bus, the system can manage all available buses based on the required bus route. When a bus is about to arrive at the stop where user needs to take the bus, or the destination where user should get off, it can issue a voice reminder to alert user and prevent them from missing it.

REFERENCES

[1] dxmy, “History of Public Transport in Malaysia,” Transport Malaysia, Jun. 30, 2023. https://www.transportmalaysia.com/history-of-public-transport-in-malaysia/

[2] Admin, “The Evolution of Malaysia’s Public Transport System: From Traditional Modes to Modern Networks - Ghoomo Global Blog,” Ghoomo Global Blog, Feb. 23, 2025. https://ghoomoglobal.com/blog/the-evolution-of-malaysias-public-transport-system-from-traditional-modes-to-modern-networks/

[3] G. of Malaysia, “Public Transportation | data.gov.my,” data.gov.my. https://data.gov.my/dashboard/public-transportation

[4] “Prasarana Malaysia Berhad Official Corporate Website,” Prasarana Malaysia Berhad. https://www.prasarana.com.my

[5] “APAD - Agensi Pengangkutan Awam Darat,” Bas.my, 2025. https://bas.my/pengenalan.php (accessed Sep. 11, 2025).

[6] “About Us,” Causeway Link, May 15, 2019. https://www.causewaylink.com.my/company-information/about-us/

[7] Department of Statistics Malaysia, “The Population of Malaysia,” open.dosm.gov.my, Oct. 17, 2024. https://open.dosm.gov.my/dashboard/population

[8] Department of Statistics Malaysia, 2025. [2060] Population Projection. OpenDOSM. Available at: https://open.dosm.gov.my/publications/population_projection_2060 [Accessed 27 August 2025].

[9] Ministry of Transport Malaysia, “NATIONAL TRANSPORT POLICY 2019-2030,” 2019. Available: https://www.pmo.gov.my/wp-content/uploads/2019/10/National-Transport-Policy-2019_2030EN.pdf

[10] T. Dey, “GPS Applications in Transportation System,” Academia.edu, Jul. 05, 2013. [Online] Available: https://www.academia.edu/3862629/GPS_Applications_in_Transportation_System

[11] Malaysia Tourism Statistics, “Malaysia Tourism Statistics,” Tourism.gov.my, 2025. https://data.tourism.gov.my/

[12] Hayder Waleed and M. Marizwan, “The Relationship between Population Growth and Vehicle Growth in Malaysia: A Review,” Feb. 08, 2024. https://www.researchgate.net/publication/378108716_The_Relationship_between_Population_Growth_and_Vehicle_Growth_in_Malaysia_A_Review

[13] M. Poovaneswary, “Assessing public perception and satisfaction within public transportation services in Penang - UUM Electronic Theses and Dissertation [eTheses],” Uum.edu.my, 2024, doi: https://etd.uum.edu.my/11353/1/depositpermission.pdf.

[14] “Google Maps Platform,” Google Maps Platform, 2025. https://mapsplatform.google.com/?_gl=1 (accessed Apr. 18, 2025)

[15] W. Todd, “How does Google Maps routing algorithm work?,” Oct. 07, 2024. [Online] Available: https://www.ncesc.com/geographic-pedia/how-does-google-maps-routing-algorithm-work/

[16] N. H. Prickett, “The Graph Neural Network Behind Your ETA,” The Next Platform, Aug. 31, 2021. [Online] Available: https://www.nextplatform.com/2021/08/31/the-graph-neural-networks-behind-your-eta/

[17] A. Derrow-Pinion et al., “ETA Prediction with Graph Neural Networks in Google Maps,” Proceedings of the 30th ACM International Conference on Information & Knowledge Management, Oct. 2021, doi: https://doi.org/10.1145/3459637.3481916.

[18] “Driving Directions, Traffic Reports & Carpool Rideshares by Waze,” www.waze.com. https://www.waze.com/company (accessed Apr 19,2025)

[19] “Traffic data,” Waze Discuss, Mar. 12, 2025. https://www.waze.com/discuss/t/traffic-data/378884 (accessed Apr 19, 2025).

[20] “How navigation app Waze works behind the scenes,” Waze Belgium. https://www.wazebelgium.be/how-navigation-app-waze-works/ (accessed Apr 19, 2025).

[21] R. Sasson, “ETA prediction at Waze using Deep Learning Models,” Roy Sasson Home Page, Apr. 02, 2023. [Online] Available: https://www.datascienceeconomics.com/post/eta-prediction-at-waze-using-deep-learning-models

[22] “MyRapid - Bus Kiosk,” myrapidbus.prasarana.com.my. https://myrapidbus.prasarana.com.my/kiosk (accessed Apr 19, 2025).

[23] “Pulse Mobile App,” MyRapid. https://myrapid.com.my/pulse/mobile-app/ (accessed Apr 19, 2025)

[24] “The World’s Most Popular Urban Mobility App,” Moovit. https://moovitapp.com/ (accessed Apr 19, 2025)

[25] “Moovit: Bus & Public Transit,” Moovit. https://play.google.com/store/apps/details?id=com.tranzmate&hl=en (accessed Apr 19, 2025)

[26] “Arrival Time Accuracy Guide,” Help Center, Aug. 28, 2024. https://support.moovitapp.com/hc/en-us/articles/12476396200978-Arrival-Time-Accuracy-Guide (accessed Apr 19, 2025).

[27] Google Maps, “Google Maps,” Google Maps, 2019. https://www.google.com/maps/@4.3250325 (accessed Apr 19, 2025).

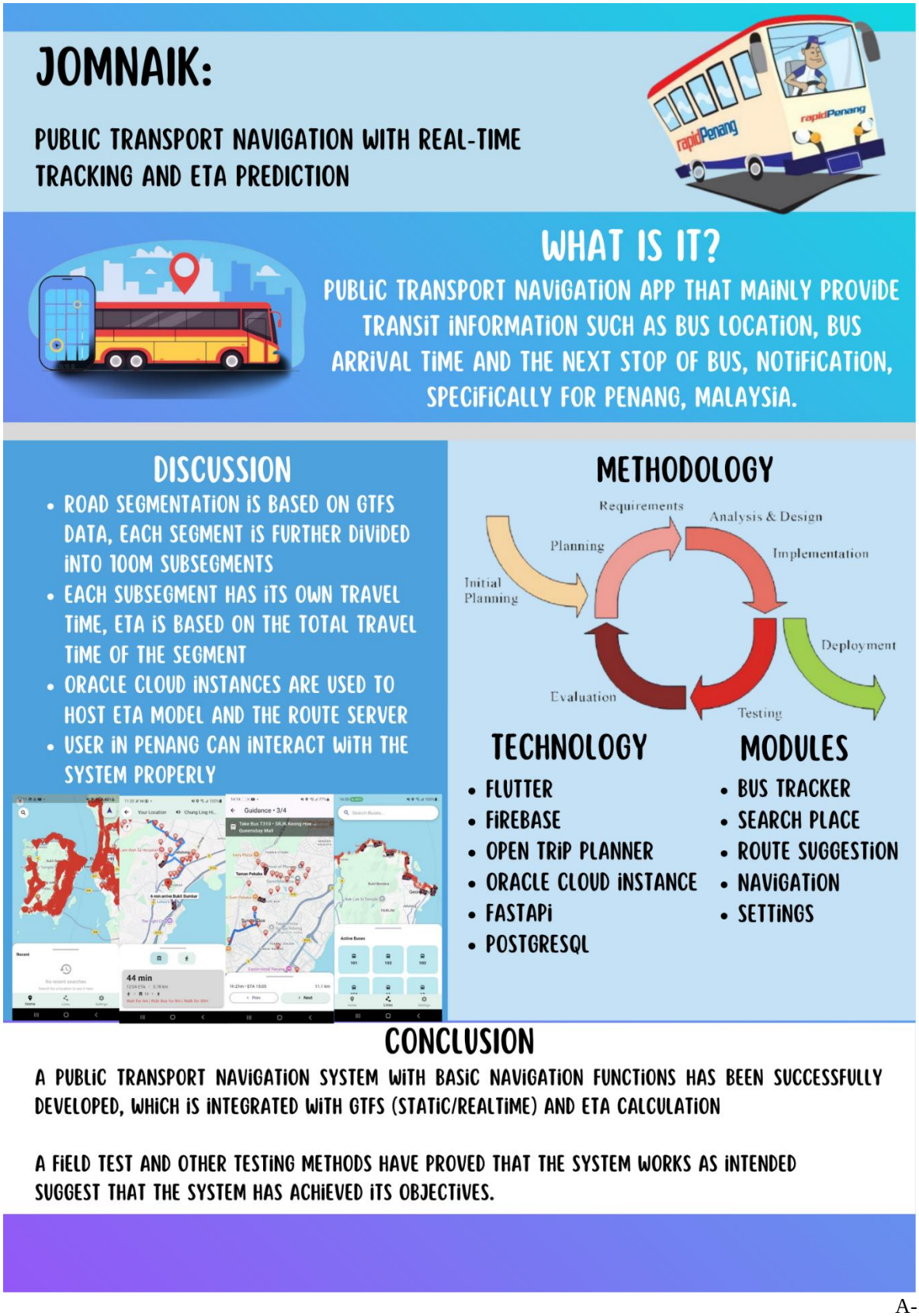
[28] “GTFS Realtime - Malaysia’s Official Open API,” Data.gov.my, 2024. https://developer.data.gov.my/realtime-api/gtfs-realtime#understanding-the-data (accessed Apr 19, 2025).

[29] GeeksforGeeks, “Software Engineering | Incremental process model - GeeksforGeeks,” GeeksforGeeks, May 28, 2018. [Online] Available: https://www.geeksforgeeks.org/software-engineering-incremental-process-model/

[30] “George Town traffic report | TomTom Traffic Index,” TomTom Traffic Index, 2026. https://www.tomtom.com/traffic-index/city/george-town

APPENDICES

Appendix A: Poster



Appendix B: View SQL Query Definition

- **Segment_eta_view**

```
CREATE VIEW segment_eta_view AS

WITH buckets AS (

SELECT DISTINCT rt_subsegments.time_bucket

FROM rt_subsegments

UNION

SELECT DISTINCT historical_subsegments.time_bucket

FROM historical_subsegments

),

rt_agg AS (

SELECT rt_subsegments. subsegment_id,

rt_subsegments. time_bucket,

sum(rt_subsegments.avg_travel_time_sec) AS rt_sum,

count(rt_subsegments.avg_travel_time_sec) AS rt_cnt

FROM rt_subsegments

GROUP BY rt_subsegments.subsegment_id, rt_subsegments.time_bucket

),

h_agg AS (

SELECT historical_subsegments.subsegment_id,

historical_subsegments.time_bucket,

AVG(historical_subsegments.avg_travel_time_sec) AS h_sum,

count(DISTINCT record_date) AS h_cnt

FROM historical_subsegments
```

```
GROUP BY historical_subsegments. subsegment_id,
historical_subsegments.time_bucket
),
base AS (
SELECT s.segment_id,
s.shape_id,
b.time_bucket,
count(DISTINCT s.subsegment_id) AS total_count,
sum(rt.rt_cnt) AS realtime_count,
sum(rt.rt_sum) AS realtime_sum,
sum(
CASE
WHEN rt.rt_sum IS NOT NULL THEN rt.rt_sum
WHEN h.h_sum IS NOT NULL THEN h.h_sum
ELSE 46::double precision
END) AS filled_total_sum,
sum(
CASE
WHEN rt.rt_sum IS NOT NULL THEN s.distance_m
ELSE NULL:: double precision
END) AS realtime_distance,
sum(s.distance_m) AS total_distance,
sum(h.h_sum) AS raw_historical_sum,
sum(h.h_cnt) AS historical_count,
count(DISTINCT s.subsegment_id) * 46 AS default_sum
```

```
FROM subsegments s

CROSS JOIN buckets b

LEFT JOIN rt_agg rt ON s. subsegment_id = rt. subsegment_id AND b.
time_bucket = rt.time_bucket

LEFT JOIN h_agg h ON s.subsegment_id = h. subsegment_id AND b. time_bucket
= h.time_bucket

GROUP BY s.segment_id, b.time_bucket, s. shape_id

),

processed_base AS (

SELECT base.segment_id,

base. shape_id,

base. time_bucket,

base. total_count,

base.realtime_count,

base.realtime_sum,

base. filled_total_sum,

base.realtime_distance,

base.total_distance,

base.raw_historical_sum,

base.historical_count,

base.default_sum,

CASE

WHEN base.historical_count < base. total_count THEN
COALESCE(base.raw_historical_sum, 0 :: double precision) + ((base.
total_count - base.historical_count) * 46) :: double precision

ELSE base.raw_historical_sum
```

```
END AS historical_sum

FROM base

)

SELECT segment_id,

time_bucket,

shape_id,

realtime_count:: double precision / total_count::double precision AS
coverage,

CASE

WHEN realtime_count = total_count::numeric THEN realtime_sum

WHEN (realtime_count::double precision / total_count::double precision) >=
0.8::double precision THEN filled_total_sum

WHEN (realtime_count :: double precision / NULLIF(total_count, 0)::double
precision) > 0 :: double precision THEN (realtime_sum /
NULLIF(realtime_distance, 0 :: double precision) * total_distance +
filled_total_sum) / 2::double precision

WHEN historical_sum IS NOT NULL THEN historical_sum

ELSE default_sum:: double precision

END AS estimated_eta

FROM processed_base;
```

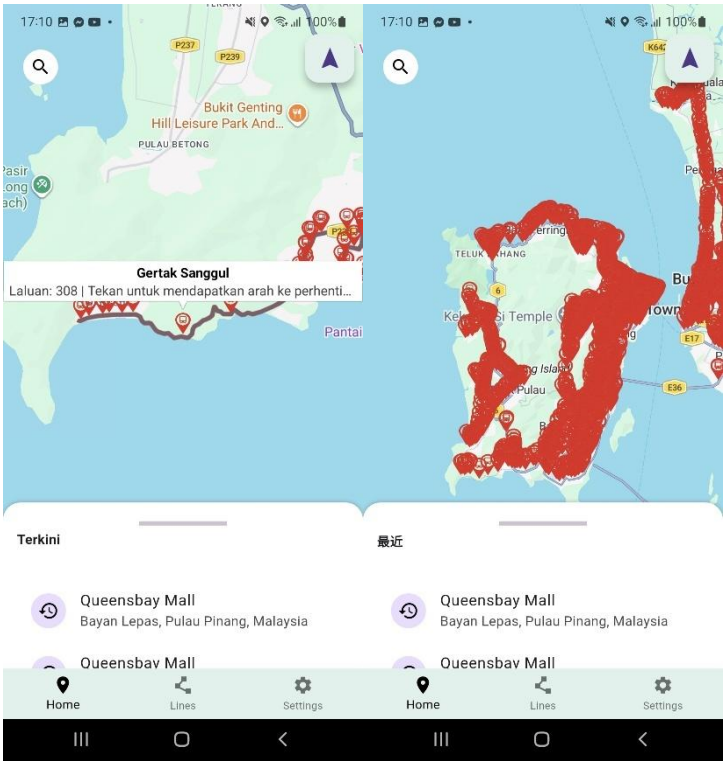
- **Subsegment_eta_view**

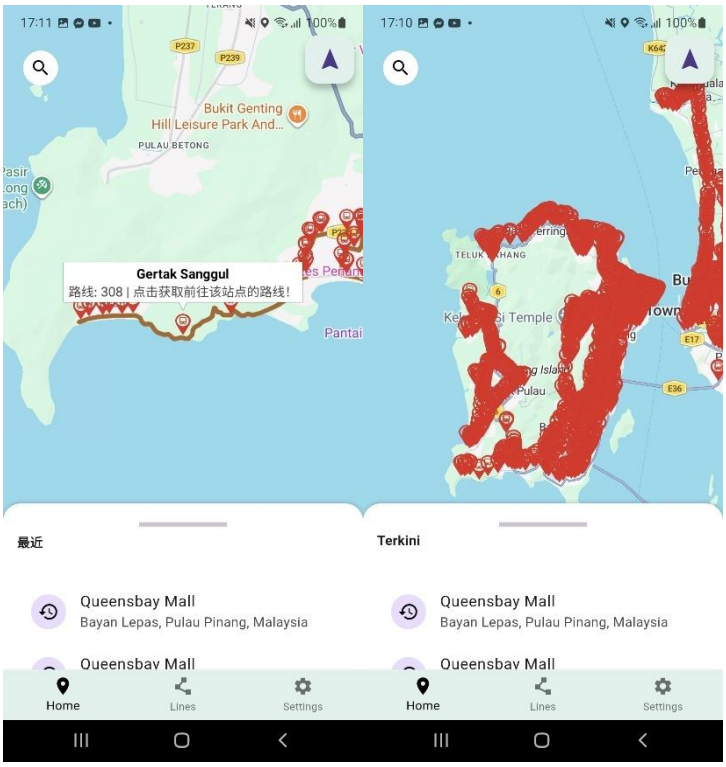
```
CREATE OR REPLACE VIEW subsegment_eta_view AS

WITH all_comb AS (

      SELECT

      s.shape_id,
```

```
    s.subsegment_id,
      t.time_bucket
      FROM subsegments s
      CROSS JOIN (
            SELECT DISTINCT time_bucket FROM historical_subsegments
      ) t
)
SELECT
    base.shape_id,
    base.subsegment_id,
      AVG(COALESCE(rt.avg_travel_time_sec,
             hist.avg_travel_time_sec,
             46
    )) AS eta_sec,
      base.time_bucket
FROM all_comb base
LEFT JOIN rt_subsegments rt
    ON base.subsegment_id = rt.subsegment_id
LEFT JOIN historical_subsegments hist
    ON base.subsegment_id = hist.subsegment_id
      AND base.time_bucket = hist.time_bucket
GROUP BY  base.shape_id, base.subsegment_id, base.time_bucket
```

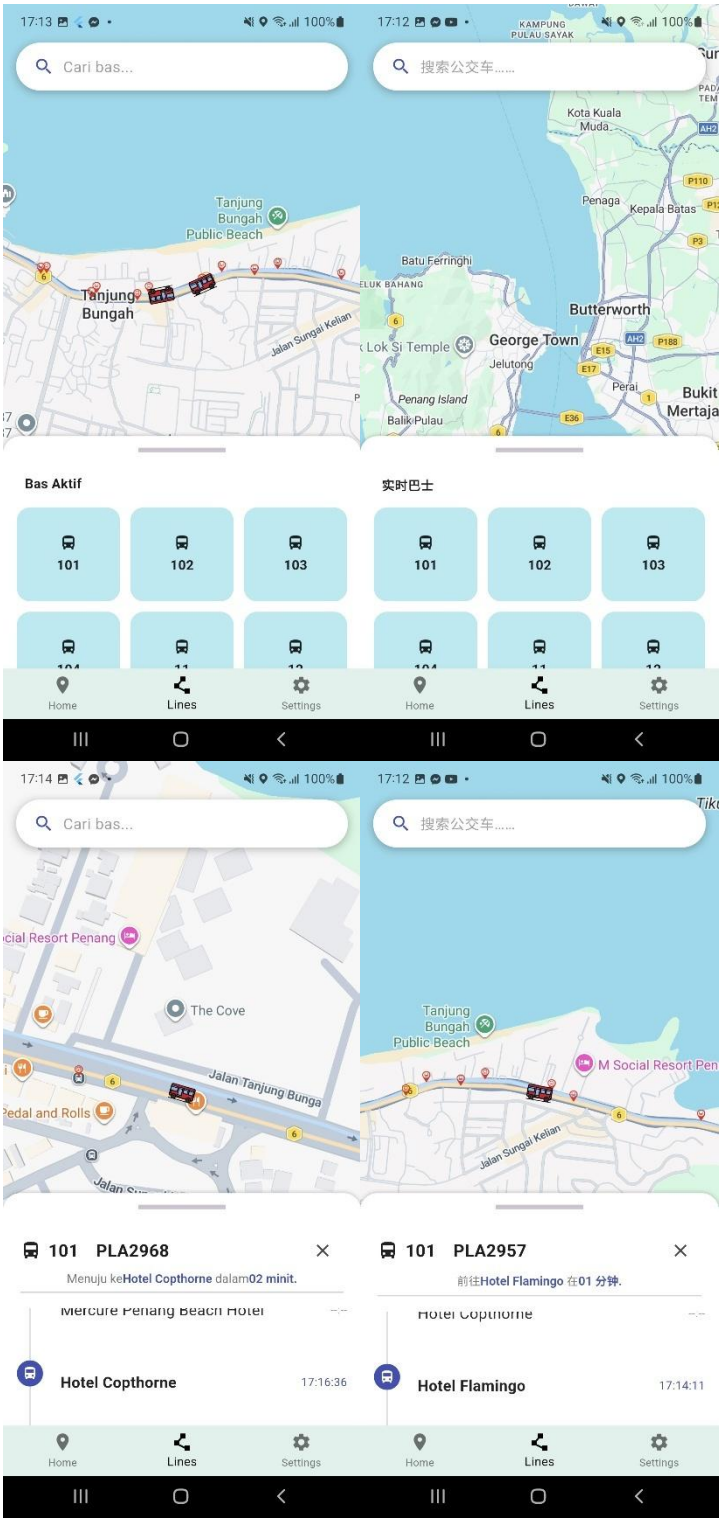
Appendix C: JomNaik User Interface with Different Languages

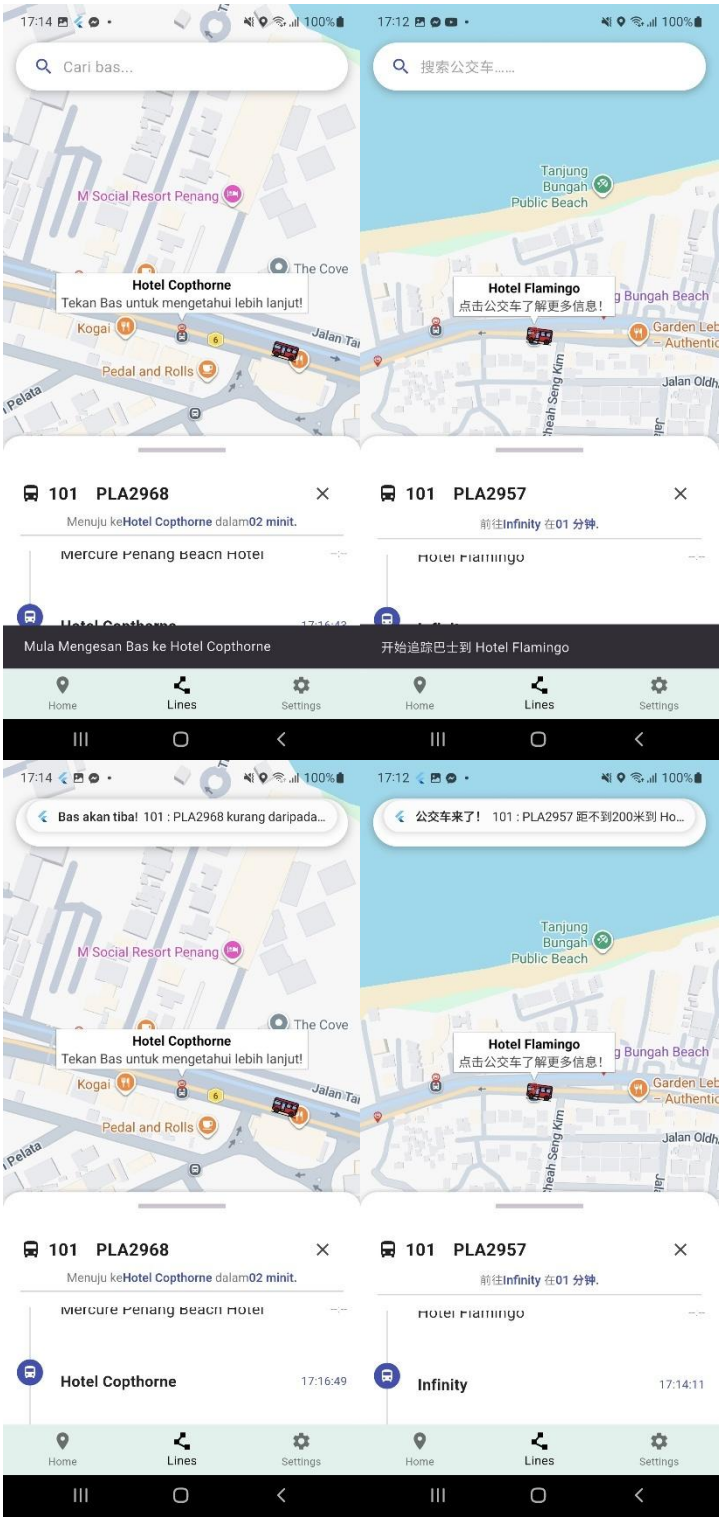
- Home Page



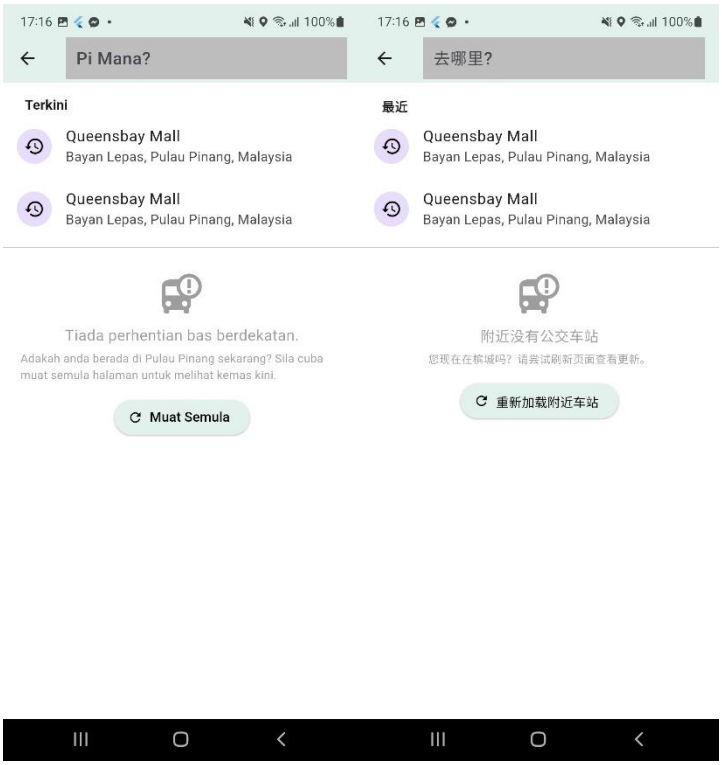


- Lines Page

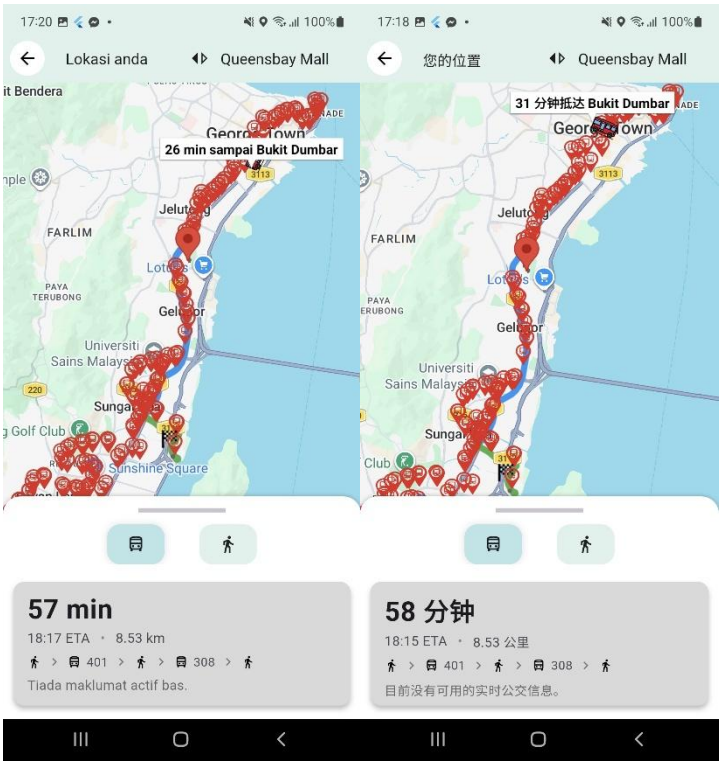

Cari bas...
搜索公交车.....
Tanjung Bungah Public Beach
Jalan Sungai Kelian
Batu Ferringhi
Butterworth
George Town
Lok Si Temple
Jelutong
Penang Island
Balik Pulau
Bas Aktif
实时巴士
101
102
103
Home
Lines
Settings
The Cove
Jalan Tanjung Bunga
M Social Resort Pen
101 PLA2968
Menuju keHotel Copthorne dalam02 minit.
Hotel Copthorne
17:16:36
101 PLA2957
前往Hotel Flamingo 在01 分钟.
Hotel Flamingo
17:14:11

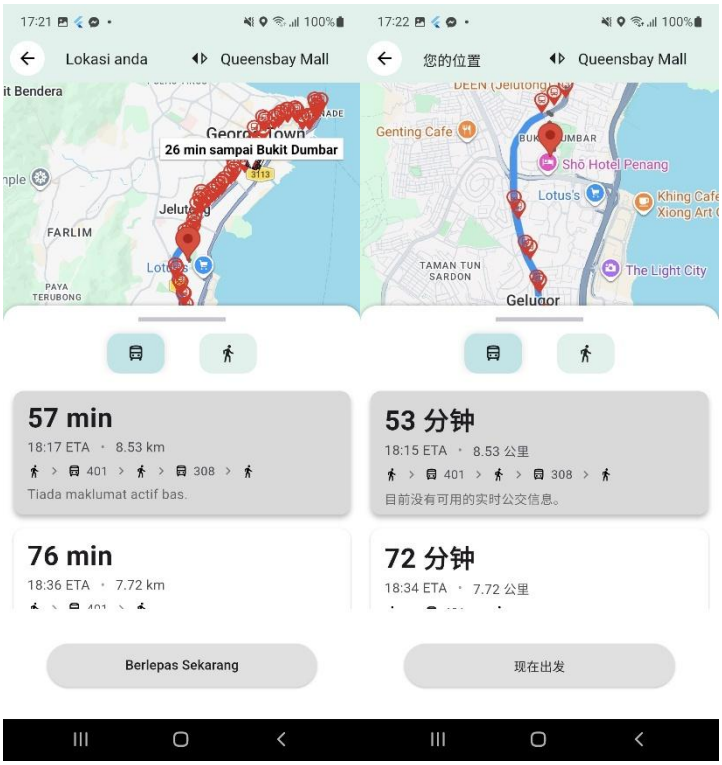

Cari bas...
M Social Resort Penang
The Cove
Hotel Copthorne
Tekan Bas untuk mengetahui lebih lanjut!
Kogai
Pedal and Rolls
搜索公交车.....
Tanjung Bungah Public Beach
Hotel Flamingo
点击公交车了解更多信息！
101 PLA2968
Menuju keHotel Copthorne dalam02 minit.
Mercure Penang Beach Hotel
101 PLA2957
前往Infinity 在01 分钟.
Mula Mengesan Bas ke Hotel Copthorne
开始追踪巴士到 Hotel Flamingo
Home
Lines
Settings
Bas akan tiba! 101 : PLA2968 kurang daripada...
公交车来了！ 101 : PLA2957 距不到200米到 Ho...
Hotel Copthorne
17:16:49
Infinity
17:14:11

- Search Page

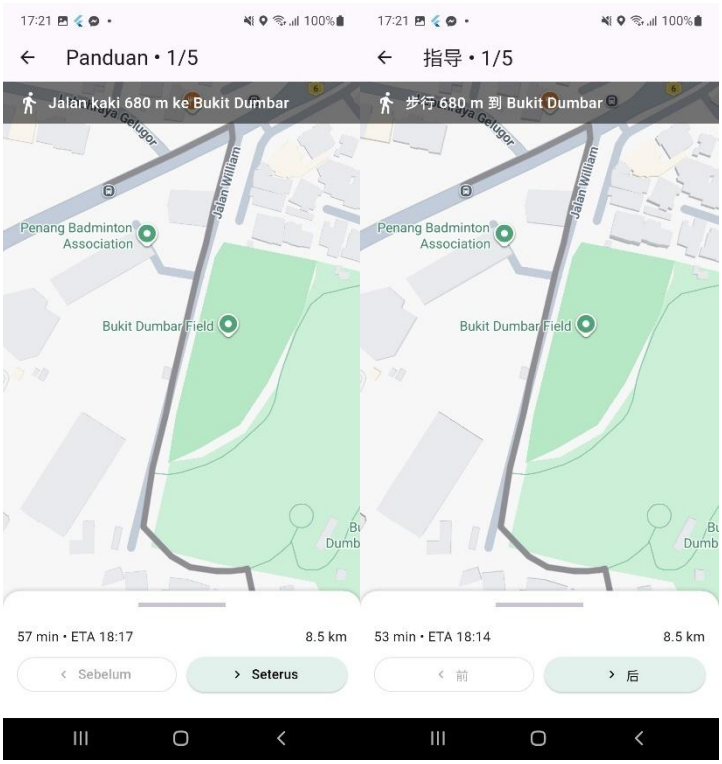


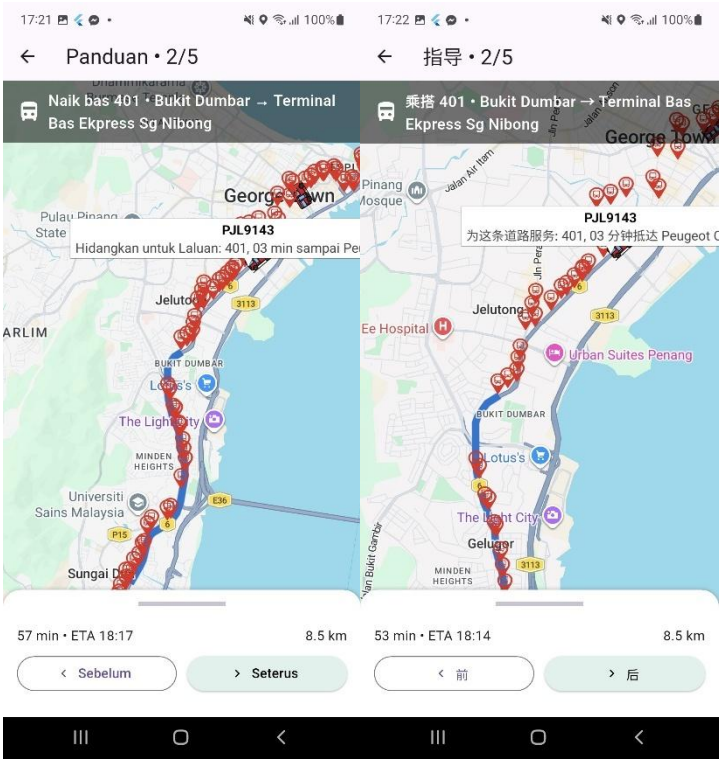
- Route Suggestion Page



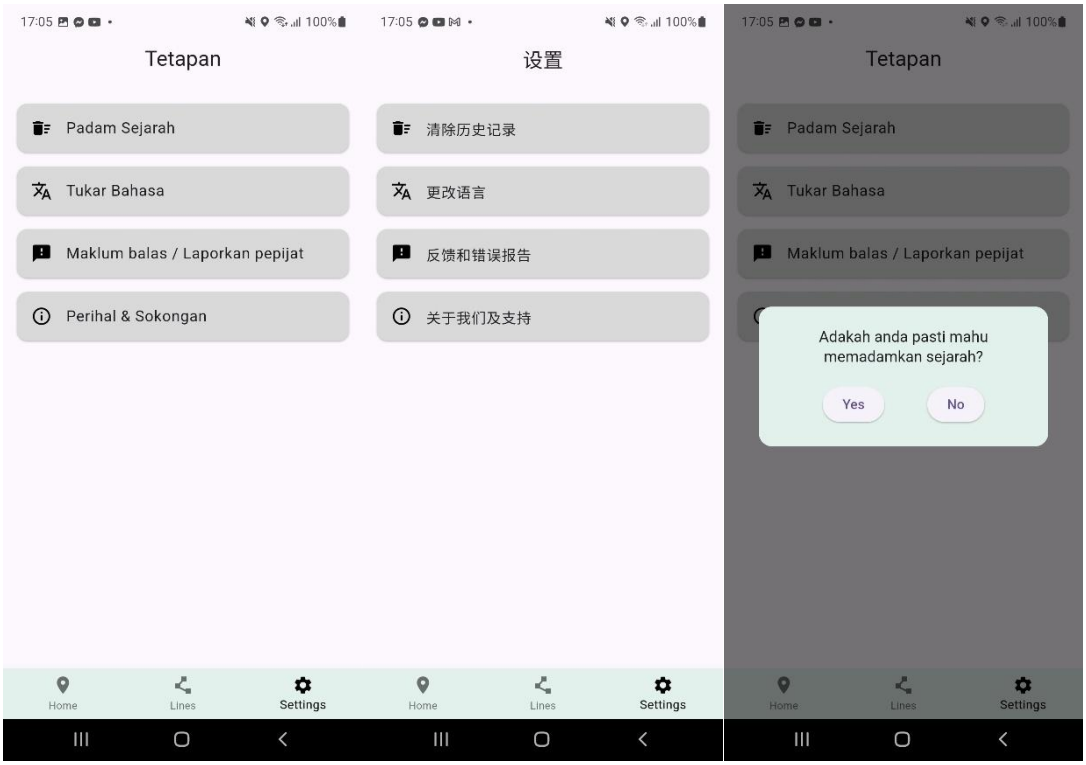


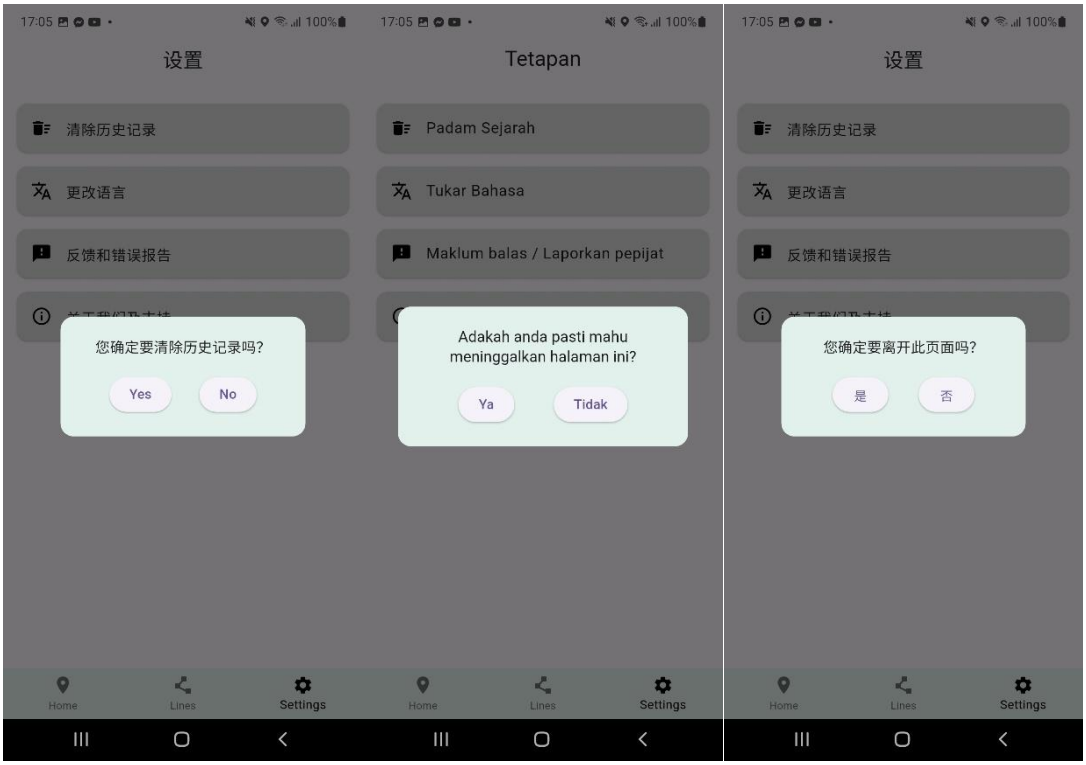
- Navigation Page

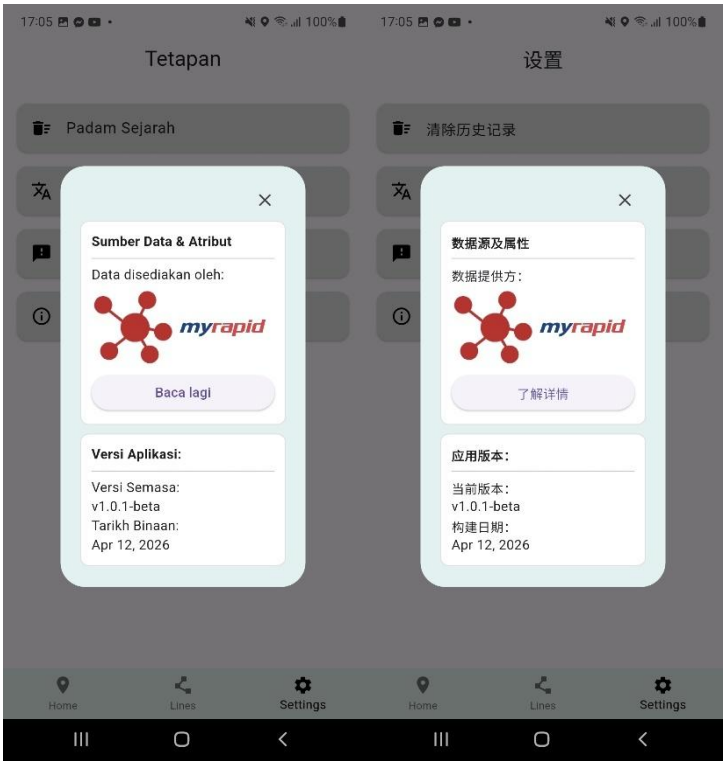




- Settings Page

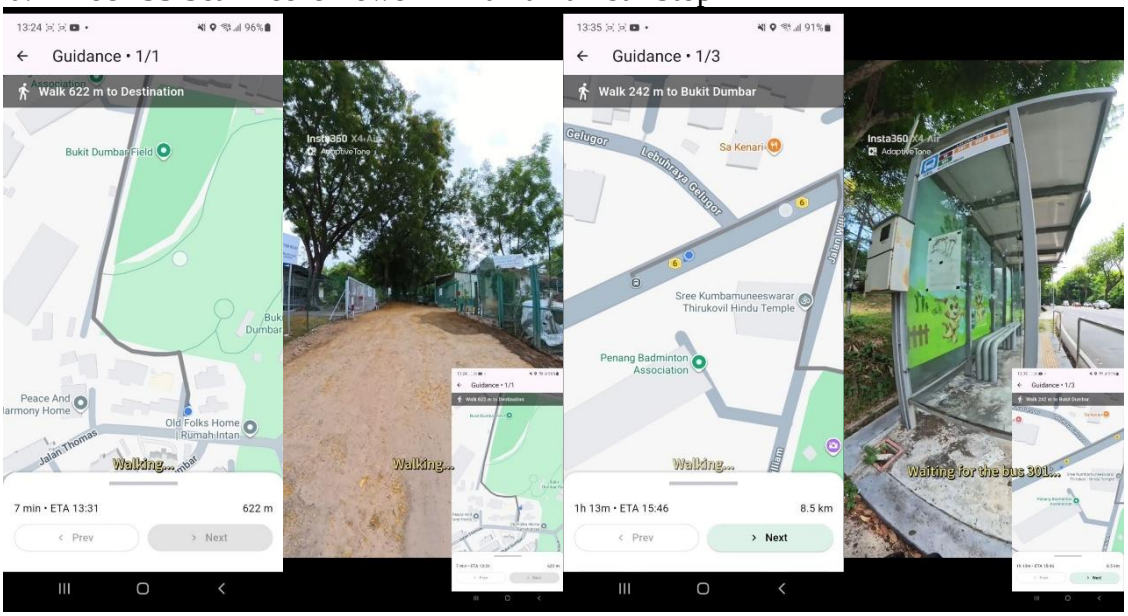



设置
清除历史记录
更改语言
反馈和错误报告
您确定要清除历史记录吗?
Yes
No
Tetapan
Padam Sejarah
Tukar Bahasa
Maklum balas / Laporkan pepijat
Adakah anda pasti mahu meninggalkan halaman ini?
Ya
Tidak
设置
清除历史记录
更改语言
反馈和错误报告
您确定要离开此页面吗?
是
否
Home
Lines
Settings

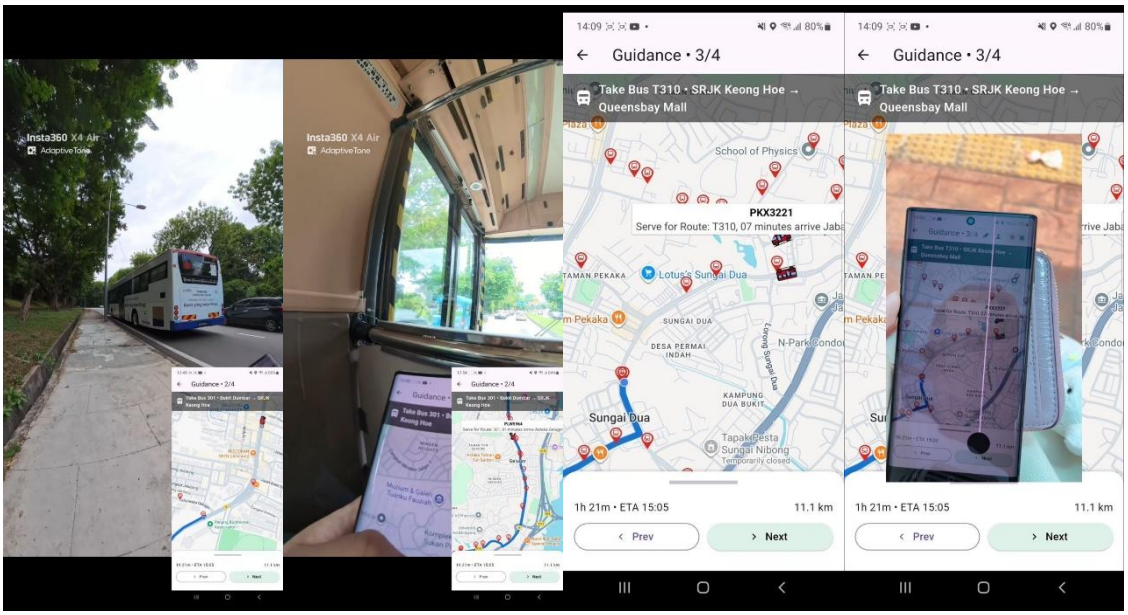

Tetapan
Padam Sejarah
Sumber Data & Atribut
Data disediakan oleh:
myrapid
Baca lagi
Versi Aplikasi:
Versi Semasa:
v1.0.1-beta
Tarikh Binaan:
Apr 12, 2026
设置
清除历史记录
数据源及属性
数据提供方:
myrapid
了解详情
应用版本:
当前版本:
v1.0.1-beta
构建日期:
Apr 12, 2026
Home
Lines
Settings

Appendix D: Image Prove for Field Test

a. Block 95 Sea Breeze Tower -> Bukit Dumbar Stop



b. Bukit Dumbar Stop -> SRJK Keong Hoe Stop



c. SRJK Keong Hoe Stop-> Queensbay Mall Stop