**UNIVERSITI MALAYSIA PAHANG**

**DECLARATION OF THESIS AND COPYRIGHT**

| | | |
|---|---|---|
| Author's Full Name | : | Nur Afifah Binti Akbar |
| Date of Birth | : | |
| Title | : | Development of a Database and E-wallet System for GPS Expressway Tolling System |
| Academic Session | : | 2021/2022 |

I declare that this thesis is classified as:

| | | |
|---|---|---|
| ☐ | CONFIDENTIAL | (Contains confidential information under the Official Secret Act 1997)* |
| ☐ | RESTRICTED | (Contains restricted information as specified by the organization where research was done)* |
| ☐ | OPEN ACCESS | I agree that my thesis to be published as online open access (Full Text) |

Certified by:

_______________________ (Student's Signature)

_______________________ (Supervisor's Signature)

Date: 14 FEBRUARY 2022

_______________________
PROF. MADYA TS DR. HADI BIN MANAP
Date: 14 FEBRUARY 2022

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

**MAKLUMAT PANEL PEMERIKSA PEPERIKSAAN LISAN**

*(only for Faculty of Computer's student)*

Thesis ini telah diperiksa dan diakui oleh
*This thesis has been checked and verified by*

Nama dan Alamat Pemeriksa Dalam :
*Name and Address Internal Examiner*

Nama dan Alamat Pemeriksa Luar :
*Name and Address External Examiner*

Nama dan Alamat Pemeriksa Luar :
*Name and Address External Examiner*

Disahkan oleh Penolong Pendaftar IPS :
*Verified by Assistant Registrar IPS*

Tandatangan :
*Signature*

Tarikh :
*Date*

Nama :
*Name*

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

## SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor Technology in Electronic Engineering (Computer System) with Honors.

________________________________

(Supervisor's Signature)

| | |
|---|---|
| Full Name | : PROF. MADYA TS DR. HADI BIN MANAP |
| Position | : SUPERVISOR |
| Date | : 14 FEBRUARY 2022 |

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

## STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_______________________________

(Student's Signature)

| | |
|---|---|
| Full Name | : NUR AFIFAH BINTI AKBAR |
| ID Number | : TG18032 |
| Date | : 14 FEBRUARY 2022 |

# DEVELOPMENT OF A DATABASE AND E-WALLET SYSTEM FOR GPS EXPRESSWAY TOLLING SYSTEM

NUR AFIFAH BINTI AKBAR

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Electronics Engineering Technology with Honors

Faculty of Technology Electrical & Electronics Engineering

UNIVERSITI MALAYSIA PAHANG

FEBRUARY 2022

## ACKNOWLEDGEMENTS

First and foremost, thanks to the Almighty God, Allah S.W.T with his willingness, I had accomplished my final year project in my fourth-year study at Universiti Malaysia Pahang as it is compulsory and a requirement for graduation and award for Bachelor Degree of Electronics Engineering Technology with Honors.

I want to express my sincerest appreciation to my supervisor Prof Madya Ts. Dr. Hadi bin Manap. He deserves my heartfelt gratitude and thanks for his constant advice and encouragement in making this study possible. Despite the numerous flaws I showed throughout the study period, he always had faith in me and my abilities. He influenced me greatly, not only in terms of improving my thesis but also in terms of caring for me as a whole. His encouragement has given me the drive and motivation to keep pushing towards completion, and I am grateful and humbled to be guided by him.

In addition, I'd want to express my gratitude to Universiti Malaysia Pahang for providing us with the resources and equipment we required to successfully finish this senior design project. I admire and appreciate my group members' dedication to making the vision a reality. We spent the whole semester developing our skills and knowledge, working till the wee hours of the morning to ensure that the project was finished and ready for the final presentation.

I'd also like to express my gratitude to my family for their unwavering support, emotional empathy, and affection. Without their help, I would not be where I am now. In their persistent prayers and faith in me, they kept me strong, especially when I felt like giving up. I'd also like to express my gratitude to my teammates for their support and assistance throughout the semester.

# ABSTRAK

Jalan tol telah menyediakan sebahagian besar pembiayaan rangkaian lebuh raya. Dengan membina jalan bertol, adalah mungkin untuk menawarkan pelbagai perkhidmatan yang dipertingkatkan kepada pengguna jalan raya, yang semuanya boleh dikira untuk memastikan mereka mencapai hasil yang diinginkan. Pihak berkepentingan utama dalam bidang jalan tol, bertanggungjawab untuk membangunkan projek tol dan menyediakan perkhidmatan operasi menggunakan teknologi berkaitan sekali gus mencapai matlamat utama membina tol, iaitu untuk memuaskan hati pelanggan. Setiap hari, tol di pusat bandar dan kawasan perindustrian dijangka menyaksikan jumlah lalu lintas yang besar yang terdiri daripada rangkaian kenderaan dari kenderaan peribadi hingga kenderaan komersial ringan dan berat, trak pelbagai gandar, dan sebagainya. Selain itu, masalah lalu lintas juga kritikal pada musim cuti terutamanya pada musim perayaan. Malah hakikat bahawa lebuh raya baharu masih dibina di Malaysia masih tidak banyak berubah dan peningkatan jumlah trafik secara beransur-ansur tidak lagi dapat dipenuhi semata-mata dengan pembinaan lebuh raya baharu..Membangunkan sistem tol lebuh raya GPS yang mampu menggantikan sistem tol semasa dengan mengurangkan bilangan sumber manusia yang bekerja di stesen tol dan menggunakan pengesanan GPS bagi semua kenderaan yang masuk dan keluar tol. Untuk memudahkan pengguna, galakkan penggunaan e-dompet dalam sistem tol. Mengelak kesesakan lalu lintas di pondok tol membolehkan pemanduan lancar tanpa perlu risau untuk membayar tol. Tambahan pula, dari permulaan dalam bab satu menerangkan pengenalan sistem Tol Lebuhraya GPS dan sebab latar belakang projek itu memberi tumpuan kepada mengatasi sistem tol semasa. Untuk memudahkan lagi pengguna, applikasi ini galakkan penggunaan e-dompet dalam sistem tol. Anda boleh memandu tanpa risau membayar tol dengan mengelak kesesakan lalu lintas di pondok tol. Kajian ini, menekankan perbezaan dengan sistem tol lain dan bagaimana sistem Tol GPS berbeza dan mampu mengatasi banyak masalah yang sistem Tol sebelum ini tidak dapat menyelesaikannya. Akhir sekali, dan sememangnya tidak kurang pentingnya, gambaran keseluruhan utama Projek SDP itu sendiri (Sistem Tol Lebuhraya GPS) dan cara ia berkerja dengan penekanan yang lebih mendalam tentang cara ia berfungsi dan perkara yang mampu dicapai dalam jangka masa panjang..

## ABSTRACT

Toll roads have provided the majority of highway network funding. By constructing a toll road, it is possible to offer a variety of enhanced services to road users, all of which can be calculated to ensure that they achieve the desired results. The major stakeholders in the toll road field, are responsible for developing the toll project and providing operational services using associated technology thus achieving the primary goal of constructing a toll, which is to satisfy customers. Every day, the toll in the city center and industrial area is expected to see a large amount of traffic consisting of a range of vehicles ranging from personal vehicles to light and heavy commercial vehicles, multi-axle trucks, and so on. In addition, traffic problems are also critical during the holiday season, especially during festive seasons. Even the fact that new highways are still being built in Malaysia still can't change much and the gradual rise in traffic volume can no longer be met solely by the construction of new highways. Develop a GPS expressway tolling system capable of replacing the current tolling system by reducing the number of human resources working at toll stations and utilizing GPS tracking of all vehicles entering and exiting the toll. To make it easier for end-users, encourage the use of e-wallets in tolling systems. Avoiding traffic jams at toll booths allows for smooth driving without having to worry about paying the toll. Furthermore, from the start in chapter one explains the introduction of the GPS Expressway Tolling system and the background reason why the project is focusing on overcoming the current tolling system. To make it easier for end users, encourage the usage of e-wallets in tolling systems. You can drive without worrying about paying the toll by avoiding traffic congestion at toll booths. This study, emphasizes the difference with other tolling systems and how the GPS Tolling system is different and able to overcome many problems that the previous Tolling system wasn't able to settle the score with. Finally, and certainly not least, the main overview of the SDP Project itself (GPS Expressway Tolling System) and how it runs and functions with more in-depth emphasis on how it works and what it's able to accomplish in the long run.

# TABLE OF CONTENT

## LIST OF TABLES

# LIST OF FIGURES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| SDP | Senior Design Project |
| PLUS | Projek Lebuhraya Utara Selatan |
| GPS | Global Positioning System |
| RM | Ringgit Malaysia |
| LMR | Limit of Maintenance Responsibility |
| UI/UX | User Interface/ User Experience |
| RFID | Radio Frequency Identification |

## LIST OF APPENDICES

# CHAPTER 1

# INTRODUCTION

## 1.1 Project Background

Historically, toll roads have provided the majority of highway network funding. By constructing a toll road, it is possible to offer a variety of enhanced services to road users, all of which can be calculated to ensure that they achieve the desired results. The major stakeholders in the toll road field, are responsible for developing the toll project and providing operational services using associated technology thus achieving the primary goal of constructing a toll, which is to satisfy customers.

The large majority of Malaysian Expressways and Highways are tolls roads. Each day, roughly 1.5 million vehicles pass the plaza toll according to the PLUS statistics. Toll fares are implemented on most vehicles using toll roads following government policy guidelines. It is crucial to provide high-quality road infrastructure and other necessary facilities to road users. Every day, the toll in the city center and industrial area is expected to see a large amount of traffic consisting of a range of vehicles ranging from personal vehicles to light and heavy commercial vehicles, multi-axle trucks, and so on. In addition, traffic problems are also critical during the holiday season, especially during festive seasons. Even the fact that new highways are still being built in Malaysia still can't change much and the gradual rise in traffic volume can no longer be met solely by the construction of new highways.

Therefore, we developed an application, GPS Expressway Tolling System which can automatically track when the vehicles enter the tolling area based on GPS and automatically deduct user money from user credit after the exit area, also saving the cost from building the tolling gates. This should be very helpful because it can improve the traffic flow and make it easier by using a cashless payment method. At the same time, users do not have to worry about their money deduction because this application will notify the users of the amount that has been deducted from their credit.

## 1.2 Problem Statement

In Malaysia, there is a toll system on every expressway and highway, which is either closed or open. The Malaysian Ringgit is used in all transactions (RM). Users are only required to pay a fixed sum at some toll plazas within the open system range. North-South Expressway issues PLUS Transit cards, and other closed toll expressways such as East Coast Expressway and South Klang Valley Expressway issue transit cards) Users collect toll tickets or touch in with their touch n go card. When entering the expressway, pay the toll or touch out with the same touch n go card at the exit toll plaza, plus the distance from the plaza to the Limit of Maintenance Responsibility (LMR). In this scheme, the toll rate is determined by the distance traveled.

The PLUS Transit reusable transit cards have been in use at all PLUS expressways since June 18, 2013, to replace the transit ticket. Due to the forthcoming complete electronic toll collection at all PLUS locations, the PLUS Transit Card will no longer be issued as of April 26, 2017. Closed system; customers must swipe the same card in and out. Nowadays the number of road users is increasing, by still using the old tolling system becomes the cause of traffic since users need to enter the toll gate pay using cash or 'touch and go' and wait for the toll gate to open. It takes a lot of times for one vehicle to pass the gate.

**Figure 1.2: Example situation at Highway Toll**

Source: CHAN, J. A. D.E. (2017, April 27). *PLUS highway Toll.* The Star Malaysia.

By applying this GPS Expressway Tolling System for every toll in Malaysia it will help improve the traffics and makes paying toll easier by the auto money deduction from credit. As the result, it will save the cost from build the tolling gates and reduce the potential of traffic during festival seasons.

## 1.3 **Objective of Project**

The main aim of this project is to develop an effective expressway tolling system for road users. To accomplish this, the following objectives must be achieved.

1. To design and an effective e-credit system for GPS Expressway Tolling system.
2. To develop applications that display the area user enter and the amount of money that user needs to pay and notify users when the money is deducted from user credit.
3. Develop a database for GPS Expressway Tolling system.

## 1.4 Significance Research

A few key points in developing more advanced technologies for expressway tolling systems are proposed in this study. Indicates how the analysis contributes to the refinement, revision, or extension of established expertise in the field of study. The aim of the design is based on design software. The following are some of our significant studies:

### 1.4.1 Software Design

1. The Arduino Integrated Development Environment (IDE) software is used to program, code- editor, identify the commands and perform the appropriate action.
2. The Blynk software is used to program, integrate computation, visualization, and perform connections with google MAP.
3. Django is a high-level Python web framework used for building database frames to bind with UI/UX for the application.
4. The Vuetify software is used for UI/UX design and visualization.

# CHAPTER 2

# LITERATURE REVIEW

Several options must be considered based on our study and review of the new expressway tolling scheme. Chapter 2 focuses on literature analysis of the previous toll collection project's design and operation, which will be compared to our " GPS Expressway Tolling System " project. Note, that GPS is used in many different types of applications to track and trace the location of the vehicles. However, this project falls somewhere within this area. This application will be a smart application that simplifies the users with a GPS tracker and cashless method payment. The final methodology and the rest of the system are shown in Chapter 3.

## 2.1 **Comparison with Other Product**

Based on our research and analysis about the Tolling System, decided to compare it with various other tolling systems that are currently being used and get a rough idea on what can be innovated/improved upon in our project. At the end of the day, the objective of this project is to succumb to a more efficient tolling system.

### **2.1.1 Open Toll System**

On an open toll system, all vehicles stop at various locations along the highway to pay a toll. While this may save money from the lack of need to construct tolls at every exit, it can cause traffic congestion, and drivers may be able to avoid tolls by exiting and re-entering the highway.

### 2.1.2 Close Toll System

With a closed system, vehicles collect a ticket when entering the highway. In some cases, the ticket displays the toll to be paid on exit. Upon exit, the driver must pay the amount listed for the given exit. Should the ticket be lost, a driver must typically pay the maximum amount possible for travel on that highway. In a variant of the closed toll system, mainline barriers are present at the two endpoints of the toll road, and each interchange has a ramp toll that is paid upon exit or entry. In this case, a motorist pays a flat fee at the ramp toll and another flat fee at the end of the toll road; no ticket is necessary.

### 2.1.3 Short Toll System

Short toll roads with no intermediate entries or exits may have only one toll plaza at one end, with motorists traveling in either direction paying a flat fee either when they enter or when they exit the toll road.

### 2.1.4 Electronic Toll System

In an all-electronic system, no cash toll collection takes place, tolls are usually collected with the use of a transponder placed before the Gate as soon as the vehicle reaches near the Transponder the amount is deducted and the gate will be opened customer account which is debited for each use of the toll road. On some road's automobiles and light trucks without transponders are permitted to use the road a bill for the toll due is then sent to the registered owner of the vehicle by mail; by contrast, some tollways require all vehicles to be equipped with a transponder.

### 2.1.5 RFID Toll System

A complete RFID system consists of a transponder (tag), reader/writer, antenna, and computer host. The transponder, better known as the tag, is a microchip combined with an antenna system in a compact package. The microchip contains memory and logic circuits to receive and send data back to the reader. These tags are classified as either active or passive tags. Active tags have internal batteries that allow a longer reading range, while passive tags are powered by the signal from their reader and thus have a shorter reading range. Passive RFID has no internal power source and uses external power to operate. These tags are powered by the electromagnetic signal received from a reader. The received electromagnetic signal charges an internal capacitor on the tags, which in turn, acts as a power source and supplies the power to the chip.

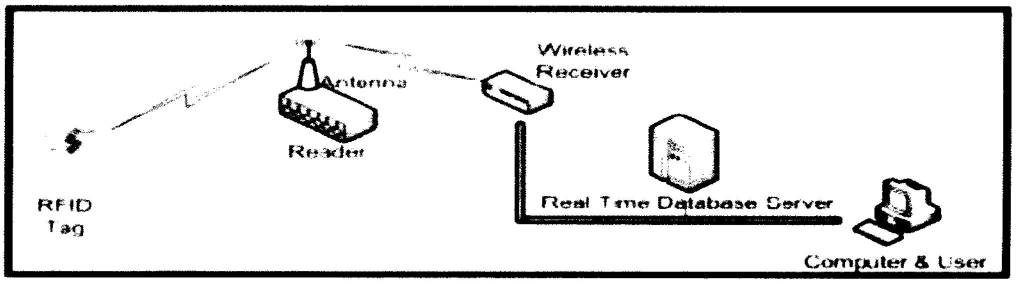


**Figure 2.1: General operation of RFID Toll System**

## 2.2 Critical Argues Findings and or Methods from Previous Work and Suggesting Potential Solution.

The above-mentioned systems for collecting toll tax are time-consuming methods. Chances of escaping the payment of toll tax are there. It leads to queuing up of following vehicles. Suppose the manual toll collection system is very efficient than for one vehicle to stop and pay taxes total time taken is 50 seconds and suppose 200 vehicles cross the toll plaza. Then, the time taken by 1 vehicle with 60 second average stop in a month is 50x30= 1500 seconds. Yearly total time taken = 1500x12 = 18000 seconds = 5.0 hours. On average each vehicle that passes through the toll plaza has to wait 5.0 hours in engine start condition yearly. The figure is staggering if on average we take 200 vehicles to pass through the toll plaza each day, then yearly 72000 vehicles pass through the toll plaza. And each year 72000 vehicles just stand still for 5.0 hours in engine start condition thereby aiding pollution and wasting fuel and money. This study is if the system is very efficient but what if the vehicle has to wait for 5 minutes? This is a figure considering one toll plaza. If considering 50 toll systems the above figure will drastically increase and the wastage of fuel, money will increase and pollution will also increase.

## 2.3 Suggesting Potential Solution

Each consumer will download the app, the GPS Expressway Toll app needs the user to sign up to have a unique ID. This unique ID can be assigned to the vehicle user used by the authorized body of the country like we can have this ID to check the user's route toll history. Once the GPS on the consumer device is turned on, able to start tracking the user via satellite using Google MAP. When the vehicle reaches the toll booth, the receiver will detect these user IDs from the app, and deduct the money from that person's e-wallet without the hassle of waiting for the barricade to open, but the consumer of the new GPS Expressway Tolling System can drive without halting anymore. Moreover, the app itself will display the deduction of e-money from that person's account once passing the specific toll.

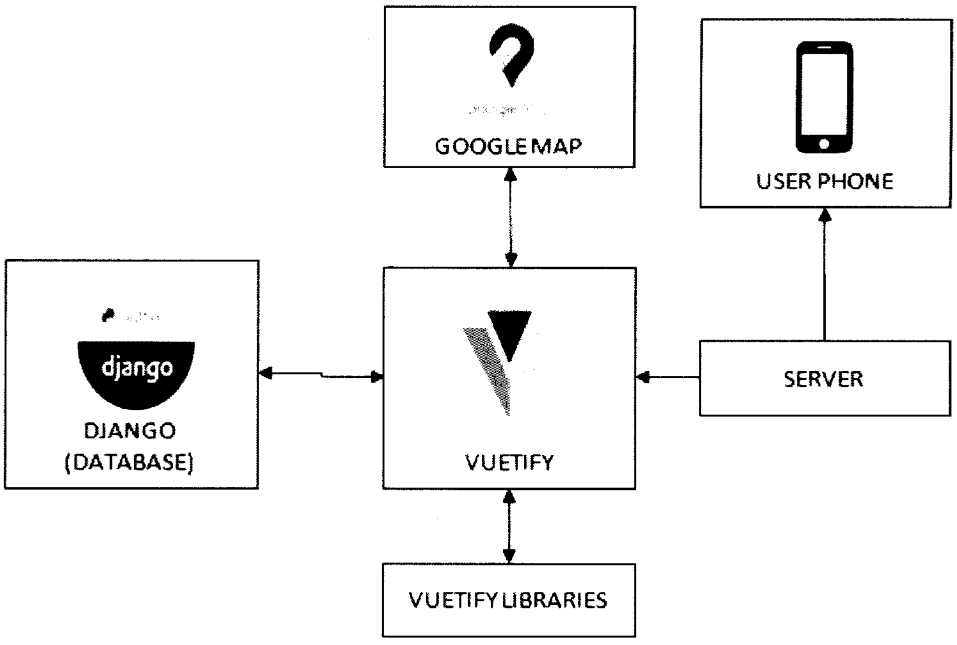


**Figure 2.2: GPS Expressway Tolling System Block Diagram**

## 2.4 Summarization of The GPS Expressway Tolling System

To implement a contemporary system of "GPS EXPRESSWAY TOLLING SYSTEM" the embedded systems platform should be a test run on the field. For this purpose, a new technology based on app-based implemented and will overcome the previous predecessor of the tolling system. By executing through using an application, able to have the best solution over money loss at toll plaza by reducing the manpower required for the collection of money and also can reduce the traffic indirectly resulting in a reduction of time at the toll plaza.

# CHAPTER 3
# METHODOLOGY

## 3.1 Component Description

### 3.1.1 Django

Django is a high-level Python web framework for building secure and maintainable websites quickly. Django is a web framework built by experienced developers that take care of a lot of the heavy lifting so you can focus on developing your app instead of reinventing the wheel. It's free and open-source, with a vibrant and active community, and excellent documentation. A high-level Web framework is a piece of software that makes creating dynamic Web sites easier. It abstracts typical Web development issues and provides shortcuts for commonly performed programming operations.

**Figure 3.1: Django**

### 3.1.2 Vuetify

Vuetify is a full user interface framework based on Vue.js. The project's purpose is to give developers the tools they need to create rich, compelling user experiences. Vuetify, unlike other frameworks, is built from the ground up to be simple to learn and rewarding to master, with hundreds of carefully developed Material Design components. Vuetify uses a mobile-first design strategy, which means your app will run on any device right out of the box, whether it's a phone, tablet, or desktop computer. Every component of Vuetify has been deliberately created to be modular, responsive, and performance by Material Design specifications. Customize your application with unique and dynamic Layouts and SASS variables to change the styles of your components.

**Figure 3.2: Vuetify**

### 3.1.3 Google MAP

Google Maps is Google's web mapping platform and consumer application. It provides satellite images, aerial photography, street maps, 360° interactive panoramic views of streets (Street View), real-time traffic conditions, and route planning for walking, driving, biking, flying (in beta), and taking public transit. Google Maps was utilized by approximately 1 billion people per month throughout the world as of 2020. Google Maps originated as a C++ desktop tool created at Where 2 Technologies by brothers Lars and Jens Rasmussen. Google purchased the firm in October 2004, and it was transformed into an online application. After successive purchases of a geographic data visualization firm and a realtime traffic analyzer, Google Maps was introduced in February 2005. The front end of the service makes use of JavaScript, XML, and Ajax. Google Maps provides an API that allows maps to be integrated on third-party websites, as well as a locator for companies and other organizations in a variety of nations worldwide. Google Map Maker, which allowed users to collaborate to develop and update the service's mapping globally, was discontinued in March 2017. However, the business indicated that crowdsourced contributions to Google Maps will be moved to the Google Local Guides initiative rather than being canceled.

**Figure 3.3: Google Maps**

## 3.2 Method Description

### 3.2.1 Django

To work with Django in the VS Code terminal, editor, and debugger, you construct this Django in the context of Visual Studio Code. Django is use to working with data models and building an administrative interface It provides a way to map requested URLs to code that responds to them. It makes displaying, validating, and re-displaying HTML forms simple. Because HTML forms are the most common means for Web users to provide data, a Web framework should make it simple to show them and handle the tiresome code of form display and re-display (with errors highlighted). It takes user input and turns it into data structures that can be easily manipulated. It uses a template system to help isolate information from the presentation. It easily interfaces with storage layers, such as databases, but it isn't necessarily required to use one.

### 3.2.2 Vuetify

Vuetify was utilized for the user interface since it was a full UI framework developed on top of Vue.js. Vuetify also allows unskilled users to construct or design their interfaces without any prior knowledge. It can interpret the frontend developer vscode to the user-pleasant interface by using yarn to install node modules and establish the libraries for each of the components within. The models within the vuetify were defined by adding a node for each function utilised. It may be used to compile all of the vscode that the developer wrote by running yarn serve. The program's body consists of src, node modules, public, yarn.lock, vue.config.js, and package.json. The assets, components, plugins, router, store, views, app. vue, and main.js are all stored under src. All of the images used to display in the interface were stored in the assets. Then, all of the component parts were saved in the components file and were linked to the views, which were used to show all of the components. Following that, the plugins saved all of the variables used to mount components. The router was used to hold all router-linked particular components. The store served as a local storage location for data entered by the user. The app. vue serves as the body, displaying all of the data stored in src. The main.js file was used to contain all of the variables defined by the developer.

The node modules deployed within Vuetify then allow the developer to utilize yarn as their package management. The yarn was utilised since it is faster in terms of performance and can install several packages at once. After all of the vscode has been saved to a file. The developer used the command prompt to execute yarn serve to create addresses for the interface's local host and network. The interface may then be accessed by the developer via the address link and onto the website browser for testing. The generated interface has not yet been deployed to the server. To allow users to access the interface, developers must publish the software to a server. Vultr.com was the server utilized. The server will then produce an address for the user interface.

### 3.2.3 Google MAP

For starters, when it comes to the GPS side of things, the use of the Google Maps API using javascript which is optimized with vuetify which can be seen within the backend coding side of things. Moreover, most of the bases of the display Google Maps is mostly the base standard of it. However, when determining the checkpoint is by making a circle radius as the point which can be seen in the live side of the UI and at the balance department. Furthermore, it will display the current location which can be seen in the balance due to setting up the coding to display the current location within the UI in vuetify. Vue.js applications can now use these APIs thanks to vue-google-maps, which has been updated for Vue.js 2. It includes a comprehensive set of components for interacting with Google Maps, as well as data binding in both directions. Create a map that uses autocomplete to display a location using the vue2-google-maps library. You can install Node.js locally by following and Creating a Local Development Environment. It may be beneficial to have some experience with setting up a Vue.js project and using Vue.js components. A key for the Google Maps javascript API. This will necessitate the use of a Google account. Using the Google Cloud Platform Console to log in. Making a new project is the first step. Enabling the Google Maps JavaScript API and the Places API for the project, as well as creating API Key credentials.

## 3.3 Flowchart

### 3.3.1 Project Process Flowchart

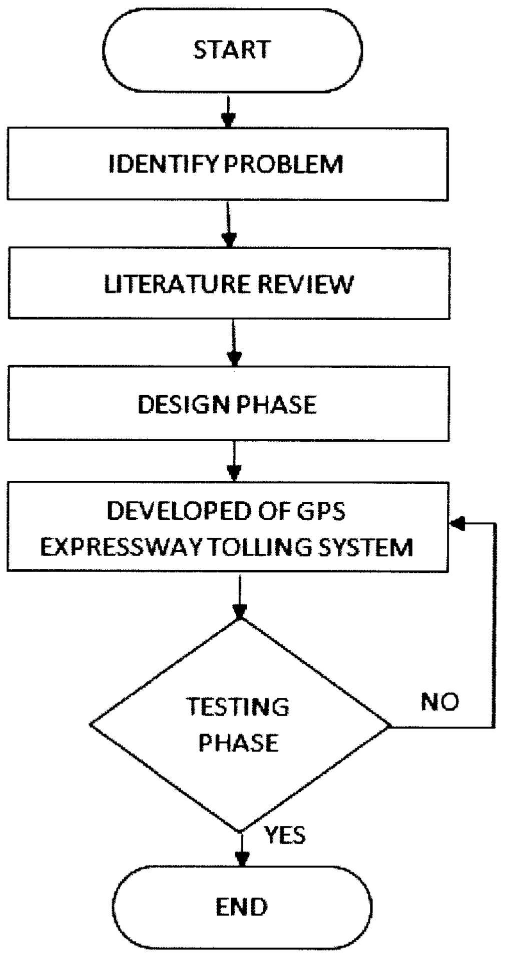


**Figure 3.4: Project Process Flowchart**

## 3.3.2 System Flowchart

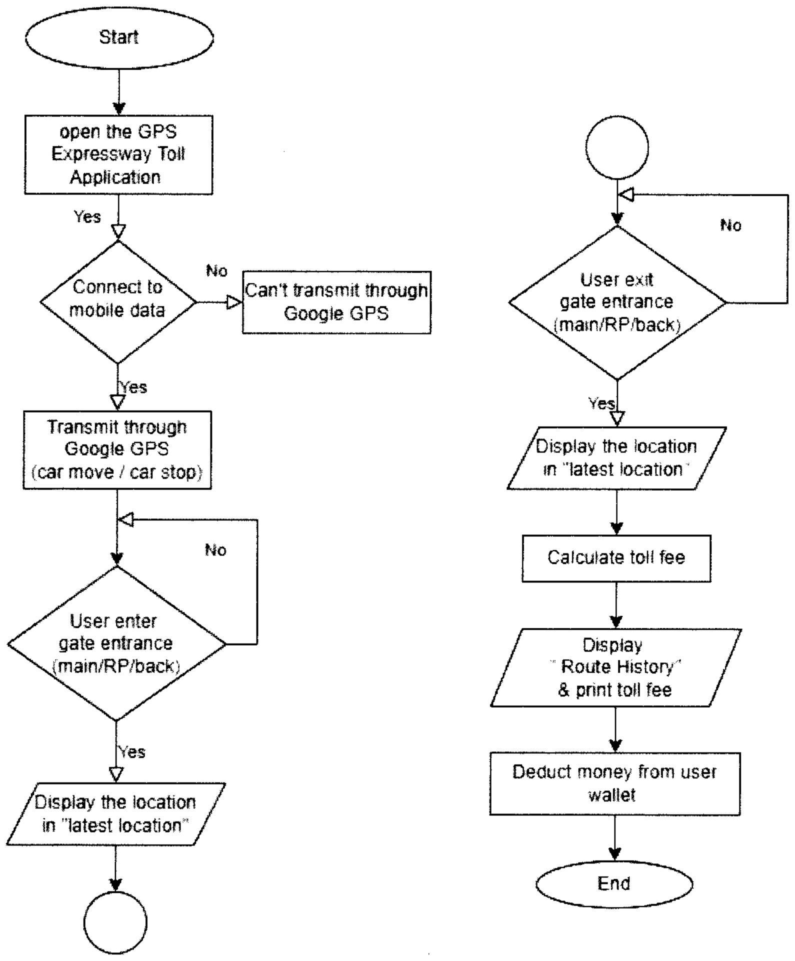

Start
open the GPS Expressway Toll Application
Yes
Connect to mobile data
No
Can't transmit through Google GPS
Yes
Transmit through Google GPS (car move / car stop)
No
User enter gate entrance (main/RP/back)
Yes
Display the location in "latest location"
No
User exit gate entrance (main/RP/back)
Yes
Display the location in "latest location"
Calculate toll fee
Display " Route History" & print toll fee
Deduct money from user wallet
End

**Figure 3.5: System Flowchart of the GPS Tolling System**

The above flowchart shows the flow of the system. The GPS expressway application turns on and connects to mobile data. Transmit to GPS in Google MAP. By using the application, users may detect the location toll without the existence of toll gates. As the user goes through the toll point, the GPS will transmit data through mobile data and satellite as it is already connected by it. The data will be transmitted to a database of The GPS expressway toll application. The application will display the user's current toll location. Then, after the user passes the second point, The app will calculate the toll fee.

It will display the current toll location and the toll fee. The application will automatically display the printout of the toll fee and deduct money from the user e-wallet.

### 3.3.3 E-Wallet System Flowchart

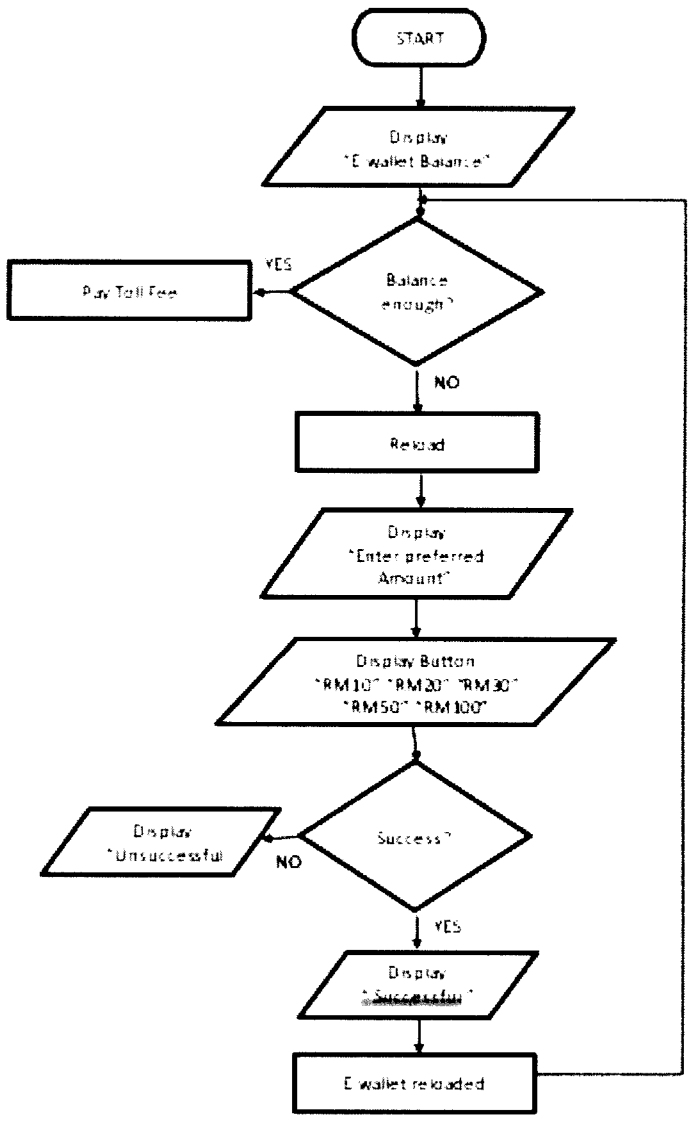


**Figure 3.6: E-Wallet Flowchart of the GPS Tolling System**

## 3.4 System Architecture

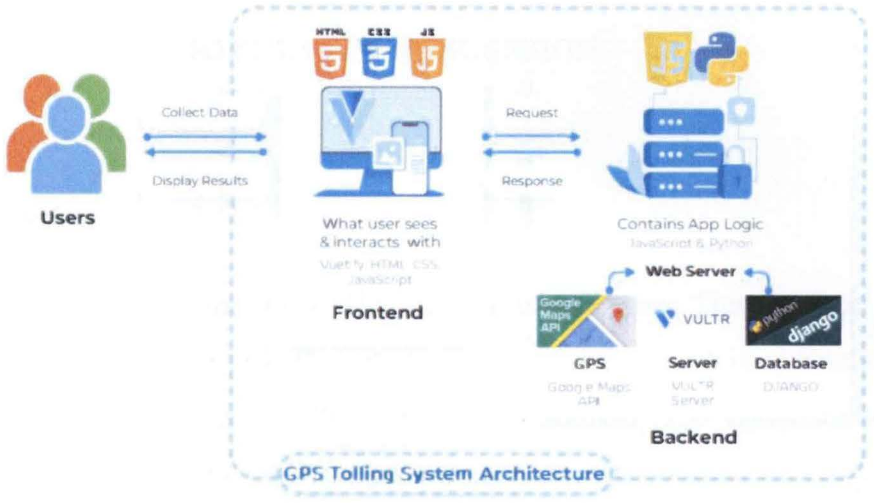


**Figure 3.7: System Architecture of the GPS Tolling System**

The above diagrams show how the GPS tolling system works. Using the user's phone, users may easily track their real-time location. The user's phone must be connected to the internet to connect to the server. Using Google Maps, it can detect the current position. The information will be saved in a Django database. The user's location will then be sent straight to Vuetify as they pass through the toll coordinate using Google Maps. It also allows the user to determine their speed of movement.

# CHAPTER 4

# RESULT AND DISCUSSION

## 4.1 Introduction

This chapter presents the study's findings and commentary. The interface as well as the figures are described in full. The interface will be written in Visual Studio Code and displayed using the Django localhost or network address. The visual studio code contained in the Visual Studio Code program is used in every model and component designed in this database. When the localhost and network addresses are received using the command prompt, the database will be shown. In a text-based user interface screen for a Windows operating system or software, a command prompt is an input area. Using a preliminary data model sketch, all of the models for each of the database's functions were obtained. Django is a database-building web framework. A model is present in a database. A model is a data source that serves a single purpose. It contains the data's most important fields and actions. Each model corresponds to a single database table, in most cases.

## 4.2 Application Behaviour

A key role of information systems is to manage huge amounts of structured and unstructured data. Data models explain the structure, manipulation, and integrity of data contained in relational databases and other data management systems. Data models serve as a foundation for the creation of information systems by defining and formatting data. In a conclusion, specifying the application behavior is required when creating a data model.

### 4.2.1 Application Behaviour for Toll User

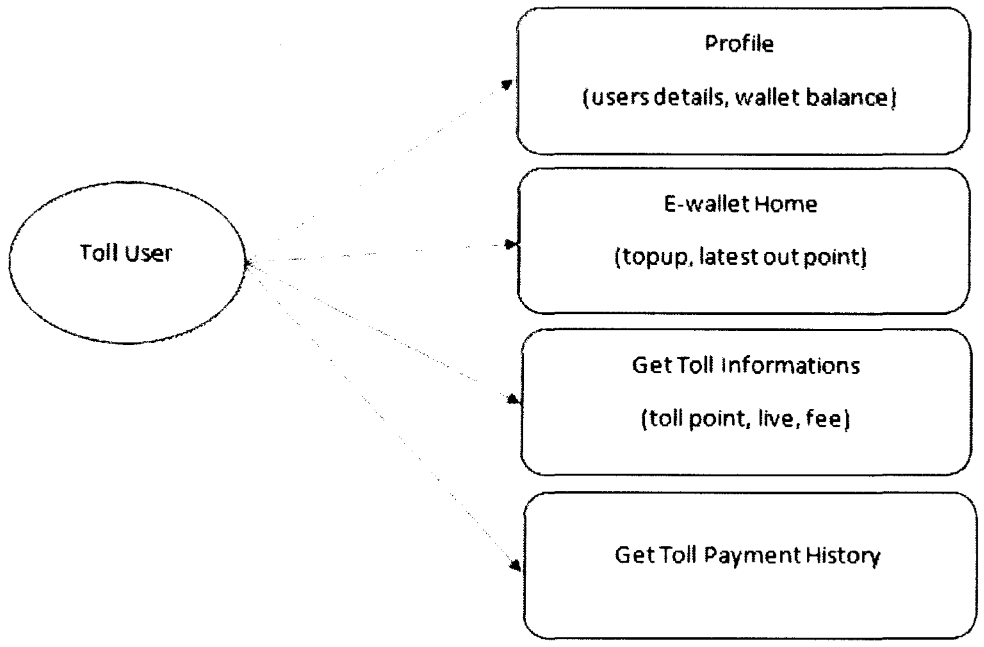


**Figure 4.1: Application behaviour for toll user**

The application behaviour for the user is to show what the user can access in the GPS Expressway Tolling System application. From figure 11, its showed that in GPS Expressway Tolling System application user can access their profile where it contain user details such as full name, phone number, IC no, gender, and wallet balance. Second is the E-wallet home where this part it's essential for the users since its use to reload E-wallet, check wallet balance and know the latest out toll point. Third, is the user can get their toll information that includes toll point, toll fees, and the live road. Lastly, is the toll payment history to show the histories of payments that the user has made.

### 4.2.2 Application Behaviour for Toll Admin

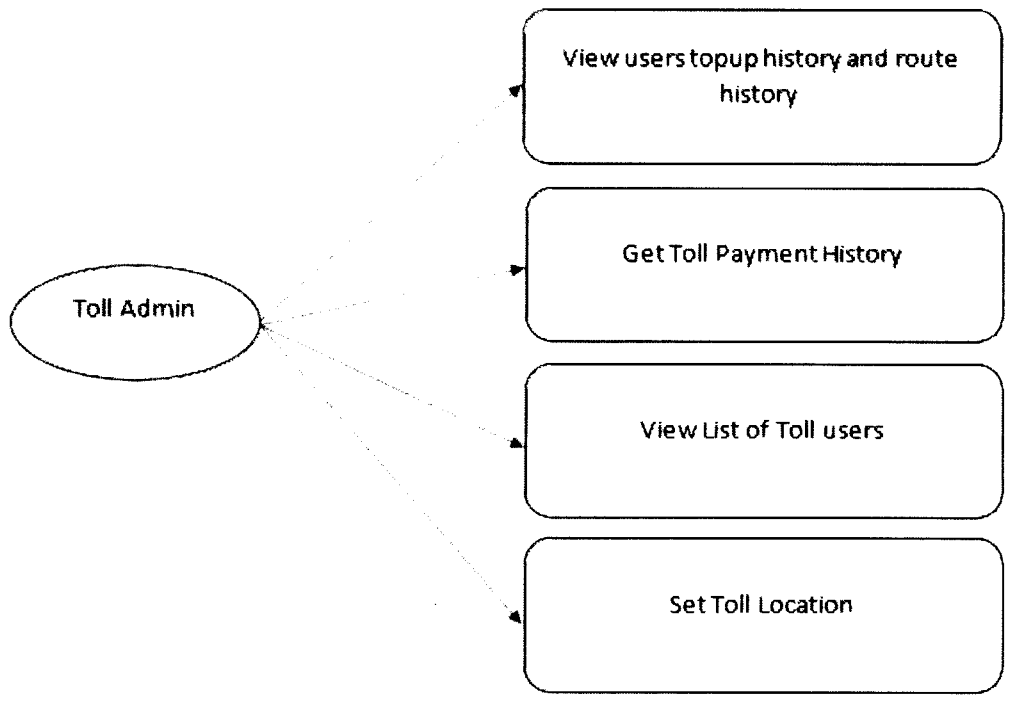


**Figure 4.2: Application behaviour for toll admin**

The application behaviour for admin is to show what toll admin can access from GPS Expressway Tolling System application. From figure 12, it showed that toll admin can view users' top-up history, route history, toll payment history, list of toll users. This part is for toll admin to monitor so if there is a report from a user toll admin can go through this data. Toll admin also can set a toll location from the admin side.

## 4.3 **Database**

A database is a collection of data that has been organized to allow for easy access, management, and updating. Data records or files containing information, such as sales transactions, customer data, financials, and product information, are often stored in computer databases. Databases are used to store, manage, and retrieve any type of data. They gather data about people, locations, and things. That data is collected in one location so that it can be viewed and analysed. Databases can be viewed as a logically structured collection of data. Another equally important style is the caption. All captions for figures, tables, and equations are formatted using their respective styles prepared in this template.

### 4.3.1 Data Model

The act of generating a visual representation of an entire information system or parts of it in order to express linkages between data points and structures is known as data modelling. The purpose is to show the many types of data that are used and stored in the system, as well as the links between them, how the data can be categorised and arranged, and its formats and features.

The Toll User data model depicts the link between database tables that contain user information. Users' traits can be classified into one of three types of data: -

1. personal Information: Name, email address, phone number, IC number, Gender, and other information.
2. Login information: Username, password, and any third-party authentication tokens such as email address or phone number.
3. Foreign keys: One to an Accounts table, most likely, to link a user to an account.

**Table 4.3.1: Toll User Table**

| ID | FULL NAME | PHONE NUMBER | IC NUMBER | Gender | Email | Password | Vehicle id |
|---|---|---|---|---|---|---|---|
| 1 | MR. DILAN | 0123445678 | 860101075342 | Male | Dilan@gmail.com | ***** | 1 |
| 2 | MISS AMIRA | 0113425678 | 950202035421 | Female | Mira@gmail.com | ***** | 2 |
| 3 | MRS. LUI HEE | 0143328756 | 001212013216 | Male | Lui@gmail.com | ***** | 3 |
| 4 | MR. Muru | 0134455622 | 991130075421 | Male | Muru@gmail.com | **** | 2 |
| 5 | Miss Alia | 0112356489 | 740603036524 | Female | Alia@gmail.com | **** | 1 |

The tolls must have the entrance and exit point, since the GPS Expressway Tolling System is tested in UMP so the point entrance is set in UMP area as shown in Table 4.2.

**Table 4.3.2: Toll Table**

| ID | Point name |
|---|---|
| 1 | UMP Main Gate Entrance |
| 2 | UMP RP Gate Entrance |
| 3 | UMP Back Gate Entrance |

The most common forms of personally driven vehicles are classified as Class 1. Vehicles having a maximum of two axles and an overall3 to 4 wheels fall into this category. Sedans, coupes, SUVs, MPVs, vans, and motorbikes (400cc and up) are all classified as Class 1 vehicles. Vehicles classified as Class 2 have two axles and 5 to 6 wheels. These are the categories in which trucks the size of the Fuso Canter fall. Lastly for vehicles classified in Class 3 are the vehicles with more than 2 axles by this the bus and lorry fall in this class.

**Table 4.3.3: Vehicle Class Table**

| ID | Category name | No of Wheels | No Of Axles |
|---|---|---|---|
| 1 | Class 1 | 3 or 4 | 2 |
| 2 | Class 2 | 5 or 6 | 2 |
| 3 | Class 3 | 5 and 6 | 3 and more |

The Toll Regulatory Board (TRB) assigns different toll costs to different vehicles class, and this classification is also applied to the GPS Expressway Tolling System as shown in Table 4.4.

**Table 4.3.4: Route Table (1)**

| Point | | OUT | | | Vehicle category |
|---|---|---|---|---|---|
| | | UMP Main Gate Entrance | UMP Lake | UMP Back Gate Entrance | |
| IN | UMP Main Gate Entrance | ■ | 1.00 | 2.00 | Class 1 |
| | | ■ | 1.20 | 2.20 | Class 2 |
| | | ■ | 1.40 | 2.40 | Class 3 |
| | UMP Lake | 1.00 | ■ | 1.00 | Class 1 |
| | | 1.20 | ■ | 1.20 | Class 2 |
| | | 1.40 | ■ | 1.40 | Class 3 |
| | UMP Back Gate Entrance | 2.00 | 1.00 | ■ | Class 1 |
| | | 2.20 | 1.20 | ■ | Class 2 |
| | | 2.40 | 1.40 | ■ | Class 3 |

**Table 4.3.5: Route Table (2)**

| ID | In_point_id | Out_point_id | Vehicle_id | charge |
|---|---|---|---|---|
| **1** | | | 1 | RM1.00 |
| **2** | 1 | 2 | 2 | RM1.20 |
| **3** | | | 3 | RM1.40 |
| **4** | | | 1 | RM1.00 |
| **5** | 2 | 1 | 2 | RM1.20 |
| **6** | | | 3 | RM1.40 |
| **5** | | | 1 | RM2.00 |
| **7** | 1 | 3 | 2 | RM2.20 |
| **8** | | | 3 | RM2.40 |
| **9** | | | 1 | RM2.00 |
| **10** | 3 | 1 | 2 | RM2.20 |
| **11** | | | 3 | RM2.40 |
| **12** | | | 1 | RM1.00 |
| **13** | 2 | 3 | 2 | RM1.20 |
| **14** | | | 3 | RM1.40 |
| **15** | | | 1 | RM1.00 |
| **16** | 3 | 2 | 2 | RM1.20 |
| **17** | | | 3 | RM1.40 |

Route history table is to track the histories of tolls that user has pass through. By this toll user will fully aware with the toll fees user has paid or need to pay. Route history data also important for toll admin to trace users who didn't pay their toll fees.

**Table 4.3.6: Route History Table**

| ID | TollUser ID | Vehicle ID | Plate Number | In point id | Out point id | Time In | Time Out | charge |
|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 2 | GTR 2211 | 1 | 2 | 4PM | 6.05PM | 1.20 |
| **2** | 2 | 1 | WX 3344 | 3 | 2 | 2PM | 4.10PM | 1.00 |
| **3** | 3 | 1 | PT 3321 | 1 | 3 | 8AM | 2.30PM | 2.00 |
| **4** | 4 | 3 | WU 2187 | 2 | 1 | 6AM | 1.07PM | 1.40 |
| **5** | 5 | 1 | PTN 9966 | 3 | 1 | 1PM | 1.15PM | 2.00 |

An e-wallet is a secure money management system or online platform that enables you to make payments in the GPS Expressway Tolling System. Table 4.6 in the database is required to distinguish each user's e-wallet so that users can transfer or send money as in Table 4.7, follow their transactions as in Table 4.8 and track users debt as in Table 4.9 .

**Table 4.3.7: Wallet Table**

| ID | TollUser ID | Balance |
|---|---|---|
| **1** | 1 | 4.00 |
| **2** | 2 | 5.00 |
| **3** | 3 | 0 |
| **4** | 4 | 1.00 |
| **5** | 5 | 0.50 |

**Table 4.3.8: Top-up Table**

| ID | Wallet ID | Created Time | Created Date | Amount | Status Credited wallet |
|---|---|---|---|---|---|
| 1 | 3 | 4.50PM | 21/5/2021 | 5.00 | Successful |
| 2 | 4 | 4.58PM | 22/5/2021 | 4.00 | Successful |
| 3 | 5 | 3PM | 22/5/2021 | 3.00 | Successful |

**Table 4.3.9:User Payment Table**

| ID | Wallet ID | Created Time | Created Date |
|---|---|---|---|
| 1 | 1 | 5PM | 21/5/2021 |
| 2 | 2 | 4PM | 23/5/2021 |
| 3 | 3 | 4PM | 21/5/2021 |
| 4 | 4 | 3PM | 22/5/2021 |
| 5 | 5 | 8PM | 22/5/2021 |

**Table 4.3.10: Debt Table**

| ID | TollUser ID | Debt Amount |
|---|---|---|
| 1 | 1 | RM4 |
| 2 | 2 | RM5 |
| 3 | 3 | RM4 |

### 4.3.2 Django Model

A model is the sole, authoritative source of data information. It includes all of the fields and actions that the data you're storing requires. In most cases, each model corresponds to a single database table. Each model is a subclass of django.db.models in Python. Each model attribute corresponds to a database field. Django provides an automatically built database-access API as a result of all of this; see Making queries. The most important part of a model – and the only required part of a model – is the list of database fields it defines. Fields are specified by class attributes. Field names should be chosen with care to avoid conflicts with the models API, such as clean, save, and delete.

#### 4.3.2.1 Toll User

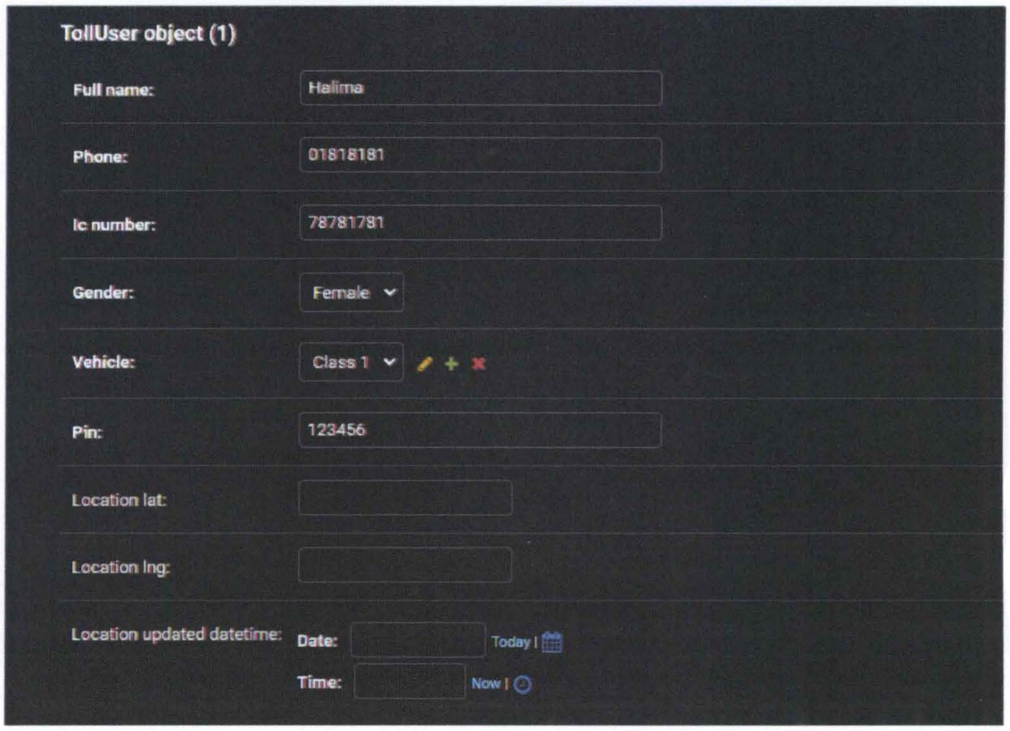
This model defines a User, which has a full_name, phone, ic_number, gender and vehicle:

```
from django.db import models
from model_utils.models import TimeStampedModel


class TollUser(TimeStampedModel):

    GENDER_CHOICES = (('M', 'Male'),
                      ('F', 'Female'))

    full_name = models.CharField(max_length=64)
    phone = models.CharField(max_length=64, unique=True)
    ic_number = models.CharField(max_length=14)
    gender = models.CharField(max_length=1, choices=GENDER_CHOICES,
default='M')
    vehicle = models.ForeignKey('Vehicle', related_name='toll_users',
on_delete=models.PROTECT)
```

full_name, phone, ic_number, gender and vehicle are fields of the model. Each field is specified as a class attribute, and each attribute maps to a database column:

```
{

   "count": 1,

   "next": "http://g3gts-api.student.my/toll/toll_users?page=2",

   "previous": null,

   "results": [
{
    "id": 1,
    "wallet": {
        "id": 1,
        "balance": 354560,
        "toll_user": 1
    },
    "created": "2021-11-21T16:42:29.774252+08:00",
    "modified": "2021-11-21T16:42:29.774252+08:00",
    "full_name": "Halima",
    "phone": "01818181",
    "ic_number": "78781781",
    "gender": "F",
    "pin": "12345,
    "vehicle": 1
},
```

The above User model would create a database table like this:


TollUser object (1)

| | |
|---|---|
| Full name: | Halima |
| Phone: | 01818181 |
| Ic number: | 78781781 |
| Gender: | Female |
| Vehicle: | Class 1 |
| Pin: | 123456 |
| Location lat: | |
| Location lng: | |
| Location updated datetime: | Date: Today \| Time: Now \| |

### 4.3.2.2 Toll Point

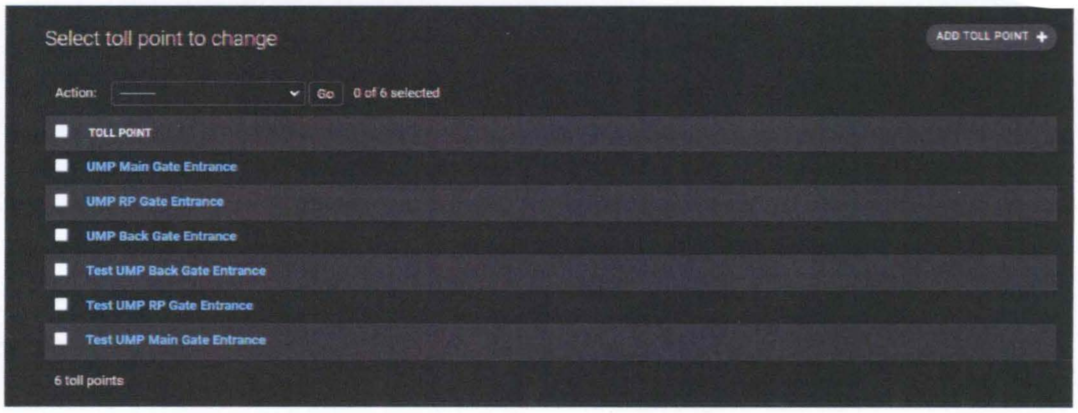
This model defines a Toll Point, which has a name, lat, lng, and radius:

```
class TollPoint(models.Model):
    name = models.CharField(max_length=64)
    lat = models.DecimalField(max_digits=22, decimal_places=16,
blank=True, null=True)
    lng = models.DecimalField(max_digits=22, decimal_places=16,
blank=True, null=True)
    radius = models.PositiveIntegerField(default=50)
```

lat, lng and radius are fields of the model. Each field is specified as a class attribute, and each attribute maps to a database column:

```
"count": 3,

"next": null,

"previous": null,

"results": [

    {

        "id": 4,

        "name": "UMP Back Gate Entrance",

        "lat": "3.5426700000000000",

        "lng": "103.4277100000000000",

        "radius": 30,

        "index": 3,

        "enabled": true

    },

    {

        "id": 5,

        "name": "UMP RP Gate Entrance",

        "lat": "3.5345150000000000",

        "lng": "103.4325040000000000",

        "radius": 30,

        "index": 2,

        "enabled": true

    },

    {

        "id": 6,

        "name": "UMP Main Gate Entrance",

        "lat": "3.5454670000000000",
            "lng": "103.434545000000000",
            "radius": 30,
            "index": 1,
            "enabled": true
        }
    ]
}
```

The above TollPoint model would create a database table like this:


Select toll point to change

ADD TOLL POINT +

Action: ——— Go 0 of 6 selected

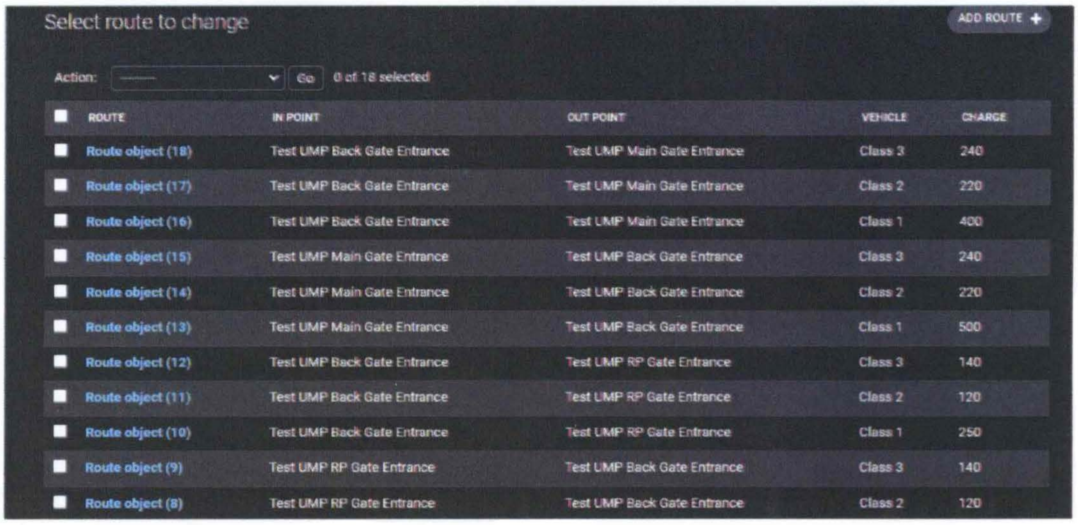
| TOLL POINT |
|---|
| UMP Main Gate Entrance |
| UMP RP Gate Entrance |
| UMP Back Gate Entrance |
| Test UMP Back Gate Entrance |
| Test UMP RP Gate Entrance |
| Test UMP Main Gate Entrance |

6 toll points

From toll point model, we can relate it to Route model as shown below:

```
class Route(models.Model):
    in_point = models.ForeignKey('TollPoint', related_name='routes_in',
on_delete=models.CASCADE)
    out_point = models.ForeignKey('TollPoint', related_name='routes_out',
on_delete=models.CASCADE)
    vehicle = models.ForeignKey('Vehicle', related_name='routes',
on_delete=models.CASCADE)
    charge = models.PositiveIntegerField()
```

The above Route model would create a database table like this:



Select route to change

ADD ROUTE +

Action: --------- Go 0 of 18 selected

| ROUTE | IN POINT | OUT POINT | VEHICLE | CHARGE |
|---|---|---|---|---|
| Route object (18) | Test UMP Back Gate Entrance | Test UMP Main Gate Entrance | Class 3 | 240 |
| Route object (17) | Test UMP Back Gate Entrance | Test UMP Main Gate Entrance | Class 2 | 220 |
| Route object (16) | Test UMP Back Gate Entrance | Test UMP Main Gate Entrance | Class 1 | 400 |
| Route object (15) | Test UMP Main Gate Entrance | Test UMP Back Gate Entrance | Class 3 | 240 |
| Route object (14) | Test UMP Main Gate Entrance | Test UMP Back Gate Entrance | Class 2 | 220 |
| Route object (13) | Test UMP Main Gate Entrance | Test UMP Back Gate Entrance | Class 1 | 500 |
| Route object (12) | Test UMP Back Gate Entrance | Test UMP RP Gate Entrance | Class 3 | 140 |
| Route object (11) | Test UMP Back Gate Entrance | Test UMP RP Gate Entrance | Class 2 | 120 |
| Route object (10) | Test UMP Back Gate Entrance | Test UMP RP Gate Entrance | Class 1 | 250 |
| Route object (9) | Test UMP RP Gate Entrance | Test UMP Back Gate Entrance | Class 3 | 140 |
| Route object (8) | Test UMP RP Gate Entrance | Test UMP Back Gate Entrance | Class 2 | 120 |

From data model of route in Table 4.4 there is 17 route object with different in_point, out_point, vehicle class and charge.

#### 4.3.2.3 Route History

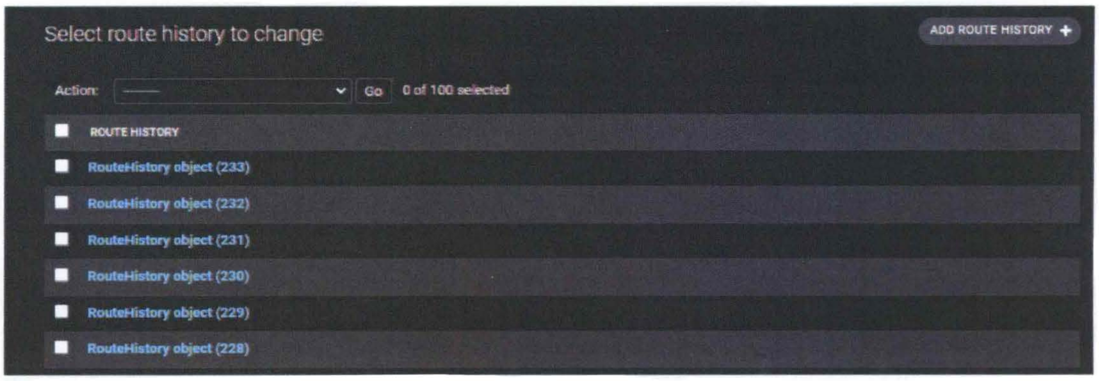
This model defines a Route History, which has a toll_user, related_name, plate_number, and charge:

```
class RouteHistory(TimeStampedModel):
    toll_user = models.ForeignKey('TollUser',
related_name='route_histories', on_delete=models.CASCADE)
    plate_number = models.CharField(max_length=10)
    charge = models.PositiveIntegerField(default=0)
    route = models.ForeignKey('route', related_name='route_histories',
on_delete=models.PROTECT)
```

toll_user, plate_number, charge and route are fields of the model. Each field is specified as a class attribute, and each attribute maps to a database column:

```
{
        "id": 17,
"route": {
    "id": 7,
    "in_point": {
        "id": 2,
        "name": "Test UMP RP Gate Entrance",
        "lat": "3.5400620000000000",
        "lng": "103.4300950000000000",
        "radius": 10,
        "index": 2,
        "enabled": false
    },
    "out_point": {
        "id": 3,
        "name": "Test UMP Back Gate Entrance",
        "lat": "3.5394990000000000",
        "lng": "103.4298839000000000",
        "radius": 10,
        "index": 3,
        "enabled": false
    },
    "charge": 250,
    "vehicle": 1
},
"created": "2021-12-05T12:25:28.340592+08:00",
"modified": "2021-12-05T12:25:28.340592+08:00",
"plate_number": "ABD 123",
        "charge": 100,
        "toll_user": 1
    },
```

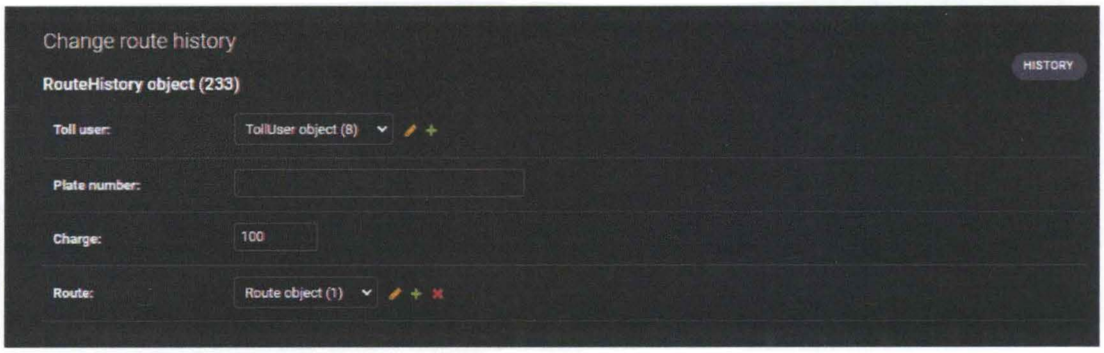
The above RouteHistory model would create a database table like this:



Select route history to change

Action: ——— Go 0 of 100 selected

| ROUTE HISTORY |
|---|
| RouteHistory object (233) |
| RouteHistory object (232) |
| RouteHistory object (231) |
| RouteHistory object (230) |
| RouteHistory object (229) |
| RouteHistory object (228) |



Change route history

RouteHistory object (233)

Toll user: TollUser object (8)

Plate number:

Charge: 100

Route: Route object (1)

Route History model is use to trace the route user has by pass and the amount of toll fee user paid for both toll user and toll admin.

### 4.3.2.4 E-wallet

This model defines a E-wallet, which has a toll_user, and balance:

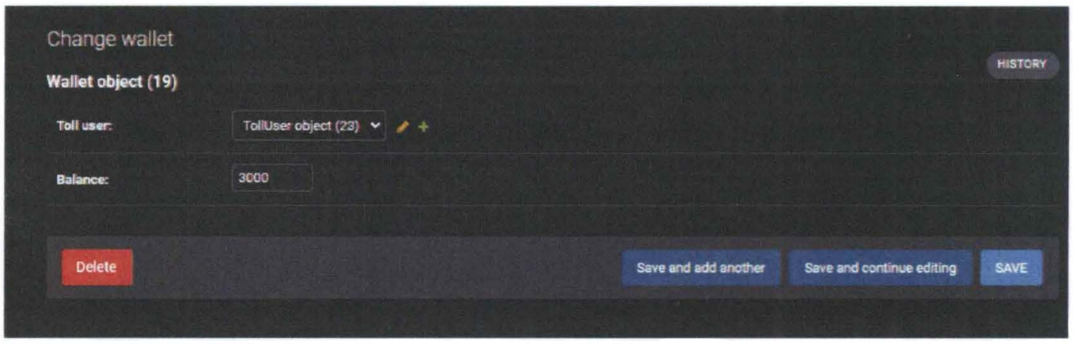
```
class Wallet(models.Model):
    toll_user = models.OneToOneField('toll.TollUser',
on_delete=models.CASCADE)
```

```
    balance = models.PositiveIntegerField(default=0)
```

toll_user and balance are fields of the model. Each field is specified as a class attribute, and each attribute maps to a database column:

```
{
        "id": 1,
        "balance": 354560,
        "toll_user": 1
    },
```

The above Wallet model would create a database table like this:


Change wallet

Wallet object (19)

Toll user: TollUser object (23)

Balance: 3000

This database show how much balance user have in their E-wallet.

#### 4.3.2.5 Topup

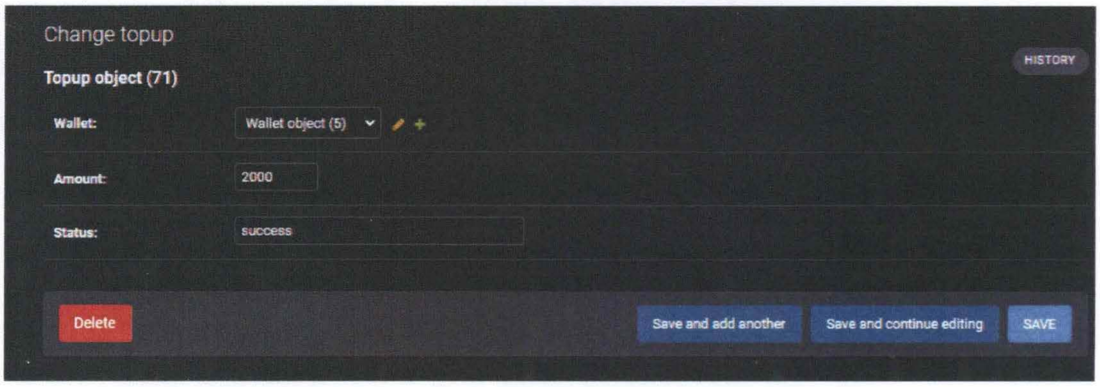
This model defines a Topup, which is related to Wallet model it has wallet, amount and status:

```
class Topup(TimeStampedModel):
    wallet = models.ForeignKey('Wallet', related_name='topups',
    on_delete=models.CASCADE)
    amount = models.PositiveIntegerField()
    status = models.CharField(max_length=16, default='success')
```

wallet, amount and status are fields of the model. Each field is specified as a class attribute, and each attribute maps to a database column:

```
        "id": 1,
        "created": "2021-11-21T18:12:25.142167+08:00",
        "modified": "2021-11-21T18:12:25.142167+08:00",
        "amount": 1000,
        "status": "success",
        "wallet": 1
    },
```

The above Topup model would create a database table like this:



Change topup

Topup object (71)

|  |  |
|---|---|
| Wallet: | Wallet object (5) |
| Amount: | 2000 |
| Status: | success |

This database show how much amount that user has reload and a status of reload either its “success” or “fail”.

In topup viewset as shown below:

```
user_id = post_data.get('user_id')
        amount = post_data.get('amount')
        wallet_user = Wallet.objects.filter(toll_user_id=user_id).first()
        wallet_user.balance = wallet_user.balance + amount
        wallet_user.save()

        obj = Topup(wallet_id=wallet_user.id, amount=amount)
        obj.save()

        resp = {
            "amount": wallet_user.balance,
        }
```

We define the equation:

Wallet = balance + amount

### 4.3.3 Django Serializers

Complex data, such as querysets and model instances, can be translated to native Python datatypes and then rendered into JSON, XML, or other content types using serializers. After validating the incoming data, serializers also enable deserialization, which allows parsed data to be transformed back into complicated types.

The REST framework's serializers are extremely similar to Django's Form and ModelForm classes. We include a Serializer class that provides a strong, generic approach to manage the output of your replies, as well as a ModelSerializer class that provides a convenient shortcut for serialising model instances and querysets.

defining a serializer for serialising and deserializing data corresponding to Comment objects. Declaring a serializer resembles defining a form:

```
from rest_framework import serializers
from .models import  TollUser, TollPoint, Route, RouteHistory
from wallet.serializers import WalletSerializer
```

### 4.3.4 Django Viewsets

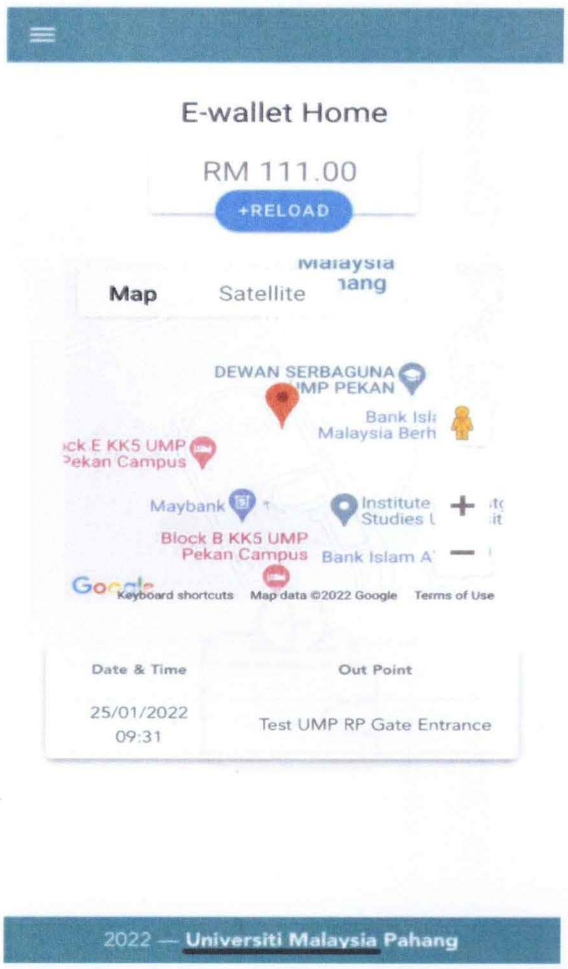
The ViewSet class in the Django REST framework allows you to aggregate the logic for a group of related views into a single class. Other frameworks may include implementations called 'Resources' or 'Controllers' that are essentially similar. A ViewSet class is a form of class-based View that doesn't have any method handlers like .get() or .post(), but instead has actions like .list() and .create().

The ViewSet's method handlers are only tied to the correct actions when the view is finalised with the.as_view() method. Rather than registering the views in a viewset explicitly in the urlconf, you'll usually register the viewset with a router class, which will automatically generate the urlconf for you.

## 4.4 **E-wallet System**

E-wallet is a form of electronic card that may be used to make online payment using a computer or smartphone. It functions similarly to a credit or debit card. To make payments, an E-wallet must be linked to the user's bank account. An e-wallet is a form of pre-paid account that allows a user to save money for future online transactions. A password is used to secure an E-wallet. An E-wallet can be used to pay for groceries, online purchases, and plane tickets, among other things.

E-wallet in GPS Expressway Tolling System application has two fundamental components which is software and information of an e-wallet. Personal information is stored in the software component, which also provides data security and encryption. The information component is a database of the user's information, such as their name, phone number, email, gender, payment method and payment amount. To make a payment toll user need to go to E-wallet page as shown in Figure 4.4.1



| Date & Time | Out Point |
|---|---|
| 25/01/2022 09:31 | Test UMP RP Gate Entrance |

**Figure 4.3: E-Wallet Page**

Figure 4.4.1 display the amount of your balance in E-wallet. If user want to reload their E-wallet user can press the reload button it will bring user to topup page as in Figure 4.4.2

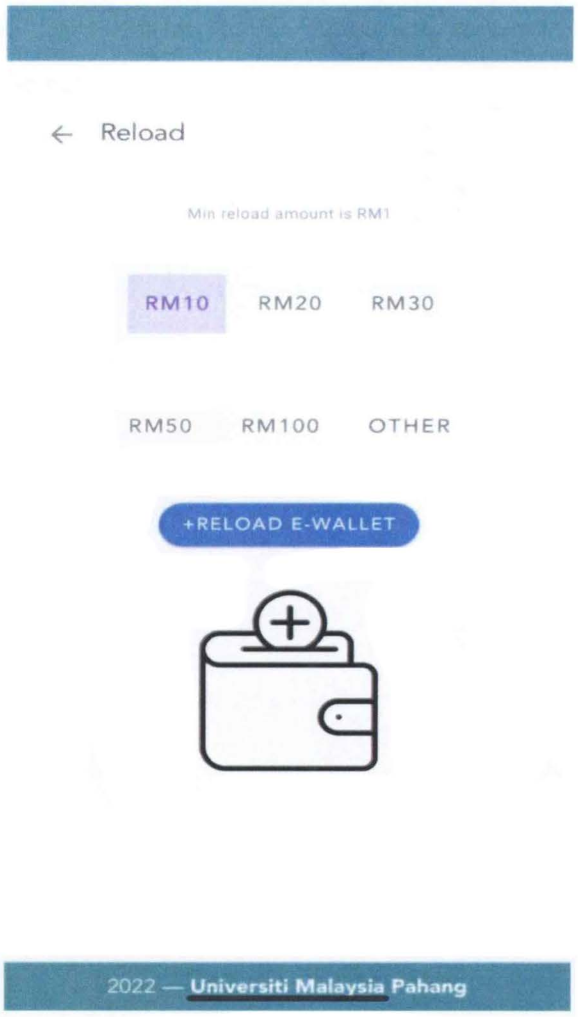


**Figure 4.4: Top-up Page**

In Topup page as in Figure 4.4.2 user can choose the amount they want to reload with the quick button or chose either button to enter the amount they want to reload. After choose the amount user want to reload to their E-wallet and press reload E-wallet button, user will get notification the current balance in their E-wallet as shown in figure-

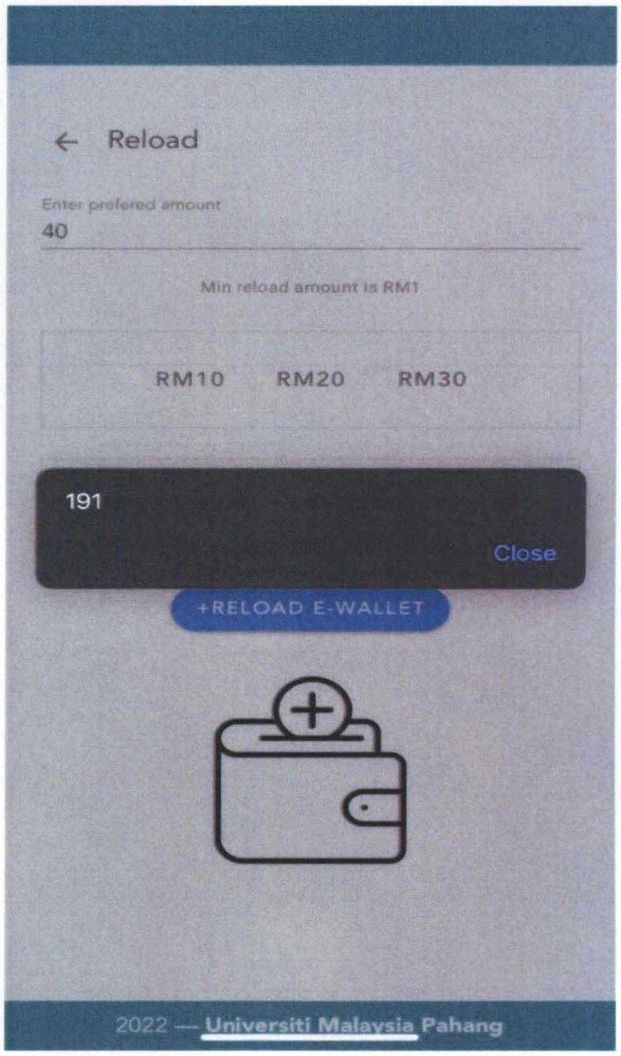


Figure 4.5: Top-up success notification

By this user has success reload their E-wallet. The development of this app only cost rm4 since only server that we need to pay the other software is an open source software.

# CHAPTER 5

# CONCLUSION AND RECOMENDATION

## 5.1 Introduction

The overall findings and results of the research, as well as the study's findings, will be concluded in this chapter. Other than the conclusion of the general data from the previous chapter, provide some recommendations or propose actions for improving the system. The constraints and obstacles encountered during the research process were also discussed, as well as suggestions for future research.

## 5.2 Conclusion

In summary, the SDP project intends to improve Malaysia's tolling system by implementing the GPS Expressway Tolling System, decreasing human resources, reducing traffic congestion, and promoting a modern tolling system that outperforms previous systems. Finally, be able to create and manufacture a simple and accurate GPS Expressway Tolling system for road users. Then inside the application, the database has construct to store and organize data for the GPS Expressway Tolling System application. the database played a significant role since every page in the application was binding to the database to pull and push the data from the API framework. Moreover, the e-wallet system in GPS Expressway Tolling System application is crucial for alerting customers to the amount of money they must pay, the level of money in their e-wallet, and when money is removed from their credit. With the development of a GPS expressway tolling system capable of replacing the current tolling system, limiting the number of human resources working at toll stations, and using GPS tracking of every vehicle that enters and exits the toll, the number of human resources working at toll stations will be reduced. Encourage the use of e-wallets in tolling systems to make it easier for end users and reduce congestion at toll stations, resulting in smooth driving without the stress of toll payment. To ensure the system more secured, we can install camera sensor to detect number plate of cars that go through the toll gate. If the cars did not turn on the apps, we can detect the car's number plates through the database.

## REFERENCES

J. Chou, L. Chen, H. Ding, J. Tu and B. Xu, "A Method of Optimizing Django Based on Greedy Strategy," 2013 10th Web Information System and Application Conference, 2013, pp. 176-179, doi: 10.1109/WISA.2013.41

S. M. Alzahrani, "Sensing for the Internet of Things and Its Applications," in 5th International Conference on Future Internet of Things and Cloud Workshops, 2017.

Me, G. 2003. Payment security in mobile environment. Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, Tunis, Tunisia, July 14–18.

Hilavuo, S. 2005. Business evolution of mobile services. Managing mobile services-technologies and business practices, Chichester. 17-45.

Misra, S.a.W., N. . 2004. Security of a mobile transaction: A trust model. Electronic Commerce Research. 4, 4, 359-372.

Yusop, N., Kamalrudin, M., Yusof, M. M., & Sidek, S. (2016). Meeting Real Challenges in Eliciting Security Attributes for Mobile Application Development. Journal of Internet Computing and Services, 17(5), 25–32. https://doi.org/10.7472/jksii.2016.17.5.25

Ahmad, A., Li, K., Feng, C., Asim, S. M., Yousif, A., & Ge, S. (2018). An Empirical Study of Investigating Mobile Applications Development Challenges. IEEE Access, 6, 17711–17728. https://doi.org/10.1109/ACCESS.2018.2818724

CHAN, J. A. D. E. (2017, April 27). *PLUS highway Toll.* The Star Malaysia. Retrieved January 31, 2022, from https://www.pressreader.com/malaysia/the-star-malaysia/20170427/281663959899443

## APPENDICES

Appendix A: Programming code for Toll model.py file to build database

```
from django.db import models
from model_utils.models import TimeStampedModel

# Create your models here.

class TollUser(TimeStampedModel):

    GENDER_CHOICES = (('M', 'Male'),
                      ('F', 'Female'))

    full_name = models.CharField(max_length=64)
    phone = models.CharField(max_length=64, unique=True)
    ic_number = models.CharField(max_length=14)
    gender = models.CharField(max_length=1, choices=GENDER_CHOICES,
default='M')
    vehicle = models.ForeignKey('Vehicle', related_name='toll_users',
on_delete=models.PROTECT)
    pin = models.CharField(max_length=6, default='123456')

    location_lat = models.DecimalField(max_digits=22, decimal_places=16,
blank=True, null=True)
    location_lng = models.DecimalField(max_digits=22, decimal_places=16,
blank=True, null=True)
    location_updated_datetime = models.DateTimeField(null=True,
blank=True)


class TollPoint(models.Model):
    name = models.CharField(max_length=64)
    lat = models.DecimalField(max_digits=22, decimal_places=16,
blank=True, null=True)
    lng = models.DecimalField(max_digits=22, decimal_places=16,
blank=True, null=True)
```

```
    radius = models.PositiveIntegerField(default=50) # metre

    index = models.PositiveIntegerField(default=0)
    enabled = models.BooleanField(default=True)

    def __str__(self):
        return self.name

class Vehicle(models.Model):
    class_name = models.CharField(max_length=10)

    def __str__(self):
        return self.class_name

class Route(models.Model):
    in_point = models.ForeignKey('TollPoint', related_name='routes_in',
on_delete=models.CASCADE)
    out_point = models.ForeignKey('TollPoint', related_name='routes_out',
on_delete=models.CASCADE)
    vehicle = models.ForeignKey('Vehicle', related_name='routes',
on_delete=models.CASCADE)
    charge = models.PositiveIntegerField()

    class Meta:
        constraints = [
            models.UniqueConstraint(fields = ['in_point_id',
'out_point_id','vehicle_id'], name='unique in_point, out_point, vehicle')
        ]


class RouteHistory(TimeStampedModel):
    toll_user = models.ForeignKey('TollUser',
related_name='route_histories', on_delete=models.CASCADE)
    plate_number = models.CharField(max_length=10)
    charge = models.PositiveIntegerField(default=0)
    route = models.ForeignKey('route', related_name='route_histories',
on_delete=models.PROTECT)
    # in_date ..
    # ...
    class Meta:
        verbose_name_plural = "Route Histories"
```

Appendix B: Programming code for Wallet model.py file to build database

```
from django.db import models
from model_utils.models import TimeStampedModel

# Create your models here.
class Wallet(models.Model):
    toll_user = models.OneToOneField('toll.TollUser',
on_delete=models.CASCADE)
    balance = models.PositiveIntegerField(default=0)


class Topup(TimeStampedModel):
    #wallet_id = models.CharField('wallet.wallet_id',
on_delete=models.CASCADE)
    wallet = models.ForeignKey('Wallet', related_name='topups',
on_delete=models.CASCADE)
    amount = models.PositiveIntegerField()
    status = models.CharField(max_length=16, default='success')
```

## Appendix C: Programming code for Toll serializers.py file to build database

```
from rest_framework import serializers
from .models import  TollUser, TollPoint, Route, RouteHistory
from wallet.serializers import WalletSerializer

class TollUserSerializer(serializers.ModelSerializer):
    wallet = WalletSerializer(read_only=True)
    class Meta:
        model = TollUser
        fields = '__all__'

class TollPointSerializer(serializers.ModelSerializer):
    class Meta:
        model = TollPoint
        fields = '__all__'

class RouteSerializer(serializers.ModelSerializer):
    in_point = TollPointSerializer()
    out_point = TollPointSerializer()
    class Meta:
        model = Route
        fields = '__all__'

class RouteHistorySerializer(serializers.ModelSerializer):
    route = RouteSerializer()
    class Meta:
        model = RouteHistory
        fields = '__all__'

class WalletSerializer(serializers.ModelSerializer):
    class Meta:
        model = Wallet
        fields = '__all__'


class TopupSerializer(serializers.ModelSerializer):
    class Meta:
        model = Topup
        fields = '__all__'
```

## Appendix D: Programming code for Wallet serializers.py file to build database

```
from rest_framework import serializers
from .models import Wallet, Topup

class WalletSerializer(serializers.ModelSerializer):
    class Meta:
        model = Wallet
        fields = '__all__'


class TopupSerializer(serializers.ModelSerializer):
    class Meta:
        model = Topup
        fields = '__all__'
```

Appendix E: Programming code for Toll viewset.py file to build database

```
from datetime import date, datetime
from django.utils import timezone
from rest_framework import viewsets, response, status
from .models import TollUser, RouteHistory, Vehicle, Route, TollPoint
from wallet.models import Wallet
from .serializers import TollUserSerializer, RouteHistorySerializer,
TollPointSerializer

class TollUserViewSet(viewsets.ModelViewSet):
    """ """
    serializer_class = TollUserSerializer
    queryset = TollUser.objects.all()


class TollPointViewSet(viewsets.ModelViewSet):
    """ """
    serializer_class = TollPointSerializer
    queryset = TollPoint.objects.filter(enabled=True).all()


class LoginViewSet(viewsets.ViewSet):
    def create(self, request):
        post_data = request.data # Tarik data yang di POST

        phone = post_data.get('phone')
        pin = post_data.get('pin')

        user = TollUser.objects.filter(pin=pin, phone=phone).first()

        if user is None:
            return response.Response({'error': 'invalid account'},
status=400)

        serializer = TollUserSerializer(user)
        return response.Response(serializer.data)


class RouteHistoryViewSet(viewsets.ModelViewSet):
    """ """
    serializer_class = RouteHistorySerializer
    queryset = RouteHistory.objects.all()
    filterset_fields = ['toll_user']


class PayTollViewSet(viewsets.ViewSet):
    def create(self, request):
        post_data = request.data # Tarik data yang di POST

        user_id = post_data.get('user_id') # tarik user_id dari post_data
        in_point_id = post_data.get('in_point_id')
        out_point_id = post_data.get('out_point_id')
        vehicle = Vehicle.objects.filter(toll_users=user_id).first()
        #print(vehicle)
        # Tarik data dari model Route based on in/out point

        route = Route.objects.filter( in_point=in_point_id,
out_point=out_point_id, vehicle=vehicle).first()

        if route is None:
            return response.Response(data="Route not exist",
status=status.HTTP_400_BAD_REQUEST)

        # Prevent duplicate
        now = timezone.now().astimezone(timezone.get_default_timezone())
        prev = RouteHistory.objects.filter(toll_user_id=user_id,
route=route).order_by('-created').first()

        if prev and (now - prev.created).seconds < 10:
            resp = {
                "status": "KO",
                "message": "Duplicate"
            }
            return response.Response(resp)


        #print(route.charge)
        charge = route.charge

        # check wallet balance
        wallet_user = Wallet.objects.filter(toll_user_id=user_id).first()
        #print(wallet_user)
       # print(wallet_user.balance)

        if charge > wallet_user.balance:
            return response.Response(data="Balance not enough",
status=status.HTTP_400_BAD_REQUEST)

        wallet_user.balance = wallet_user.balance - charge
        wallet_user.save()
        obj = RouteHistory(
            toll_user_id=user_id,
            plate_number='',
            charge=charge,
            route=route)

        obj.save()

        resp = {
            "status": "OK",
            "message": f"Successfully paid RM {charge/100:.2f}"
        }
        return response.Response(resp)
```

Appendix F:    Programming code for Wallet viewset.py file to build database

```
from rest_framework import viewsets, response, status
from .models import Wallet, Topup
from .serializers import WalletSerializer, TopupSerializer

class WalletViewSet(viewsets.ModelViewSet):
    serializer_class = WalletSerializer
    queryset = Wallet.objects.all()
    #search_fields = ['code', 'description']


#class TopupViewSet(viewsets.ViewSet):
class TopupViewSet(viewsets.ModelViewSet):

    serializer_class = TopupSerializer
    queryset = Topup.objects.all()
    filter_fields = ['wallet']


    def create(self, request):
        post_data = request.data # Tarik data yang di POST

        user_id = post_data.get('user_id') # tarik user_id dari post_data
        amount = post_data.get('amount')

        wallet_user = Wallet.objects.filter(toll_user_id=user_id).first()
        wallet_user.balance = wallet_user.balance + amount
        wallet_user.save()

        obj = Topup(wallet_id=wallet_user.id, amount=amount)
        obj.save()

        resp = {
            "amount": wallet_user.balance,
        }

        return response.Response(resp)
```