# ECOMMERCE SHIPPING SOLUTION FOR MOBILE PLATFORM

BY

WONG JUN WEI

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfilment of the requirements
for the degree of
BACHELOR OF INFORMATION SYSTEMS (HONS) INFORMATION SYSTEMS ENGINEERING
Faculty of Information and Communication Technology
(Kampar Campus)

MAY 2020

UNIVERSITI TUNKU ABDUL RAHMAN

# REPORT STATUS DECLARATION FORM

**Title**: ECOMMERCE SHIPPING SOLUTION FOR MOBILE PLATFORM

**Academic Session**: MAY 2020

I WONG JUN WEI

**(CAPITAL LETTER)**

declare that I allow this Final Year Project Report to be kept in Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

**Address**:
37, Jalan Punai,
Taman Chaah Baru
85400, Chaah, Segamat, Johor.

Yeck Yin Ping
Supervisor's name

**Date**: 10/9/2020

**Date**: 10/9/2020

# ECOMMERCE SHIPPING SOLUTION FOR MOBILE PLATFORM

BY

WONG JUN WEI

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfilment of the requirements
for the degree of
BACHELOR OF INFORMATION SYSTEMS (HONS) INFORMATION SYSTEMS
ENGINEERING
Faculty of Information and Communication Technology
(Kampar Campus)

MAY 2020

# DECLARATION OF ORIGINALITY

I declare that this report entitled "**ECOMMERCE SHIPPING SOLUTION FOR MOBILE PLATFORM**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : ______________________________

Name : ______Wong Jun Wei________

Date : ___ 10/9/2020___________

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisors Mr. Yeck Yin Ping who has given me this bright opportunity to engage in E-commerce shipping solution for mobile application. The guidance, comments and suggestions given from you are invaluable for me.

Also, I would like to thanks my family for being support me and always encourage me. A million thanks to you all.

# ABSTRACT

This project is about an ecommerce shipping solution for mobile platform that aims to reduce the shipping fee charge and also decrease the time waiting for the buyer to receive their parcel. Nowadays, the innovation of the internet made everyone can become a seller on the internet. People starting to promote their products and sell it through a social networking platform because there is no any charging fee with using social networking platforms to sell an item. However, this way also brings a problem to the seller. They need to find a courier service company that offers a cheaper price to help them to ship the item for their buyer. Therefore, ecommerce shipping solution for mobile platform is planned to develop in order to help the online seller solve their problem. ecommerce shipping solution for mobile platform allows user to find out the nearby courier delivery companies that located around them. Besides, the estimated shipping fee and distance between courier delivery companies also will be listed together on the map to let the seller know which courier delivery company is the closest and provides the cheapest shipping price. Lastly, this project also helps the seller find the nearby active driver and send their parcel to the selected courier company

## TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| GPS | Global Positioning System |
| UPS | United Parcel Service |
| RAM | Random Access Memory |
| OS | Operating System |
| B2C | Business-to-Consumer |
| B2B | Business-to-Business |
| API | Application Programming Interface |
| RM | Ringgit Malaysia |

# Chapter 1 Introduction

## 1.1 Project Background

The innovation of e-commerce nowadays has totally changed the way how we trade. E-commerce is an activity of buying or selling products on online services over the Internet. Through e-commerce website, we can buy most of the things that we want. Not only that, we also may sell our product by using an e-commerce website. Today's e-commerce marketplace is no more just an online platform where people can buy or sell things but much more, from being hyperlocal to infusing artificial intelligence to offering even intangible goods and services like GST compliance or short-term loans, e-commerce marketplace has evolved rapidly (Garg, 2018). The effectiveness and efficiency of e-commerce could bring a huge advantage to benefit a business. Many of the merchants saw the new business opportunity from e-commerce and make them turn into online seller.

With the increase of online seller, problems such as price offer, time of delivery and rating comparison also come along with. The traditional delivery workflow begins with the parcel carrier collect all parcels from an e-commerce fulfilment centre. Then, the orders will be sorted at the carrier's local parcel handling facility and next in a regional hub. Lastly, assort the parcel again at the local or regional destination facility before final delivery to the customer. This multi-step process consumes highly of the cost and time, therefore increase the burden of the whole supply chain. Today, there are too many options out there for the seller to select using which courier delivery company to ship their parcel. The time takes to compare the shipping fees among courier delivery company and self-sending the parcel to the courier centre makes the buyer have to wait a longer time to receive their parcel. The seller is currently seeking a way to cut down the total cost for their products and also shipping faster to the customer.

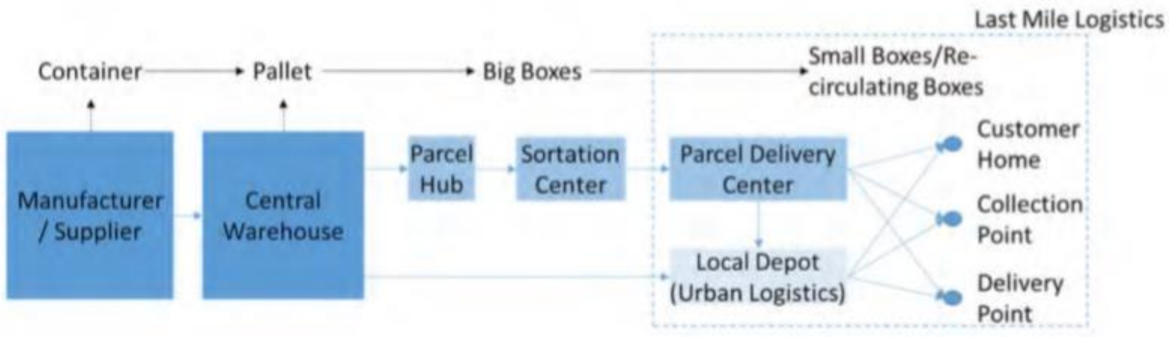


Figure 1.1 Sample of eCommerce logistics activity

The competitive of e-commerce shipping has become more intense in the e-commerce business. The large retailers such as Amazon, Shopee and Lazada making shopping faster and cheaper for their customer. The large retailers have the ability and power to make negotiation with the third party couriers company in order to get the discount price from them. But such way no suitable for small or medium size retailer because they are no able to do so. Thus, they need to use other strategies such as free shipping, fast delivery and lower price than the large retailer to compete with the large retailer. As the result, the business costs within the whole supply chain will increase and they did not gain many benefits from the business.

The problem of e-commerce shipping not only affect the suppliers but also the customers. The customers that live in rural areas will need to pay more charge fees when they purchasing online compare with urban areas. This is because the distance and time to deliver will be longer. The majority of the total charges their need to pay comes from transportation fees. Besides, this also caused the courier company unable to utilize their assets probably and lowering down their productivity. As the conclusion, the traditional delivery workflow affecting the whole supply chain and even consumer.

The reason of this problem is so important is to ensure all related party in e-commerce can gain the benefit. In this project, an ecommerce shipping solution for mobile platform was proposed to solve the problem. It is used to reduce the true business costs for supply chain and boost the delivery time of the customer.

## 1.2 Problem Statement

Many vendors use the courier delivery service provided by an e-commerce website to sell goods to their customer in order to run the online business. After reviewed the research paper and article, there was some problem on the courier delivery service that would bring a negative impact on consumer and vendor. Below are the list of the problems:

i. **Carrier-specific systems bring a loss of the money to vendor and customer**

   The e-commerce company will provide the courier delivery service to help their vendors deliver the parcel to customer. Vendors normally will choose one of the delivery company as their main shipping methods. The courier company will give a discount offer to vendor if they agree to sign a multiyear contract for cooperation. The vendor will potentially be locked into multiyear contracts and constantly using the same courier delivery company and unconsciously increased the cost of delivery (Moreno, et. al., 2017). The other courier delivery company might provide a lower shipping cost compared with currently used. Lack of courier delivery optional for the vendor will lead to inefficiently and resulting in the increasing of the shipping costs.

ii. **Low efficiency of shipping process increases the waiting time of customer**

   The customer needs to wait for a longer period to receive their parcel because the coverage of the pickup service offered by courier delivery company is limited. The courier service needs to make an improvement to increase infrastructure for innovation (Hasnan et al., 2014). If a vendor is located in the area which does not have a pickup service especially in rural areas, they have to send the parcel to the courier delivery company by themself. The vendor will not send the parcel one by one to increase their business cost. Thus, they will wait for the order to reach a certain amount then send it to the courier delivery company

iii. **Increase true business costs within the whole supply chain to meet customer expectation**

Most of the customer expectation on delivery service is fast and free delivery costs. To meet the demands, vendor will select the courier delivery company that can provide a faster delivery service to ship the parcel. As the result, the vendor may add the delivery costs into the total costs that consumer need to pay or they would shoulder the cost if customers are less willing to foot a delivery fee (Battan, et. al., 2016).

## 1.3 Project Objectives

The following are the objectives of this project:

- To provide a GPS search function to find out the nearby courier delivery companies around the vendor location and suggest it to the vendor. The vendor will able to choose the nearest and cheaper price offer courier delivery company to ship the parcel for the purpose of offering the lower shipping costs to the customer. This is because the nearest centre will have a shorter delivery distance and lesser transport charge compare with others.

- To develop an ecommerce shipping solution for mobile platform that allows vendor can have multiple options on choosing a courier delivery company and request a driver sends their parcel to the selected courier delivery company. The delivery fee will be count using the distance of the vendor location and selected courier delivery company.

- To provide a price and distance estimation framework to help vendor evaluate which courier delivery company is the best choice compare with other courier delivery companies to ship the parcel. The vendor will able to study the estimated charging price offer and the distance between courier delivery companies to decide whether which courier can bring the most benefit to them.

## 1.4 Project Scopes

This project is to develop an ecommerce shipping solution for mobile platform to benefit the vendor and customer. The application will allow vendors to have multiple options on selecting the courier delivery company to ship the parcel. The application will cooperate with multiple courier delivery company to help the vendor to ship their parcel to the customer.

This application will be built in a global positioning system (GPS) function to find out all the nearby courier delivery companies around the vendor location. The courier delivery companies location will show on the map and the shipping price and distance to reach will be included as well. Therefore, the vendor can evaluate which courier delivery companies will bring the most benefits to them.

Besides showing courier delivery companies location, the rating and the open status of them also will be attached together. Lastly, the application will help vendors to find a driver to send their parcel to the selected courier delivery company with extremely low route fare charging to reduce the burden of the vendor.

## 1.5 Contributions

The development of this project will impact and contribute to the e-commerce vendors and also its clients. It will bring the benefit to the vendor to lower down the delivery fee charge and make the delivery timeline become shorter as well as meet the customer expectation.

The high reliability of this software will help e-commerce business growth because vendor can request driver helps them to send the parcel and resulting decrease burden and increase the sales of business. The customer will found out the e-commerce business that using this software will provide a lower total cost need to pay and receiving their parcel faster. The customer will become loyally to the company when they feel treated nicely.

Moreover, it also helps the driver users of this application to optimum utilization of transportation use. The person who has a vehicle and valid license can apply to become the driver of the application. It helps them to gain extra income when they have free time to work with.

# Chapter 2 Literature Review

## 2.1 Shipping As Strategy: How Small And Midsize Retailers Can Best Meet Customers' Delivery Expectations In The Age Of Amazon

### 2.1.1 Summary

This white paper discusses the problem rising expectation of customer around shipping for small and midsize retailers and how it can be overcome and managed with shipping strategy. For instance, using automation and a mix of carriers. The U.S. domestic e-commerce marketplace is dominated by Amazon and large retailers (Brendish, et. al., 2017). The clout of Amazon and large retailer has increased the customer expectation of shorter delivery time. To compete with them, small and midsize retailers have to rely heavily upon commercial carriers such as UPS and FedEx to speed up the delivery time. This had made retailers are likely left paying more than necessary, and potentially locked into multiyear contracts that will escalate shipping costs if target volumes are not met, even for periods as short as one week (Brendish, et. al., 2017). If the retailers continuously using the same parcel shipping system, they will lose money in their business by failing to use the multiple carriers as their competitive advantage. To solve the problem, the retailer should use an automated multicarrier system as their strategy in e-business to make sure their profit.

### 2.1.2 Strength

The automated multi-carrier system has the advantage of comparing the strength and skills of each carrier. It can evaluate which carrier offers a better service, price rates as well as the delivery times at different areas. The system is served by software that enables merchants to access all those carriers from one system. It can import orders from e-commerce sales channel into view and simplifies the decision making such as order and shipping labels. As the result, the merchant can finish the order faster and shipping the parcel with cost-effectively and also meeting the customers' expectation.

### 2.1.3 Weakness

The automated multi-carrier system needs to provide an equivalence service or a better consumer experience that similar to the large retailer. Besides, it also needs a reliable platform to manage all of the process such as the system supports intelligent management and automate order fulfilment. The retailer also needs to have pay for the system fees and maintenance fees.

### 2.1.4 Recommendation

The merchants should look for a reliable third-party shipping platform that can provide additional guarantees, discounts and better services and partnering with them. For example, the merchants can use the application that is easy to work with and favourable pricing and payment terms to save the cost of business.

## 2.2 The 21st Century Spice Trade

### 2.2.1 Summary

This research paper discusses the opportunity in cross-border e-commerce and provides a guideline for retailers and manufacturers around the world to help them succeed in the e-commerce business. The innovation of e-commerce changed the way of trade between people. Not only that, it is still growing rapidly becoming more convenience to fulfil the consumer demand. When consumers are comfortable with purchasing through e-commerce, they will become loyally and increasing the usage of e-commerce. E-commerce has totally changed the parcel and express industry ever since. It is shifting from business to business (B2B) focus to business to customer (B2C). Moreover, e-commerce business not just only can run on a local country, it is belonging globally business. With this huge demand, e-commerce business faces another new challenge. They have to solve the problem of cost, convenience, volume and value, fast movement and constant improvement. Thus, DHL Express provides Premium logistics services for those retailers and manufacturers. Service Premium is the ideal solution to reduce international shipping transit times. It ensures particularly short and calculable delivery times, including transport insurance, for international shipments to business (B2B) or private (B2C) recipients (DHLde ,2019). It can solve the e-commerce business with having international trade problem and provides high-speed time-definite international solutions.

### 2.2.2 Strength

Premium logistics services can speed up the parcel delivery process that waiting for slow-turning in inventory in order to fulfil consumers' demand for faster delivery. The reliability of Premium shipping is high because of suitable for all types of online seller including small and medium Enterprises. Moreover, online seller which offering the premium shipping will have more opportunity to grow faster compared with other. It helps small manufacturers to compensate for parts of the natural disadvantages compared to large manufacturers such as provide faster delivery. This is because larger manufacturers having generally a higher international brand strength, more resources, and valuable knowledge on further building their cross-border business.

### 2.2.3 Weakness

The consumer needs to pay an extra surcharge in order to enjoy the premium shipping delivery. It does not meet the expectation of consumer and they will seek another way to receive their parcel. Besides, e-commerce Company such as Amazon have set some requirement for online seller before enabling them to use the premium shipping delivery services. For example, online seller must achieve valid tracking rate of 100% and on-time delivery rate of 97% or greater (Amazon 2019).

### 2.2.4 Recommendation

E-commerce company may abolish the achievement require for e-tailor to use the premium shipping service. This can increase more online seller to use e-commerce site to sell their product and achieve both win-win situations for e-commerce Company and online seller. The e-commerce company will have an increase in profitability and the online seller also fulfil the customer needs and gain the sales profit.

## 2.3 Parcel delivery The future of last mile

### 2.3.1 Summary

The extensive popularization and application of the Internet bring up the growth of electronic communication and digital information. The E-commerce business automatically becomes the first choice of the merchant. However, the last-mile delivery is the problem that always troubled them to gain their competitive advantage. In order to maintain their business, some of the merchants often shoulder significant labour cost disadvantages that cause by last-mile delivery. Thus, this paper proposes the new technology that will implement in the coming next ten years to replace the traditional delivery method which is called autonomous delivery models. It will develop the new automated vehicles such as autonomous aircraft which is called as Drones, autonomous ground vehicles and use it to deliver the parcel automated. The Autonomous vehicles including drones will able to deliver close to 80 percent of all items and only a few items will be delivered by the bike couriers to the relative small instant delivery segment. The remaining parcel will be assigned to use the traditional delivery method.

### 2.3.2 Strength

The new autonomous delivery models will have a cost advantage compare with today's conventional last-mile delivery. This is because it cut off a lot of the cost of the labour force and replace with an automated vehicles. Besides, it can also enable service providers to create superior value for customers and make the profit with additional fees from new services such as overnight pickup and Sunday delivery services. This additional services not only bring the profit and also help the parcel service providers save the high real estate cost that use to store parcels.

### 2.3.3 Weakness

There will be much costly to offer delivery within a specified time window or on the same day with any kind of driving vehicle due to the large distances that need to be covered as especially in rural areas. Although the new idea of automated vehicles can offer a solution to this problem, it will derivative new problems that need to solve. This is because there is a weight limit for automated vehicles and also there will be hard to find a suitable landing site in urban areas.

### 2.3.4 Recommendation

The new autonomous delivery models might first focus in the developed countries at early adoption to ensure the return of investment of this project and observe the reliability. Besides, they will need to implement a fully-fledged parcel network that allows for high degree of consolidation to realize the project.

## 2.4 E-Commerce Trends And Challenges: A Logistics And Supply Chain Perspective

### 2.4.1 Summary

As the E-commerce business is growing fast nowadays, varying of new business models also emerged in order to satisfy the customer needs. Most of the models are focusing on introducing new intermediaries and accelerate the arrival time of a parcel. In fact, such way will rebound and increase true business costs for other partners or within the whole supply chain. This is because it is not only implicated removing the time and cost and also consists of increase responsiveness and efficiency of the whole supply chain. Poor transport infrastructure, lack of warehouse readiness and inefficient last mile delivery are the root causes that decrease the logistics performance. Thus, this project proposed a concept of E-Commerce Logistics Management (ELM) to tackle last-mile logistics inefficiency. It aims to work with different stakeholders to collaborate for cost-effective deliveries. It covers the process from the time the order is made by the customer until the delivery is received by the customer.

### 2.4.2 Strength

E-Commerce Logistics Management can benefit not only customers but also for companies that serve the e-commerce last mile deliveries. It applies the consolidation concept and achieves the multiple deliveries at the same time to optimum utilization of transportation use and lead to a decrease in the cost of transportation. Besides, it also adopts dynamic delivery scheduling and real-time tracking to optimize last-mile e-commerce deliveries. The dynamic scheduling and routing can optimize the fleet travel time, minimize traffic congestion and reduce the logistics cost.

### 2.4.3 Weakness

With using the consolidation shipping concept, the customer lives in rural areas might need to pay an extra cost to third-party transporters to receive their parcel. This is because some of the pick-up points are far away with the customer’s address or hard to reach and the transportation company will unload the goods in one of its warehouses that nearby the customer's address. Besides, the complexity of consolidation shipments might not be accepted by all of the carriers and they more prefer using the traditional way to run their business.

### 2.4.4 Recommendation

Before carrying out a consolidated shipment, the entire process should be plan appropriately. Having a complete proposal can help to convince the courier company and get their co-partnership. After that, be sure the charge rates are being well informed to make sure the benefit get.

## 2.5 Automating e-Commerce from Click to Pick to Door

### 2.5.1 Summary

Technology is driving changes in the e-commerce business and it also impacts the way we shop. The e-commerce landscape has been changing throughout the years. This report paper discussed the problems and effects on automation driven by e-commerce for physical goods in the coming future. The invention of automation technology over the forthcoming decades will directly impact and threaten the workforce. The work in transportation, warehousing, and logistics are susceptible to automation. The automated order fulfilment, inventory management, and delivery when consumers shop online were predicted to implement and launch in the near future. It will have a high success possibility due to the growing fast of e-commerce. The automation technology not only beneficial to the merchant but also fulfil the needs and expectation of customers. This is because automation can help to reduce the time needed for a distribution centre to process an order, thereby allowing the merchant to offer and promise the shorter delivery times for their customer.

### 2.5.2 Strength

With using the automation technology, the warehousing can run the operation whole day long without interruption and improve productivity. The retailer is able to reduce the cost of hiring more staff that have limited work time. Besides, it also can eliminate the manual data entry between the systems and reduce the error information entry.

### 2.5.3 Weakness

Mechanical is the core item to process automation. Thus, mechanical failures will cause unpredictable effects on total processes such as a huge amount of money loss. Due to the increase in the productivity of e-commerce, there will have a need for increasing the warehouse to handle the huge amount of parcel and also need to look for warehouse locations which enjoy proximity to the customers themselves. This has led to affect the real estate market, as the desire for the location of the warehouses exceeds the land resources available.

### 2.5.4 Recommendation

The mechanical should involve proactive data monitoring and real-time processing system function to detect the error of the stock data. This will also help to ensure that will not happen a big loss of profit because it is protected by using syncs data function to make sure it is the same in the database.

## 2.6 Zone Skipping Strategies to Reduce E-Commerce Shipping Costs

### 2.6.1 Summary

With the world are going digitally, the number of e-commerce company is also increasing day by day. The parameters such as product variety, price and delivery options are the keys for them to compete with others. Nowadays, the customer is more preferable to use the e-commerce company which can provide a lower total delivered price, fast shipping and order accuracy as their first choice at online purchasing. All e-commerce company is seeking ways to reduce the shipping costs to gain profitability and customer retention. However, this is very hard to achieve due to the operators face the dimensional weight (DIM) and other pricing pressures from parcel carriers. Thus, this research paper purpose a sortation strategy is known as zone skipping allows fulfilment centres to assume some of the handling steps typically required of carriers in order to reduce shipping costs and increase delivery speed. With using zone skipping, the orders are pre-sorted at the e-commerce fulfilment centre according to final destination before being turned over to the carrier. It allows the orders to bypass parcel carrier sortation at local or regional hubs close to the fulfilment centre and instead be delivered closer to the customer address.

### 2.6.2 Strength

Zone skipping can reduce the burden on parcel carriers and boost the delivery speed by getting parcel as close to the final delivery address as fast as possible. Such a way also helps to reduce the total cost of the parcel.

### 2.6.3 Weakness

In order to implement the zone skipping, the fulfilment centre must contain a sufficient area to store and divert the large volume of parcels. Besides, the system design also cannot simply implement to all fulfilment centre. This is because different divert centre have a different maximum order size to handle parcels. For example, a larger parcel will not suit for a smaller divert centre.

### 2.6.4 Recommendation

The zone skipping may work together with third party parcel Carriers Company to ship the parcel. The shipper can consolidate the parcels and send straight to a local parcel carrier’s terminal for last-mile delivery.

## 2.7 Comparison of the features in table form

| Author | Characteristic | | | | | | |
|---|---|---|---|---|---|---|---|
| | Optimum Asset Utilization | Time Saving | Cost-effectively | Flexibility | Reliability | Heterogeneous Shipping | Availability |
| Moreno. H. S. et al. | | | ✓ | ✓ | | ✓ | ✓ |
| Allen. K | | | | ✓ | ✓ | ✓ | ✓ |
| Joerss. M. et al. | ✓ | ✓ | ✓ | | ✓ | | |
| Battan. S. et al. | ✓ | ✓ | ✓ | | ✓ | ✓ | |
| Frey. C. B. et al. | | ✓ | ✓ | ✓ | | | ✓ |
| Kraus. T | ✓ | ✓ | ✓ | | | | ✓ |
| eCommerce Shipping Solution for Mobile Platform | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 2.1 Table comparison of the feature in literature review

This project will include many characteristics to solve the problem statements. The characteristics will be use in this project are cost-effective, reliability and heterogeneous. With these characteristics, user can purchase online and pay with lower total cost and receive their parcel in shorter time. The addition features in this project are GPS Search and sorting framework. GPS is use for find out nearby courier deliver companies and price and estimation framework is use to help user to evaluate the price and distance of courier deliver company. Lastly is requesting a driver to help vendor send the parcel to the selected courier deliver company to reduce the burden of vendor.

# Chapter 3 System Design

## 3.1 Design Specification

### 3.1.1 Methodology and General Work Procedures

This project applies the agile method of the project management methodology to implement the system. The agile method is highly interactive and allows for rapid adjustments throughout a project. This method offers repeatable processes, lower risk, fast turnaround, allows instant feedback and reduces the complexity to deliver high-quality software in a small team. This project purpose to develop an ecommerce shipping solution for mobile platform to lower down the total cost of parcel shipping and reduce the delivery lead time. This project can solve the inefficiency of the traditional delivery workflow. The traditional method requires a multi-step process and it will accumulate the costs and time. With the assistant of ecommerce shipping solution for mobile platform, vendors allow cooperating with multiple couriers deliver companies, it uses GPS features to find out the nearby courier delivery companies, price and distance estimation framework to vendor evaluate the best fit courier delivery companies and also help vendor decrease the burden with using driver to help them ship their parcel to the selected couriers deliver company.

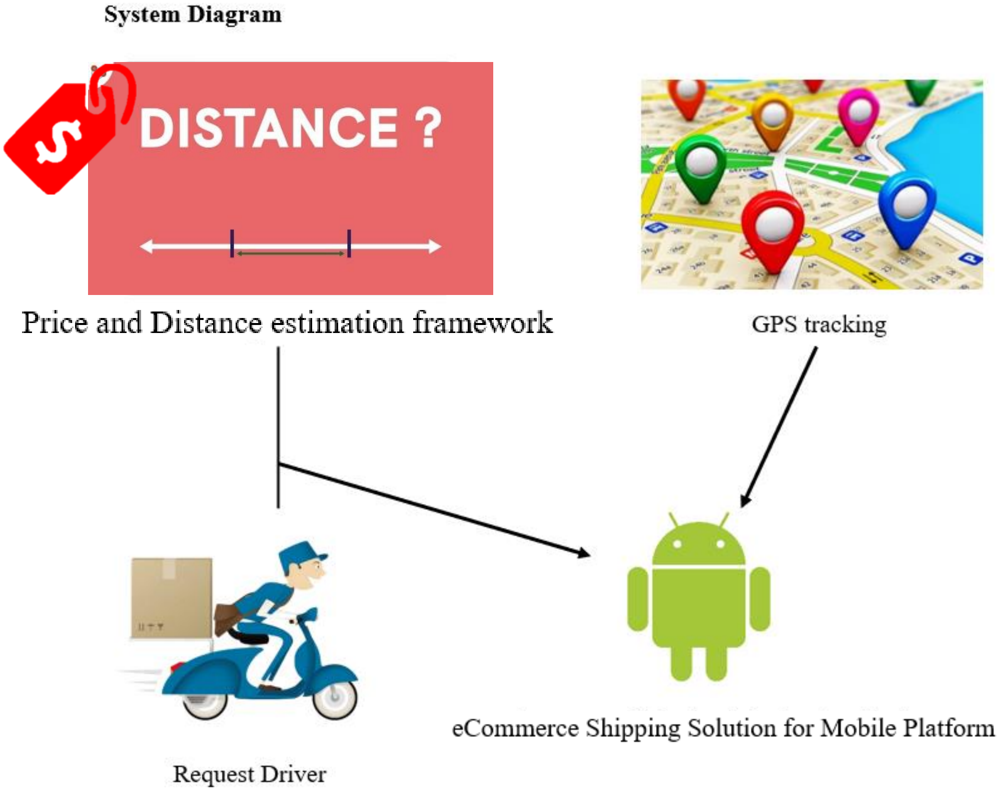

System Diagram
DISTANCE ?
Price and Distance estimation framework
GPS tracking
Request Driver
eCommerce Shipping Solution for Mobile Platform

Figure 3.1 Diagram of eCommerce Shipping Solution for Mobile Platform

In Figure 3.1, the ecommerce shipping solution for mobile platform will have the features such as delivery service, price and distance estimation framework and GPS search for multi couriers. Through the application, the vendor can have the multiple couriers' options to use to ship the parcel to their customers. It works together with the price and distance estimation framework and delivery service. The price and distance estimation framework help the user evaluate the fastest delivery distance and lowest price charge offered by the courier company. The delivery time is predicted by using the GPS feature. Thus, the user can choose the best-fit courier delivery company to ship their parcel based on their requirement.

**System Framework**

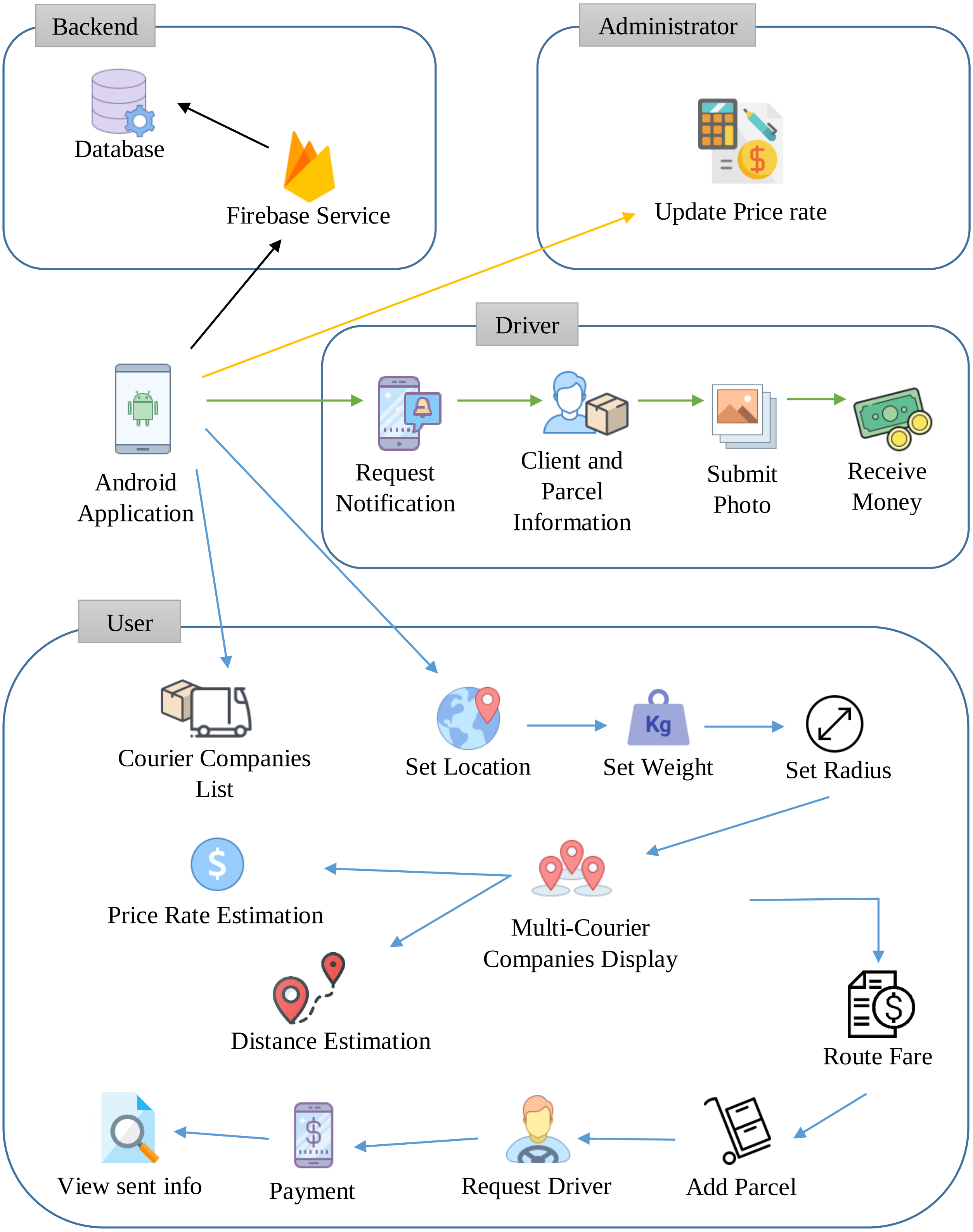


Figure 3.2 Framework in this system

Figure 3.2 contains 4 types of color lines and each of them represents different activities in this application. The black lines are the backend activity of the application. The application will access the database using Firebase Service. The data such as user information and courier price rate can retrieve and update through the application. The yellow line is the activity performed by the administrator. The administrator is responsible to update the price rate of the courier company listed in the application. The updated price rate will be store into the database.

The green lines are the activity for driver users. First, the driver will receive the nearby user request notification. Once the driver accepts the request, the client and parcel information will be shown to them and the information will lead the driver to reach a user's location. After receiving the parcel from the user, the driver needs to send the parcel to the courier company that the user requested. In order to finish the order, the driver is required to snap a photo and submit tracking number to prove that the parcel has successfully received by the courier company. The money will credit to the driver after they complete the order.

The blue lines are the activity for users. The user needs to set the weight of the parcel and also the radius to find the courier company around them. Then, the system will use the information set to find out multiple courier companies and show courier information such as company name, address, rating, distance and price offer to the user. The route fare will be calculated and display after selecting the courier company. The user needs to add parcels information before request a driver to come and collect the parcel. Lastly, user needs to pay the ride fare when the driver finish sent the parcel to the selected courier company in order to view the consignment info.

## 3.2 System Design / Overview

### 3.2.1 Use case diagram

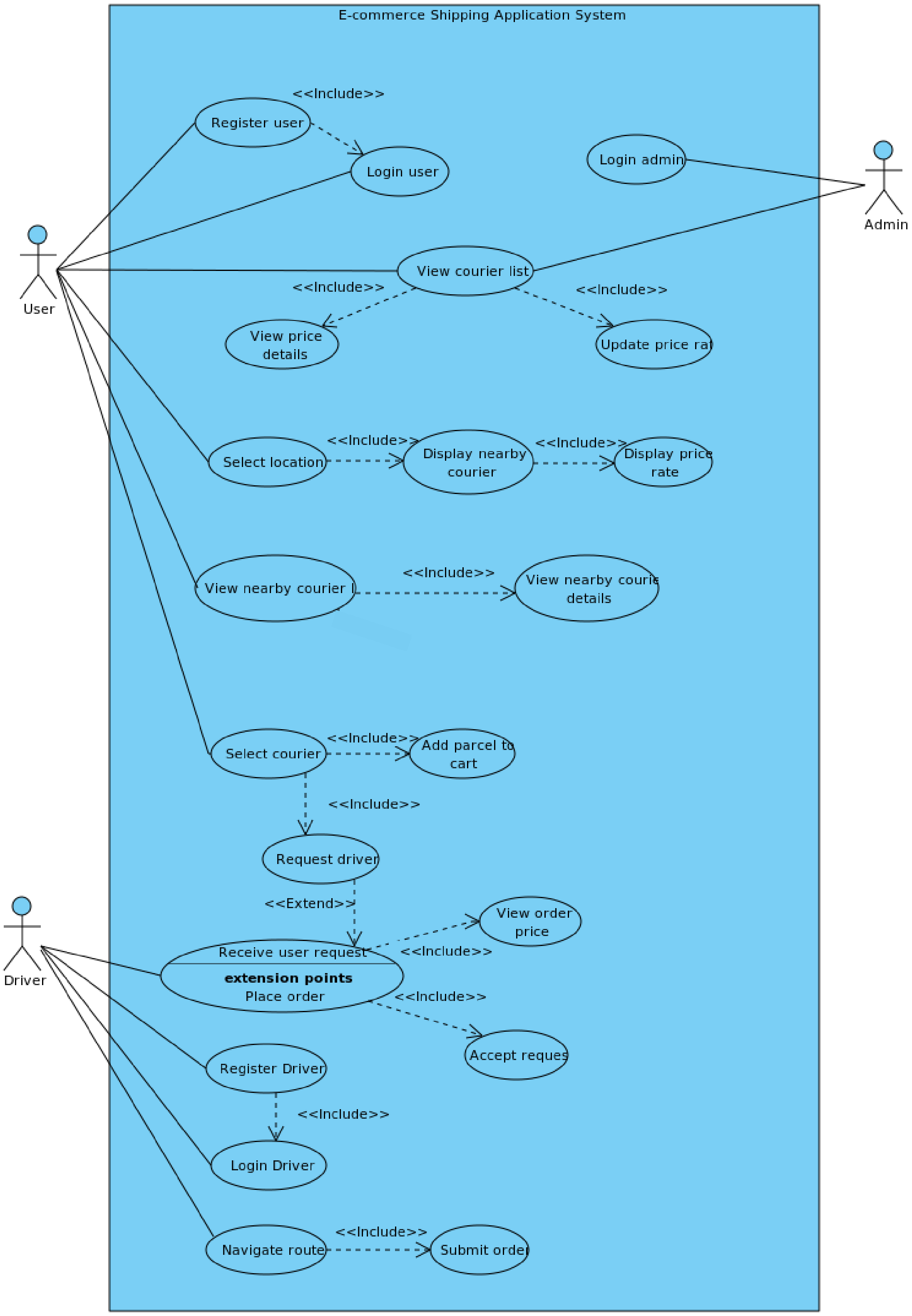


Figure 3.3 Use case diagram of eCommerce Shipping Solution for Mobile Platform

The use case diagram contains three actors which are User, Admin and Driver. All of them can login to the system. User and Driver can make a register to the system. Besides, User and Admin can get the courier companies list from the system. Moreover, User can select the location to find out the nearby courier companies and its price rate offer. The user also can view the estimated price and distance to decide which courier company to use before request the driver. The driver will receive a user request while a user placed the order. Once the driver accepts the order, they will get the navigation route from the system.

#### 3.2.1.1 Register

User and Driver need to make a registration to the system. If register as User, they have to submit the information such as phone number, username and password. If register as Driver, they will have to submit not only personal information like User but also their vehicle information such as plat number.

#### 3.2.1.2 Login

The application will authenticate the input data to identify the current user is under which category. The input data such as phone number and password are used for verifying the current user.

#### 3.2.1.3 View courier list

The application will show a listing of the courier companies and their price rate that stores inside the database. User can only view the courier companies and price rate but the Admin is allowed to make an update on the price rate.

#### 3.2.1.4 Select Location

This module is used to let the User specifying their current location, parcel weight and the radius to search the courier companies around them. The price offered by courier companies will be displayed when the result show.

### 3.2.1.5 View nearby courier list

This module will link with the courier companies that display on the map and show one by one in a card list. The information of courier companies such as address, rating, operating status, distance and estimate time to reach also will be shown together with it.

### 3.2.1.6 Select courier

This module will let the user select a courier company to ship their parcel. The parcel information also needs to add in before send a driver request.

### 3.2.1.7 Add parcel to cart

This module requires the user to fill-up the form parcel information such as the number of parcels, parcel's description and the size of the parcels. User can make a request to a driver only when completely fill-up the form.

### 3.2.1.8 Receive user request

Driver will receive an incoming user request when the user sends a driver request. It contains the user and parcel information such as the user's location, selected courier company address, number of parcels and route price.

### 3.2.1.9 Accept request

This module is used to let the driver accept or reject the user request. If the driver did not respond in a certain time to the request then it will automatically reject the request.

### 3.2.1.10 Navigate route

This module will display the address of the user location and also the address of the courier company that needs to be sent.

### 3.2.1.11 Submit order

The driver needs to upload the photo of the parcel recipe and submit in order to prove they have successfully sent the parcel to the courier company.

## 3.2.2 Activity Diagram

### 3.2.2.1 Activity Diagram for Login

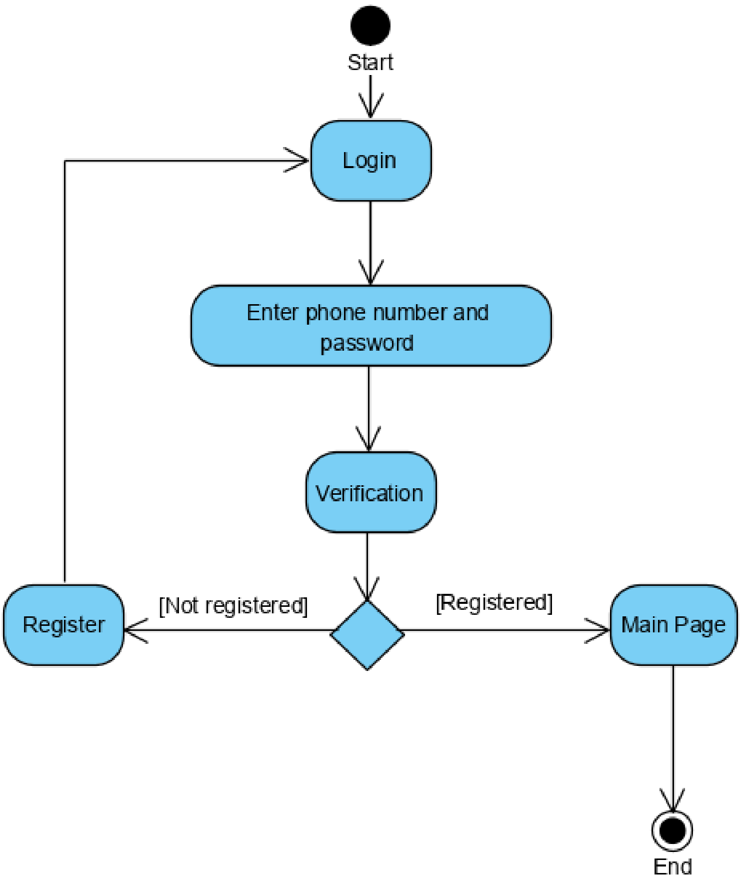


Figure 3.4 Activity diagram for login

In order to login to the system, a user needs fill-up the phone number and password they registered at the registration page. The system will verify the phone number and password enter are matched with the data at the database. The user is allowed to go to the main page of the system if the data is matched else the system will display an error message to let the user try again enter the correct phone number or password.

### 3.2.2.2 Activity Diagram for Register

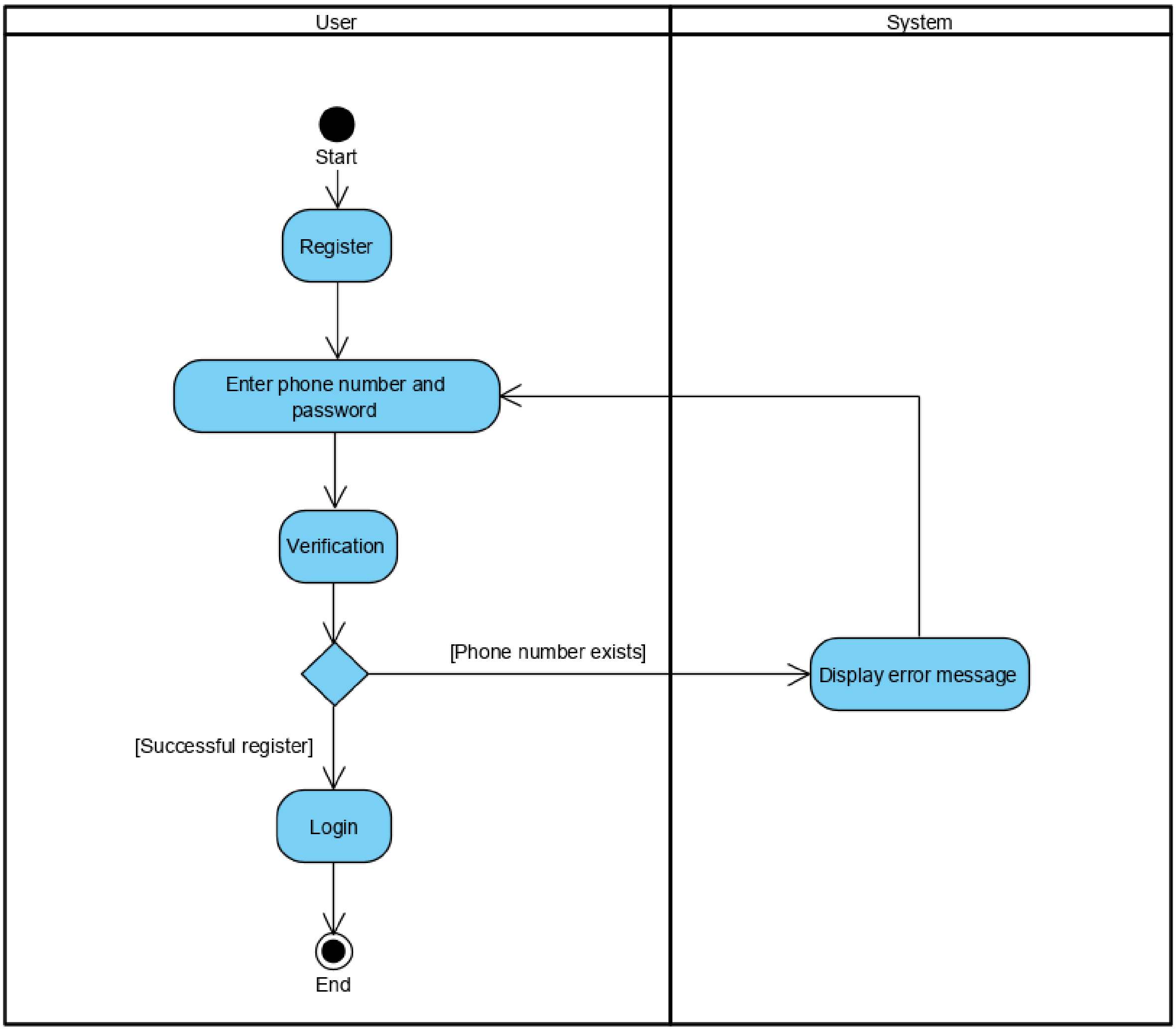


Figure 3.5 Activity Diagram for Register

At the registration page, a user is required to enter the phone number and password. The system will verify the phone number entered exists in the database or not. If the phone number exists in the database, the system will display an error message and the user needs to try with another phone number. The system will lead the user to the login page once the registration is successful.

### 3.2.2.3 Activity Diagram for Courier company list (user)

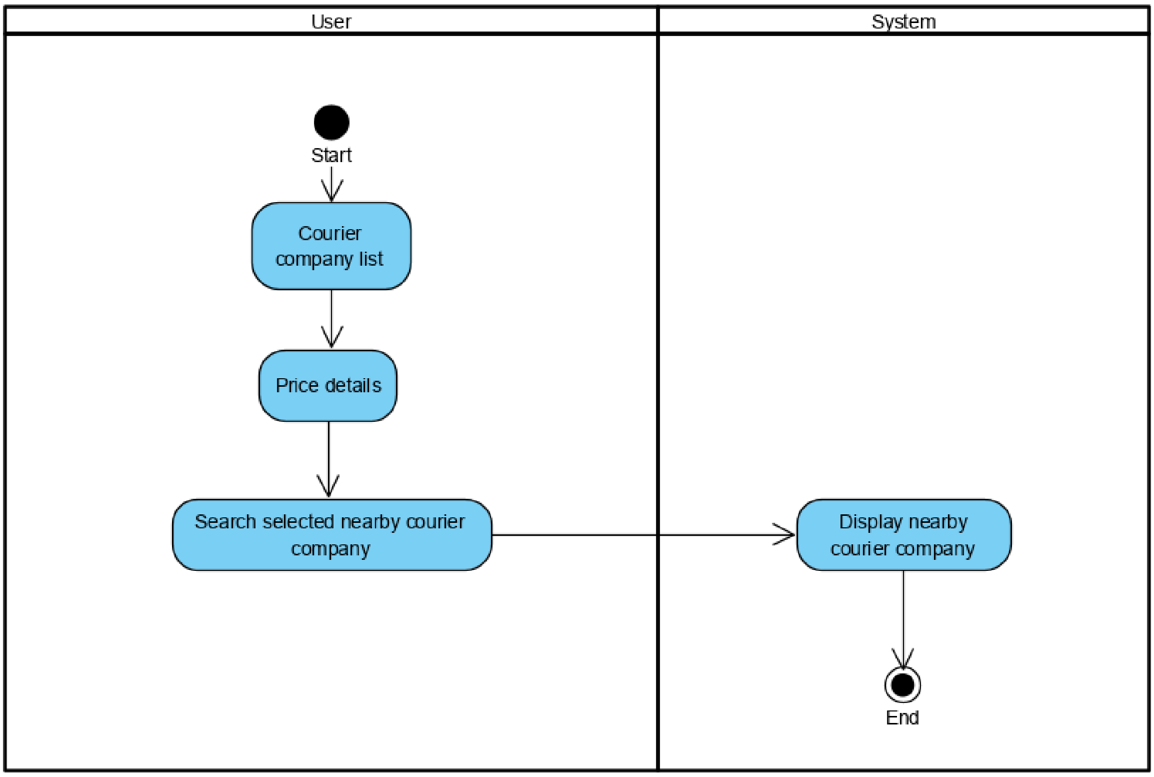

User
System
Start
Courier company list
Price details
Search selected nearby courier company
Display nearby courier company
End

Figure 3.6 Activity Diagram for Courier company list (user)

The system will display a courier company list data which retrieve from the database. The user is allowed to select a courier company from the list and view the latest price rate offer from them. Besides, the user also can directly make a nearby search of the selected courier company and show it on the map.

### 3.2.2.4 Activity Diagram for Courier company list (admin)

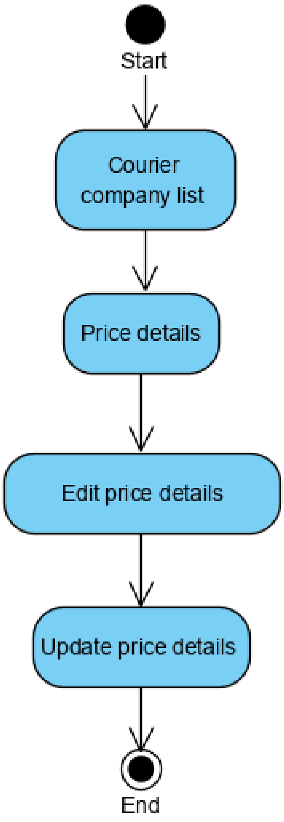


Figure 3.7 Activity Diagram for Courier company list (admin)

At first, the system will check the login account is a user account or admin account. If the login account is detected as the admin account, the admin is able to edit the price details of the courier company that in the list. After the admin is done editing the price details, they can click the update button to save the edited price to the database.

### 3.2.2.5 Activity Diagram for Select location

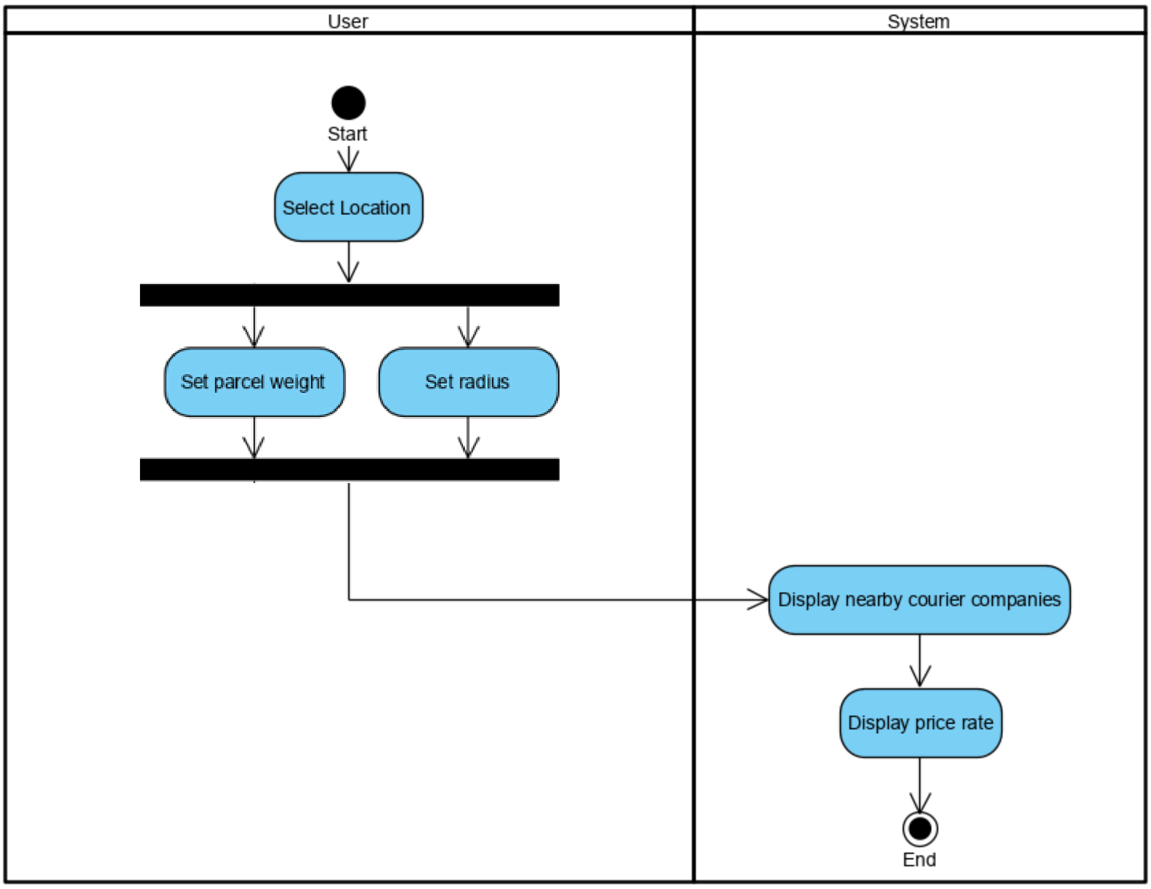


Figure 3.8 Activity Diagram for Select location

The system provides a form to let the user set their parcel weight and the radius of searching to find out the courier companies which meet the user request. The courier companies will be displayed on the map as well as showing the distance between and price offered by them. The price shown is calculated by using the price rate store in the database.

### 3.2.2.6 Activity Diagram for Select courier

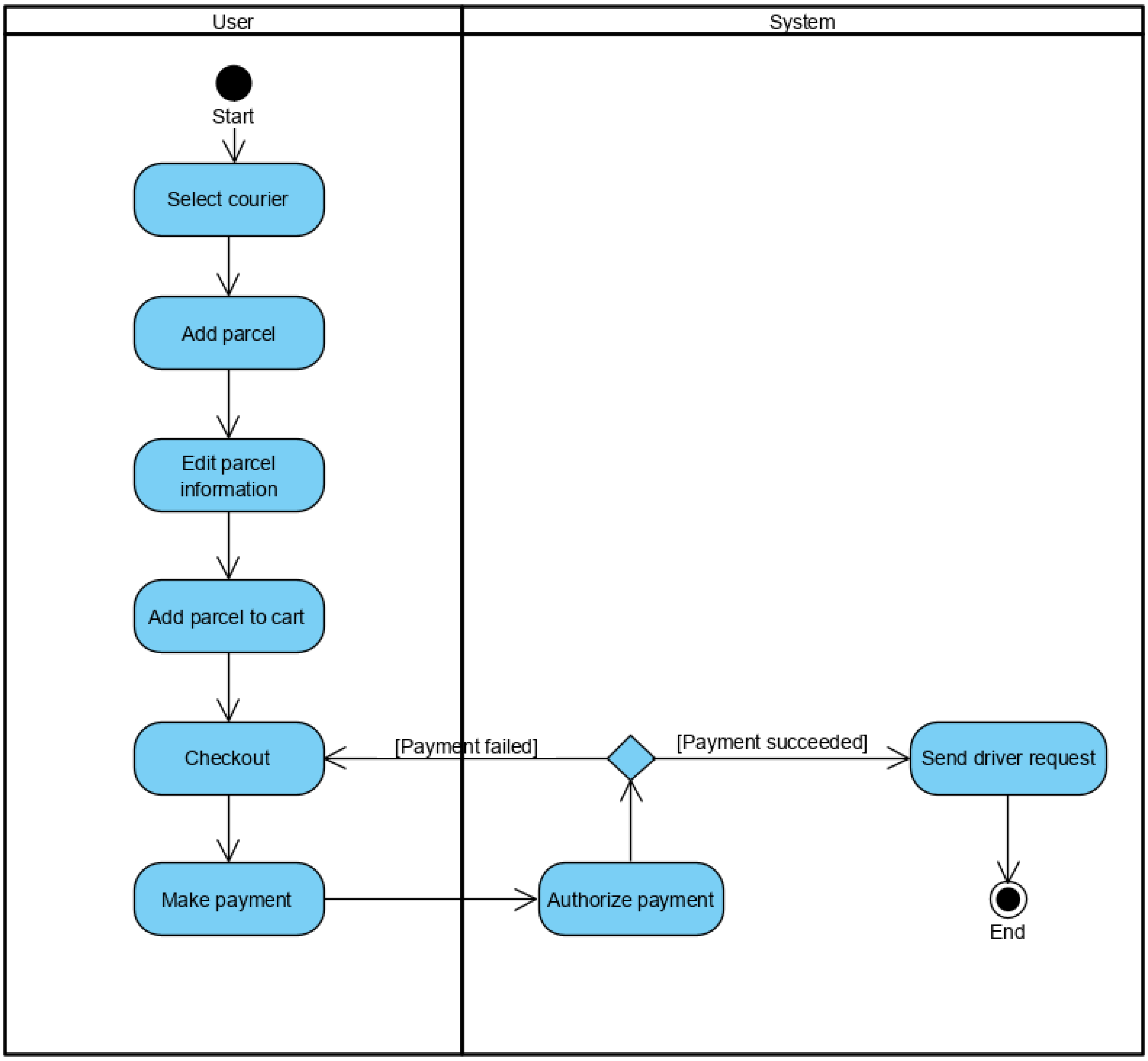


Figure 3.9 Activity Diagram for Select courier

The system requires the user to select a courier company from the list to send a driver request in order to ship their parcel. The user needs to click on add parcel button and fill up the information required such as shipping address, receiver address and etc. After that, the user needs to add the parcel into their cart and proceed checkout. Make payment is the last step to request a driver. The system will require the user to pay an amount of shipping fee calculated by the system. The system will start to find a nearby driver and send request to them.

### 3.2.2.7 Activity Diagram for Receive user request

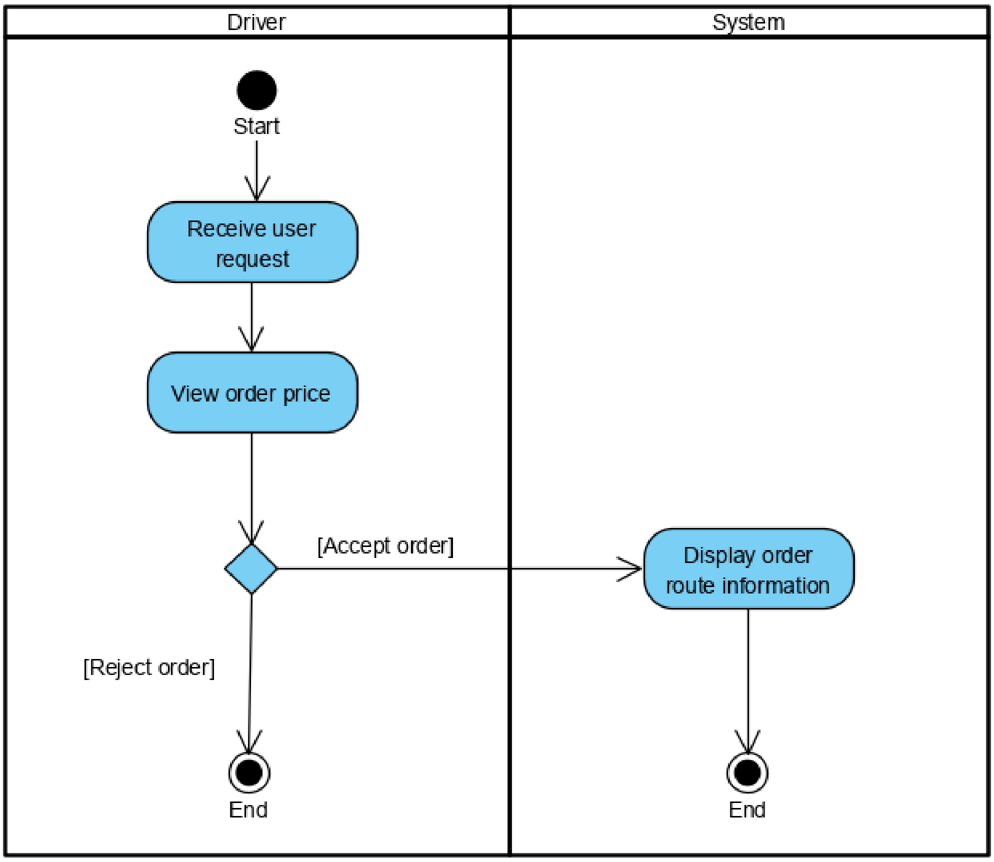


Figure 3.10 Activity Diagram for Receive user request

The nearest working driver will receive a user request notification. The driver can choose whether they want to accept or reject the request. Once the driver accepts the request, the system will show the route information. The information will lead the driver to reach the user's location.

### 3.2.2.8 Activity Diagram for Navigate route

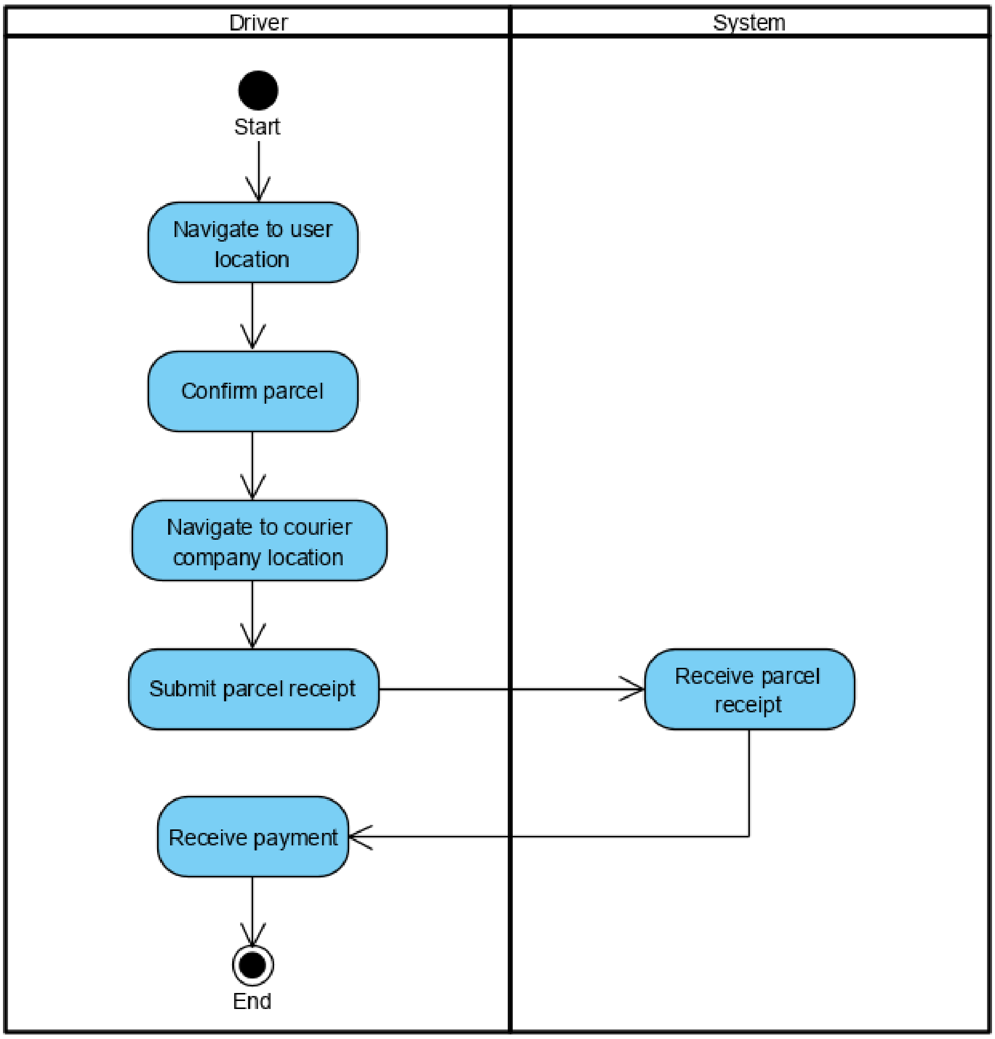


Figure 3.11 Activity Diagram for Navigate route

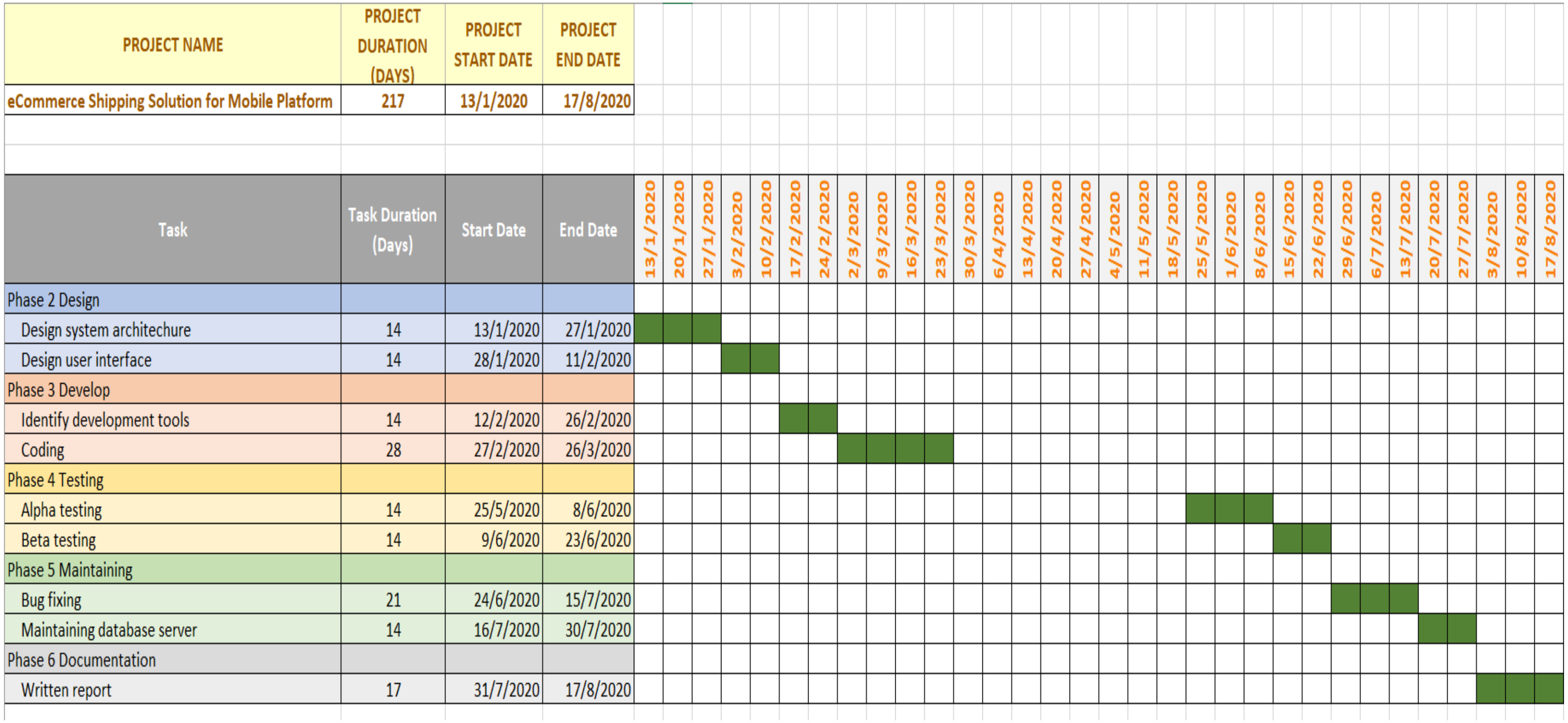
The system will provide the user address to the driver after they accept the request. The driver can use the information given and reach user's location. The driver needs to press the confirm button after receiving the parcel from the user. After that, the system will provide the courier company's address to the driver in order to help driver reach there. The driver requires to submit the photo of parcel receipt and the amount of shipping to the system as the evidence to receive their payment.

### 3.3 Implementation Issues and Challenges

There are some implementation issues happen during the development of the application. The hardware used which is running with the AMD processor does not support SVM function in Android Studio. A virtual device cannot be used to test the application caused an inconvenience of development. An android smartphone is needed to plug into the laptop to perform USB debugging through developer mode in order to test the application on the smartphone.

Besides, the challenge is met when developing the application by using the API of the Google Map Platform. It does not support directly get the result of nearby courier delivery companies because the courier company is not included in their category lists. In order to get the result of nearby courier delivery company, the name of the courier delivery company needs to be specified in the code as the keyword to successfully get the result.

## 3.4 Timeline



| PROJECT NAME | PROJECT DURATION (DAYS) | PROJECT START DATE | PROJECT END DATE |
|---|---|---|---|
| eCommerce Shipping Solution for Mobile Platform | 217 | 13/1/2020 | 17/8/2020 |

| Task | Task Duration (Days) | Start Date | End Date | 13/1/2020 | 20/1/2020 | 27/1/2020 | 3/2/2020 | 10/2/2020 | 17/2/2020 | 24/2/2020 | 2/3/2020 | 9/3/2020 | 16/3/2020 | 23/3/2020 | 30/3/2020 | 6/4/2020 | 13/4/2020 | 20/4/2020 | 27/4/2020 | 4/5/2020 | 11/5/2020 | 18/5/2020 | 25/5/2020 | 1/6/2020 | 8/6/2020 | 15/6/2020 | 22/6/2020 | 29/6/2020 | 6/7/2020 | 13/7/2020 | 20/7/2020 | 27/7/2020 | 3/8/2020 | 10/8/2020 | 17/8/2020 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Phase 2 Design | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Design system architechure | 14 | 13/1/2020 | 27/1/2020 | 🟩 | 🟩 | 🟩 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Design user interface | 14 | 28/1/2020 | 11/2/2020 | | | | 🟩 | 🟩 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Phase 3 Develop | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Identify development tools | 14 | 12/2/2020 | 26/2/2020 | | | | | | 🟩 | 🟩 | | | | | | | | | | | | | | | | | | | | | | | | | |
| Coding | 28 | 27/2/2020 | 26/3/2020 | | | | | | | | 🟩 | 🟩 | 🟩 | 🟩 | | | | | | | | | | | | | | | | | | | | | |
| Phase 4 Testing | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Alpha testing | 14 | 25/5/2020 | 8/6/2020 | | | | | | | | | | | | | | | | | | | | 🟩 | 🟩 | 🟩 | | | | | | | | | | |
| Beta testing | 14 | 9/6/2020 | 23/6/2020 | | | | | | | | | | | | | | | | | | | | | | | 🟩 | 🟩 | | | | | | | | |
| Phase 5 Maintaining | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bug fixing | 21 | 24/6/2020 | 15/7/2020 | | | | | | | | | | | | | | | | | | | | | | | | | 🟩 | 🟩 | 🟩 | | | | | |
| Maintaining database server | 14 | 16/7/2020 | 30/7/2020 | | | | | | | | | | | | | | | | | | | | | | | | | | | | 🟩 | 🟩 | | | |
| Phase 6 Documentation | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Written report | 17 | 31/7/2020 | 17/8/2020 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 🟩 | 🟩 | 🟩 |

Figure 3.12 Gantt Chart

Figure 3.13 shows the Gantt Chart timeline of ecommerce shipping solution for mobile platform. The project duration plan is 217 days to finish this project in two semesters. Using the Agile Development Life Cycle methodology, the project is separate into 6 different phases. Phase 2 and phase 3 plan to be finished in the current semester. Phase 4, phase 5 and phase 6 will be conduct in next semester. The first 4 weeks will use to design the system architecture and also the user interface of the system. Week 5 and 6 will use to identify suitable development tools. The coding will be conducted continued until the current semester break. When the next semester is started, the coding of the application should be done and start to perform testing. The testing phase is planned to be finished in 4 weeks. After the testing phase is done, the following 6 weeks are used to maintaining the application to check the application is stable and perform well. Finally, there will have 17 days to finish the written report of the project.

# Chapter 4 Implementation

## 4.1 Technology and Tools Involved

### 4.1.1 Software involved

- **Android Studio**
  Android Studio is Android's official Integrated Drive Electronics (IDE) from Google, Android Studio includes everything that is needed to build an app, including an intelligent code editor and debugger, performance analysis tools, emulators, and more. It is purpose-built for Android to accelerate the development and help the developer to build the highest-quality apps for every Android device.

- **Firebase**
  Firebase is a mobile and web app development platform that provides developers with a plethora of tools and services to help them develop high-quality apps, grow their user base and earn more profit. Firebase provides functionality like analytics, databases, messaging and crash reporting for the administrator to have better control on their users.

### 4.1.2 Hardware involved

- **Mobile Computer**
  This project will use a laptop computer run with Windows 10, AMD Ryzen 5 3550H processor, 12GB RAM and 64-bit OS to develop the system. The user needs the laptop computer to access the software involved and perform the task.

- **Smartphone device**
  This project will use smartphone specifications with Qualcomm SDM845 Snapdragon 845 chipset, 6GB RAM, Android 8.1 (Oreo) OS and 64GB internal storage. The user needs to install the application in this smartphone device and perform testing.

## 4.2 Modules

### 4.2.1 User Authentication

#### 4.2.1.1 MainActivity.java

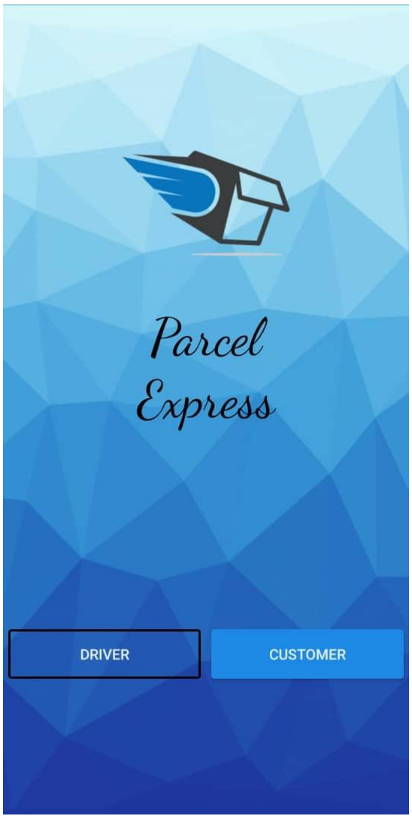
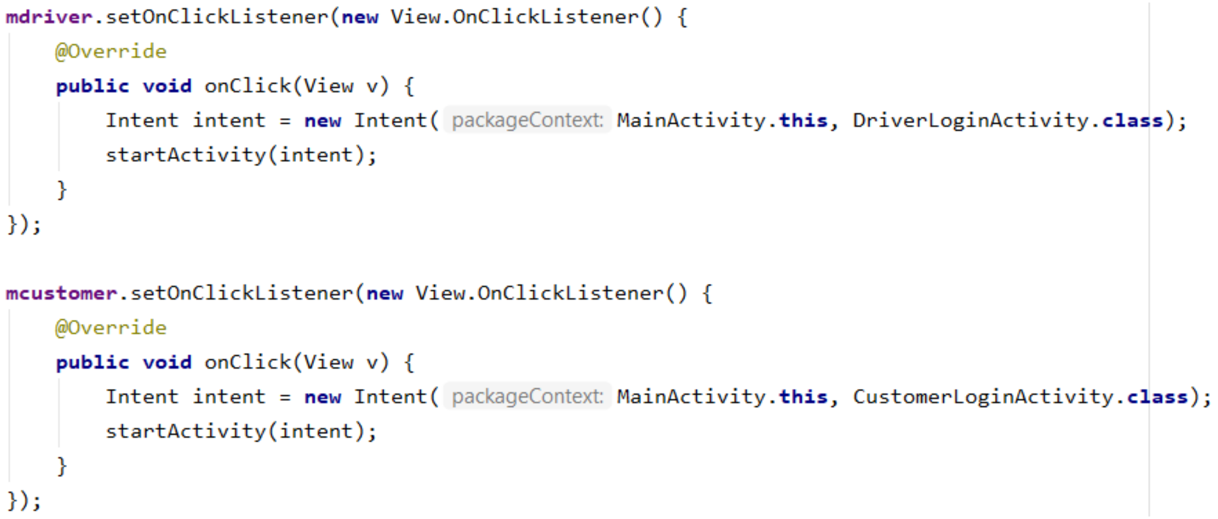


Figure 4.1 Screenshot of main page

This is the first page of the application. There are two buttons on this page which are driver and customer. Both of them will take a user to different page. If a user wants to earn money from this application, he or she may go to driver page. A driver in this application is responsible to help the customer sends their parcel to the selected location. However, if a user wants to send a parcel, he or she may go to customer pag



```
mdriver.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent( packageContext: MainActivity.this, DriverLoginActivity.class);
        startActivity(intent);
    }
});

mcustomer.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent( packageContext: MainActivity.this, CustomerLoginActivity.class);
        startActivity(intent);
    }
});
```

Figure 4.2 Screenshot of driver and customer button code

These are the code of both buttons. When user clicks driver button, the system will bring the user to driver login page. When user clicks customer button, the system will bring the user to customer login page.

### 4.2.1.2 CustomerLoginActivity.java

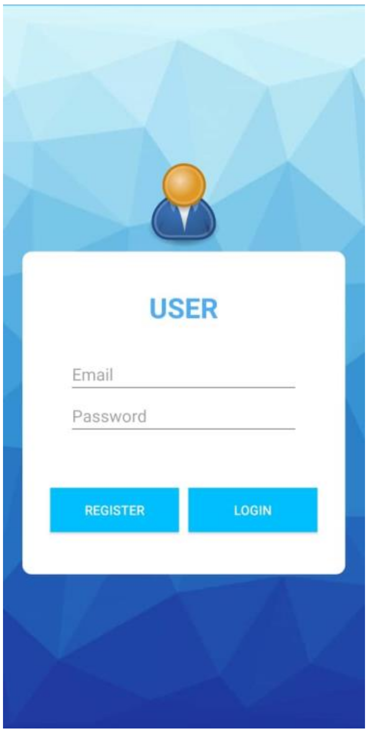
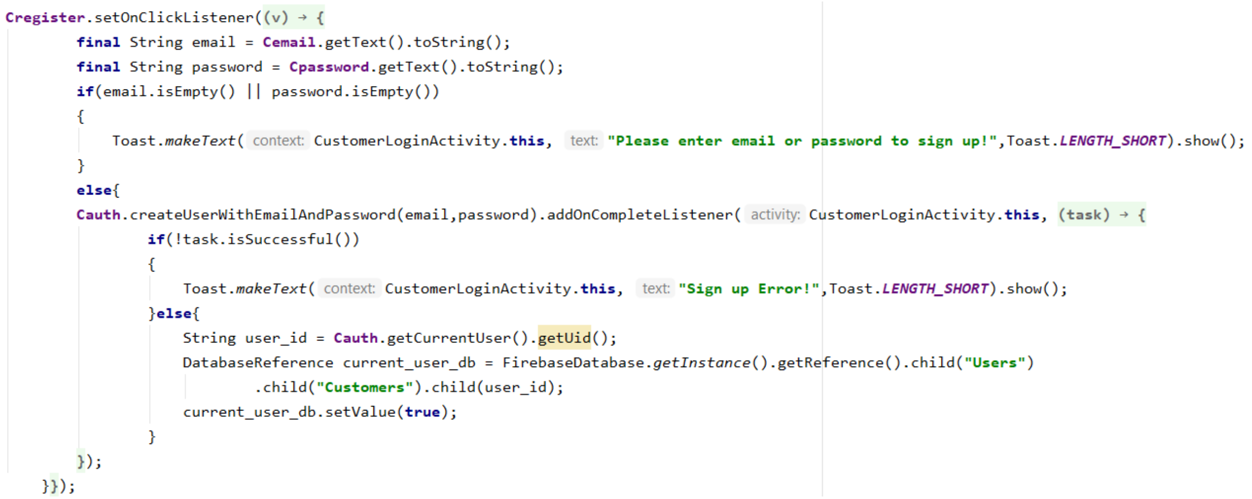


Figure 4.3 Screenshot of customer login and register page

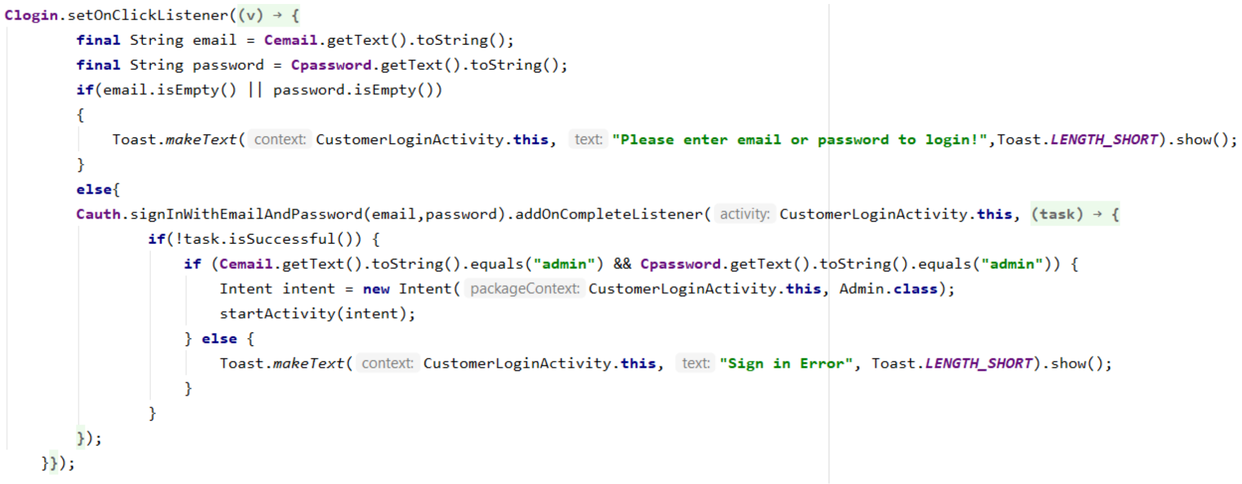
This is the page that will show up when a user clicks the customer button. User can login to the application with the registered email and password. The system only will give access to the user with enter a correct email and password matched with the database. If user is the first time using the application, he or she can register an account through fill up the email and password and click the register button. The system will register an account for the user if the email entered not yet register in the database.



```
Cregister.setOnClickListener((v) → {
        final String email = Cemail.getText().toString();
        final String password = Cpassword.getText().toString();
        if(email.isEmpty() || password.isEmpty())
        {
            Toast.makeText( context: CustomerLoginActivity.this,  text: "Please enter email or password to sign up!",Toast.LENGTH_SHORT).show();
        }
        else{
        Cauth.createUserWithEmailAndPassword(email,password).addOnCompleteListener( activity: CustomerLoginActivity.this, (task) → {
                if(!task.isSuccessful())
                {
                    Toast.makeText( context: CustomerLoginActivity.this,  text: "Sign up Error!",Toast.LENGTH_SHORT).show();
                }else{
                    String user_id = Cauth.getCurrentUser().getUid();
                    DatabaseReference current_user_db = FirebaseDatabase.getInstance().getReference().child("Users")
                            .child("Customers").child(user_id);
                    current_user_db.setValue(true);
                }
        });
    }});
```

Figure 4.4 Screenshot of user register button

These are the code of the register button. When a user clicked the register button, the system will check the fields of the email and password whether both of the fields are empty or not. If any field is empty, the system will show an error message to warn user. Other than that, it will also show an error message if the email entered is not a valid format. The system will save the email and password to the database when user fill up all the fields with the correct format.



```
Clogin.setOnClickListener((v) → {
        final String email = Cemail.getText().toString();
        final String password = Cpassword.getText().toString();
        if(email.isEmpty() || password.isEmpty())
        {
            Toast.makeText( context: CustomerLoginActivity.this,  text: "Please enter email or password to login!",Toast.LENGTH_SHORT).show();
        }
        else{
        Cauth.signInWithEmailAndPassword(email,password).addOnCompleteListener( activity: CustomerLoginActivity.this, (task) → {
                if(!task.isSuccessful()) {
                    if (Cemail.getText().toString().equals("admin") && Cpassword.getText().toString().equals("admin")) {
                        Intent intent = new Intent( packageContext: CustomerLoginActivity.this, Admin.class);
                        startActivity(intent);
                    } else {
                        Toast.makeText( context: CustomerLoginActivity.this,  text: "Sign in Error", Toast.LENGTH_SHORT).show();
                    }
                }
        });
    }});
```

Figure 4.5 Screenshot of user login button

After user is registered an account, he or she may click the login button to go to the next activity. When login button is clicked, the system will check whether all the fields are empty or not. It will show error message if one of the fields are empty. The system will check the email and password entered are existed in the database or not. The system will go to next activity if entered correct email and password else it will show an error message. Other than that, if the email and password fields both entered admin and click login, the system will go to the admin page.

### 4.2.1.3 DriverLoginActivity.java

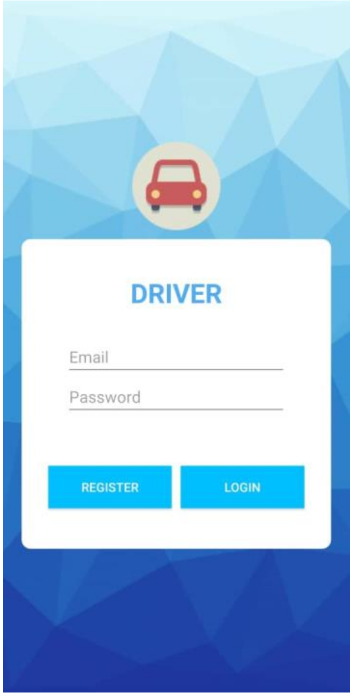
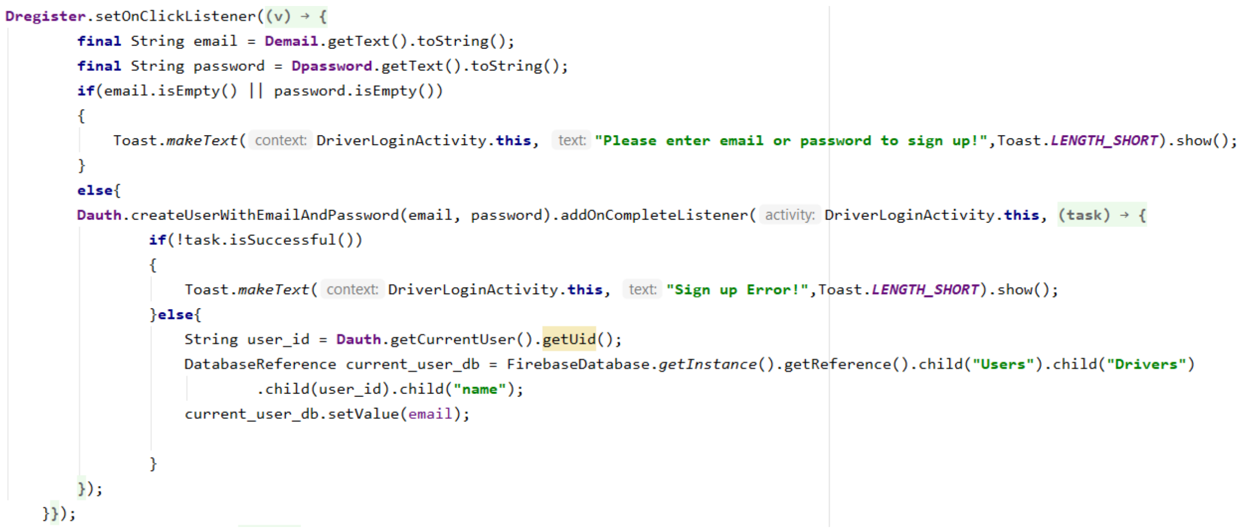


Figure 4.6 Screenshot of driver login and register page

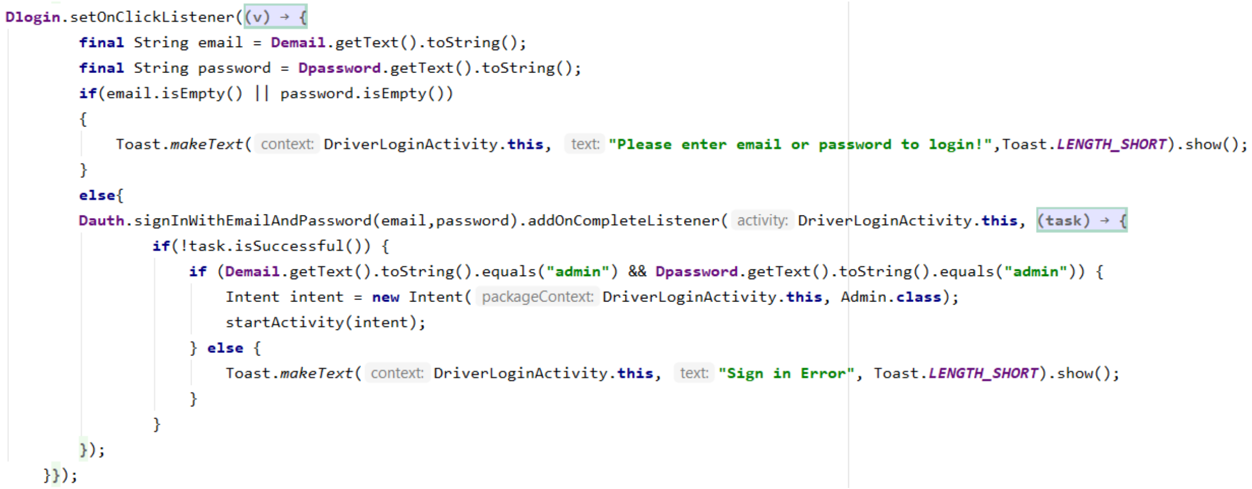
This is the page that will show up after a user clicks the driver button. The function of this page is the same as the customer login and register page. The only difference is checking and saving data at another database.



```
Dregister.setOnClickListener((v) → {
        final String email = Demail.getText().toString();
        final String password = Dpassword.getText().toString();
        if(email.isEmpty() || password.isEmpty())
        {
            Toast.makeText( context: DriverLoginActivity.this,  text: "Please enter email or password to sign up!",Toast.LENGTH_SHORT).show();
        }
        else{
        Dauth.createUserWithEmailAndPassword(email, password).addOnCompleteListener( activity: DriverLoginActivity.this, (task) → {
                if(!task.isSuccessful())
                {
                    Toast.makeText( context: DriverLoginActivity.this,  text: "Sign up Error!",Toast.LENGTH_SHORT).show();
                }else{
                    String user_id = Dauth.getCurrentUser().getUid();
                    DatabaseReference current_user_db = FirebaseDatabase.getInstance().getReference().child("Users").child("Drivers")
                            .child(user_id).child("name");
                    current_user_db.setValue(email);

                }
        });
    }});
```

Figure 4.7 Screenshot of driver register button

When user click the register button, the system will make sure that all the fields are entered with valid value. If the entered data not found in driver database, the system will add the email and password to the driver database. However, it will show an error message while the email is already registered in the database.



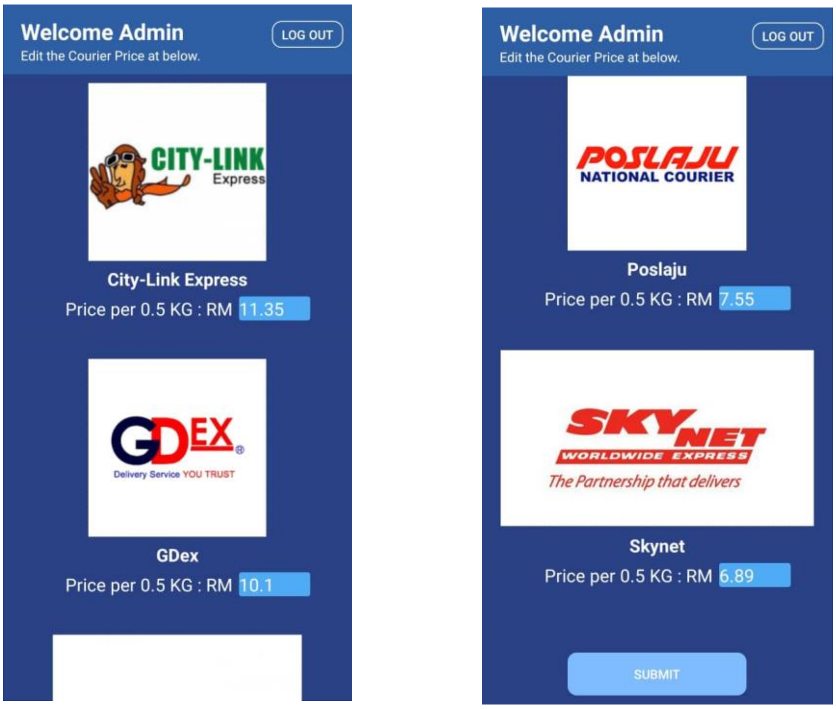
```
Dlogin.setOnClickListener((v) → {
        final String email = Demail.getText().toString();
        final String password = Dpassword.getText().toString();
        if(email.isEmpty() || password.isEmpty())
        {
            Toast.makeText( context: DriverLoginActivity.this,  text: "Please enter email or password to login!",Toast.LENGTH_SHORT).show();
        }
        else{
        Dauth.signInWithEmailAndPassword(email,password).addOnCompleteListener( activity: DriverLoginActivity.this, (task) → {
                if(!task.isSuccessful()) {
                    if (Demail.getText().toString().equals("admin") && Dpassword.getText().toString().equals("admin")) {
                        Intent intent = new Intent( packageContext: DriverLoginActivity.this, Admin.class);
                        startActivity(intent);
                    } else {
                        Toast.makeText( context: DriverLoginActivity.this,  text: "Sign in Error", Toast.LENGTH_SHORT).show();
                    }
                }
        });
    }});
```

Figure 4.8 Screenshot of driver login button

Same as register button, the login button will make sure email and password fields are not empty. If both entered matched data with the database, the system will bring user to go to the next activity. Admin page also can be accessed while email and password fields both entered admin and click login button.

## 4.2.2 Admin

### 4.2.2.1 Admin.java



Welcome Admin
Edit the Courier Price at below.

City-Link Express
Price per 0.5 KG : RM 11.35

GDex
Price per 0.5 KG : RM 10.1

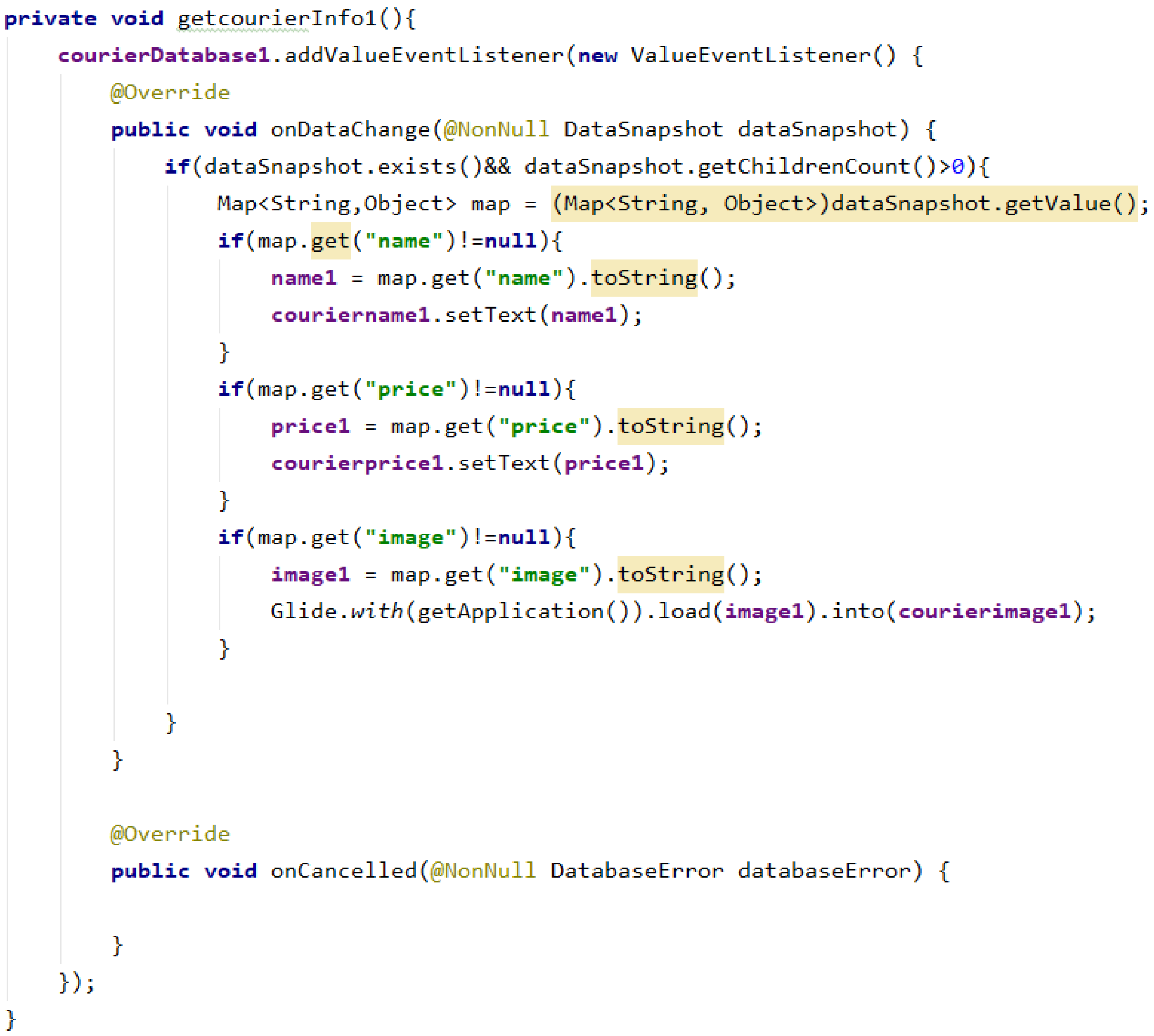
Welcome Admin
Edit the Courier Price at below.

Poslaju
Price per 0.5 KG : RM 7.55

Skynet
Price per 0.5 KG : RM 6.89

Figure 4.9 Screenshot of admin page

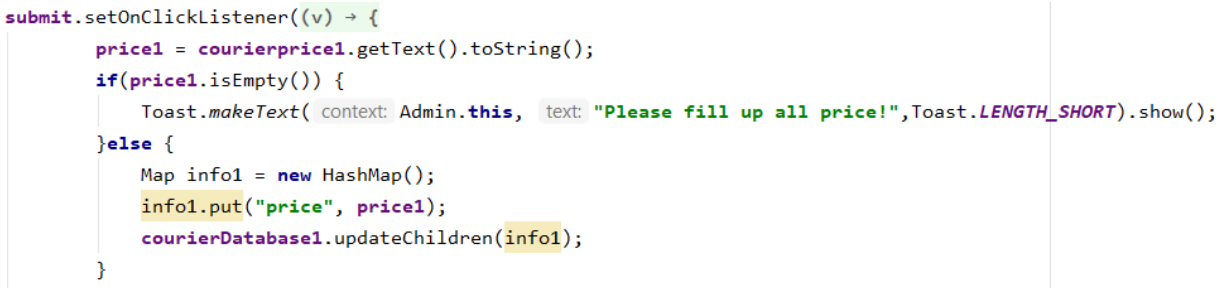
This is the page where the administrator can modify the price of courier companies that supported in this application. All the prices will be updated to the courier company database when the submit button is clicked. Each of the courier company prices must be filled up nicely by the administrator. An error message will show if a price is empty.



```
private void getcourierInfo1(){
    courierDatabase1.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()&& dataSnapshot.getChildrenCount()>0){
                Map<String,Object> map = (Map<String, Object>)dataSnapshot.getValue();
                if(map.get("name")!=null){
                    name1 = map.get("name").toString();
                    couriername1.setText(name1);
                }
                if(map.get("price")!=null){
                    price1 = map.get("price").toString();
                    courierprice1.setText(price1);
                }
                if(map.get("image")!=null){
                    image1 = map.get("image").toString();
                    Glide.with(getApplication()).load(image1).into(courierimage1);
                }

            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}
```

Figure 4.10 Screenshot of retrieve courier company data from database

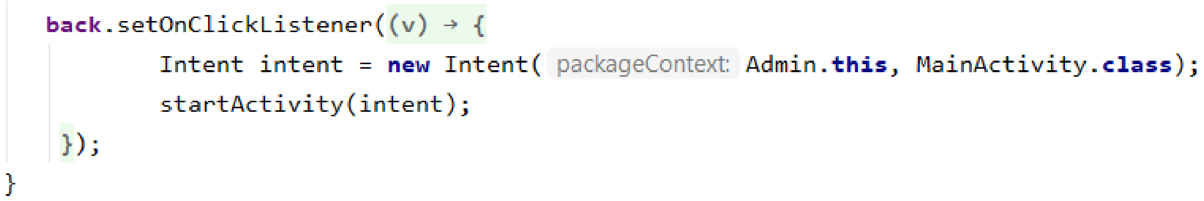
These are the code used to retrieve courier company data stored at the database and display it on the page. The data include courier's name, courier's price and courier's image.



```
submit.setOnClickListener((v) → {
        price1 = courierprice1.getText().toString();
        if(price1.isEmpty()) {
            Toast.makeText( context: Admin.this,  text: "Please fill up all price!",Toast.LENGTH_SHORT).show();
        }else {
            Map info1 = new HashMap();
            info1.put("price", price1);
            courierDatabase1.updateChildren(info1);
        }
```

Figure 4.11 Screenshot of the submit button code

These are the code of the submit button at the admin page. When admin clicks the submit button, the system will have a check and ensure all the prices have been filled up. If found price field is empty, it will show an error message to admin.



```
    back.setOnClickListener((v) → {
            Intent intent = new Intent( packageContext: Admin.this, MainActivity.class);
            startActivity(intent);
     });
}
```

Figure 4.12 Screenshot of the logout button code

When the logout button is clicked, the application will go back to the main page.

### 4.2.3 Search nearby courier company

#### 4.2.3.1 Search.java

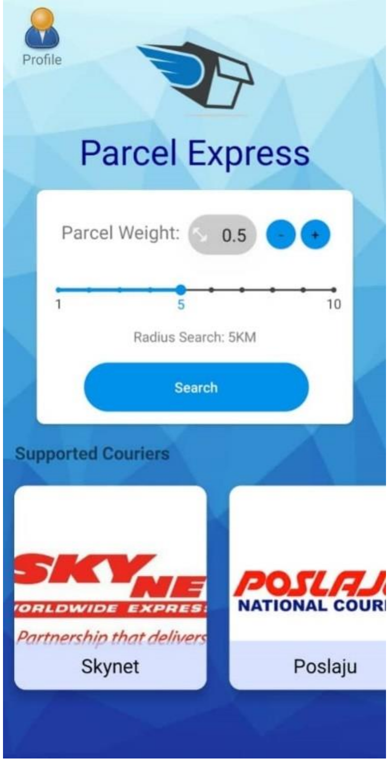
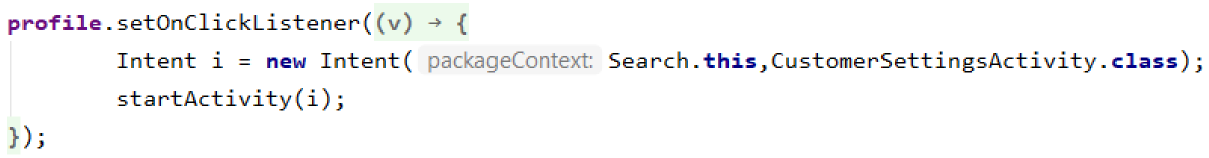


Figure 4.13 Screenshot of the customer page

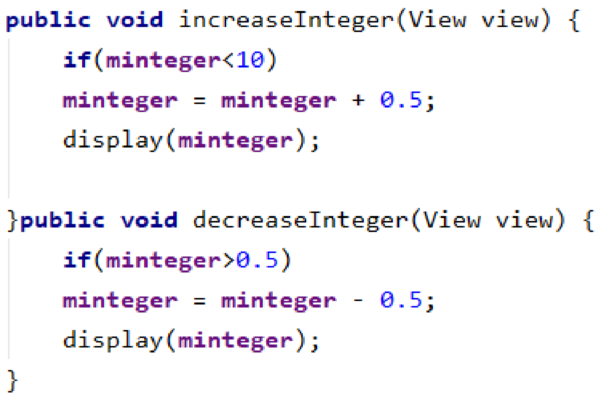
This page will be displayed to user when login successfully using customer account. The top left of the page exists a profile icon will bring users to go to their profile page after clicking. At the bottom of this page will show a list of supported courier company in this application. The user can set their parcel weight and the search radius to find out nearby courier companies around the user's current location. The default of parcel weight set will be 0.5KG and the maximum is 10.0KG. The search radius minimum is 1KM and the maximum is 10KM. After user done set the weight and search radius, the last step is to click the search button. The application will bring user to the map and show the search result.



```
profile.setOnClickListener((v) → {
        Intent i = new Intent( packageContext: Search.this,CustomerSettingsActivity.class);
        startActivity(i);
});
```

Figure 4.14 Screenshot of the profile icon button code

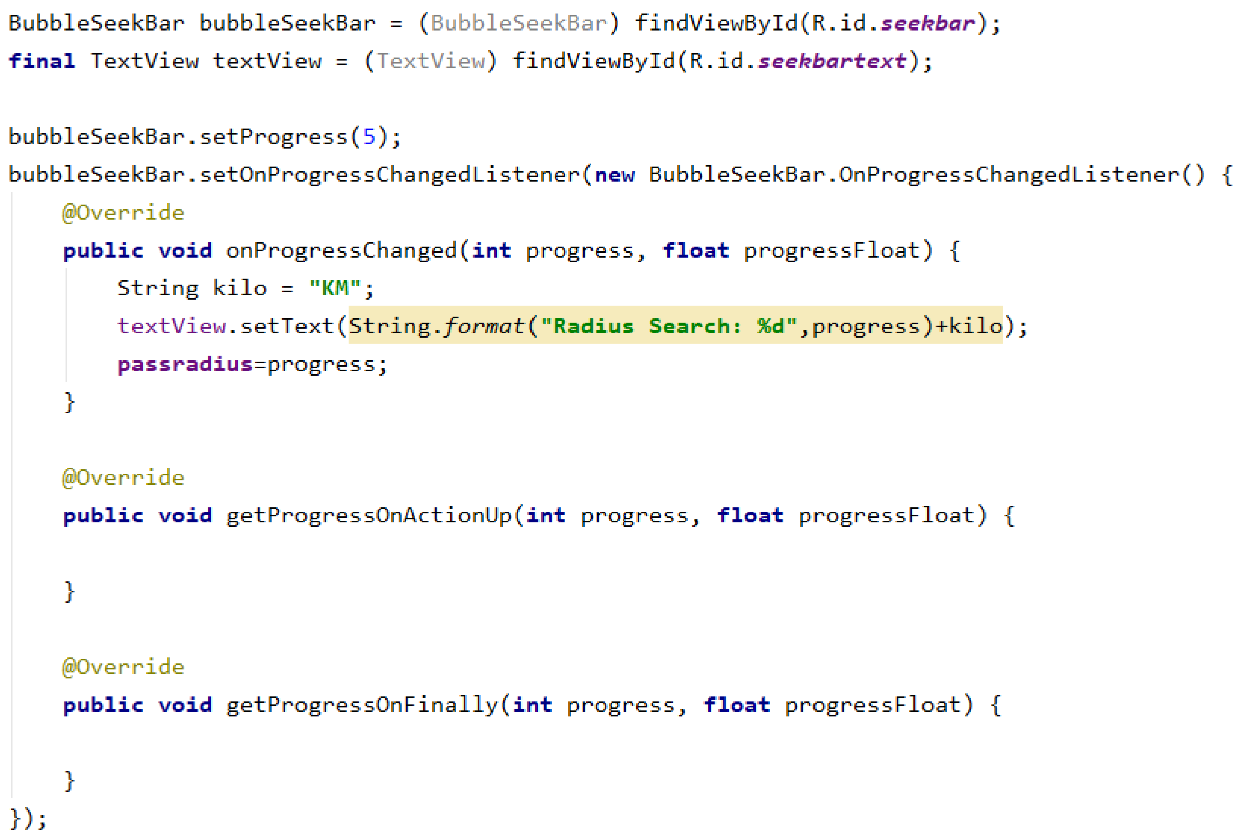
When the profile icon is clicked, the application will go to customer settings page.



```
public void increaseInteger(View view) {
    if(minteger<10)
    minteger = minteger + 0.5;
    display(minteger);

}public void decreaseInteger(View view) {
    if(minteger>0.5)
    minteger = minteger - 0.5;
    display(minteger);
}
```

Figure 4.15 Screenshot of the minus and plus button code

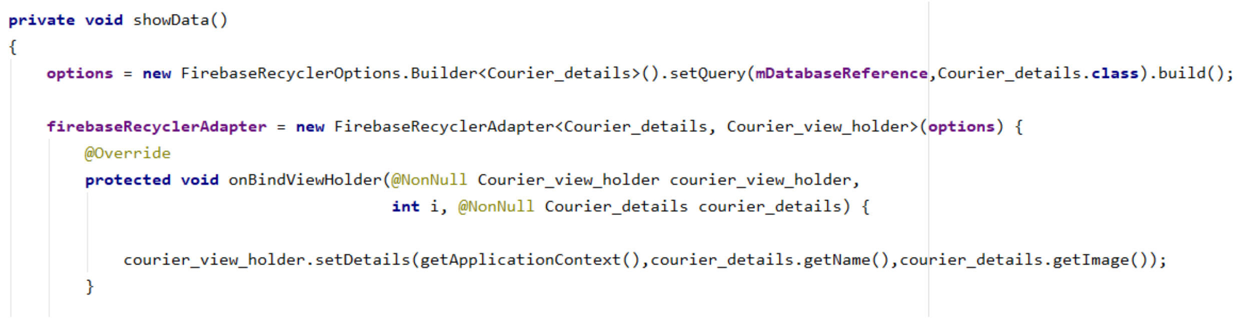
These are the code of the plus and minus buttons to set the weight of the parcel. The plus button will increase 0.5 for each time press and the minus button also will decrease 0.5 for each time press.



```
BubbleSeekBar bubbleSeekBar = (BubbleSeekBar) findViewById(R.id.seekbar);
final TextView textView = (TextView) findViewById(R.id.seekbartext);

bubbleSeekBar.setProgress(5);
bubbleSeekBar.setOnProgressChangedListener(new BubbleSeekBar.OnProgressChangedListener() {
    @Override
    public void onProgressChanged(int progress, float progressFloat) {
        String kilo = "KM";
        textView.setText(String.format("Radius Search: %d",progress)+kilo);
        passradius=progress;
    }

    @Override
    public void getProgressOnActionUp(int progress, float progressFloat) {

    }

    @Override
    public void getProgressOnFinally(int progress, float progressFloat) {

    }
});
```

Figure 4.16 Screenshot of seek bar code

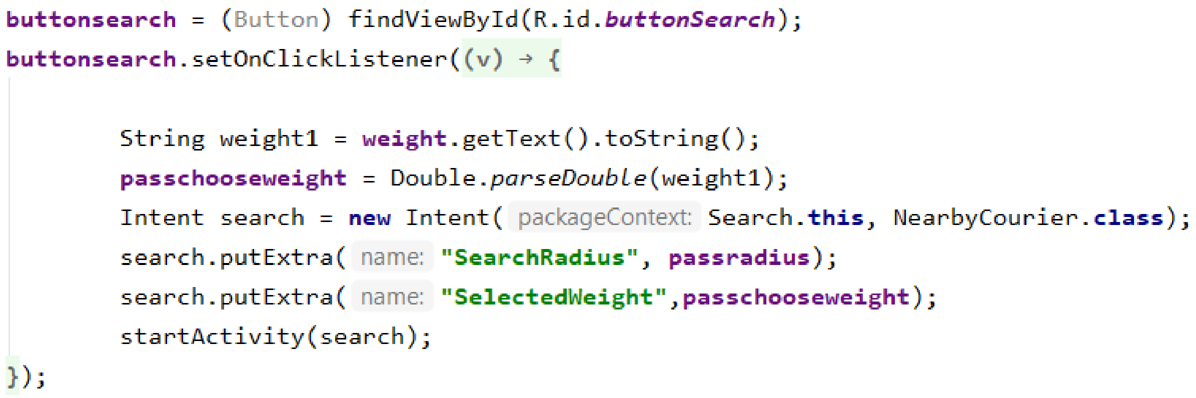
These are the code to display the seek bar and the default value of the seek bar is 5. Every time changing the seek bar value will also change the value of text view below to let user know the search radius.



```
private void showData()
{
    options = new FirebaseRecyclerOptions.Builder<Courier_details>().setQuery(mDatabaseReference,Courier_details.class).build();

    firebaseRecyclerAdapter = new FirebaseRecyclerAdapter<Courier_details, Courier_view_holder>(options) {
        @Override
        protected void onBindViewHolder(@NonNull Courier_view_holder courier_view_holder,
                                        int i, @NonNull Courier_details courier_details) {

            courier_view_holder.setDetails(getApplicationContext(),courier_details.getName(),courier_details.getImage());
        }
```

Figure 4.17 Screenshot of courier company list code

These are the code used to get the courier company data in the database and display the name and image of the courier company in the list.



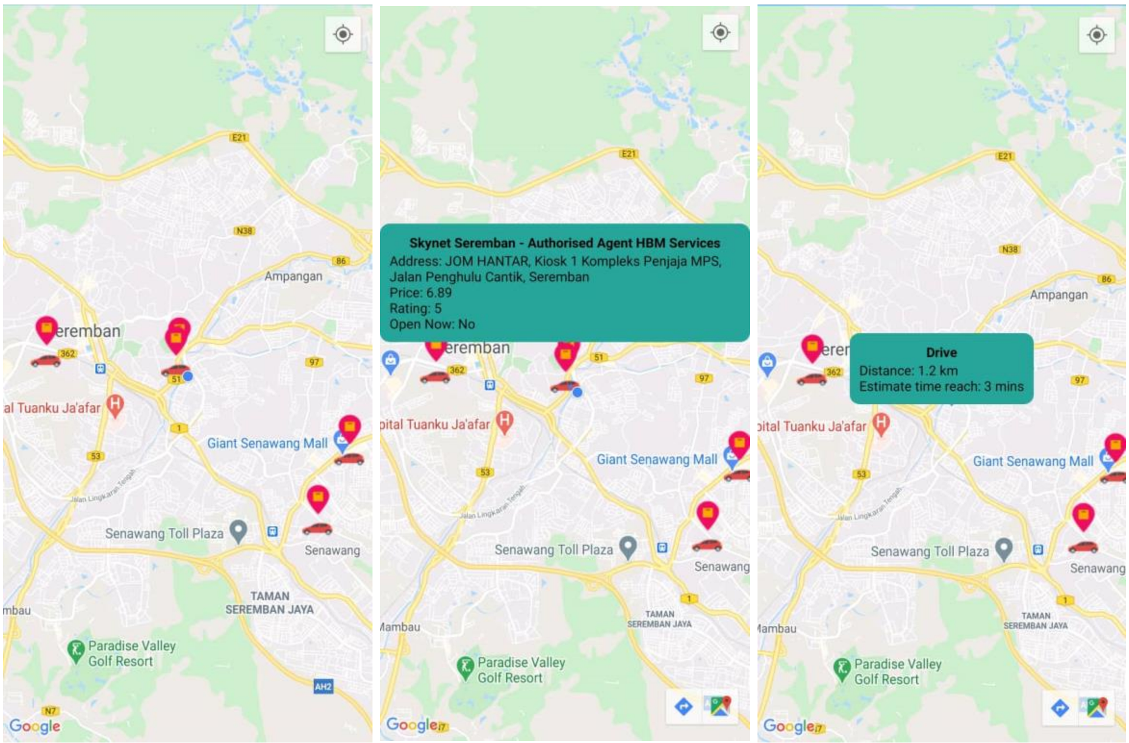
```
buttonsearch = (Button) findViewById(R.id.buttonSearch);
buttonsearch.setOnClickListener((v) → {

        String weight1 = weight.getText().toString();
        passchooseweight = Double.parseDouble(weight1);
        Intent search = new Intent( packageContext: Search.this, NearbyCourier.class);
        search.putExtra( name: "SearchRadius", passradius);
        search.putExtra( name: "SelectedWeight",passchooseweight);
        startActivity(search);
});
```

Figure 4.18 Screenshot of search button code

These are the code of the search button. When user clicks the search button, the system will pass the value of search radius and weight of parcel to the next activity.

### 4.2.4 Courier company map

#### 4.2.4.1 NearbyCourier.java

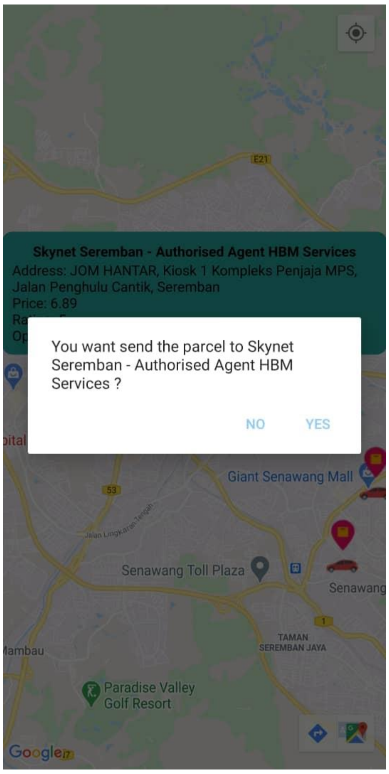


Skynet Seremban - Authorised Agent HBM Services
Address: JOM HANTAR, Kiosk 1 Kompleks Penjaja MPS, Jalan Penghulu Cantik, Seremban
Price: 6.89
Rating: 5
Open Now: No

Drive
Distance: 1.2 km
Estimate time reach: 3 mins

Figure 4.19 Screenshot of the courier company map page

This page will show all the search for nearby courier company result. The blue circle dot is the current location of the user. The red markers point on map is the location of the courier company. The car icons below the red markers are the distance information of the courier company. When user clicks the red marker point, an info window contains courier company details will be displayed. It contains the name, address, price offer, rating and current open status of the courier company. When user clicks the car icon, the distance information such as the distance between user and the courier company and also the estimated time to reach there will be shown.

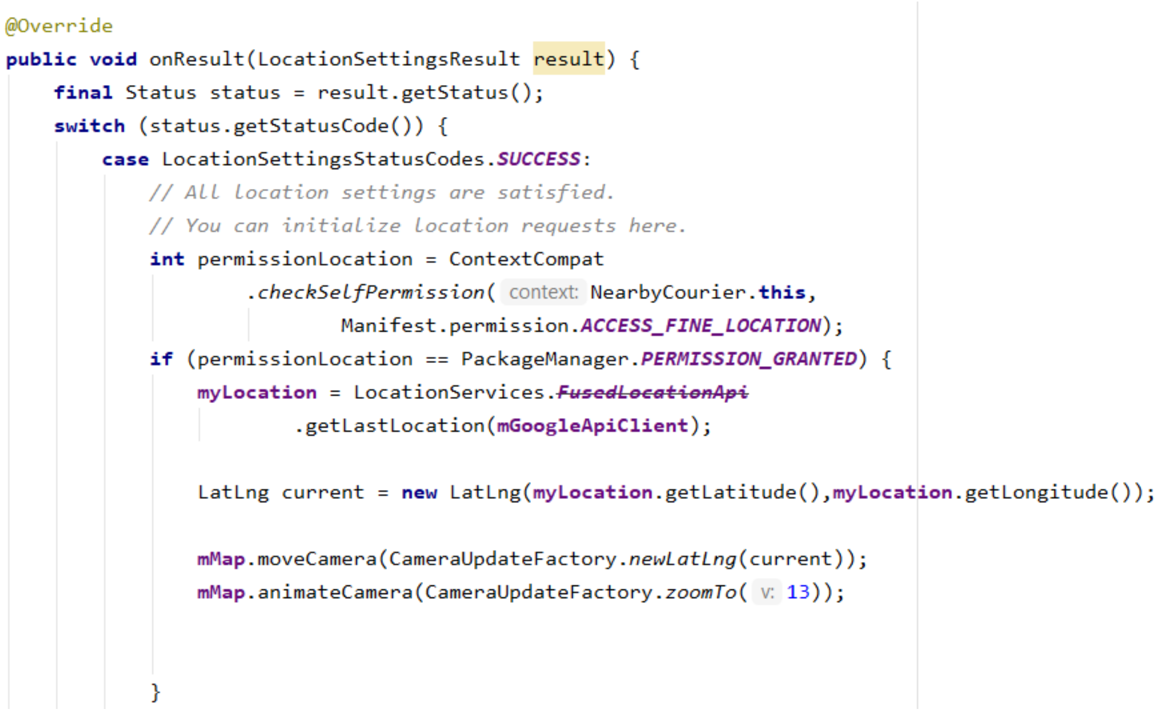


**Skynet Seremban - Authorised Agent HBM Services**
Address: JOM HANTAR, Kiosk 1 Kompleks Penjaja MPS, Jalan Penghulu Cantik, Seremban
Price: 6.89
Ra
Op

You want send the parcel to Skynet Seremban - Authorised Agent HBM Services ?

Figure 4.20 Screenshot of alert window

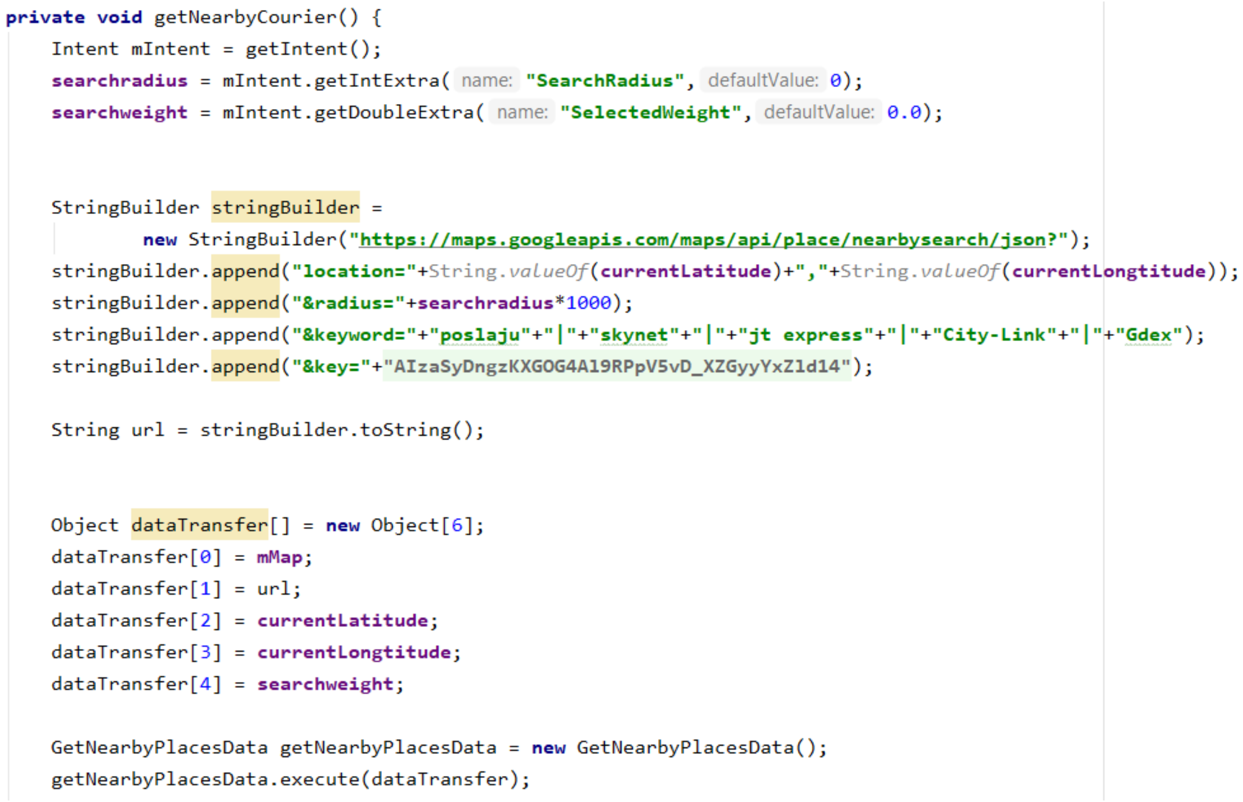
An alert window will show when user clicks the selected courier company info window. It is used to let user confirm the courier company selected is the place where they want to send their parcel.



```
@Override
public void onResult(LocationSettingsResult result) {
    final Status status = result.getStatus();
    switch (status.getStatusCode()) {
        case LocationSettingsStatusCodes.SUCCESS:
            // All location settings are satisfied.
            // You can initialize location requests here.
            int permissionLocation = ContextCompat
                    .checkSelfPermission( context: NearbyCourier.this,
                            Manifest.permission.ACCESS_FINE_LOCATION);
            if (permissionLocation == PackageManager.PERMISSION_GRANTED) {
                myLocation = LocationServices.~~FusedLocationApi~~
                        .getLastLocation(mGoogleApiClient);

                LatLng current = new LatLng(myLocation.getLatitude(),myLocation.getLongitude());

                mMap.moveCamera(CameraUpdateFactory.newLatLng(current));
                mMap.animateCamera(CameraUpdateFactory.zoomTo( v: 13));


            }
```

Figure 4.21 Screenshot of current location code

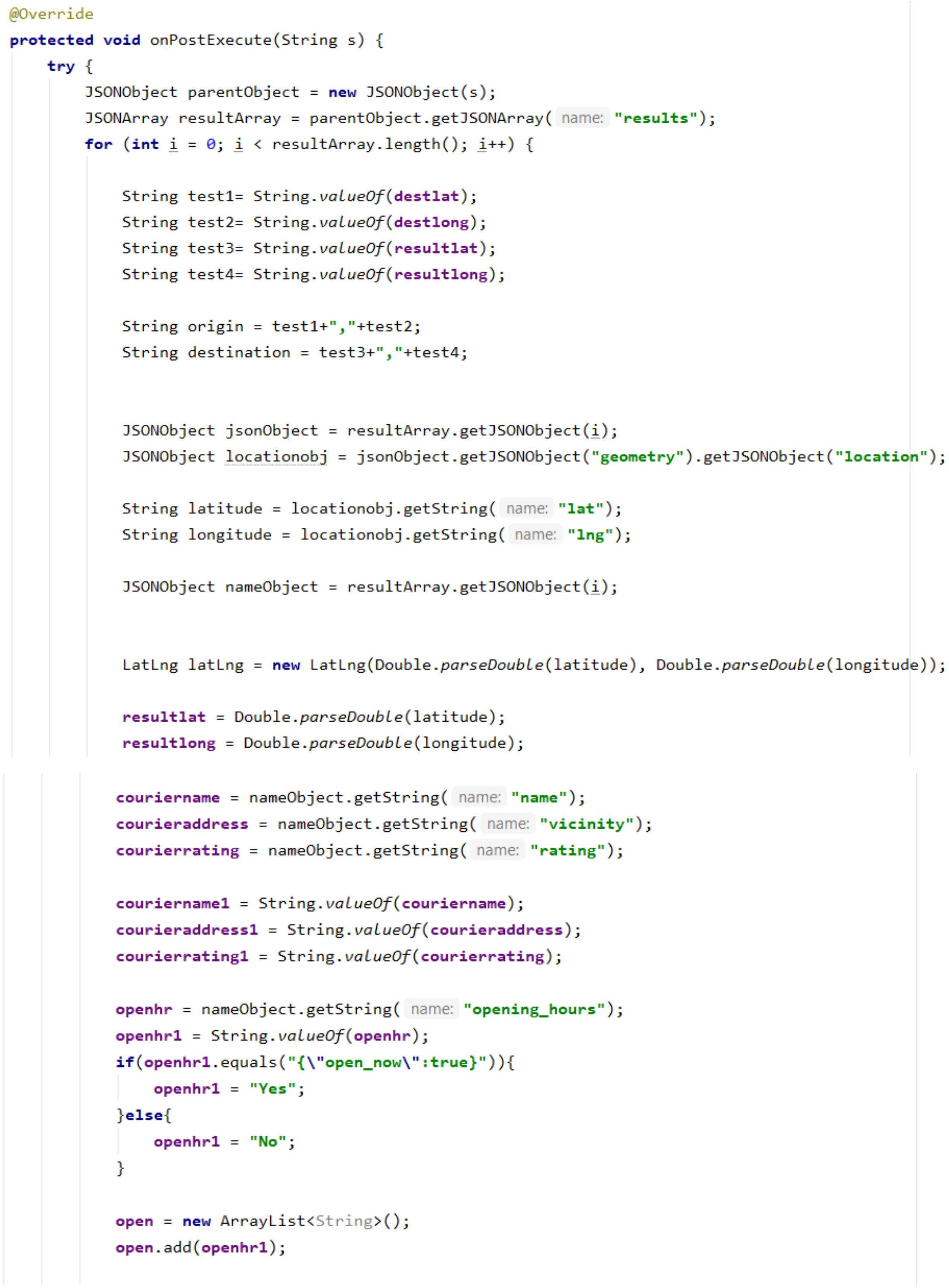
These are the code used to get the user’s current location and show on the map using a blue circle dot. It will first check the grant location permission status of application every time using the device location. If the permission is granted, the system will get the user’s location and display a blue circle dot on the map.



```
private void getNearbyCourier() {
    Intent mIntent = getIntent();
    searchradius = mIntent.getIntExtra( name: "SearchRadius", defaultValue: 0);
    searchweight = mIntent.getDoubleExtra( name: "SelectedWeight", defaultValue: 0.0);


    StringBuilder stringBuilder =
            new StringBuilder("https://maps.googleapis.com/maps/api/place/nearbysearch/json?");
    stringBuilder.append("location="+String.valueOf(currentLatitude)+","+String.valueOf(currentLongtitude));
    stringBuilder.append("&radius="+searchradius*1000);
    stringBuilder.append("&keyword="+"poslaju"+"|"+"skynet"+"|"+"jt express"+"|"+"City-Link"+"|"+"Gdex");
    stringBuilder.append("&key="+"AIzaSyDngzKXGOG4A19RPpV5vD_XZGyyYxZ1d14");

    String url = stringBuilder.toString();


    Object dataTransfer[] = new Object[6];
    dataTransfer[0] = mMap;
    dataTransfer[1] = url;
    dataTransfer[2] = currentLatitude;
    dataTransfer[3] = currentLongtitude;
    dataTransfer[4] = searchweight;

    GetNearbyPlacesData getNearbyPlacesData = new GetNearbyPlacesData();
    getNearbyPlacesData.execute(dataTransfer);
```

Figure 4.22 Screenshot of get nearby courier company code

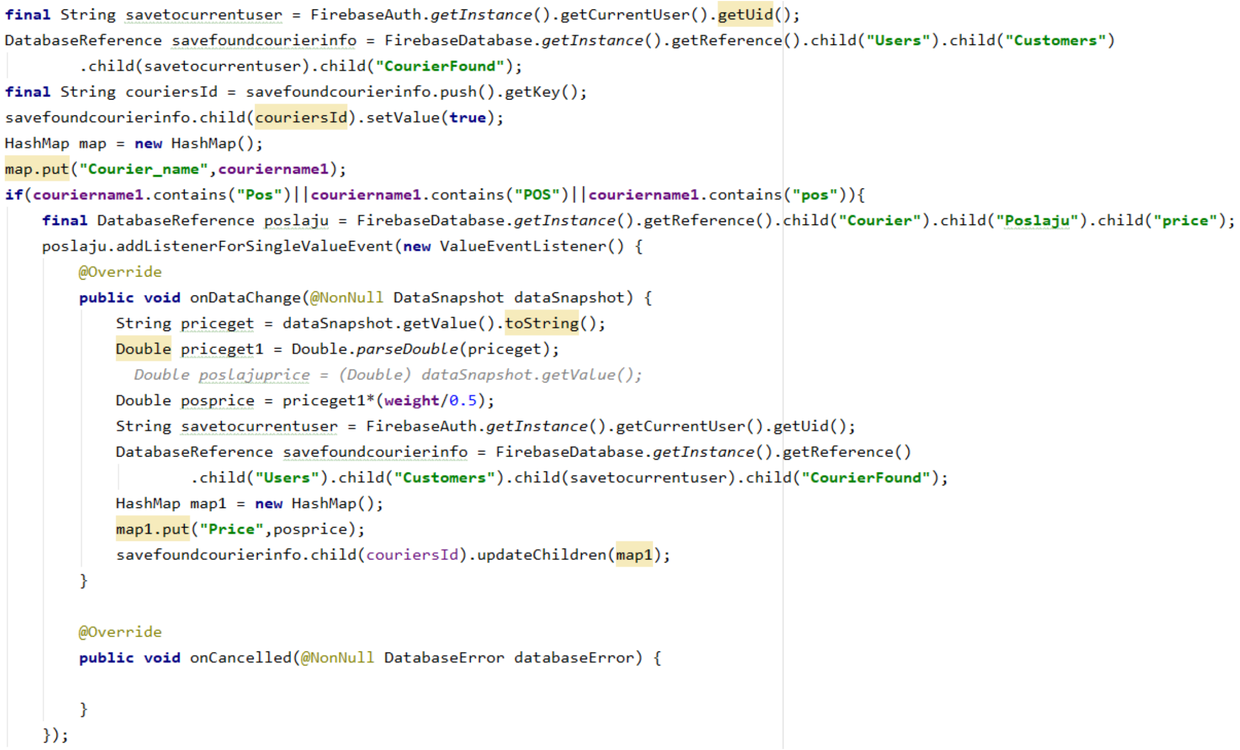
These are the code used to search nearby courier companies. The system will first get the search radius and parcel weight from the previous search activity. The system will add the search radius into the URL and call another function that use to call Location API.



```
@Override
protected void onPostExecute(String s) {
    try {
        JSONObject parentObject = new JSONObject(s);
        JSONArray resultArray = parentObject.getJSONArray( name: "results");
        for (int i = 0; i < resultArray.length(); i++) {

            String test1= String.valueOf(destlat);
            String test2= String.valueOf(destlong);
            String test3= String.valueOf(resultlat);
            String test4= String.valueOf(resultlong);

            String origin = test1+","+test2;
            String destination = test3+","+test4;


            JSONObject jsonObject = resultArray.getJSONObject(i);
            JSONObject locationobj = jsonObject.getJSONObject("geometry").getJSONObject("location");

            String latitude = locationobj.getString( name: "lat");
            String longitude = locationobj.getString( name: "lng");

            JSONObject nameObject = resultArray.getJSONObject(i);


            LatLng latLng = new LatLng(Double.parseDouble(latitude), Double.parseDouble(longitude));

            resultlat = Double.parseDouble(latitude);
            resultlong = Double.parseDouble(longitude);

            couriername = nameObject.getString( name: "name");
            courieraddress = nameObject.getString( name: "vicinity");
            courierrating = nameObject.getString( name: "rating");

            couriername1 = String.valueOf(couriername);
            courieraddress1 = String.valueOf(courieraddress);
            courierrating1 = String.valueOf(courierrating);

            openhr = nameObject.getString( name: "opening_hours");
            openhr1 = String.valueOf(openhr);
            if(openhr1.equals("{\"open_now\":true}")){
                openhr1 = "Yes";
            }else{
                openhr1 = "No";
            }

            open = new ArrayList<String>();
            open.add(openhr1);
```

Figure 4.23 Screenshot of get result of nearby courier company code

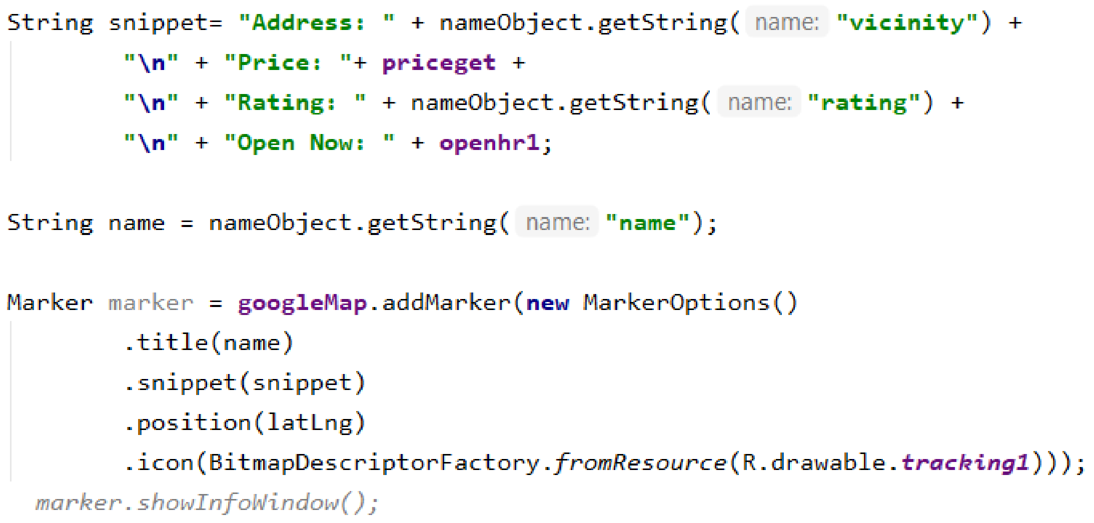
Figure 4.23 is the code use to get the result of the nearby courier company. The return result of the Location API will be in JSON format. The code will loop all the result data such as latitude, longitude, name, address, rating and open hours.



```
final String savetocurrentuser = FirebaseAuth.getInstance().getCurrentUser().getUid();
DatabaseReference savefoundcourierinfo = FirebaseDatabase.getInstance().getReference().child("Users").child("Customers")
        .child(savetocurrentuser).child("CourierFound");
final String couriersId = savefoundcourierinfo.push().getKey();
savefoundcourierinfo.child(couriersId).setValue(true);
HashMap map = new HashMap();
map.put("Courier_name",couriername1);
if(couriername1.contains("Pos")||couriername1.contains("POS")||couriername1.contains("pos")){
    final DatabaseReference poslaju = FirebaseDatabase.getInstance().getReference().child("Courier").child("Poslaju").child("price");
    poslaju.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            String priceget = dataSnapshot.getValue().toString();
            Double priceget1 = Double.parseDouble(priceget);
              Double poslajuprice = (Double) dataSnapshot.getValue();
            Double posprice = priceget1*(weight/0.5);
            String savetocurrentuser = FirebaseAuth.getInstance().getCurrentUser().getUid();
            DatabaseReference savefoundcourierinfo = FirebaseDatabase.getInstance().getReference()
                    .child("Users").child("Customers").child(savetocurrentuser).child("CourierFound");
            HashMap map1 = new HashMap();
            map1.put("Price",posprice);
            savefoundcourierinfo.child(couriersId).updateChildren(map1);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
```

Figure 4.24 Screenshot of adding courier company price code

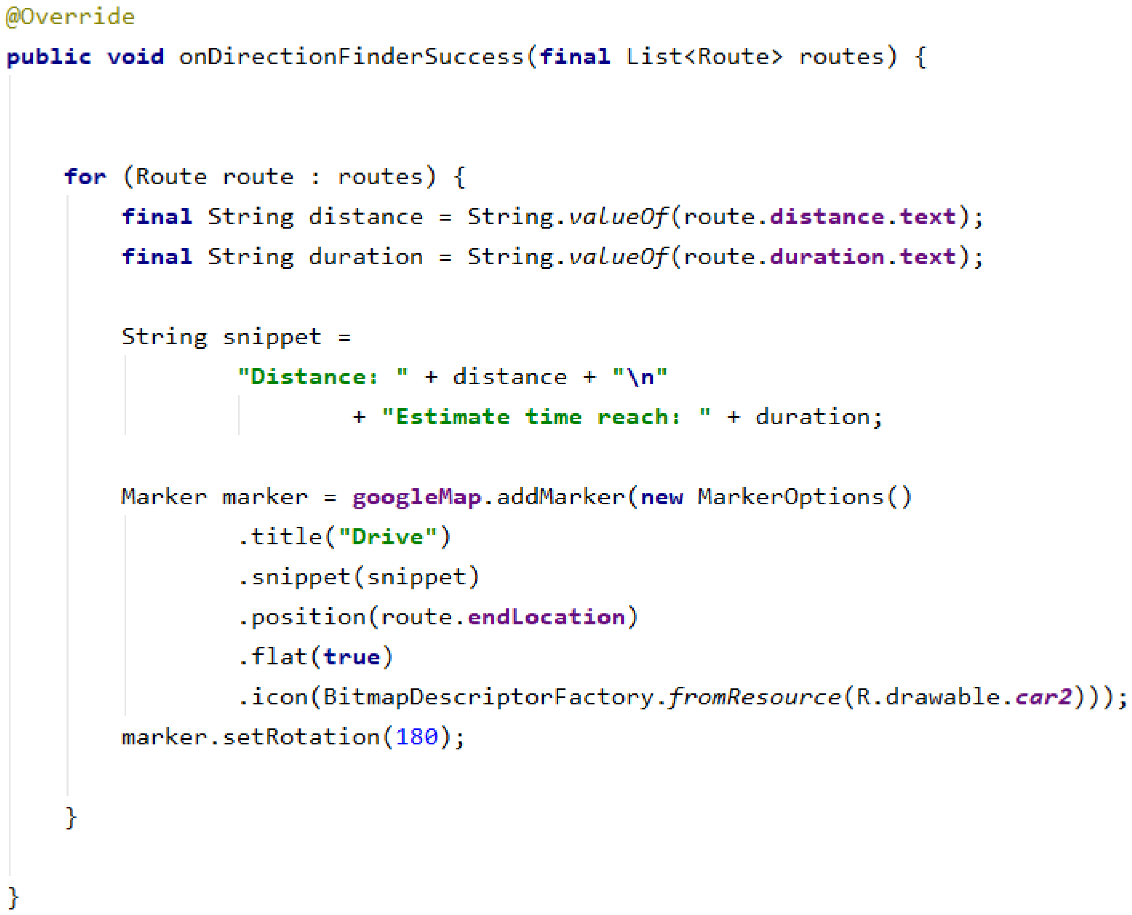
These codes will check the courier company name whether match with the courier company's name store at the database. If matched, it will add the price to the courier company. Besides, the price also will be calculated using the parcel weight entered by the user.



```
String snippet= "Address: " + nameObject.getString( name: "vicinity") +
        "\n" + "Price: "+ priceget +
        "\n" + "Rating: " + nameObject.getString( name: "rating") +
        "\n" + "Open Now: " + openhr1;

String name = nameObject.getString( name: "name");

Marker marker = googleMap.addMarker(new MarkerOptions()
        .title(name)
        .snippet(snippet)
        .position(latLng)
        .icon(BitmapDescriptorFactory.fromResource(R.drawable.tracking1)));
  marker.showInfoWindow();
```

Figure 4.25 Screenshot of adding marker on map code

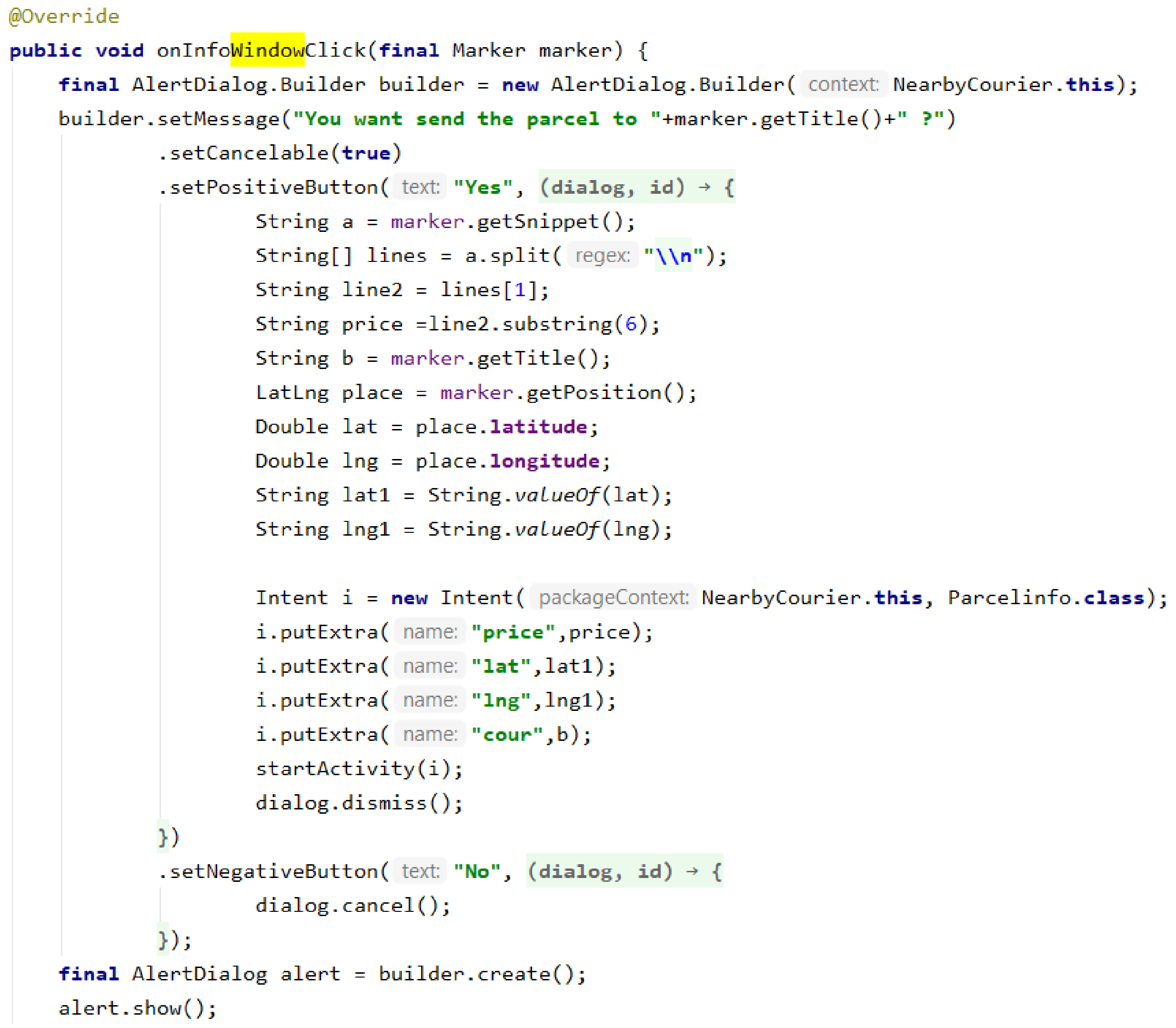
These codes will add the courier company information into the marker and display it when clicking on the marker.



```
@Override
public void onDirectionFinderSuccess(final List<Route> routes) {


    for (Route route : routes) {
        final String distance = String.valueOf(route.distance.text);
        final String duration = String.valueOf(route.duration.text);

        String snippet =
                "Distance: " + distance + "\n"
                        + "Estimate time reach: " + duration;

        Marker marker = googleMap.addMarker(new MarkerOptions()
                .title("Drive")
                .snippet(snippet)
                .position(route.endLocation)
                .flat(true)
                .icon(BitmapDescriptorFactory.fromResource(R.drawable.car2)));
        marker.setRotation(180);

    }

}
```

Figure 4.26 Screenshot of adding car icon contains distance details code

These codes will add the car icon on map that contains distance details. The distance and duration results are getting from Distance API.



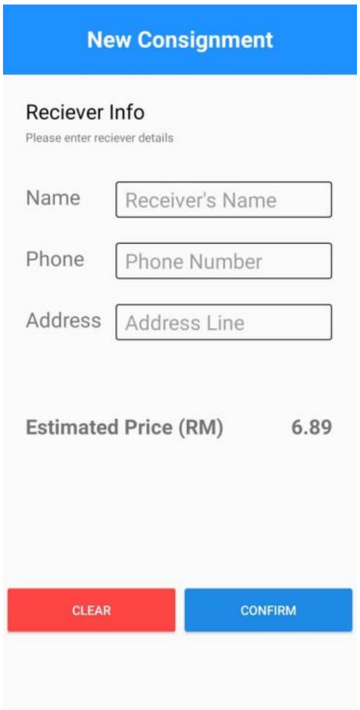
```
@Override
public void onInfoWindowClick(final Marker marker) {
    final AlertDialog.Builder builder = new AlertDialog.Builder( context: NearbyCourier.this);
    builder.setMessage("You want send the parcel to "+marker.getTitle()+" ?")
            .setCancelable(true)
            .setPositiveButton( text: "Yes", (dialog, id) → {
                    String a = marker.getSnippet();
                    String[] lines = a.split( regex: "\\n");
                    String line2 = lines[1];
                    String price =line2.substring(6);
                    String b = marker.getTitle();
                    LatLng place = marker.getPosition();
                    Double lat = place.latitude;
                    Double lng = place.longitude;
                    String lat1 = String.valueOf(lat);
                    String lng1 = String.valueOf(lng);

                    Intent i = new Intent( packageContext: NearbyCourier.this, Parcelinfo.class);
                    i.putExtra( name: "price",price);
                    i.putExtra( name: "lat",lat1);
                    i.putExtra( name: "lng",lng1);
                    i.putExtra( name: "cour",b);
                    startActivity(i);
                    dialog.dismiss();
            })
            .setNegativeButton( text: "No", (dialog, id) → {
                    dialog.cancel();
            });
    final AlertDialog alert = builder.create();
    alert.show();
```

Figure 4.27 Screenshot of alert dialog code

These codes will let the application display an alert dialog when user clicks the window info of the courier company marker. If user clicks no at the alert dialog, it will return to the map. However, when user clicks yes at the alert dialog, the system will pass the courier company information such as price, location and name to the next activity.

### 4.2.5 Parcel Info

#### 4.2.5.1 Parcelinfo.java



New Consignment

Reciever Info

Please enter reciever details

Name: Receiver's Name

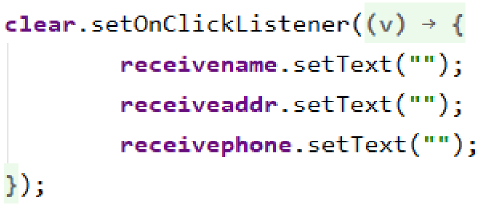
Phone: Phone Number

Address: Address Line

Estimated Price (RM) 6.89

Figure 4.28 Screenshot of parcel info page

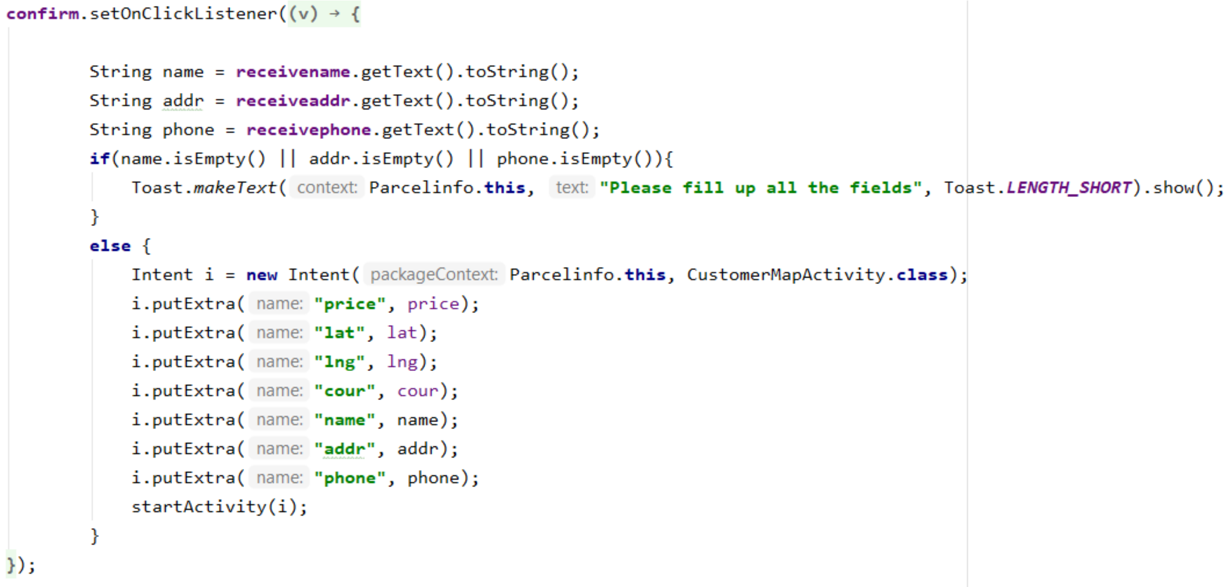
This is the parcel info page for users enter their parcel information such as receiver’s name, phone number and address. Following by the estimated price in RM calculated from the previous activity that using the courier company price multiply with the weight of the parcel. There are two buttons at the bottom of the page which is clear and confirms. Clear button is used to clear all the value entered in the fields and confirm button will bring user go to the next activity. An error message will show if the user does not fill up all the fields.



```
clear.setOnClickListener((v) → {
        receivename.setText("");
        receiveaddr.setText("");
        receivephone.setText("");
});
```

Figure 4.29 Screenshot of clear button code

When user clicks the clear button, the name, address and phone field will be clear.



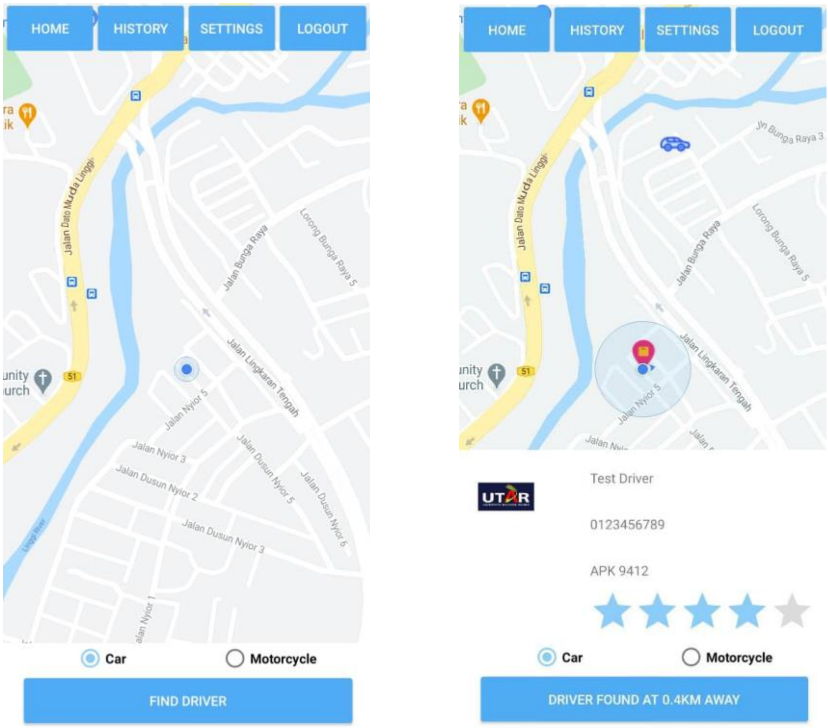
```
confirm.setOnClickListener((v) → {

        String name = receivename.getText().toString();
        String addr = receiveaddr.getText().toString();
        String phone = receivephone.getText().toString();
        if(name.isEmpty() || addr.isEmpty() || phone.isEmpty()){
            Toast.makeText( context: Parcelinfo.this,  text: "Please fill up all the fields", Toast.LENGTH_SHORT).show();
        }
        else {
            Intent i = new Intent( packageContext: Parcelinfo.this, CustomerMapActivity.class);
            i.putExtra( name: "price", price);
            i.putExtra( name: "lat", lat);
            i.putExtra( name: "lng", lng);
            i.putExtra( name: "cour", cour);
            i.putExtra( name: "name", name);
            i.putExtra( name: "addr", addr);
            i.putExtra( name: "phone", phone);
            startActivity(i);
        }
});
```

Figure 4.30 Screenshot of confirm button code

When the confirm button is clicked, the system will check all the fields to make sure it entered a value. An error message will show if one of the fields is empty. The application will pass the value of price, location, courier company name, receiver name, receiver address and receiver phone to the next activity.

### 4.2.6 Driver Request

#### 4.2.6.1 CustomerMapActivity.java



Test Driver

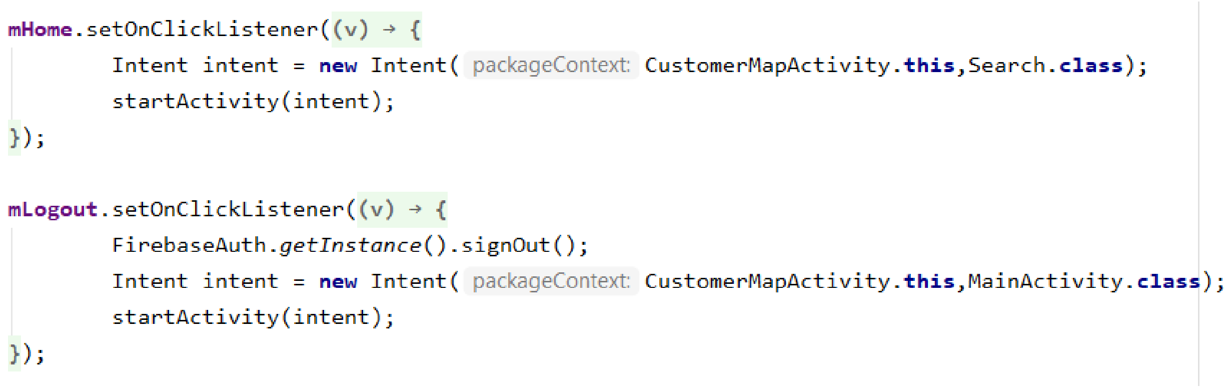
0123456789

APK 9412

★★★★☆

Figure 4.31 Screenshot of request driver page

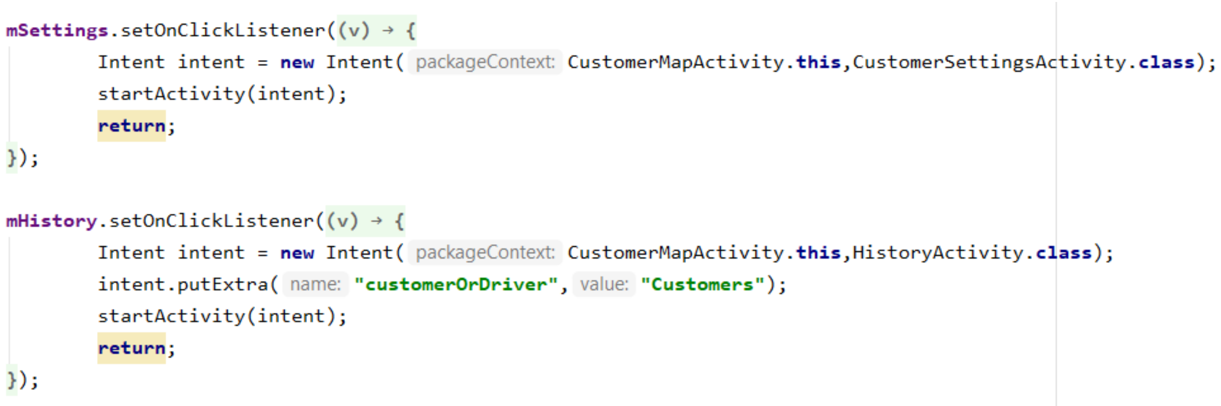
This is the page for users to find a car or motorcycle driver and the driver will send their parcel to the selected courier company location. There are four buttons at the top of the page which are home, history, settings and logout. The blue circle dot represents the user current location. When user clicks find driver button, the system will start searching the active driver around the user. The user will only get a driver within 10KM from current location. If a driver is found, a blue car icon will display and it represents the driver current location. Besides, the driver information such as picture, name, phone, rating and vehicle number will show when found a driver. User also may cancel the request by clicking the find driver button when found a driver.



```
mHome.setOnClickListener((v) → {
        Intent intent = new Intent( packageContext: CustomerMapActivity.this,Search.class);
        startActivity(intent);
});

mLogout.setOnClickListener((v) → {
        FirebaseAuth.getInstance().signOut();
        Intent intent = new Intent( packageContext: CustomerMapActivity.this,MainActivity.class);
        startActivity(intent);
});
```

Figure 4.32 Screenshot of home and logout button code

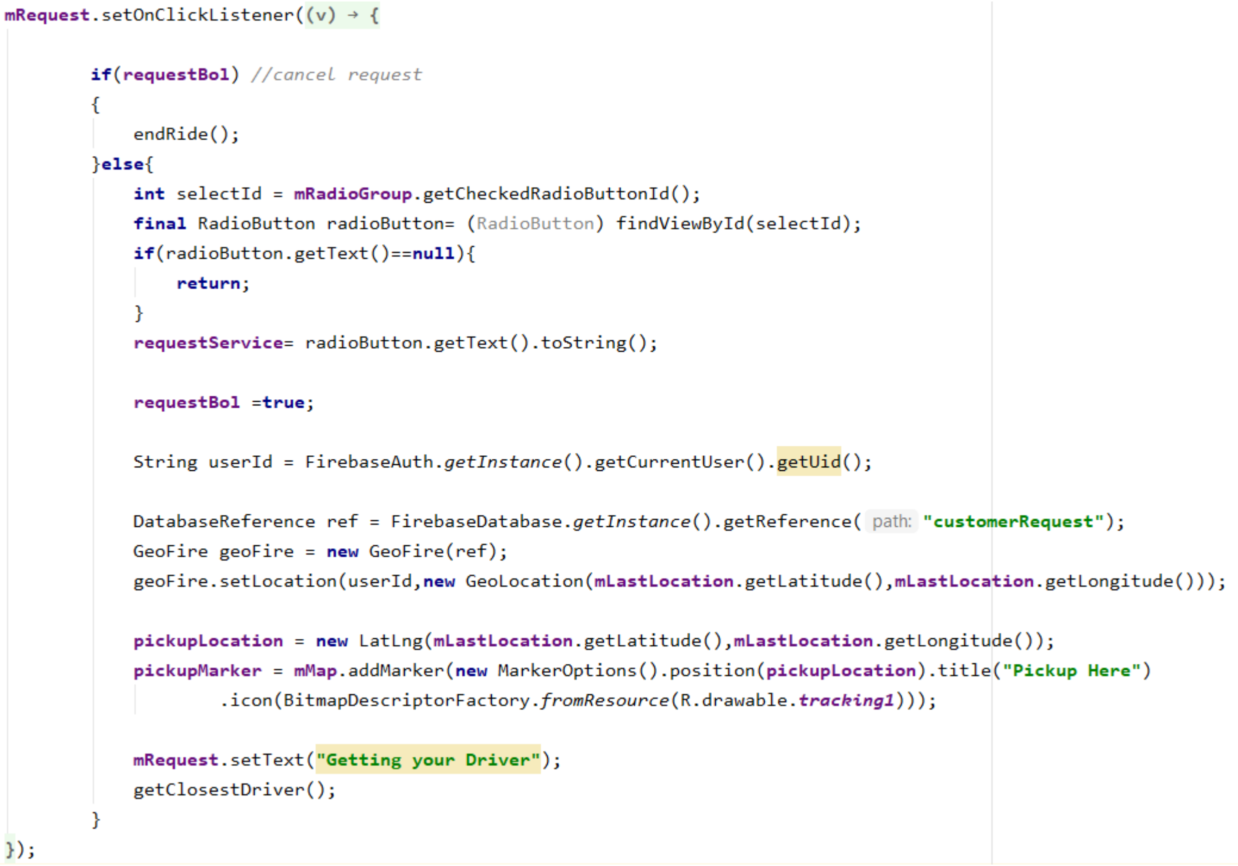
When user clicks the home button, the application will go back to the search activity. If a user wants to logout from the current account, they may click the logout button.



```
mSettings.setOnClickListener((v) → {
        Intent intent = new Intent( packageContext: CustomerMapActivity.this,CustomerSettingsActivity.class);
        startActivity(intent);
        return;
});

mHistory.setOnClickListener((v) → {
        Intent intent = new Intent( packageContext: CustomerMapActivity.this,HistoryActivity.class);
        intent.putExtra( name: "customerOrDriver", value: "Customers");
        startActivity(intent);
        return;
});
```

Figure 4.33 Screenshot of the settings and history button code

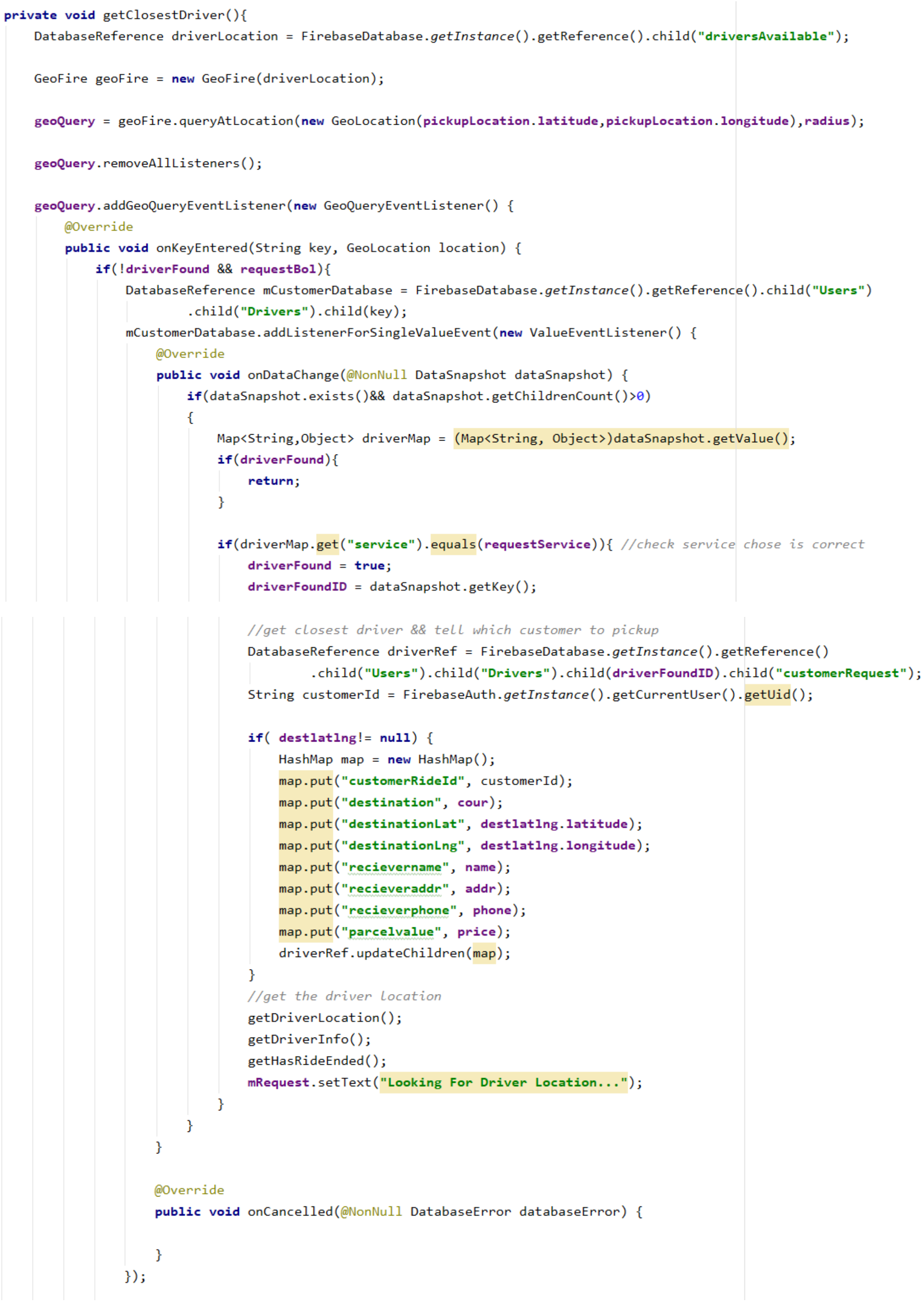
User may click the settings button to go to the account settings page and the history button will let user review their request history.



```
mRequest.setOnClickListener((v) → {

        if(requestBol) //cancel request
        {
            endRide();
        }else{
            int selectId = mRadioGroup.getCheckedRadioButtonId();
            final RadioButton radioButton= (RadioButton) findViewById(selectId);
            if(radioButton.getText()==null){
                return;
            }
            requestService= radioButton.getText().toString();

            requestBol =true;

            String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();

            DatabaseReference ref = FirebaseDatabase.getInstance().getReference( path: "customerRequest");
            GeoFire geoFire = new GeoFire(ref);
            geoFire.setLocation(userId,new GeoLocation(mLastLocation.getLatitude(),mLastLocation.getLongitude()));

            pickupLocation = new LatLng(mLastLocation.getLatitude(),mLastLocation.getLongitude());
            pickupMarker = mMap.addMarker(new MarkerOptions().position(pickupLocation).title("Pickup Here")
                    .icon(BitmapDescriptorFactory.fromResource(R.drawable.tracking1)));

            mRequest.setText("Getting your Driver");
            getClosestDriver();
        }
});
```

Figure 4.34 Screenshot of find driver button code

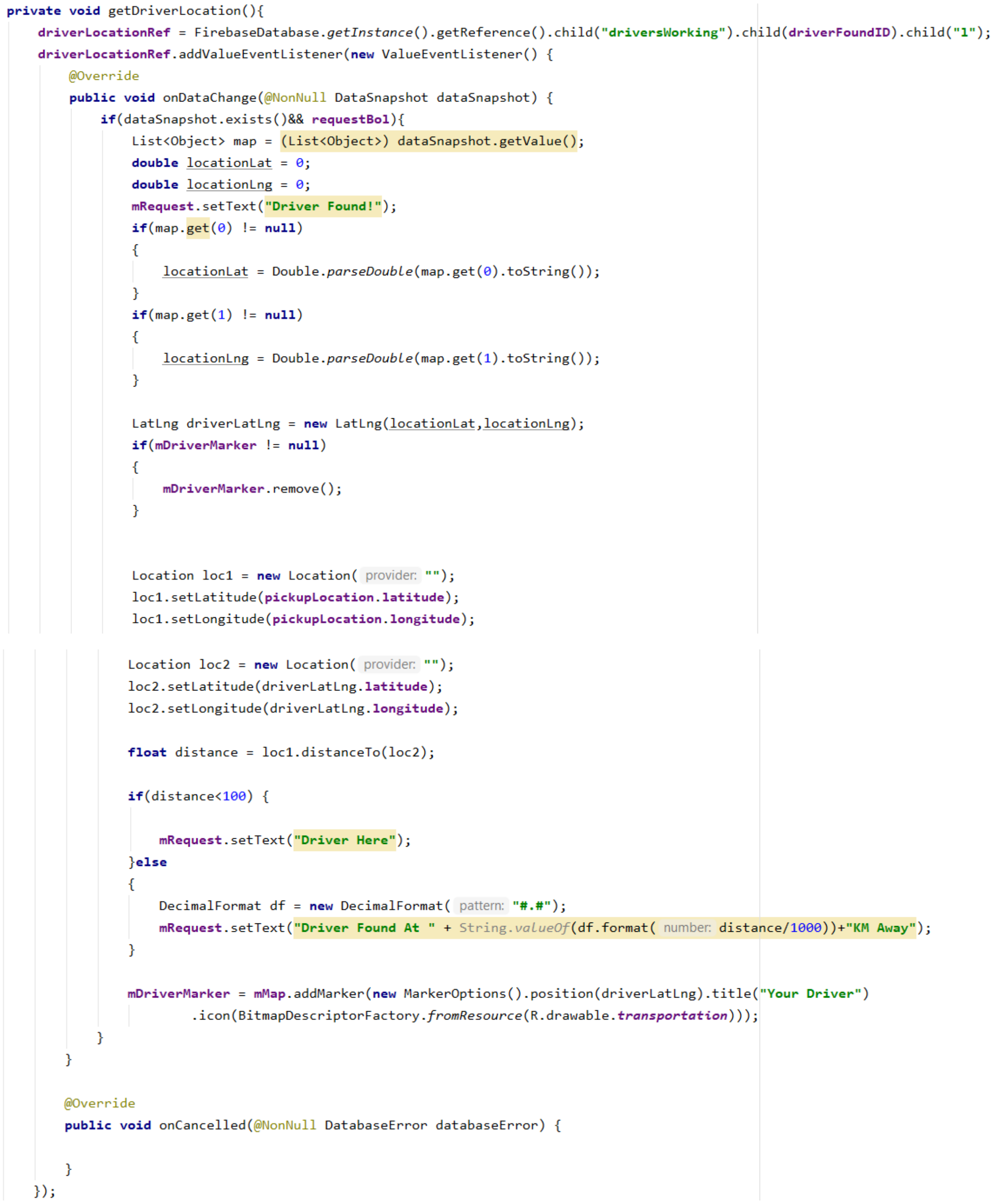
When user clicks the find driver button, the system first will check the user wants to find a car driver or motorcycle driver. After that, add the request information to the firebase database and look for an active driver.



```
private void getClosestDriver(){
    DatabaseReference driverLocation = FirebaseDatabase.getInstance().getReference().child("driversAvailable");

    GeoFire geoFire = new GeoFire(driverLocation);

    geoQuery = geoFire.queryAtLocation(new GeoLocation(pickupLocation.latitude,pickupLocation.longitude),radius);

    geoQuery.removeAllListeners();

    geoQuery.addGeoQueryEventListener(new GeoQueryEventListener() {
        @Override
        public void onKeyEntered(String key, GeoLocation location) {
            if(!driverFound && requestBol){
                DatabaseReference mCustomerDatabase = FirebaseDatabase.getInstance().getReference().child("Users")
                        .child("Drivers").child(key);
                mCustomerDatabase.addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                        if(dataSnapshot.exists()&& dataSnapshot.getChildrenCount()>0)
                        {
                            Map<String,Object> driverMap = (Map<String, Object>)dataSnapshot.getValue();
                            if(driverFound){
                                return;
                            }

                            if(driverMap.get("service").equals(requestService)){ //check service chose is correct
                                driverFound = true;
                                driverFoundID = dataSnapshot.getKey();

                                //get closest driver && tell which customer to pickup
                                DatabaseReference driverRef = FirebaseDatabase.getInstance().getReference()
                                        .child("Users").child("Drivers").child(driverFoundID).child("customerRequest");
                                String customerId = FirebaseAuth.getInstance().getCurrentUser().getUid();

                                if( destlatlng!= null) {
                                    HashMap map = new HashMap();
                                    map.put("customerRideId", customerId);
                                    map.put("destination", cour);
                                    map.put("destinationLat", destlatlng.latitude);
                                    map.put("destinationLng", destlatlng.longitude);
                                    map.put("recievername", name);
                                    map.put("recieveraddr", addr);
                                    map.put("recieverphone", phone);
                                    map.put("parcelvalue", price);
                                    driverRef.updateChildren(map);
                                }
                                //get the driver location
                                getDriverLocation();
                                getDriverInfo();
                                getHasRideEnded();
                                mRequest.setText("Looking For Driver Location...");
                            }
                        }
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError databaseError) {

                    }
                });
```

Figure 4.35 Screenshot of get the closest driver code

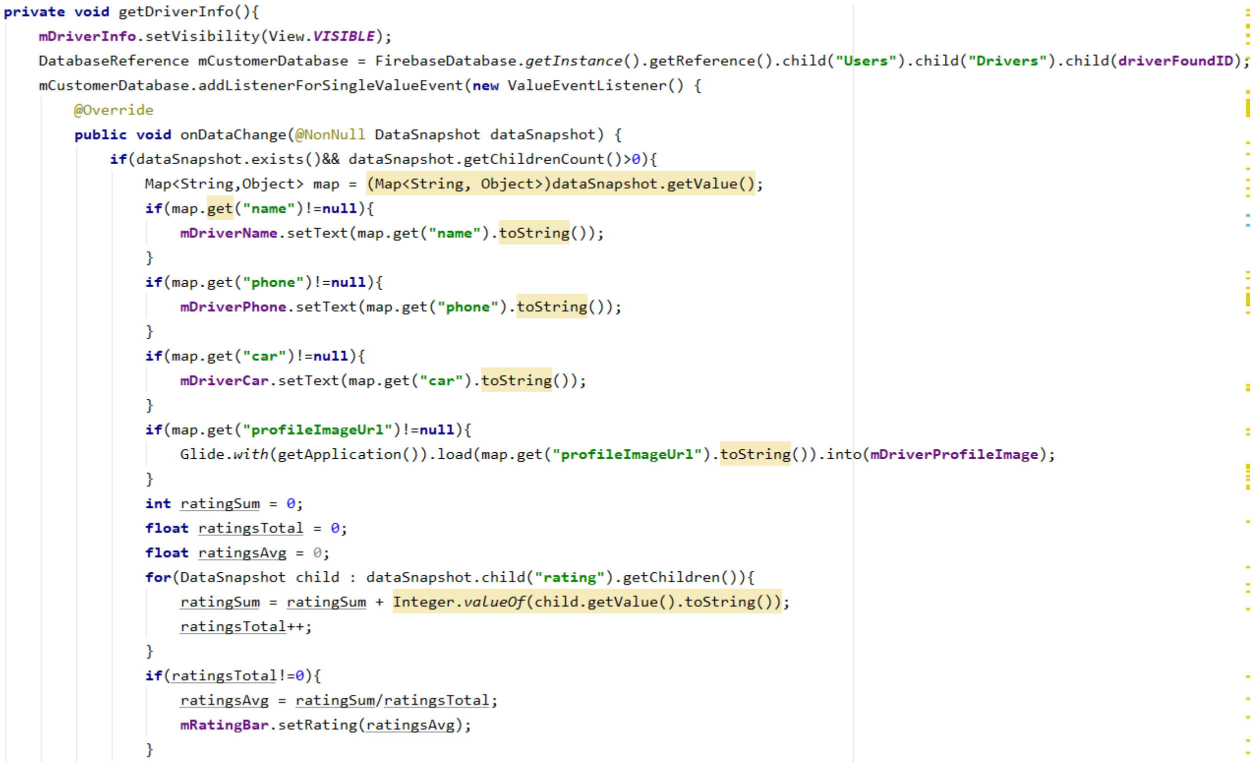
This function will help the user find the closest driver around and show the driver information. When an active driver is found in the database, the system will assign the driver to the user and send the user and parcel information to the driver database.



```
private void getDriverLocation(){
    driverLocationRef = FirebaseDatabase.getInstance().getReference().child("driversWorking").child(driverFoundID).child("1");
    driverLocationRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()&& requestBol){
                List<Object> map = (List<Object>) dataSnapshot.getValue();
                double locationLat = 0;
                double locationLng = 0;
                mRequest.setText("Driver Found!");
                if(map.get(0) != null)
                {
                    locationLat = Double.parseDouble(map.get(0).toString());
                }
                if(map.get(1) != null)
                {
                    locationLng = Double.parseDouble(map.get(1).toString());
                }

                LatLng driverLatLng = new LatLng(locationLat,locationLng);
                if(mDriverMarker != null)
                {
                    mDriverMarker.remove();
                }


                Location loc1 = new Location( provider: "");
                loc1.setLatitude(pickupLocation.latitude);
                loc1.setLongitude(pickupLocation.longitude);

                Location loc2 = new Location( provider: "");
                loc2.setLatitude(driverLatLng.latitude);
                loc2.setLongitude(driverLatLng.longitude);

                float distance = loc1.distanceTo(loc2);

                if(distance<100) {

                    mRequest.setText("Driver Here");
                }else
                {
                    DecimalFormat df = new DecimalFormat( pattern: "#.#");
                    mRequest.setText("Driver Found At " + String.valueOf(df.format( number: distance/1000))+"KM Away");
                }

                mDriverMarker = mMap.addMarker(new MarkerOptions().position(driverLatLng).title("Your Driver")
                        .icon(BitmapDescriptorFactory.fromResource(R.drawable.transportation)));
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
```

Figure 4.36 Screenshot of get the location of driver code

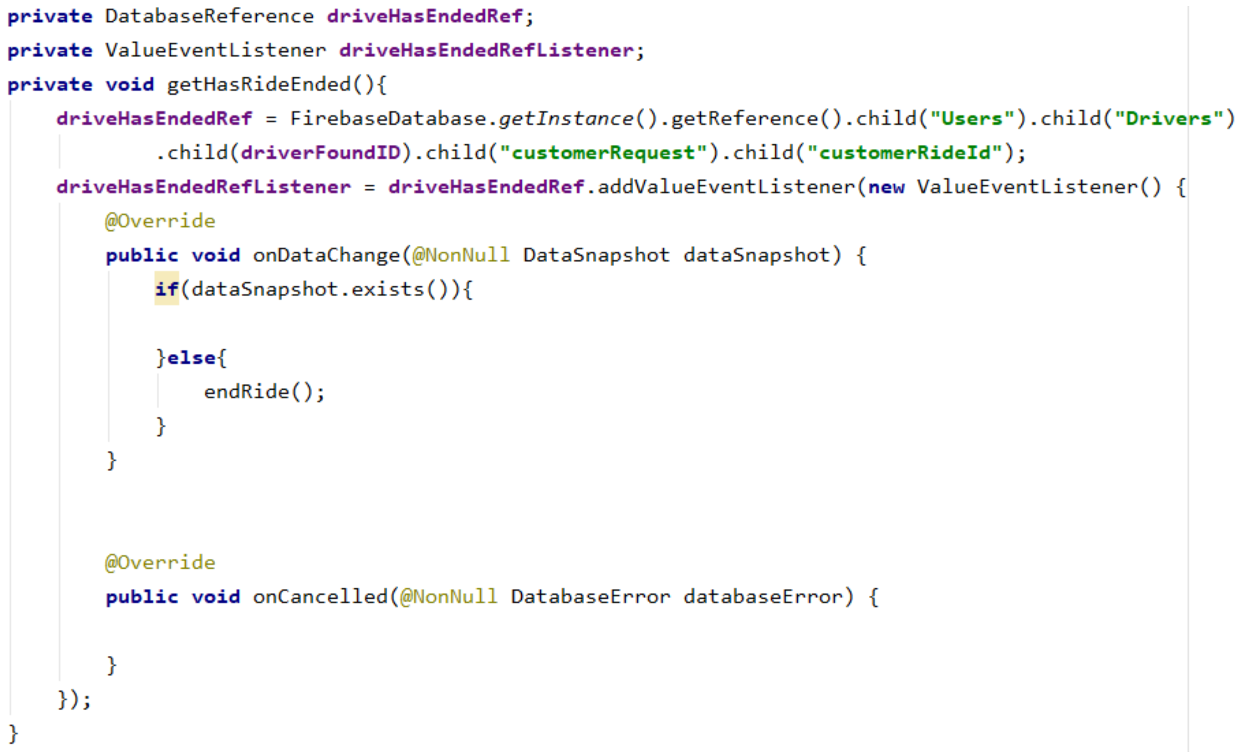
This function will help the user to get the driver current location and display a blue car icon on the map to represent the driver current location. Besides, the text of the find driver buttons also will be change to the distance between the user and driver.



```java
private void getDriverInfo(){
    mDriverInfo.setVisibility(View.VISIBLE);
    DatabaseReference mCustomerDatabase = FirebaseDatabase.getInstance().getReference().child("Users").child("Drivers").child(driverFoundID);
    mCustomerDatabase.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()&& dataSnapshot.getChildrenCount()>0){
                Map<String,Object> map = (Map<String, Object>)dataSnapshot.getValue();
                if(map.get("name")!=null){
                    mDriverName.setText(map.get("name").toString());
                }
                if(map.get("phone")!=null){
                    mDriverPhone.setText(map.get("phone").toString());
                }
                if(map.get("car")!=null){
                    mDriverCar.setText(map.get("car").toString());
                }
                if(map.get("profileImageUrl")!=null){
                    Glide.with(getApplication()).load(map.get("profileImageUrl").toString()).into(mDriverProfileImage);
                }
                int ratingSum = 0;
                float ratingsTotal = 0;
                float ratingsAvg = 0;
                for(DataSnapshot child : dataSnapshot.child("rating").getChildren()){
                    ratingSum = ratingSum + Integer.valueOf(child.getValue().toString());
                    ratingsTotal++;
                }
                if(ratingsTotal!=0){
                    ratingsAvg = ratingSum/ratingsTotal;
                    mRatingBar.setRating(ratingsAvg);
                }
```

Figure 4.37 Screenshot of get the driver details code

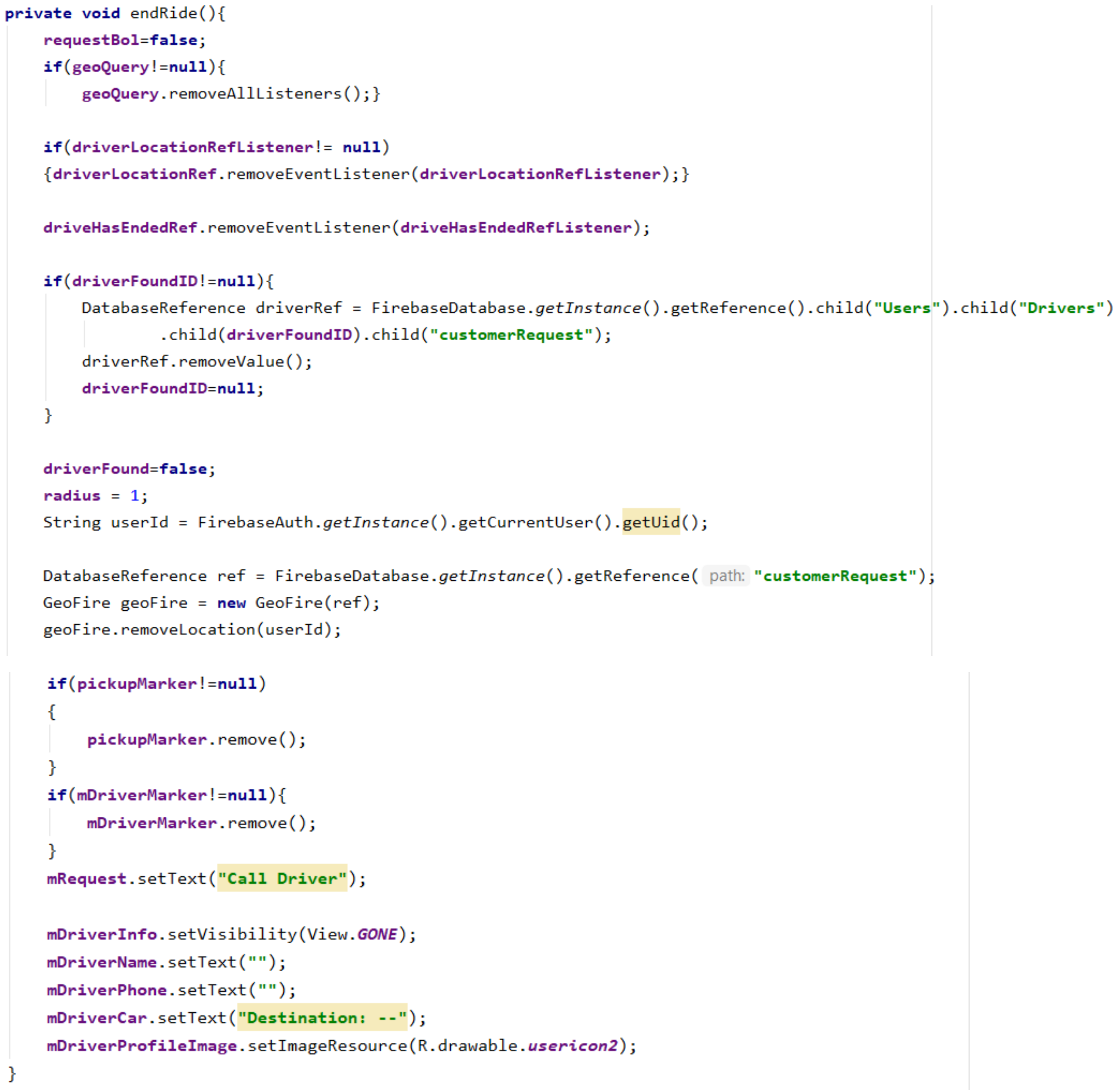
This function will help the user get the information of the driver found such as name, phone number, vehicle number, picture and also average rating value.



```
private DatabaseReference driveHasEndedRef;
private ValueEventListener driveHasEndedRefListener;
private void getHasRideEnded(){
    driveHasEndedRef = FirebaseDatabase.getInstance().getReference().child("Users").child("Drivers")
            .child(driverFoundID).child("customerRequest").child("customerRideId");
    driveHasEndedRefListener = driveHasEndedRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()){

            }else{
                endRide();
            }
        }


        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}
```

Figure 4.38 Screenshot of check driver progress code

This function will check whether the driver is completed to send the parcel to the courier company or not. The system will check the existence of the request id in the database. If the request id still exists which means the driver not yet complete send the parcel. If the request id is not found in the database, it will call endRide() function.



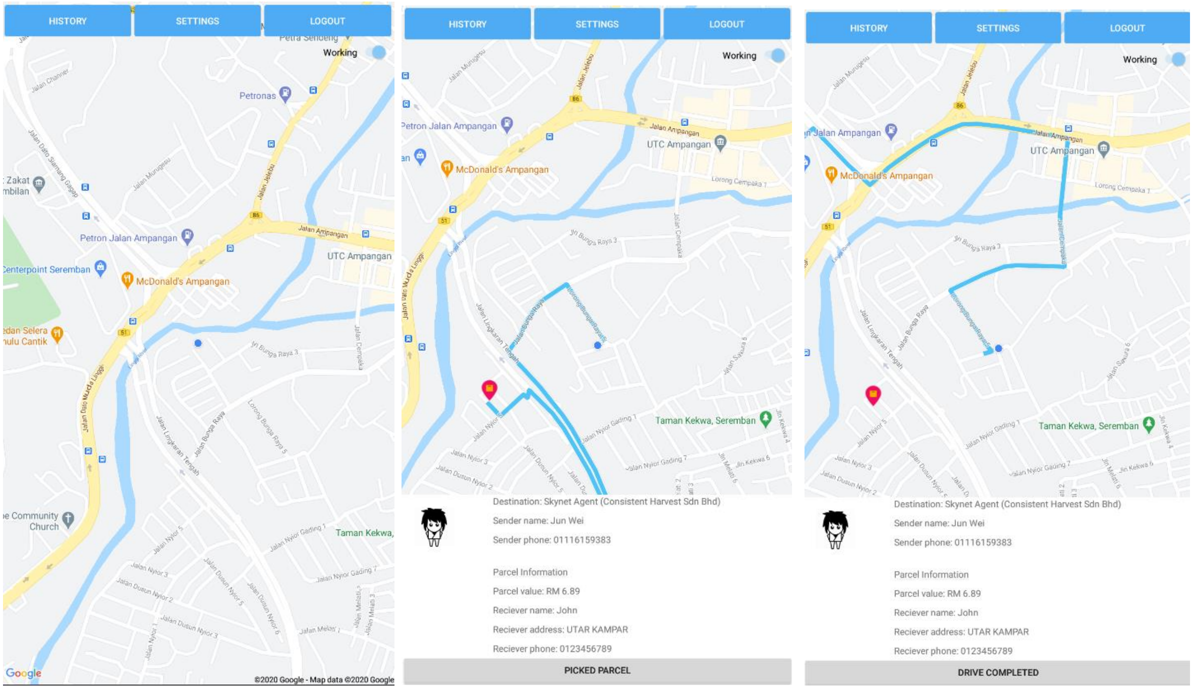
```
private void endRide(){
    requestBol=false;
    if(geoQuery!=null){
        geoQuery.removeAllListeners();}

    if(driverLocationRefListener!= null)
    {driverLocationRef.removeEventListener(driverLocationRefListener);}

    driveHasEndedRef.removeEventListener(driveHasEndedRefListener);

    if(driverFoundID!=null){
        DatabaseReference driverRef = FirebaseDatabase.getInstance().getReference().child("Users").child("Drivers")
                .child(driverFoundID).child("customerRequest");
        driverRef.removeValue();
        driverFoundID=null;
    }

    driverFound=false;
    radius = 1;
    String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();

    DatabaseReference ref = FirebaseDatabase.getInstance().getReference( path: "customerRequest");
    GeoFire geoFire = new GeoFire(ref);
    geoFire.removeLocation(userId);
    if(pickupMarker!=null)
    {
        pickupMarker.remove();
    }
    if(mDriverMarker!=null){
        mDriverMarker.remove();
    }
    mRequest.setText("Call Driver");

    mDriverInfo.setVisibility(View.GONE);
    mDriverName.setText("");
    mDriverPhone.setText("");
    mDriverCar.setText("Destination: --");
    mDriverProfileImage.setImageResource(R.drawable.usericon2);
}
```

Figure 4.39 Screenshot of remove request value in database code

This function will remove all the request data in the database created by the user and make the radius, marker become the default value.

### 4.2.7 Driver working map

#### 4.2.7.1 DriverMapNew.java



Destination: Skynet Agent (Consistent Harvest Sdn Bhd)
Sender name: Jun Wei
Sender phone: 01116159383

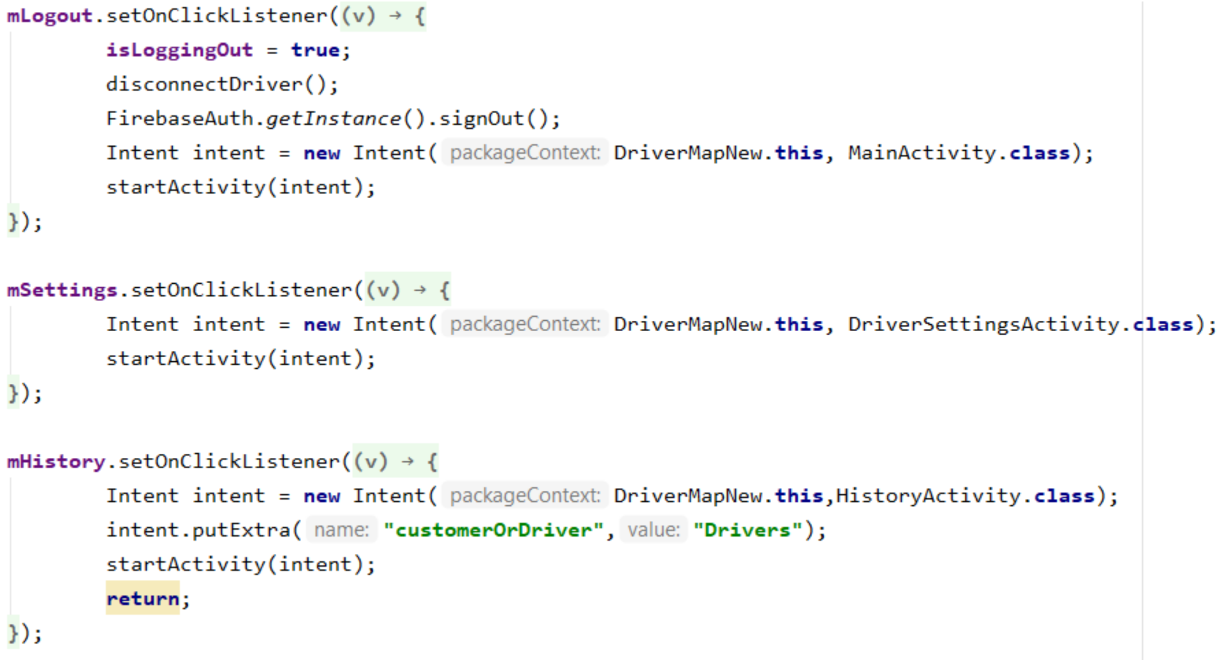
Parcel Information
Parcel value: RM 6.89
Reciever name: John
Reciever address: UTAR KAMPAR
Reciever phone: 0123456789

Destination: Skynet Agent (Consistent Harvest Sdn Bhd)
Sender name: Jun Wei
Sender phone: 01116159383

Parcel Information
Parcel value: RM 6.89
Reciever name: John
Reciever address: UTAR KAMPAR
Reciever phone: 0123456789

Figure 4.40 Screenshot of driver working map

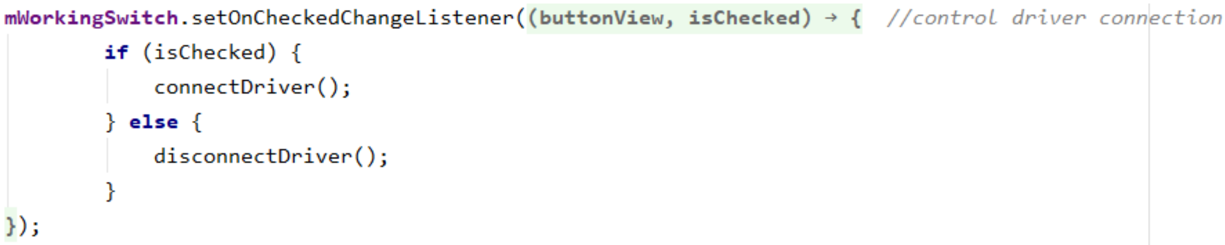
This page will be shown when a driver user successfully login through the driver login page. There are three buttons on the top of this page which are history, settings and logout. Besides, on the top right of the page have a working switch for the driver to change their working status. A driver will receive a user request while the working switch is on. When the user request is found, the page will show the user and parcel information such as destination, sender name and sender phone number. Besides, a blue line will draw on the map to help driver reach user current location and also the destination.



```
mLogout.setOnClickListener((v) → {
        isLoggingOut = true;
        disconnectDriver();
        FirebaseAuth.getInstance().signOut();
        Intent intent = new Intent( packageContext: DriverMapNew.this, MainActivity.class);
        startActivity(intent);
});

mSettings.setOnClickListener((v) → {
        Intent intent = new Intent( packageContext: DriverMapNew.this, DriverSettingsActivity.class);
        startActivity(intent);
});

mHistory.setOnClickListener((v) → {
        Intent intent = new Intent( packageContext: DriverMapNew.this,HistoryActivity.class);
        intent.putExtra( name: "customerOrDriver", value: "Drivers");
        startActivity(intent);
        return;
});
```

Figure 4.41 Screenshot of logout, driver settings and history button code

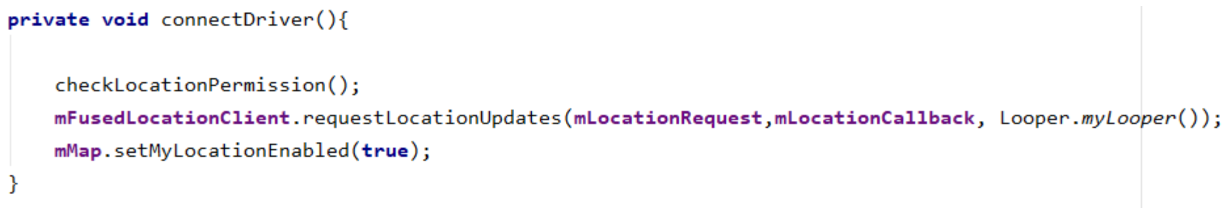
When driver clicks the logout button, the application will go to main page and run disconnectDriver() function. The settings button will bring driver go to the driver settings activity and the history button will bring driver to the history page to review their ride.



```
mWorkingSwitch.setOnCheckedChangeListener((buttonView, isChecked) → {  //control driver connection
        if (isChecked) {
            connectDriver();
        } else {
            disconnectDriver();
        }
});
```

Figure 4.42 Screenshot of working switch code

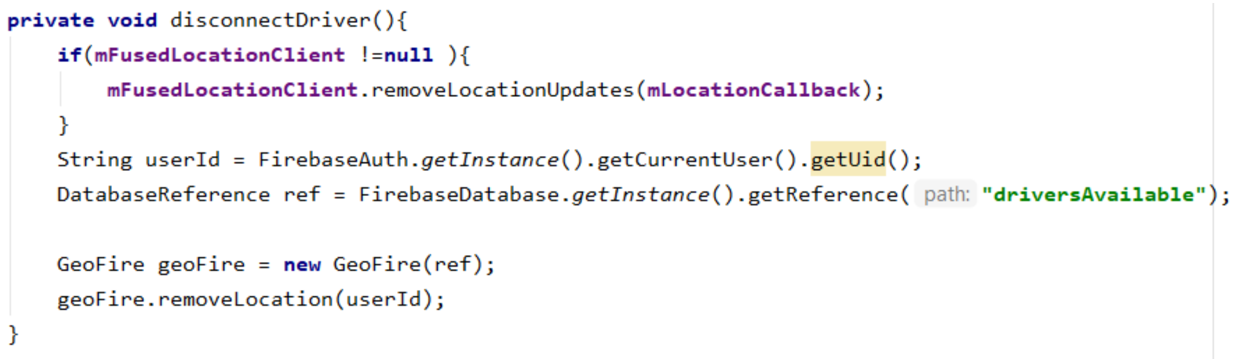
When the working switch is on, the driver status will become active in the database. If driver wants to stop working, they may turn off the working switch so they will not receive any request from user.



```java
private void connectDriver(){

    checkLocationPermission();
    mFusedLocationClient.requestLocationUpdates(mLocationRequest,mLocationCallback, Looper.myLooper());
    mMap.setMyLocationEnabled(true);
}
```

Figure 4.43 Screenshot of connect driver function code

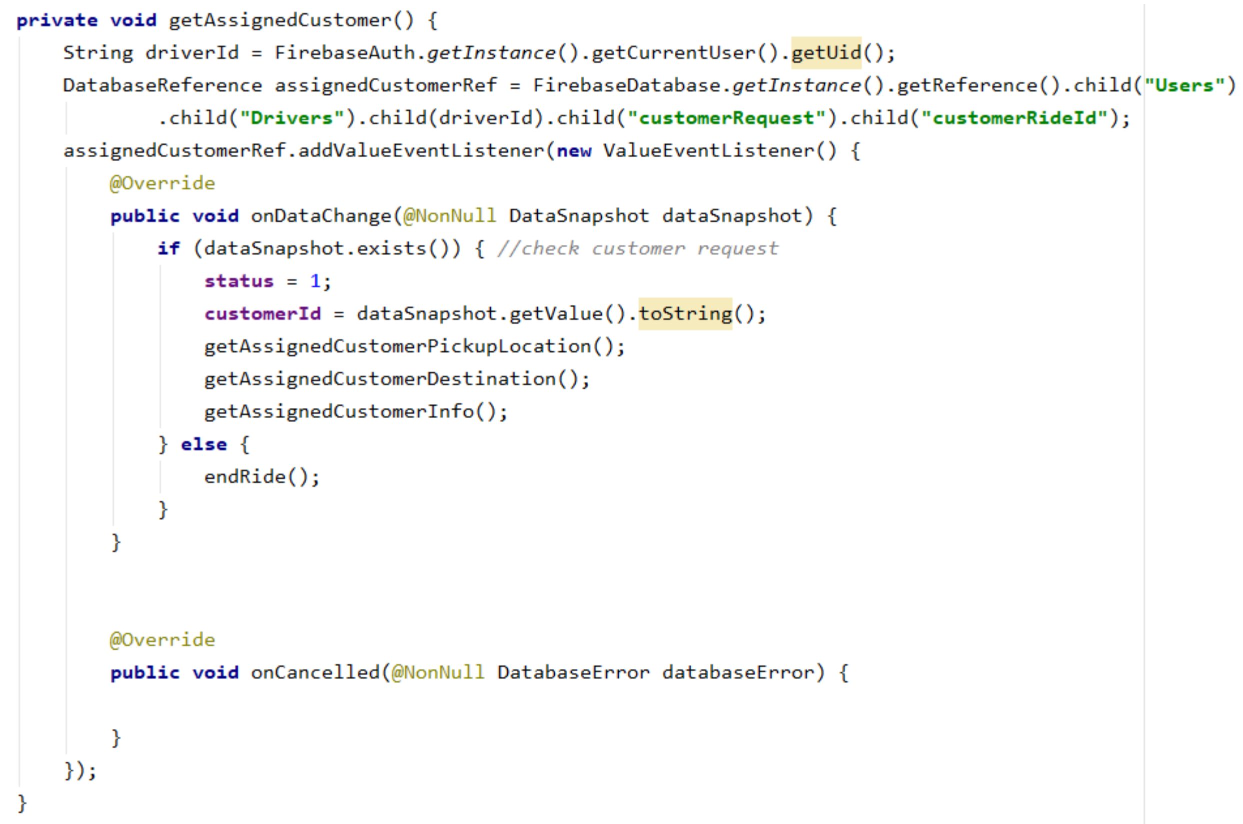
This function will first check the location permission of the device then continuously update self-location.



```java
private void disconnectDriver(){
    if(mFusedLocationClient !=null ){
        mFusedLocationClient.removeLocationUpdates(mLocationCallback);
    }
    String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();
    DatabaseReference ref = FirebaseDatabase.getInstance().getReference( path: "driversAvailable");

    GeoFire geoFire = new GeoFire(ref);
    geoFire.removeLocation(userId);
}
```

Figure 4.44 Screenshot of disconnect driver function code

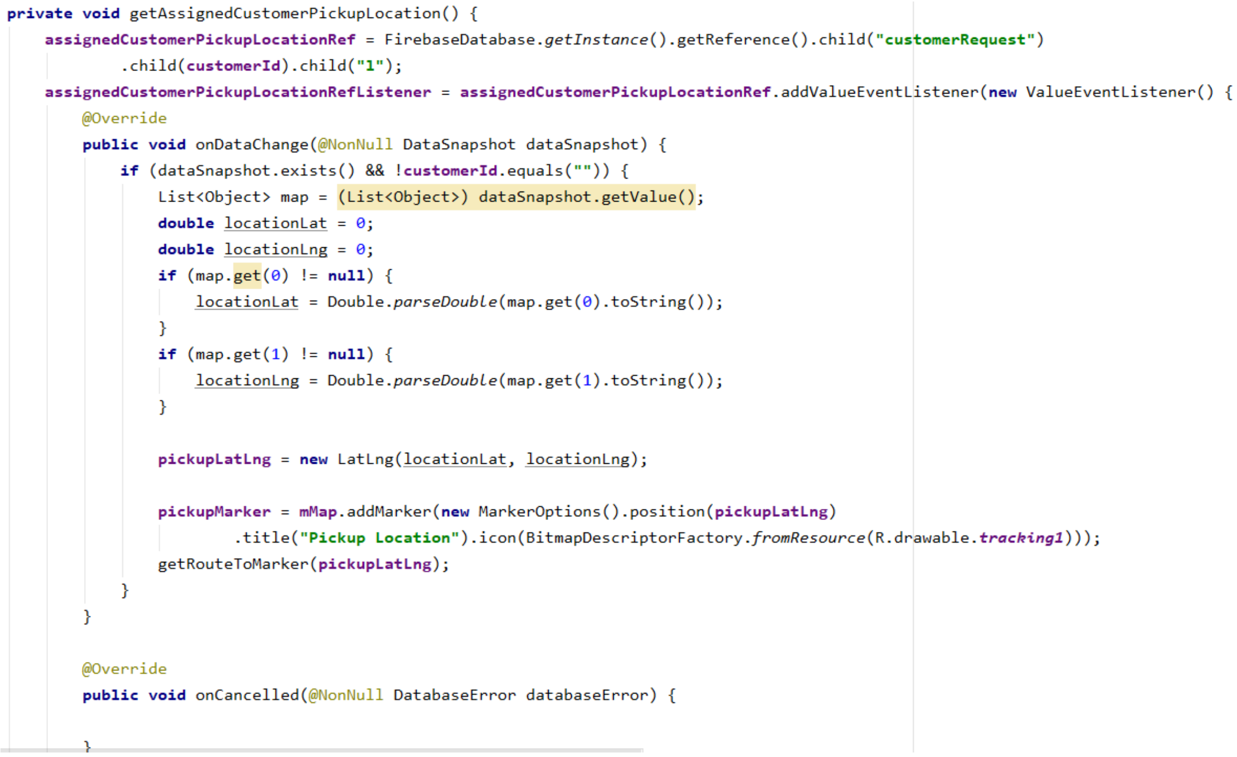
This function will remove the active status of the driver in the database so that the application will not receiving any incoming request from user.



```
private void getAssignedCustomer() {
    String driverId = FirebaseAuth.getInstance().getCurrentUser().getUid();
    DatabaseReference assignedCustomerRef = FirebaseDatabase.getInstance().getReference().child("Users")
            .child("Drivers").child(driverId).child("customerRequest").child("customerRideId");
    assignedCustomerRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (dataSnapshot.exists()) { //check customer request
                status = 1;
                customerId = dataSnapshot.getValue().toString();
                getAssignedCustomerPickupLocation();
                getAssignedCustomerDestination();
                getAssignedCustomerInfo();
            } else {
                endRide();
            }
        }


        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}
```

Figure 4.45 Screenshot of receive user request function code

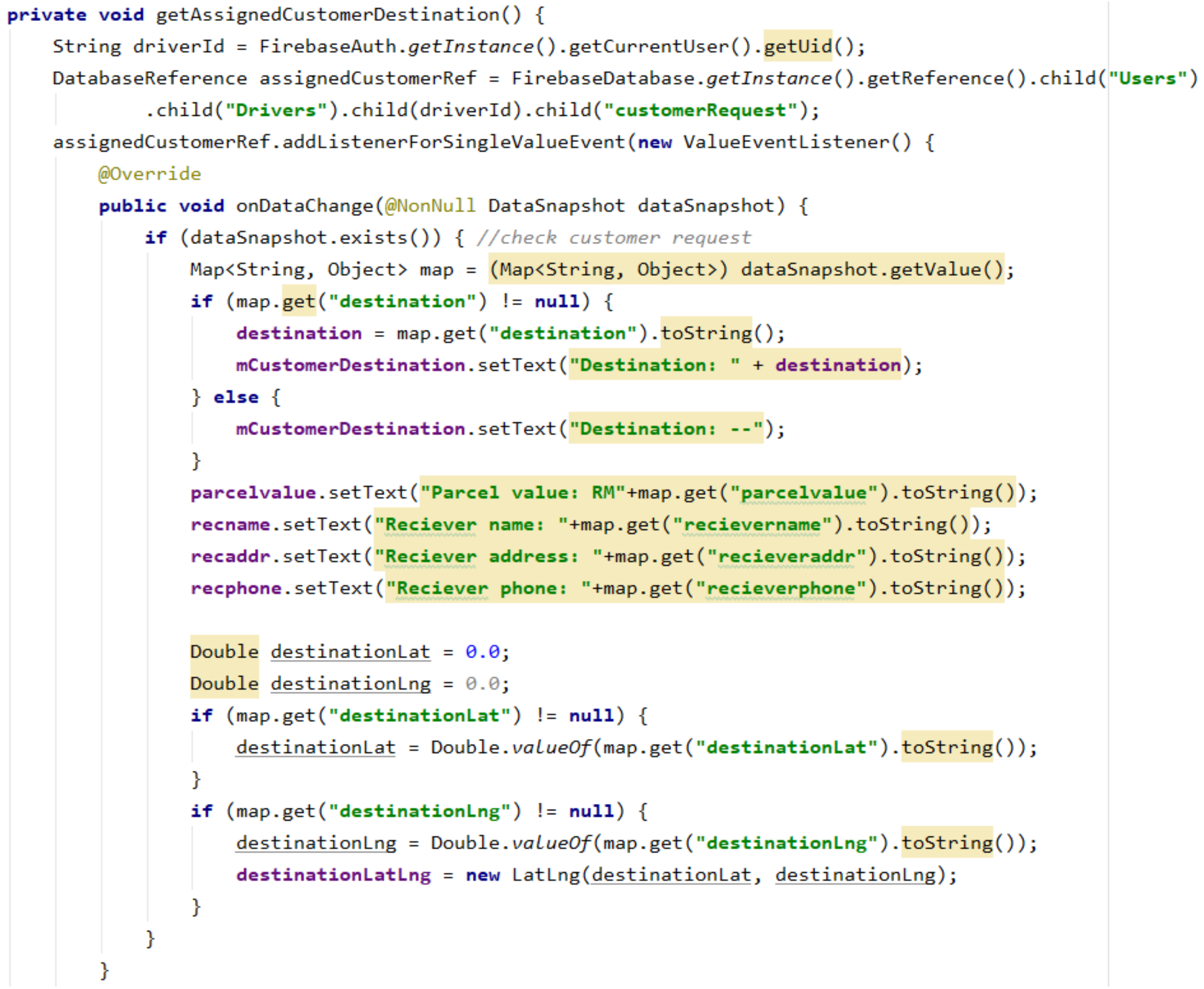
This function used to help driver check the user request data in the database. If the user request exists, then the driver will get the customer location, parcel information and customer information.



```
private void getAssignedCustomerPickupLocation() {
    assignedCustomerPickupLocationRef = FirebaseDatabase.getInstance().getReference().child("customerRequest")
            .child(customerId).child("l");
    assignedCustomerPickupLocationRefListener = assignedCustomerPickupLocationRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (dataSnapshot.exists() && !customerId.equals("")) {
                List<Object> map = (List<Object>) dataSnapshot.getValue();
                double locationLat = 0;
                double locationLng = 0;
                if (map.get(0) != null) {
                    locationLat = Double.parseDouble(map.get(0).toString());
                }
                if (map.get(1) != null) {
                    locationLng = Double.parseDouble(map.get(1).toString());
                }

                pickupLatLng = new LatLng(locationLat, locationLng);

                pickupMarker = mMap.addMarker(new MarkerOptions().position(pickupLatLng)
                        .title("Pickup Location").icon(BitmapDescriptorFactory.fromResource(R.drawable.tracking1)));
                getRouteToMarker(pickupLatLng);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
```

Figure 4.46 Screenshot of get request location function code

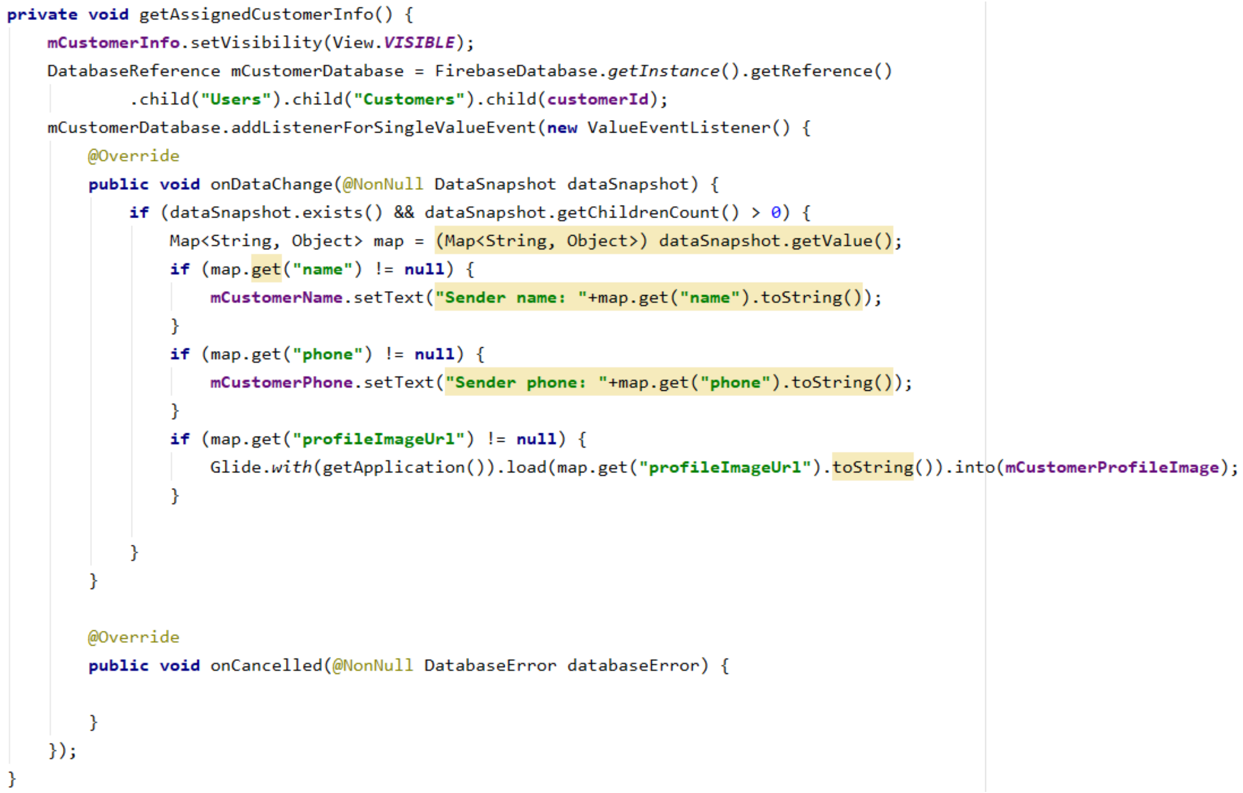
This function used to get the customer’s location that store at database and display on the map. Besides, a red marker also will point on the map to help driver easier to find the customer’s location.



```
private void getAssignedCustomerDestination() {
    String driverId = FirebaseAuth.getInstance().getCurrentUser().getUid();
    DatabaseReference assignedCustomerRef = FirebaseDatabase.getInstance().getReference().child("Users")
            .child("Drivers").child(driverId).child("customerRequest");
    assignedCustomerRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (dataSnapshot.exists()) { //check customer request
                Map<String, Object> map = (Map<String, Object>) dataSnapshot.getValue();
                if (map.get("destination") != null) {
                    destination = map.get("destination").toString();
                    mCustomerDestination.setText("Destination: " + destination);
                } else {
                    mCustomerDestination.setText("Destination: --");
                }
                parcelvalue.setText("Parcel value: RM"+map.get("parcelvalue").toString());
                recname.setText("Reciever name: "+map.get("recievername").toString());
                recaddr.setText("Reciever address: "+map.get("recieveraddr").toString());
                recphone.setText("Reciever phone: "+map.get("recieverphone").toString());

                Double destinationLat = 0.0;
                Double destinationLng = 0.0;
                if (map.get("destinationLat") != null) {
                    destinationLat = Double.valueOf(map.get("destinationLat").toString());
                }
                if (map.get("destinationLng") != null) {
                    destinationLng = Double.valueOf(map.get("destinationLng").toString());
                    destinationLatLng = new LatLng(destinationLat, destinationLng);
                }
            }
        }
```

Figure 4.47 Screenshot of get parcel information code

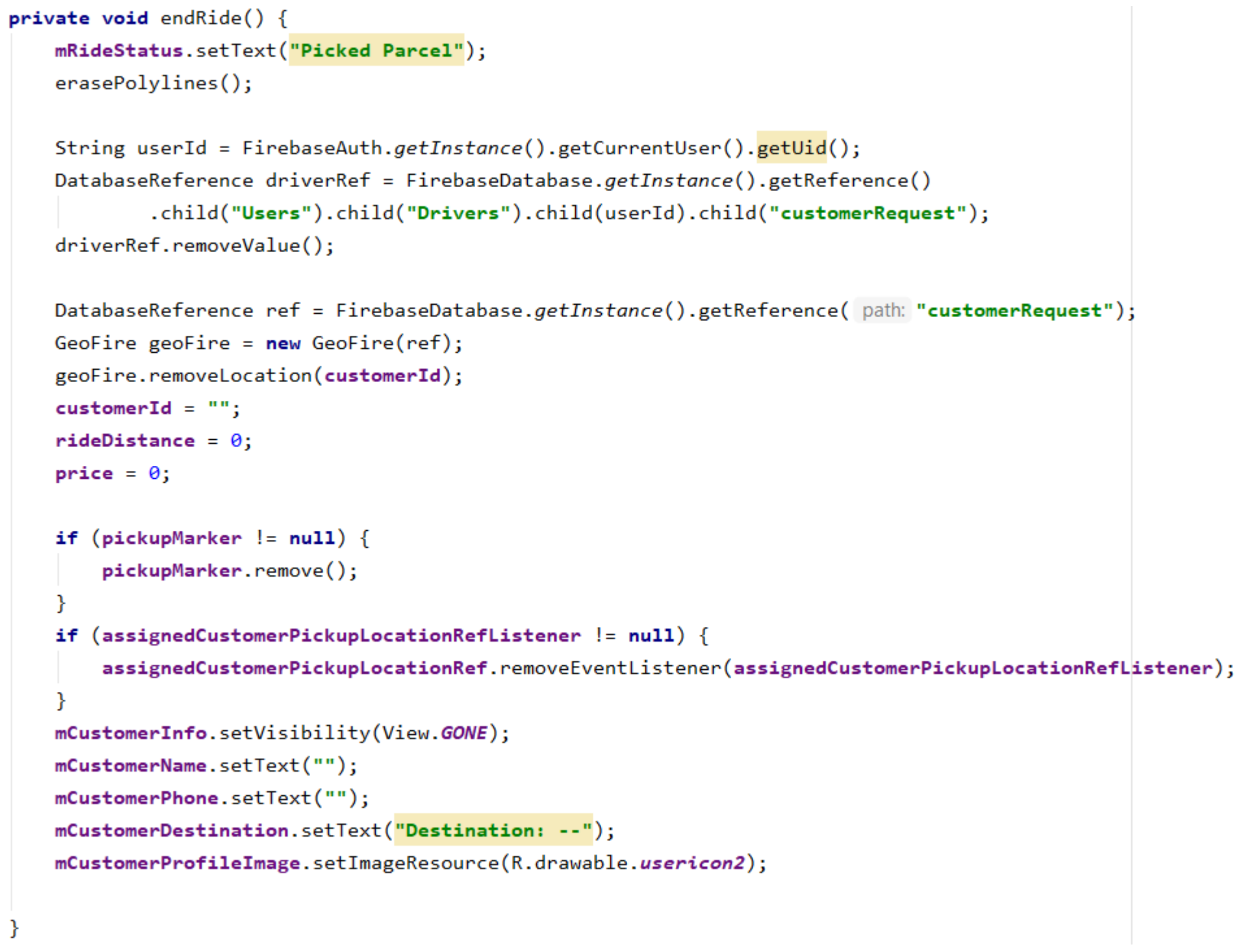
This function will be trigger when driver picks up the parcel from the user. It will get the destination of the parcel from the database and parcel information from the database and show on the working map.



```
private void getAssignedCustomerInfo() {
    mCustomerInfo.setVisibility(View.VISIBLE);
    DatabaseReference mCustomerDatabase = FirebaseDatabase.getInstance().getReference()
            .child("Users").child("Customers").child(customerId);
    mCustomerDatabase.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (dataSnapshot.exists() && dataSnapshot.getChildrenCount() > 0) {
                Map<String, Object> map = (Map<String, Object>) dataSnapshot.getValue();
                if (map.get("name") != null) {
                    mCustomerName.setText("Sender name: "+map.get("name").toString());
                }
                if (map.get("phone") != null) {
                    mCustomerPhone.setText("Sender phone: "+map.get("phone").toString());
                }
                if (map.get("profileImageUrl") != null) {
                    Glide.with(getApplication()).load(map.get("profileImageUrl").toString()).into(mCustomerProfileImage);
                }

            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}
```

Figure 4.48 Screenshot of get user information code

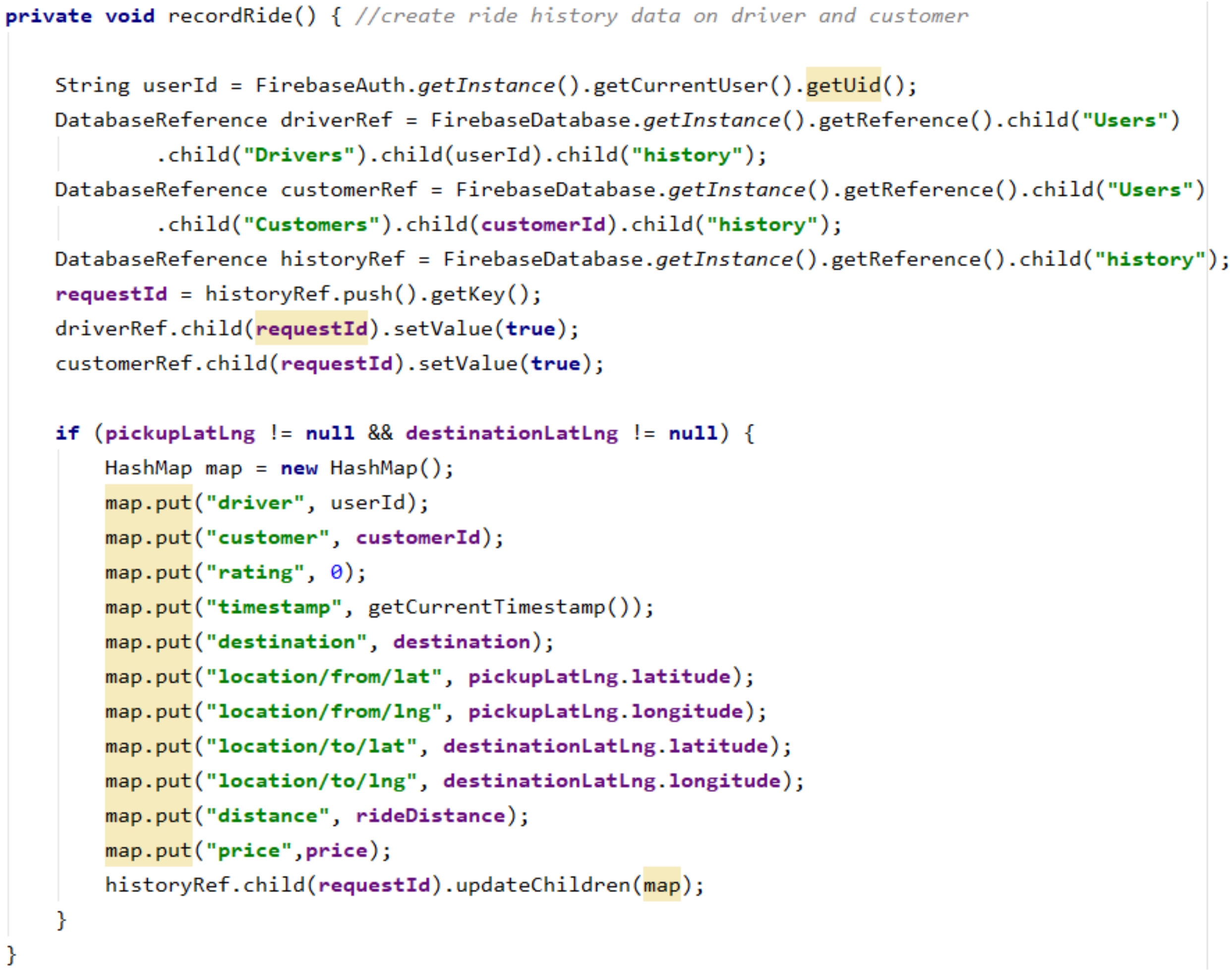
This function will get the user information of the request id such as name, phone and picture.



```
private void endRide() {
    mRideStatus.setText("Picked Parcel");
    erasePolylines();

    String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();
    DatabaseReference driverRef = FirebaseDatabase.getInstance().getReference()
            .child("Users").child("Drivers").child(userId).child("customerRequest");
    driverRef.removeValue();

    DatabaseReference ref = FirebaseDatabase.getInstance().getReference( path: "customerRequest");
    GeoFire geoFire = new GeoFire(ref);
    geoFire.removeLocation(customerId);
    customerId = "";
    rideDistance = 0;
    price = 0;

    if (pickupMarker != null) {
        pickupMarker.remove();
    }
    if (assignedCustomerPickupLocationRefListener != null) {
        assignedCustomerPickupLocationRef.removeEventListener(assignedCustomerPickupLocationRefListener);
    }
    mCustomerInfo.setVisibility(View.GONE);
    mCustomerName.setText("");
    mCustomerPhone.setText("");
    mCustomerDestination.setText("Destination: --");
    mCustomerProfileImage.setImageResource(R.drawable.usericon2);

}
```

Figure 4.49 Screenshot of finish send parcel code

This function will be trigger when driver done to send the parcel to the request location. It will remove the user request data in the database and clear all the markers on the map.



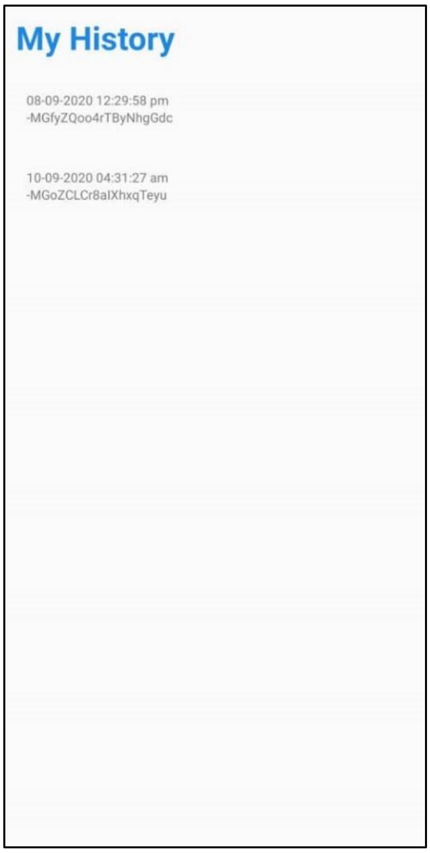
```
private void recordRide() { //create ride history data on driver and customer

    String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();
    DatabaseReference driverRef = FirebaseDatabase.getInstance().getReference().child("Users")
            .child("Drivers").child(userId).child("history");
    DatabaseReference customerRef = FirebaseDatabase.getInstance().getReference().child("Users")
            .child("Customers").child(customerId).child("history");
    DatabaseReference historyRef = FirebaseDatabase.getInstance().getReference().child("history");
    requestId = historyRef.push().getKey();
    driverRef.child(requestId).setValue(true);
    customerRef.child(requestId).setValue(true);

    if (pickupLatLng != null && destinationLatLng != null) {
        HashMap map = new HashMap();
        map.put("driver", userId);
        map.put("customer", customerId);
        map.put("rating", 0);
        map.put("timestamp", getCurrentTimestamp());
        map.put("destination", destination);
        map.put("location/from/lat", pickupLatLng.latitude);
        map.put("location/from/lng", pickupLatLng.longitude);
        map.put("location/to/lat", destinationLatLng.latitude);
        map.put("location/to/lng", destinationLatLng.longitude);
        map.put("distance", rideDistance);
        map.put("price",price);
        historyRef.child(requestId).updateChildren(map);
    }
}
```

Figure 4.50 Screenshot of record ride details code

This function will save the ride details into driver history database and user history database. The information includes driver id, customer id, parcel destination, pick up location, ride distance and ride fare.

### 4.2.8 History

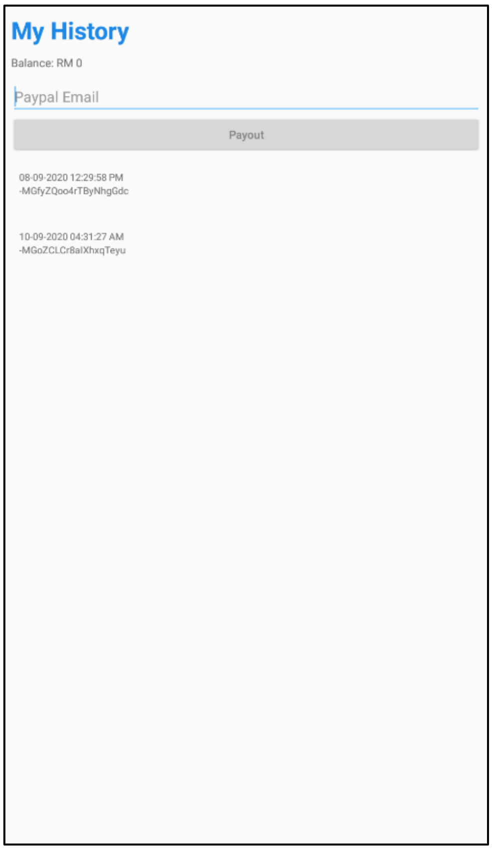
#### 4.2.8.1 HistoryActivity.java



My History

08-09-2020 12:29:58 pm
-MGfyZQoo4rTByNhgGdc

10-09-2020 04:31:27 am
-MGoZCLCr8alXhxqTeyu



My History

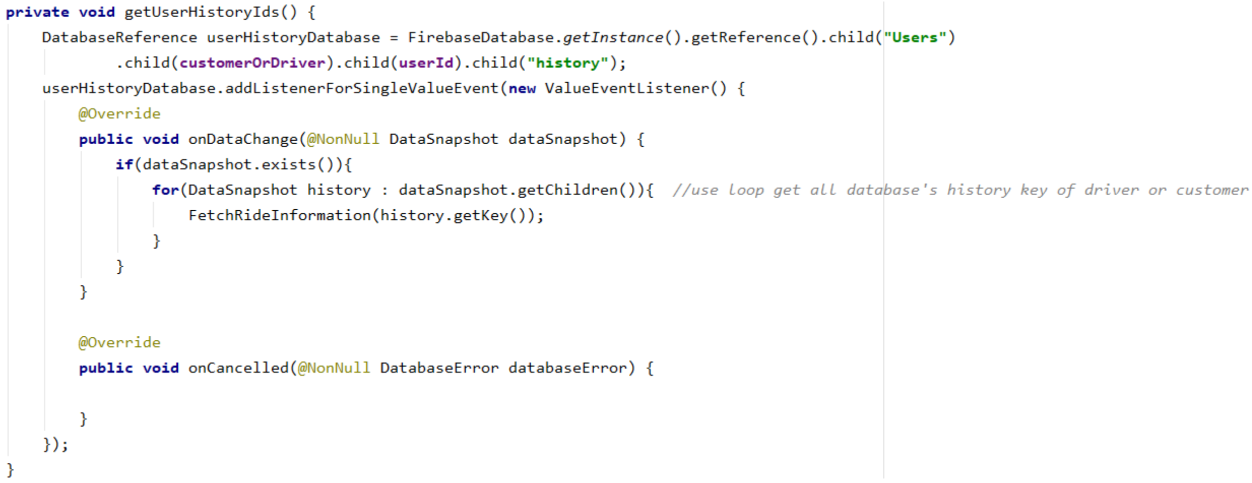
Balance: RM 0

08-09-2020 12:29:58 PM
-MGfyZQoo4rTByNhgGdc

10-09-2020 04:31:27 AM
-MGoZCLCr8alXhxqTeyu

Figure 4.51 Screenshot of user history page    Figure 4.52 Screenshot of driver history page

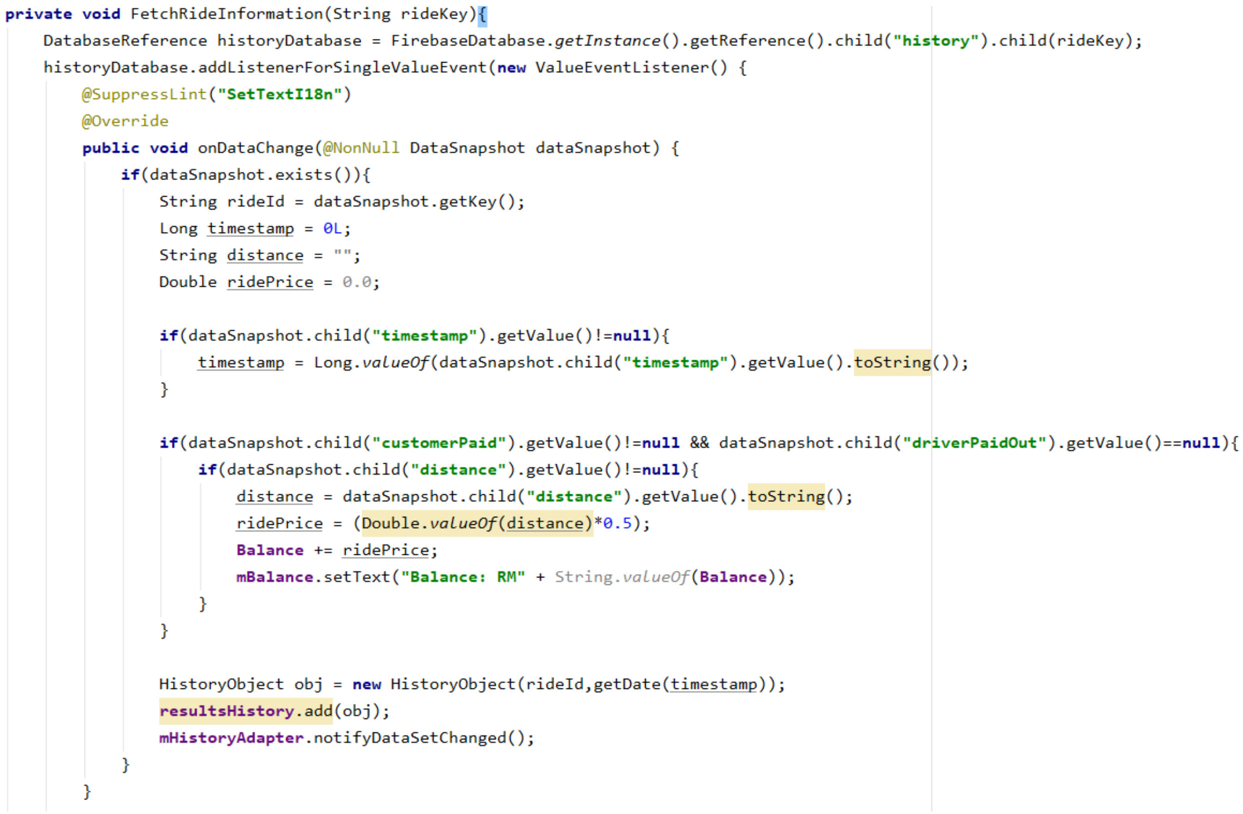
These are the history page of the user and driver. Both pages will show the request id and the time of successfully sent the parcel to the courier company. The only difference is the driver history page has the balance amount and payout function. Driver can enter their own PayPal email address to get the balance amount. The balance amount will be increased when user pays the ride fare.



```
private void getUserHistoryIds() {
    DatabaseReference userHistoryDatabase = FirebaseDatabase.getInstance().getReference().child("Users")
            .child(customerOrDriver).child(userId).child("history");
    userHistoryDatabase.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()){
                for(DataSnapshot history : dataSnapshot.getChildren()){  //use loop get all database's history key of driver or customer
                    FetchRideInformation(history.getKey());
                }
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}
```

Figure 4.53 Screenshot of get history id function code

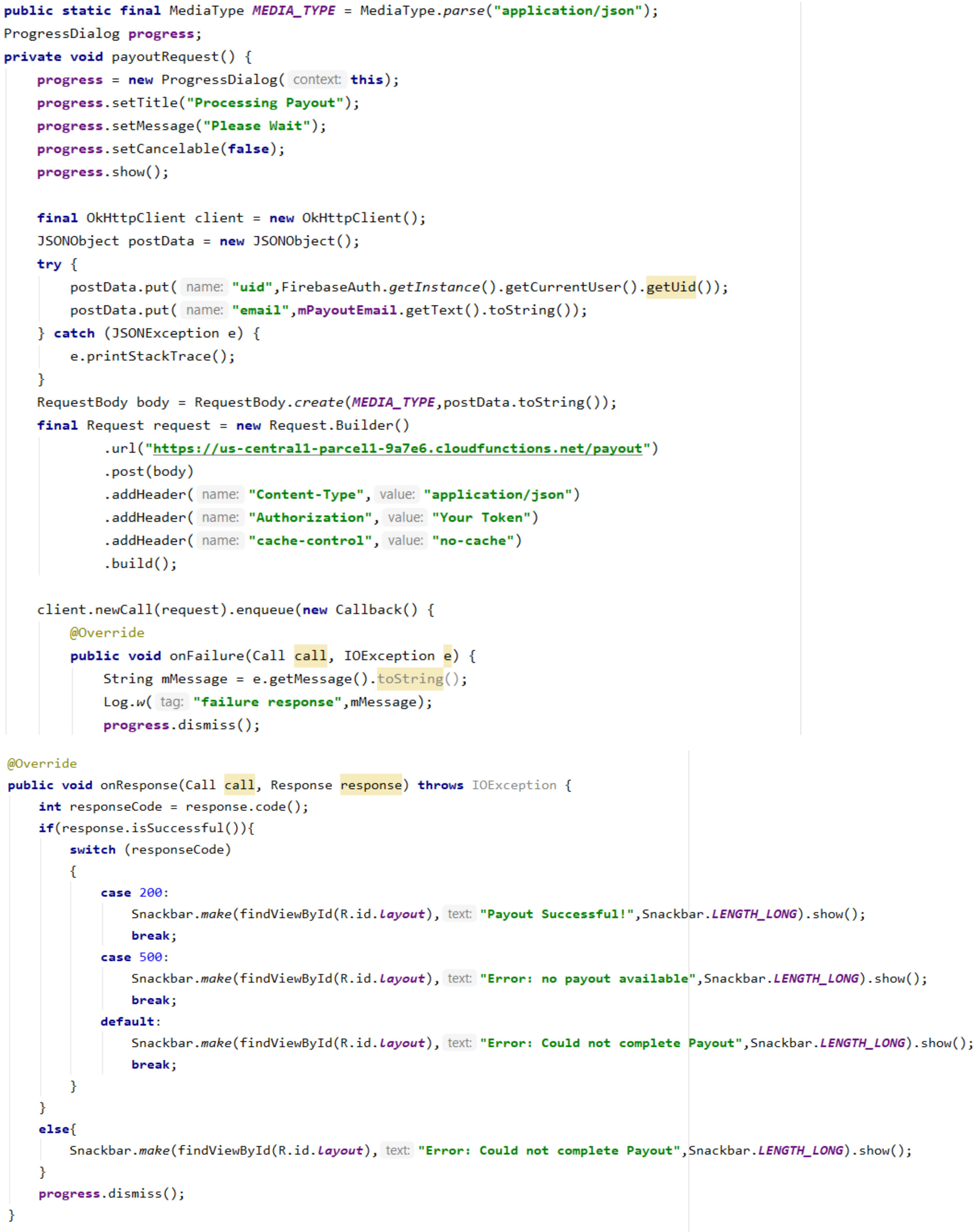
This function will loop all the database’s history id and display on page.



```
private void FetchRideInformation(String rideKey){
    DatabaseReference historyDatabase = FirebaseDatabase.getInstance().getReference().child("history").child(rideKey);
    historyDatabase.addListenerForSingleValueEvent(new ValueEventListener() {
        @SuppressLint("SetTextI18n")
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()){
                String rideId = dataSnapshot.getKey();
                Long timestamp = 0L;
                String distance = "";
                Double ridePrice = 0.0;

                if(dataSnapshot.child("timestamp").getValue()!=null){
                    timestamp = Long.valueOf(dataSnapshot.child("timestamp").getValue().toString());
                }

                if(dataSnapshot.child("customerPaid").getValue()!=null && dataSnapshot.child("driverPaidOut").getValue()==null){
                    if(dataSnapshot.child("distance").getValue()!=null){
                        distance = dataSnapshot.child("distance").getValue().toString();
                        ridePrice = (Double.valueOf(distance)*0.5);
                        Balance += ridePrice;
                        mBalance.setText("Balance: RM" + String.valueOf(Balance));
                    }
                }

                HistoryObject obj = new HistoryObject(rideId,getDate(timestamp));
                resultsHistory.add(obj);
                mHistoryAdapter.notifyDataSetChanged();
            }
        }
```

Figure 4.54 Screenshot of balance amount function code

This function will calculate the ride fare of the user request and when the user paid the ride fare it will add the balance into the balance amount of driver.



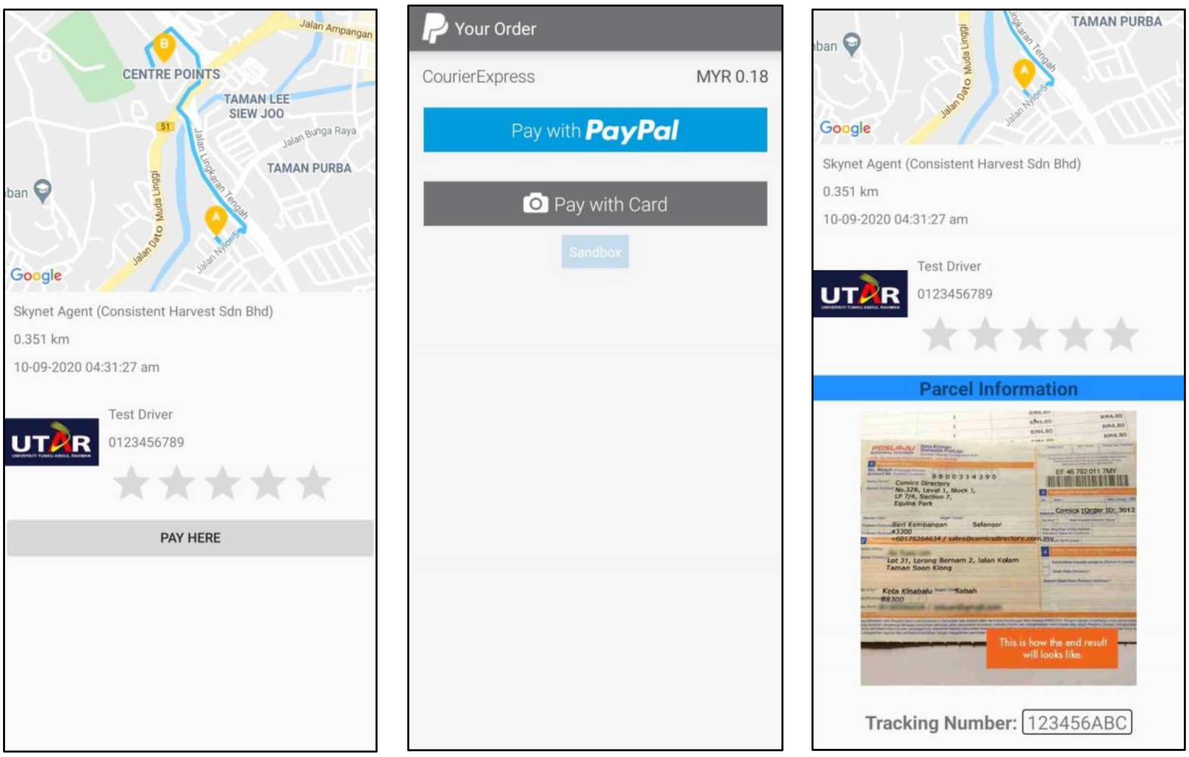
```
public static final MediaType MEDIA_TYPE = MediaType.parse("application/json");
ProgressDialog progress;
private void payoutRequest() {
    progress = new ProgressDialog( context: this);
    progress.setTitle("Processing Payout");
    progress.setMessage("Please Wait");
    progress.setCancelable(false);
    progress.show();

    final OkHttpClient client = new OkHttpClient();
    JSONObject postData = new JSONObject();
    try {
        postData.put( name: "uid",FirebaseAuth.getInstance().getCurrentUser().getUid());
        postData.put( name: "email",mPayoutEmail.getText().toString());
    } catch (JSONException e) {
        e.printStackTrace();
    }
    RequestBody body = RequestBody.create(MEDIA_TYPE,postData.toString());
    final Request request = new Request.Builder()
            .url("https://us-central1-parcel1-9a7e6.cloudfunctions.net/payout")
            .post(body)
            .addHeader( name: "Content-Type", value: "application/json")
            .addHeader( name: "Authorization", value: "Your Token")
            .addHeader( name: "cache-control", value: "no-cache")
            .build();

    client.newCall(request).enqueue(new Callback() {
        @Override
        public void onFailure(Call call, IOException e) {
            String mMessage = e.getMessage().toString();
            Log.w( tag: "failure response",mMessage);
            progress.dismiss();
```

```
@Override
public void onResponse(Call call, Response response) throws IOException {
    int responseCode = response.code();
    if(response.isSuccessful()){
        switch (responseCode)
        {
            case 200:
                Snackbar.make(findViewById(R.id.layout), text: "Payout Successful!",Snackbar.LENGTH_LONG).show();
                break;
            case 500:
                Snackbar.make(findViewById(R.id.layout), text: "Error: no payout available",Snackbar.LENGTH_LONG).show();
                break;
            default:
                Snackbar.make(findViewById(R.id.layout), text: "Error: Could not complete Payout",Snackbar.LENGTH_LONG).show();
                break;
        }
    }
    else{
        Snackbar.make(findViewById(R.id.layout), text: "Error: Could not complete Payout",Snackbar.LENGTH_LONG).show();
    }
    progress.dismiss();
}
```

Figure 4.55 Screenshot of driver payout function code

This function will be trigger when driver enter PayPal email and clicks payout. If the PayPal email entered is not valid or payout failed, an error message will show on page.

### 4.2.8.2 HistorySingleActivity.java



Skynet Agent (Consistent Harvest Sdn Bhd)
0.351 km
10-09-2020 04:31:27 am
Test Driver
0123456789

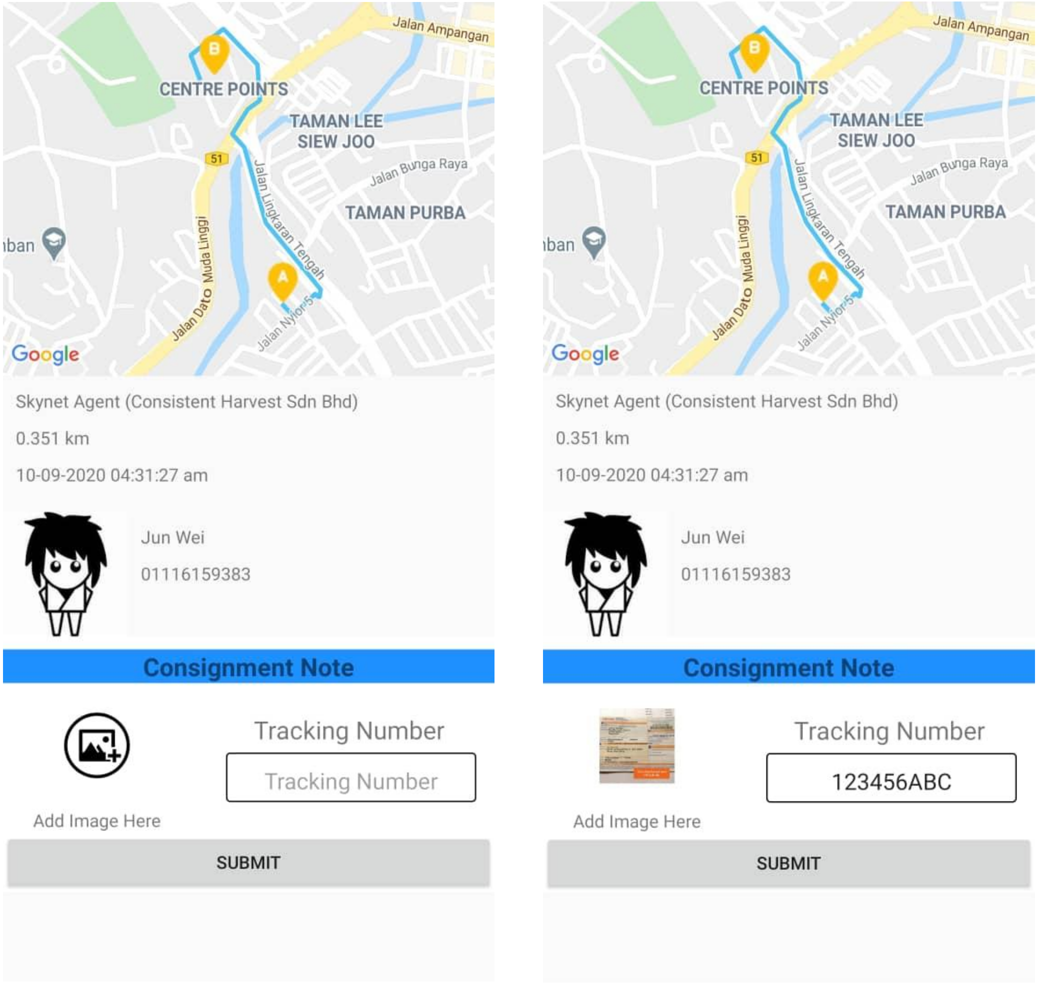
Your Order
CourierExpress MYR 0.18

Skynet Agent (Consistent Harvest Sdn Bhd)
0.351 km
10-09-2020 04:31:27 am
Test Driver
0123456789

Tracking Number: 123456ABC

Figure 4.56 Screenshot of history view page (User)

This page will show the ride details of the request driver. The information such as destination, ride distance and parcel sent time will show to user. However, the parcel information will not be display if the user not yet pay the ride fare. User needs to click the pay here button and pay the amount at the payment gateway. After the user successfully paid the amount, the parcel information such as picture and the tracking number will be shown.



Skynet Agent (Consistent Harvest Sdn Bhd)

0.351 km

10-09-2020 04:31:27 am

Jun Wei

01116159383

Consignment Note

Tracking Number: Tracking Number

Add Image Here

SUBMIT

Skynet Agent (Consistent Harvest Sdn Bhd)

0.351 km

10-09-2020 04:31:27 am

Jun Wei

01116159383

Consignment Note

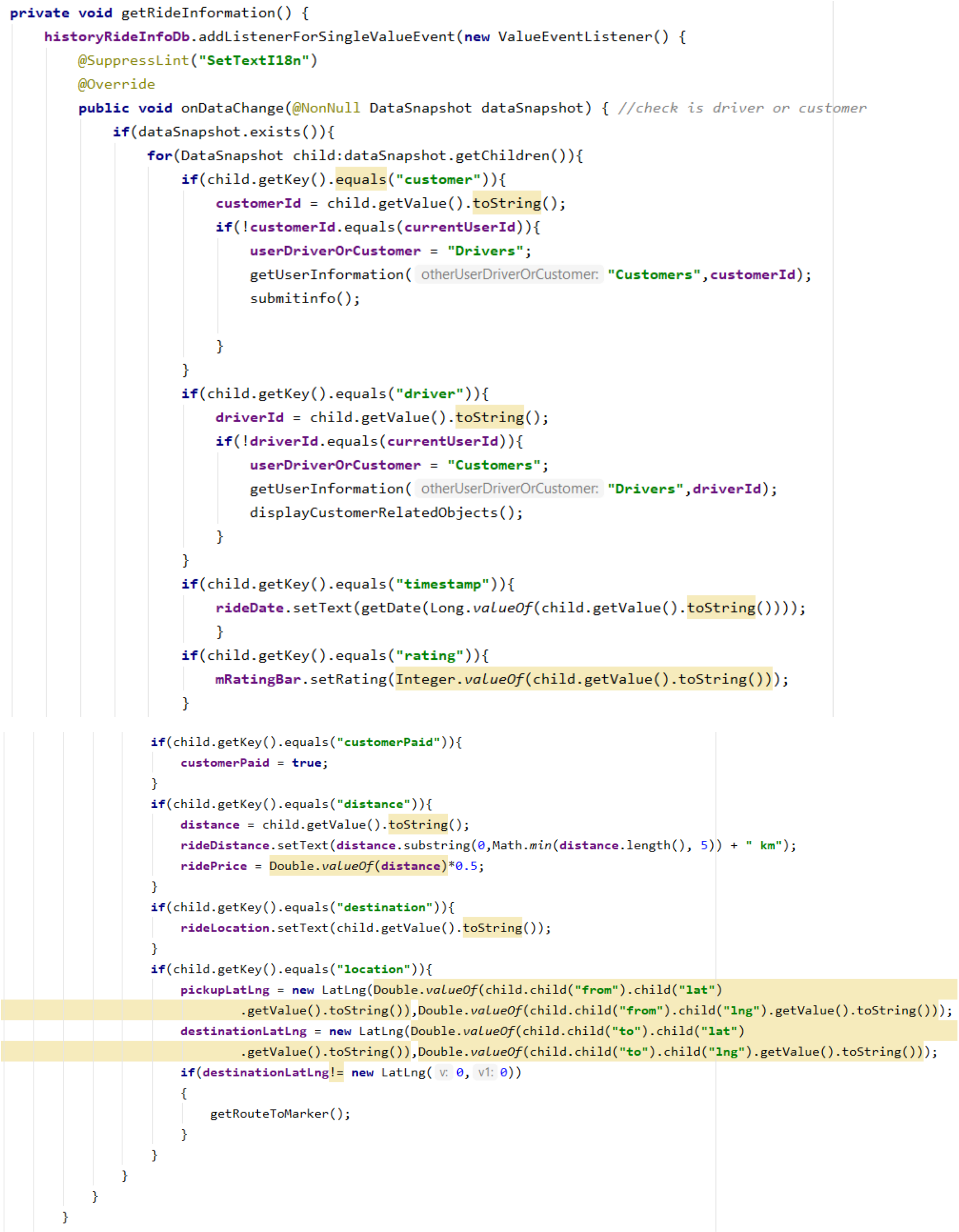
Tracking Number: 123456ABC

Add Image Here

SUBMIT

Figure 4.57 Screenshot of history view page (Driver)

This page will show the ride details and user information. Driver needs to snap a picture of the parcel and upload it to database act as evidence to prove the parcel already sent to destination. The tracking number on the consignment note also needs to be entered at the tracking number field. The information will be updated to the database when driver clicks the submit button.

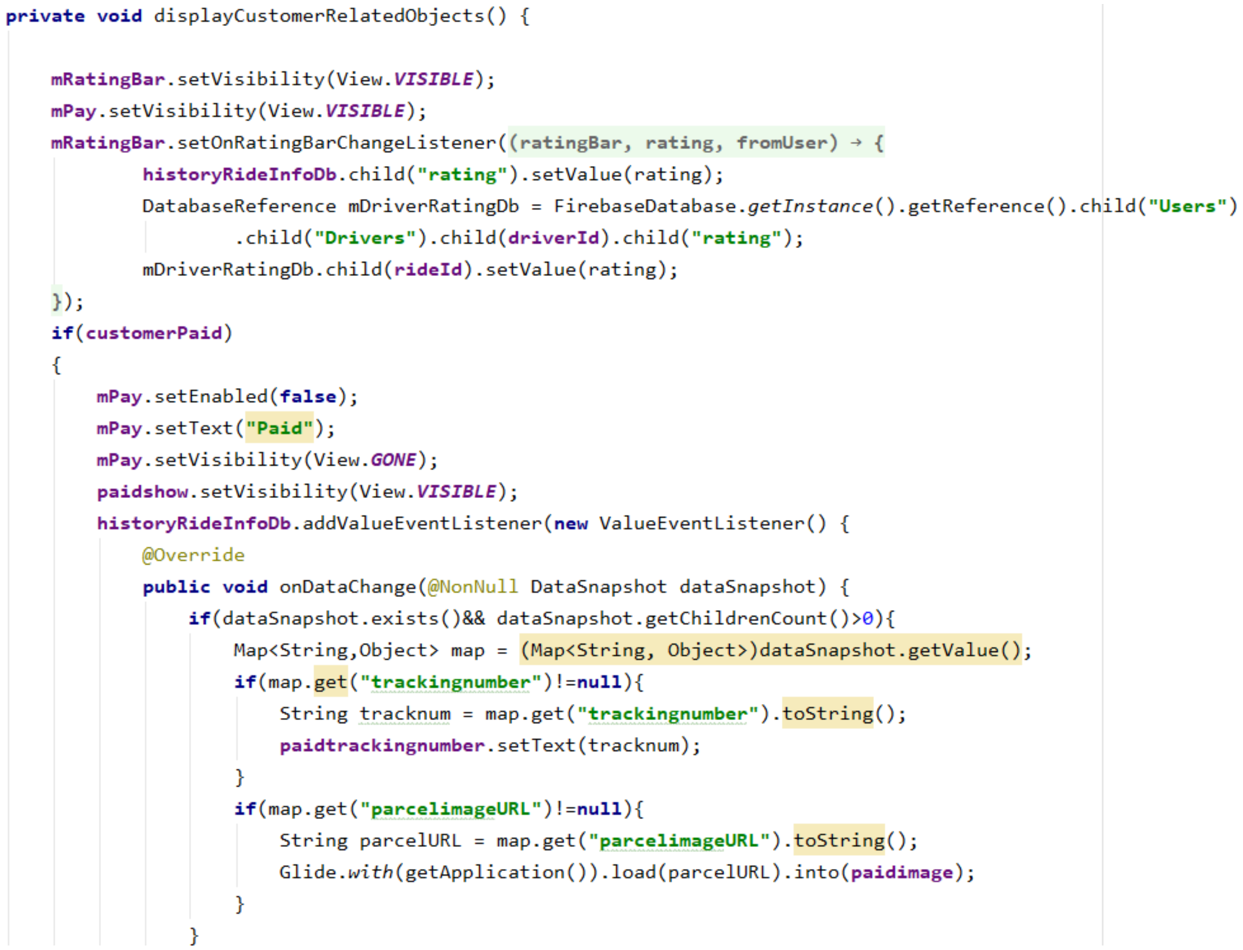


```java
private void getRideInformation() {
    historyRideInfoDb.addListenerForSingleValueEvent(new ValueEventListener() {
        @SuppressLint("SetTextI18n")
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) { //check is driver or customer
            if(dataSnapshot.exists()){
                for(DataSnapshot child:dataSnapshot.getChildren()){
                    if(child.getKey().equals("customer")){
                        customerId = child.getValue().toString();
                        if(!customerId.equals(currentUserId)){
                            userDriverOrCustomer = "Drivers";
                            getUserInformation( otherUserDriverOrCustomer: "Customers",customerId);
                            submitinfo();

                        }
                    }
                    if(child.getKey().equals("driver")){
                        driverId = child.getValue().toString();
                        if(!driverId.equals(currentUserId)){
                            userDriverOrCustomer = "Customers";
                            getUserInformation( otherUserDriverOrCustomer: "Drivers",driverId);
                            displayCustomerRelatedObjects();
                        }
                    }
                    if(child.getKey().equals("timestamp")){
                        rideDate.setText(getDate(Long.valueOf(child.getValue().toString())));
                        }
                    if(child.getKey().equals("rating")){
                        mRatingBar.setRating(Integer.valueOf(child.getValue().toString()));
                    }
```

```java
                    if(child.getKey().equals("customerPaid")){
                        customerPaid = true;
                    }
                    if(child.getKey().equals("distance")){
                        distance = child.getValue().toString();
                        rideDistance.setText(distance.substring(0,Math.min(distance.length(), 5)) + " km");
                        ridePrice = Double.valueOf(distance)*0.5;
                    }
                    if(child.getKey().equals("destination")){
                        rideLocation.setText(child.getValue().toString());
                    }
                    if(child.getKey().equals("location")){
                        pickupLatLng = new LatLng(Double.valueOf(child.child("from").child("lat")
                                .getValue().toString()),Double.valueOf(child.child("from").child("lng").getValue().toString()));
                        destinationLatLng = new LatLng(Double.valueOf(child.child("to").child("lat")
                                .getValue().toString()),Double.valueOf(child.child("to").child("lng").getValue().toString()));
                        if(destinationLatLng!= new LatLng( v: 0, v1: 0))
                        {
                            getRouteToMarker();
                        }
                    }
                }
            }
        }
```

Figure 4.58 Screenshot of display ride details function code

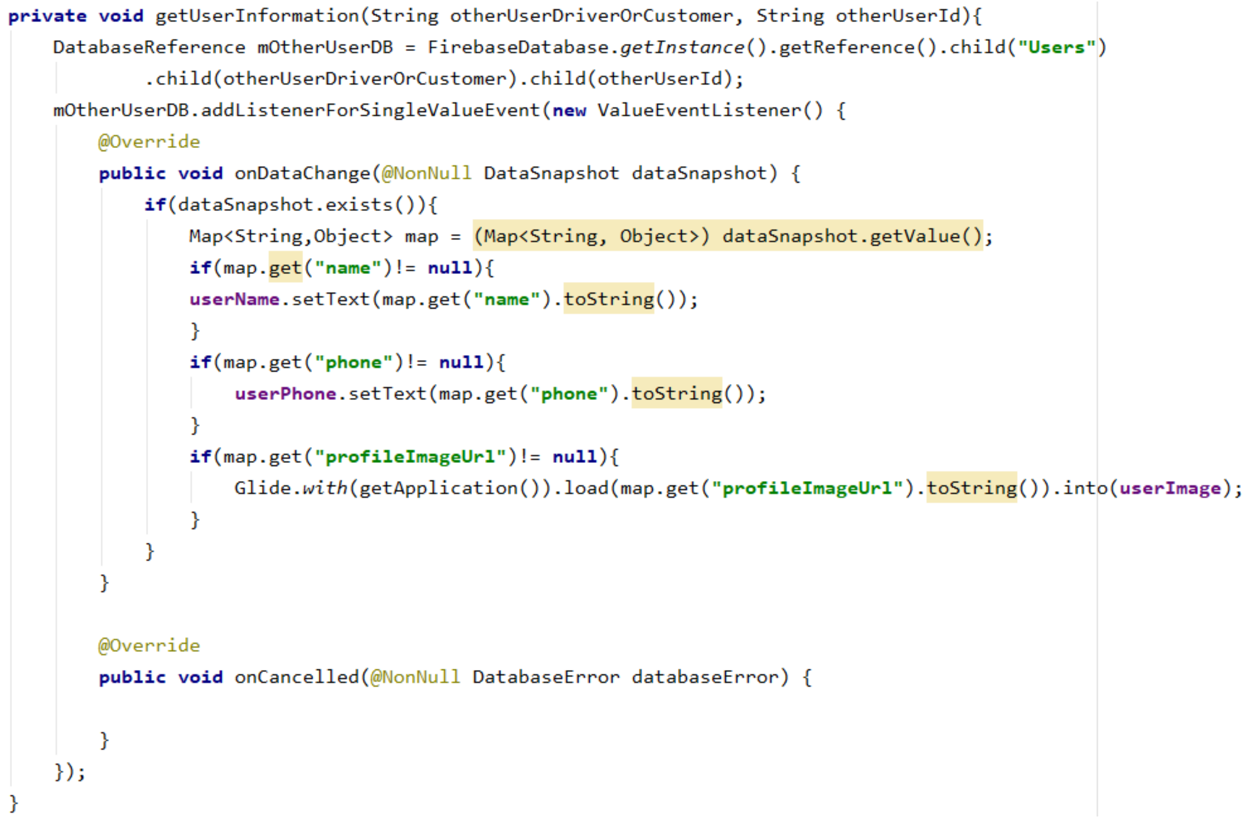
This function will first check the current user is customer or driver. Different users will display different information on the page. For example, if the current user is customer, the page will show driver details. However, if the current user is driver, the page will show customer details.



```java
private void displayCustomerRelatedObjects() {

    mRatingBar.setVisibility(View.VISIBLE);
    mPay.setVisibility(View.VISIBLE);
    mRatingBar.setOnRatingBarChangeListener((ratingBar, rating, fromUser) → {
            historyRideInfoDb.child("rating").setValue(rating);
            DatabaseReference mDriverRatingDb = FirebaseDatabase.getInstance().getReference().child("Users")
                    .child("Drivers").child(driverId).child("rating");
            mDriverRatingDb.child(rideId).setValue(rating);
    });
    if(customerPaid)
    {
        mPay.setEnabled(false);
        mPay.setText("Paid");
        mPay.setVisibility(View.GONE);
        paidshow.setVisibility(View.VISIBLE);
        historyRideInfoDb.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if(dataSnapshot.exists()&& dataSnapshot.getChildrenCount()>0){
                    Map<String,Object> map = (Map<String, Object>)dataSnapshot.getValue();
                    if(map.get("trackingnumber")!=null){
                        String tracknum = map.get("trackingnumber").toString();
                        paidtrackingnumber.setText(tracknum);
                    }
                    if(map.get("parcelimageURL")!=null){
                        String parcelURL = map.get("parcelimageURL").toString();
                        Glide.with(getApplication()).load(parcelURL).into(paidimage);
                    }
                }
```

Figure 4.59 Screenshot of display driver details and parcel information function code

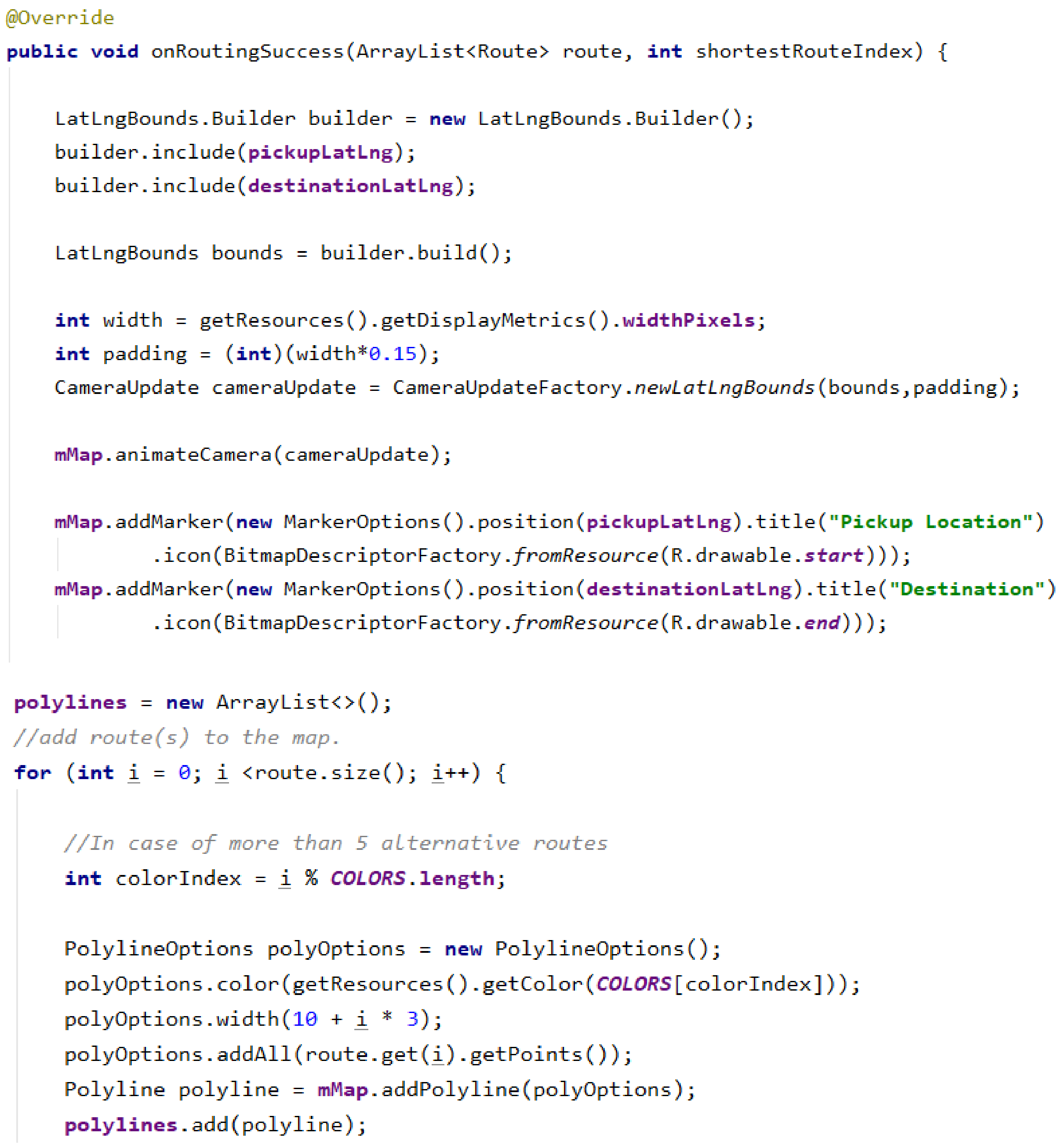
This function will be trigger when the system detected the current user is customer. It will show the details of driver and check whether the user paid the ride fare or not. The parcel information will not be display if the ride fares not yet paid.



```
private void getUserInformation(String otherUserDriverOrCustomer, String otherUserId){
    DatabaseReference mOtherUserDB = FirebaseDatabase.getInstance().getReference().child("Users")
            .child(otherUserDriverOrCustomer).child(otherUserId);
    mOtherUserDB.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()){
                Map<String,Object> map = (Map<String, Object>) dataSnapshot.getValue();
                if(map.get("name")!= null){
                userName.setText(map.get("name").toString());
                }
                if(map.get("phone")!= null){
                    userPhone.setText(map.get("phone").toString());
                }
                if(map.get("profileImageUrl")!= null){
                    Glide.with(getApplication()).load(map.get("profileImageUrl").toString()).into(userImage);
                }
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}
```

Figure 4.60 Screenshot of display customer details function code

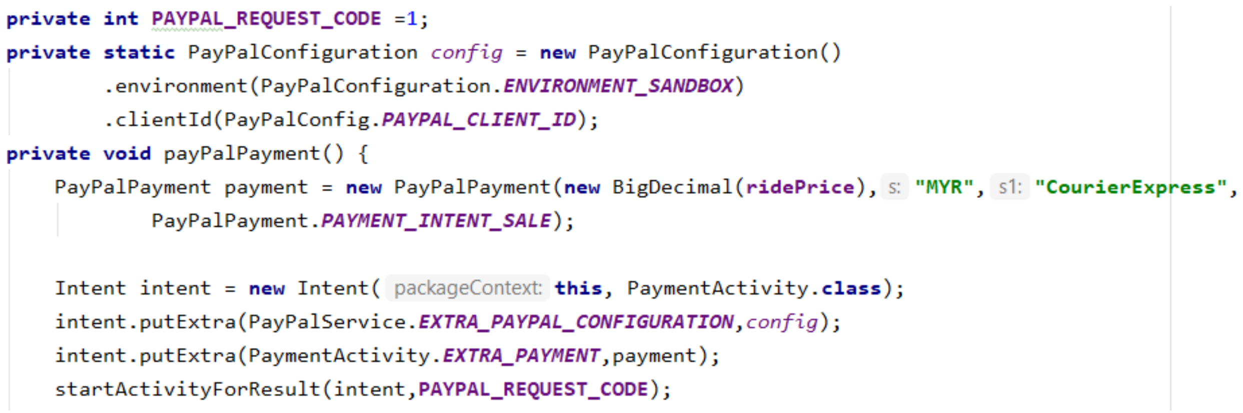
This function will be trigger when the system detected the current user is a driver. It will show the user details such as name, phone and picture.



```
@Override
public void onRoutingSuccess(ArrayList<Route> route, int shortestRouteIndex) {

    LatLngBounds.Builder builder = new LatLngBounds.Builder();
    builder.include(pickupLatLng);
    builder.include(destinationLatLng);

    LatLngBounds bounds = builder.build();

    int width = getResources().getDisplayMetrics().widthPixels;
    int padding = (int)(width*0.15);
    CameraUpdate cameraUpdate = CameraUpdateFactory.newLatLngBounds(bounds,padding);

    mMap.animateCamera(cameraUpdate);

    mMap.addMarker(new MarkerOptions().position(pickupLatLng).title("Pickup Location")
            .icon(BitmapDescriptorFactory.fromResource(R.drawable.start)));
    mMap.addMarker(new MarkerOptions().position(destinationLatLng).title("Destination")
            .icon(BitmapDescriptorFactory.fromResource(R.drawable.end)));

 polylines = new ArrayList<>();
 //add route(s) to the map.
 for (int i = 0; i <route.size(); i++) {

     //In case of more than 5 alternative routes
     int colorIndex = i % COLORS.length;

     PolylineOptions polyOptions = new PolylineOptions();
     polyOptions.color(getResources().getColor(COLORS[colorIndex]));
     polyOptions.width(10 + i * 3);
     polyOptions.addAll(route.get(i).getPoints());
     Polyline polyline = mMap.addPolyline(polyOptions);
     polylines.add(polyline);
```

Figure 4.61 Screenshot of put marker and draw blue line on map code

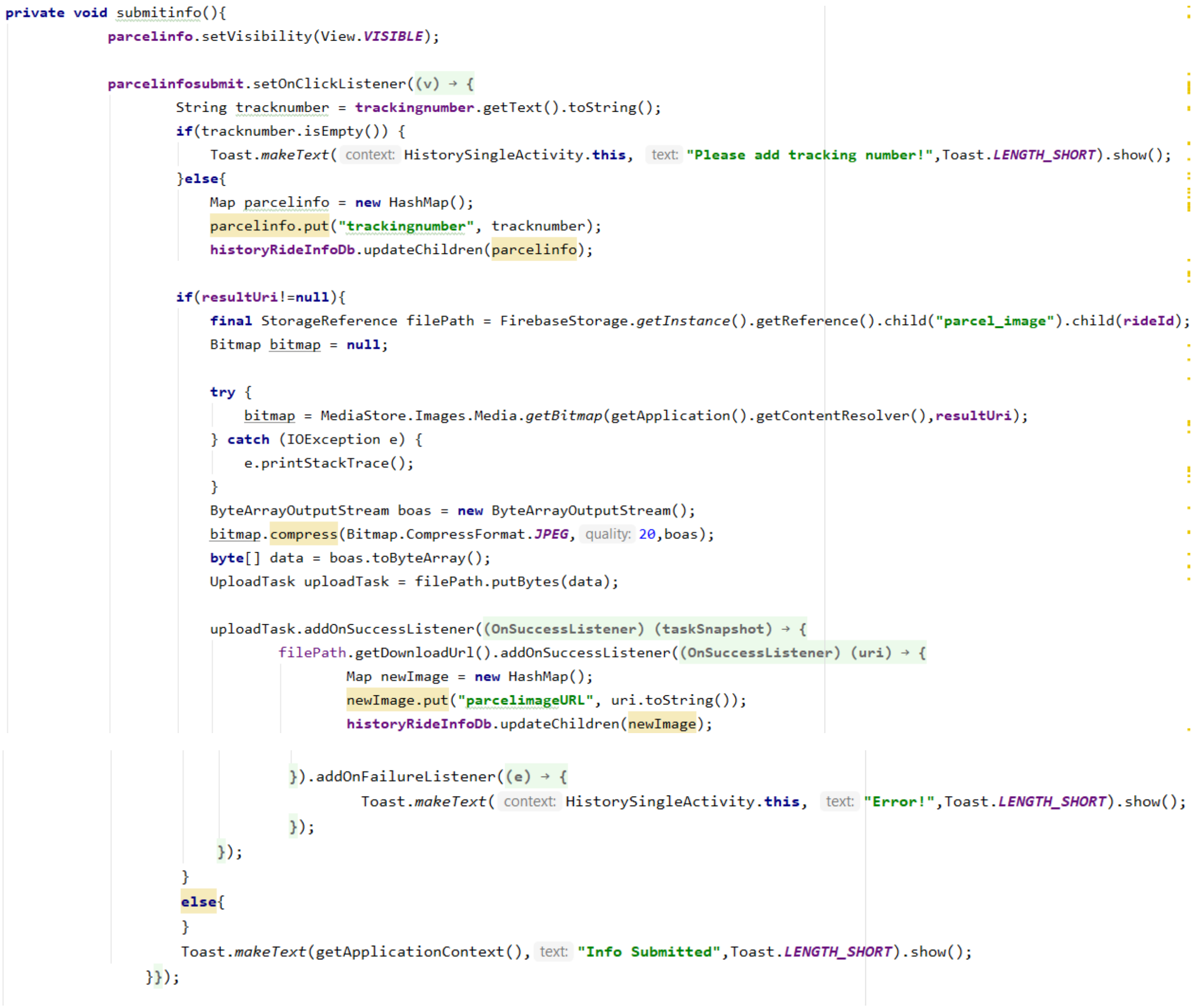
This function will get the pickup location of the parcel and the destination and place yellow marker on both locations. Besides, it will also draw a blue line route on map.



```
private int PAYPAL_REQUEST_CODE =1;
private static PayPalConfiguration config = new PayPalConfiguration()
        .environment(PayPalConfiguration.ENVIRONMENT_SANDBOX)
        .clientId(PayPalConfig.PAYPAL_CLIENT_ID);
private void payPalPayment() {
    PayPalPayment payment = new PayPalPayment(new BigDecimal(ridePrice), s: "MYR", s1: "CourierExpress",
            PayPalPayment.PAYMENT_INTENT_SALE);

    Intent intent = new Intent( packageContext: this, PaymentActivity.class);
    intent.putExtra(PayPalService.EXTRA_PAYPAL_CONFIGURATION,config);
    intent.putExtra(PaymentActivity.EXTRA_PAYMENT,payment);
    startActivityForResult(intent,PAYPAL_REQUEST_CODE);
```

Figure 4.62 Screenshot of PayPal payment gateway code

These codes will help the application open the PayPal payment gateway to let user paid the route fare.



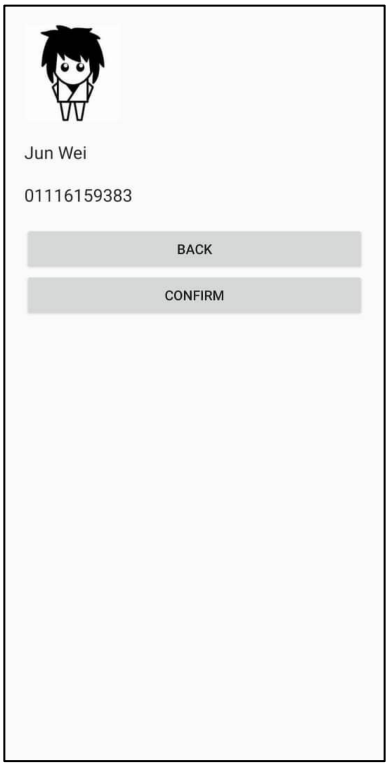
```
private void submitinfo(){
            parcelinfo.setVisibility(View.VISIBLE);

            parcelinfosubmit.setOnClickListener((v) → {
                    String tracknumber = trackingnumber.getText().toString();
                    if(tracknumber.isEmpty()) {
                        Toast.makeText( context: HistorySingleActivity.this,  text: "Please add tracking number!",Toast.LENGTH_SHORT).show();
                    }else{
                        Map parcelinfo = new HashMap();
                        parcelinfo.put("trackingnumber", tracknumber);
                        historyRideInfoDb.updateChildren(parcelinfo);

                    if(resultUri!=null){
                        final StorageReference filePath = FirebaseStorage.getInstance().getReference().child("parcel_image").child(rideId);
                        Bitmap bitmap = null;

                        try {
                            bitmap = MediaStore.Images.Media.getBitmap(getApplication().getContentResolver(),resultUri);
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                        ByteArrayOutputStream boas = new ByteArrayOutputStream();
                        bitmap.compress(Bitmap.CompressFormat.JPEG,  quality: 20,boas);
                        byte[] data = boas.toByteArray();
                        UploadTask uploadTask = filePath.putBytes(data);

                        uploadTask.addOnSuccessListener((OnSuccessListener) (taskSnapshot) → {
                                filePath.getDownloadUrl().addOnSuccessListener((OnSuccessListener) (uri) → {
                                        Map newImage = new HashMap();
                                        newImage.put("parcelimageURL", uri.toString());
                                        historyRideInfoDb.updateChildren(newImage);

                                }).addOnFailureListener((e) → {
                                        Toast.makeText( context: HistorySingleActivity.this,  text: "Error!",Toast.LENGTH_SHORT).show();
                                });
                        });
                    }
                    else{
                    }
                    Toast.makeText(getApplicationContext(), text: "Info Submitted",Toast.LENGTH_SHORT).show();
                }});
```

Figure 4.63 Screenshot of submit button code

These codes will be trigger when the driver clicks the submit button. The function will first check the tracking number field entered a value then it will send the image data and tracking number to the database. When the function successfully runs, the system will show a message.

## 4.2.9 Profile Settings

### 4.2.9.1 CustomerSettingsActivity.java

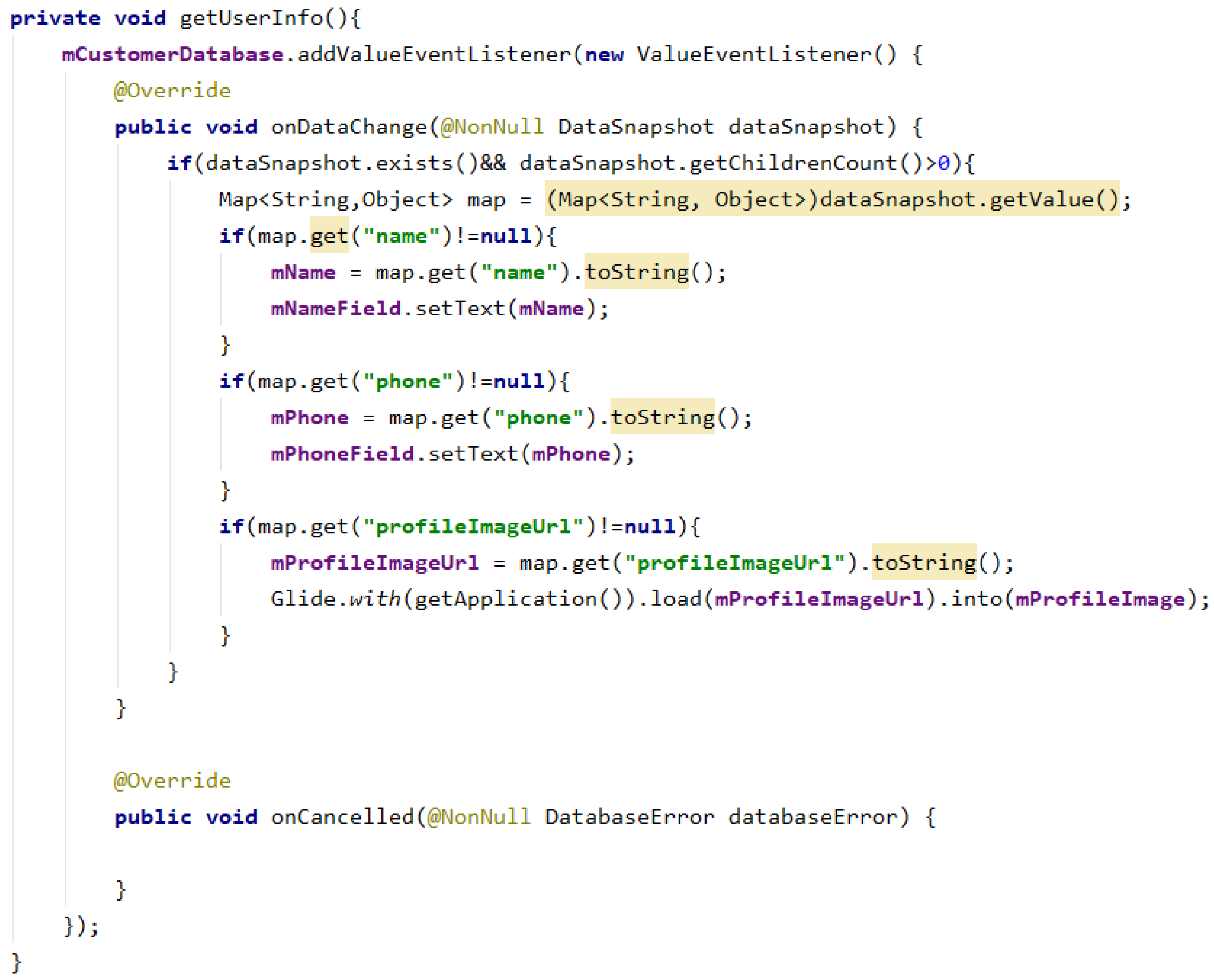


Jun Wei

01116159383

Figure 4.64 Screenshot of customer profile settings page

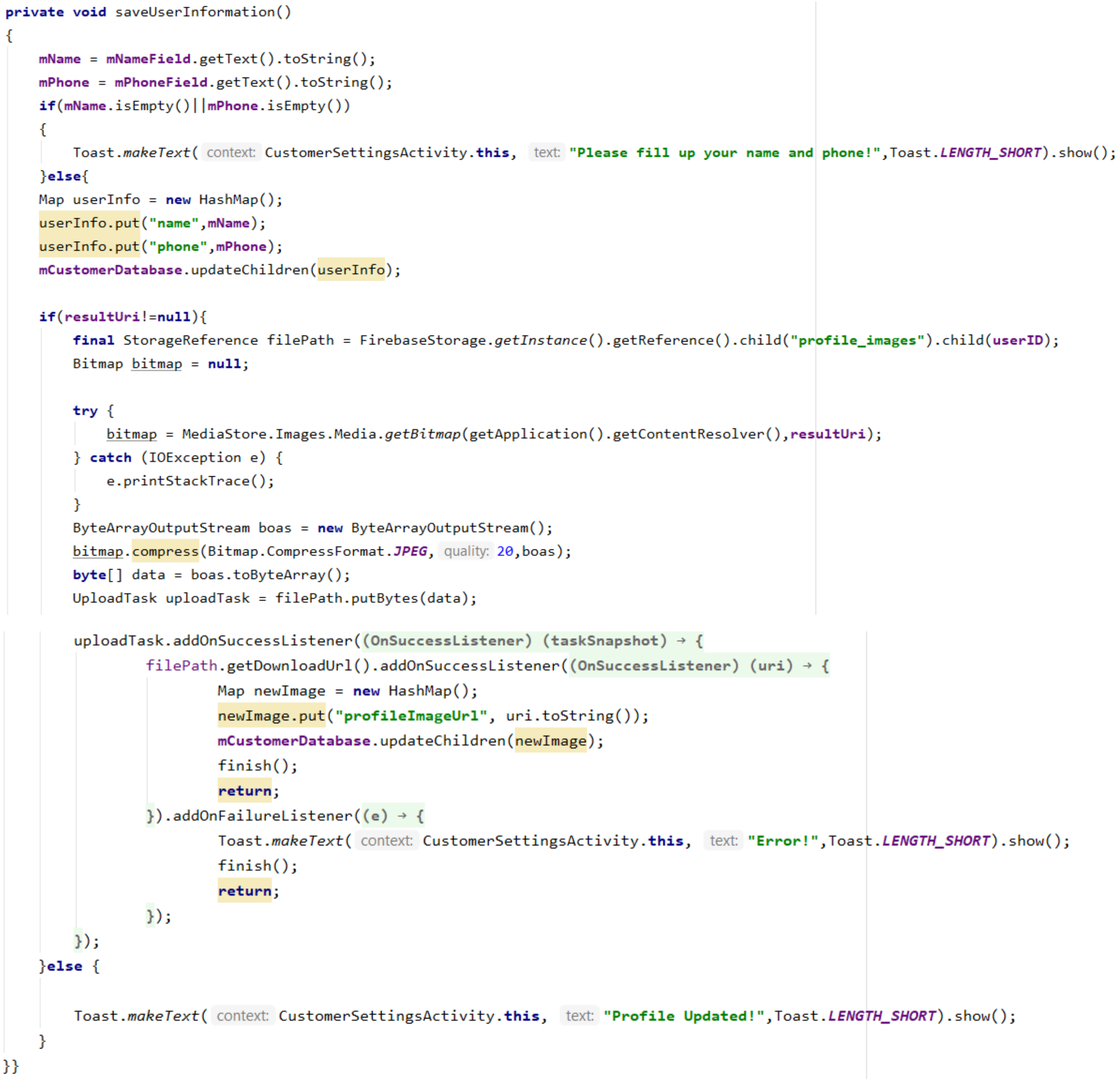
This page will show the customer details and the user can modify the details such as profile picture, user name, and user phone. The back button will bring user back to the previous activity and the confirm button will update current user information at the database when clicked.



```
private void getUserInfo(){
    mCustomerDatabase.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()&& dataSnapshot.getChildrenCount()>0){
                Map<String,Object> map = (Map<String, Object>)dataSnapshot.getValue();
                if(map.get("name")!=null){
                    mName = map.get("name").toString();
                    mNameField.setText(mName);
                }
                if(map.get("phone")!=null){
                    mPhone = map.get("phone").toString();
                    mPhoneField.setText(mPhone);
                }
                if(map.get("profileImageUrl")!=null){
                    mProfileImageUrl = map.get("profileImageUrl").toString();
                    Glide.with(getApplication()).load(mProfileImageUrl).into(mProfileImage);
                }
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}
```

Figure 4.65 Screenshot of get current user information code

These codes will check the current user id at the database and get the user data to show on the page.

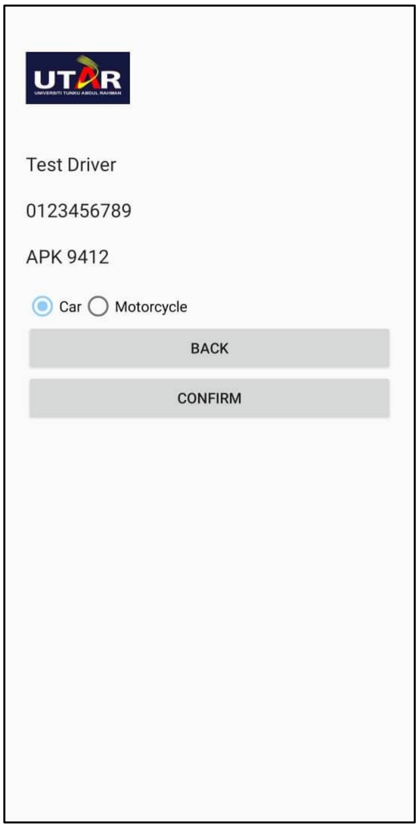


```
private void saveUserInformation()
{
    mName = mNameField.getText().toString();
    mPhone = mPhoneField.getText().toString();
    if(mName.isEmpty()||mPhone.isEmpty())
    {
        Toast.makeText( context: CustomerSettingsActivity.this,  text: "Please fill up your name and phone!",Toast.LENGTH_SHORT).show();
    }else{
    Map userInfo = new HashMap();
    userInfo.put("name",mName);
    userInfo.put("phone",mPhone);
    mCustomerDatabase.updateChildren(userInfo);

    if(resultUri!=null){
        final StorageReference filePath = FirebaseStorage.getInstance().getReference().child("profile_images").child(userID);
        Bitmap bitmap = null;

        try {
            bitmap = MediaStore.Images.Media.getBitmap(getApplication().getContentResolver(),resultUri);
        } catch (IOException e) {
            e.printStackTrace();
        }
        ByteArrayOutputStream boas = new ByteArrayOutputStream();
        bitmap.compress(Bitmap.CompressFormat.JPEG,  quality: 20,boas);
        byte[] data = boas.toByteArray();
        UploadTask uploadTask = filePath.putBytes(data);

        uploadTask.addOnSuccessListener((OnSuccessListener) (taskSnapshot) → {
                filePath.getDownloadUrl().addOnSuccessListener((OnSuccessListener) (uri) → {
                        Map newImage = new HashMap();
                        newImage.put("profileImageUrl", uri.toString());
                        mCustomerDatabase.updateChildren(newImage);
                        finish();
                        return;
                }).addOnFailureListener((e) → {
                        Toast.makeText( context: CustomerSettingsActivity.this,  text: "Error!",Toast.LENGTH_SHORT).show();
                        finish();
                        return;
                });
        });
    }else {

        Toast.makeText( context: CustomerSettingsActivity.this,  text: "Profile Updated!",Toast.LENGTH_SHORT).show();
    }
}}
```

Figure 4.66 Screenshot of save user information code

These codes will get the value of the fields and upload the data to the current user database id. An error message will show if update failed.

### 4.2.9.2 DriverSettingsActivity.java



Test Driver

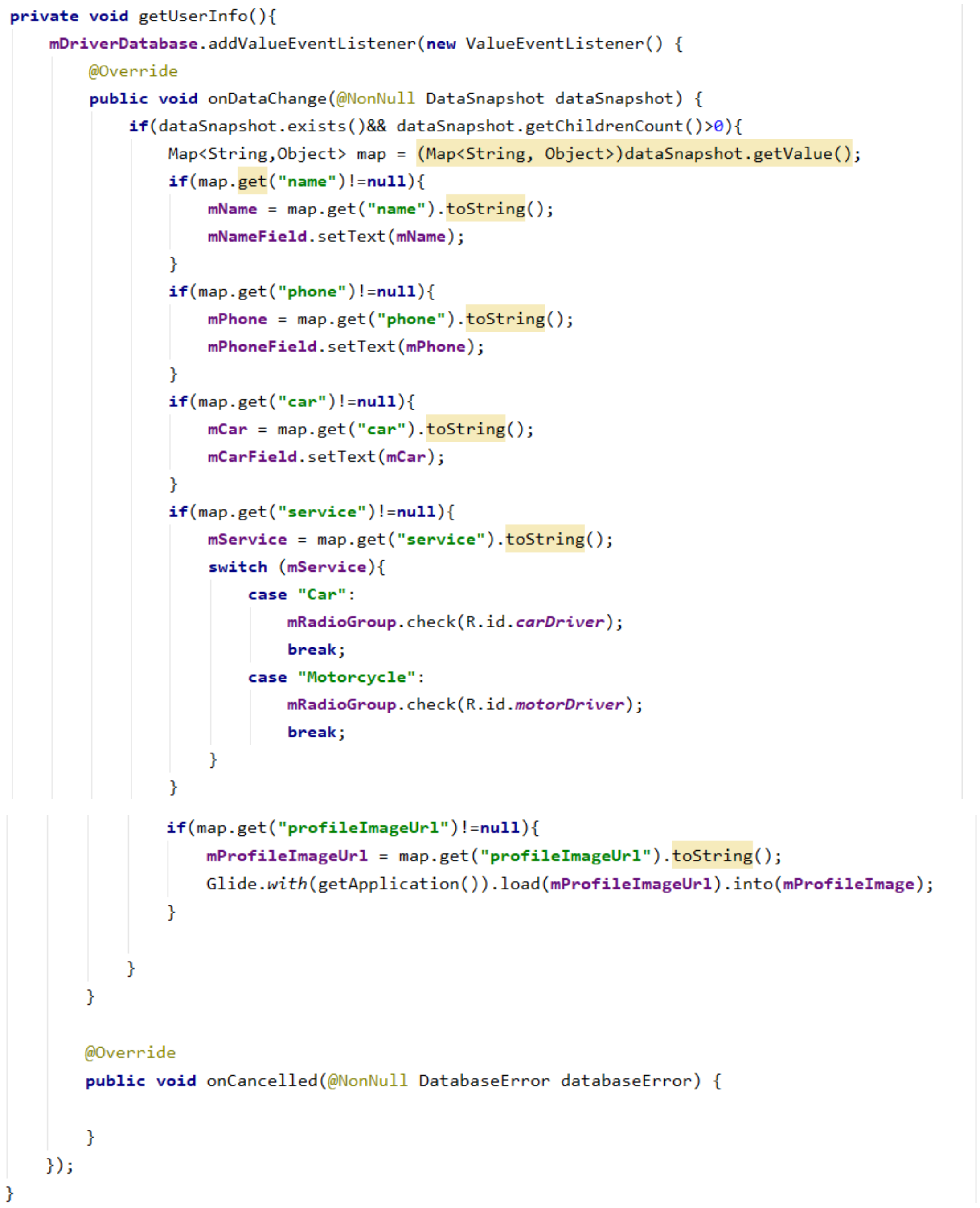
0123456789

APK 9412

● Car ○ Motorcycle

Figure 4.67 Screenshot of driver profile settings page

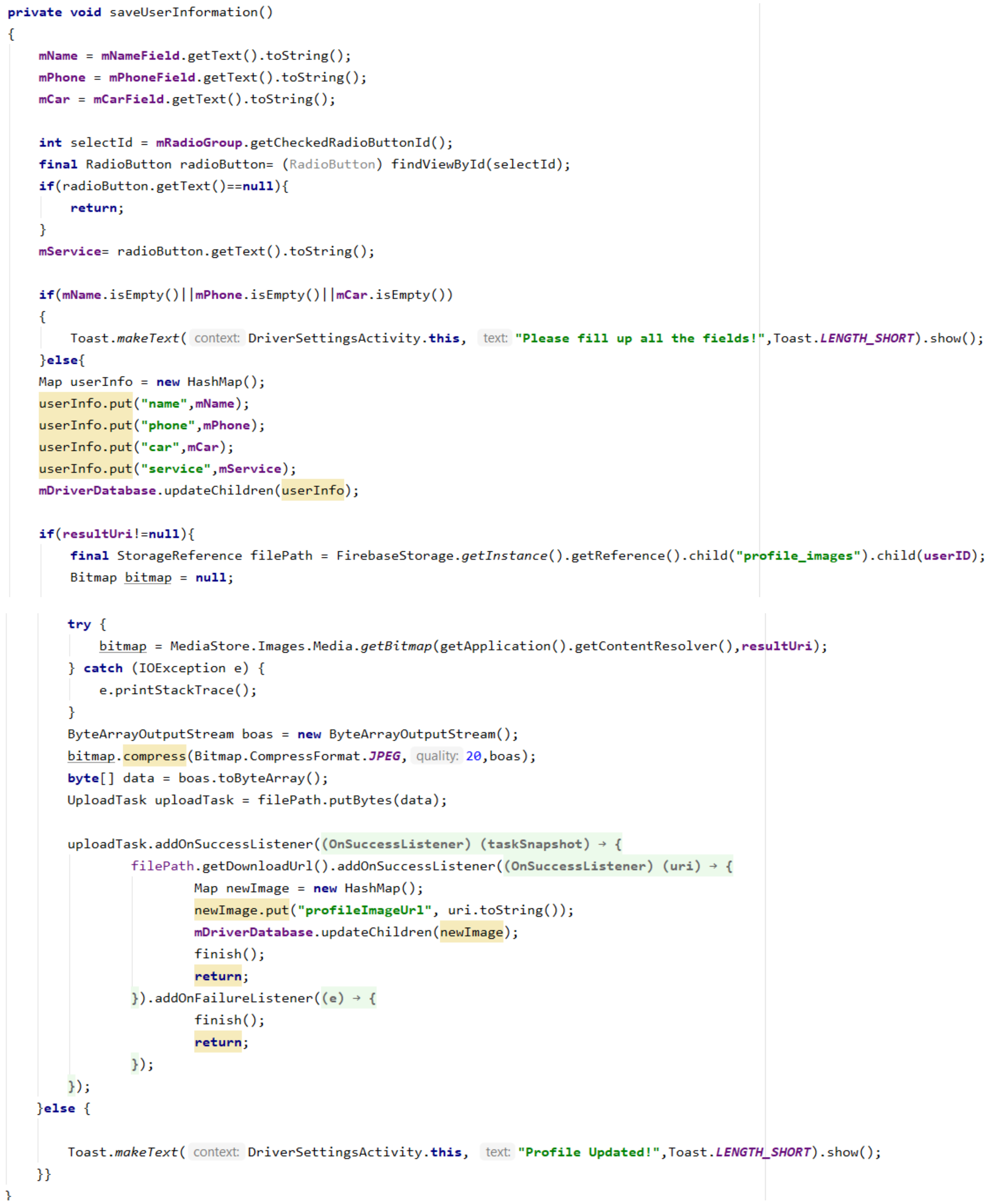
This page will show the driver details and the driver also can modify the details such as profile picture, name, phone number and vehicle number. The default service choose is the car service and the driver can change to motorcycle if using motorcycle service.



```
private void getUserInfo(){
    mDriverDatabase.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()&& dataSnapshot.getChildrenCount()>0){
                Map<String,Object> map = (Map<String, Object>)dataSnapshot.getValue();
                if(map.get("name")!=null){
                    mName = map.get("name").toString();
                    mNameField.setText(mName);
                }
                if(map.get("phone")!=null){
                    mPhone = map.get("phone").toString();
                    mPhoneField.setText(mPhone);
                }
                if(map.get("car")!=null){
                    mCar = map.get("car").toString();
                    mCarField.setText(mCar);
                }
                if(map.get("service")!=null){
                    mService = map.get("service").toString();
                    switch (mService){
                        case "Car":
                            mRadioGroup.check(R.id.carDriver);
                            break;
                        case "Motorcycle":
                            mRadioGroup.check(R.id.motorDriver);
                            break;
                    }
                }
                if(map.get("profileImageUrl")!=null){
                    mProfileImageUrl = map.get("profileImageUrl").toString();
                    Glide.with(getApplication()).load(mProfileImageUrl).into(mProfileImage);
                }

            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}
```

Figure 4.68 Screenshot of get driver details function code

These codes will get the information of the current driver at the database and display on page.



```
private void saveUserInformation()
{
    mName = mNameField.getText().toString();
    mPhone = mPhoneField.getText().toString();
    mCar = mCarField.getText().toString();

    int selectId = mRadioGroup.getCheckedRadioButtonId();
    final RadioButton radioButton= (RadioButton) findViewById(selectId);
    if(radioButton.getText()==null){
        return;
    }
    mService= radioButton.getText().toString();

    if(mName.isEmpty()||mPhone.isEmpty()||mCar.isEmpty())
    {
        Toast.makeText( context: DriverSettingsActivity.this,  text: "Please fill up all the fields!",Toast.LENGTH_SHORT).show();
    }else{
    Map userInfo = new HashMap();
    userInfo.put("name",mName);
    userInfo.put("phone",mPhone);
    userInfo.put("car",mCar);
    userInfo.put("service",mService);
    mDriverDatabase.updateChildren(userInfo);

    if(resultUri!=null){
        final StorageReference filePath = FirebaseStorage.getInstance().getReference().child("profile_images").child(userID);
        Bitmap bitmap = null;

        try {
            bitmap = MediaStore.Images.Media.getBitmap(getApplication().getContentResolver(),resultUri);
        } catch (IOException e) {
            e.printStackTrace();
        }
        ByteArrayOutputStream boas = new ByteArrayOutputStream();
        bitmap.compress(Bitmap.CompressFormat.JPEG,  quality: 20,boas);
        byte[] data = boas.toByteArray();
        UploadTask uploadTask = filePath.putBytes(data);

        uploadTask.addOnSuccessListener((OnSuccessListener) (taskSnapshot) → {
                filePath.getDownloadUrl().addOnSuccessListener((OnSuccessListener) (uri) → {
                        Map newImage = new HashMap();
                        newImage.put("profileImageUrl", uri.toString());
                        mDriverDatabase.updateChildren(newImage);
                        finish();
                        return;
                }).addOnFailureListener((e) → {
                        finish();
                        return;
                });
        });
    }else {

        Toast.makeText( context: DriverSettingsActivity.this,  text: "Profile Updated!",Toast.LENGTH_SHORT).show();
    }}
}
```

Figure 4.69 Screenshot of update driver details function code

These codes will be trigger when driver clicks the submit button, it will first check all the fields have entered value nicely. If the system found an empty field, it will display an error message. It will update the driver information to the current driver database

## Chapter 5: Experimental Result

### 5.1 Unit Testing

Unit testing is used to test on every module to validate that each unit of the software functioning well. At this stage, each module of the application designed will be taken for testing.

### 5.1.1 MainActivity.java

<table>
<tr><th colspan="5">Name of Test Case</th></tr>
<tr><td colspan="3"><b>Test Case Filename:</b> MainActivity.java</td><td colspan="2"><b>Date Created:</b> 3/5/2020</td></tr>
<tr><td colspan="3"><b>Tester(s):</b> Wong Jun Wei</td><td colspan="2"><b>Access Path:</b> C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1</td></tr>
<tr><td colspan="5"><b>Transactions:</b> CustomerLoginActivity.java / DriverLoginActivity.java</td></tr>
<tr><td colspan="5"><b>Purpose of Test Case:</b> To test user navigate to user login page and driver navigate to driver login page when click the button.</td></tr>
<tr><th>Step No.</th><th>Input Values to Execute Test Case</th><th>Expected Results</th><th>Actual Results</th><th>Pass / Fail - Comments</th></tr>
<tr><td>1.</td><td>User clicks user button</td><td>System go to user login page</td><td>System go to user login page.</td><td>Pass</td></tr>
<tr><td>2.</td><td>Driver clicks driver button</td><td>System go to driver login page</td><td>System go to driver login page</td><td>Pass</td></tr>
</table>

### 5.1.2 CustomerLoginActivity.java

| | | | | |
|---|---|---|---|---|
| **Name of Test Case** | | | | |
| **Test Case Filename:** CustomerLoginActivity.java | | **Date Created:** 5/5/2020 | | |
| **Tester(s):** Wong Jun Wei | | **Access Path:** C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1 | | |
| **Transactions:** Search.java / Admin.java | | | | |
| **Purpose of Test Case:** To test user register new account and login with registered account and admin can login to admin page. | | | | |
| **Step No.** | **Input Values to Execute Test Case** | **Expected Results** | **Actual Results** | **Pass / Fail - Comments** |
| 1. | Admin enters correct email and password of admin account | System go to admin page. | System go to admin page. | Pass |
| 2. | User empty all fields and click register button | System show error message | System show toast: Please enter email or password to sign up! | Pass |
| 3. | User empty all fields and click login button | System show error message | System show toast: Please enter email or password to login! | Pass |
| 4. | User empty email and click register button | System show error message | System show toast: Please enter email or password to sign up! | Pass |
| 5. | User empty password and click register button | System show error message | System show toast: Please enter email or password to sign up! | Pass |
| 6. | User empty email and click login button | System show error message | System show toast: Please enter email or password to login! | Pass |
| 7. | User empty password and click login button | System show error message | System show toast: Please enter email or password to login! | Pass |

| | | | | |
|---|---|---|---|---|
| 8. | User enters invalid format of email and click register button | System show error message | System show toast: Sign up error! | Pass |
| 9. | User enters invalid format of password and click register button | System show error message | System show toast: Sign up error! | Pass |
| 10. | User enters wrong email and click login button | System show error message | System show toast: Sign in error! | Pass |
| 11. | User enters wrong password and click login button | System show error message | System show toast: Sign in error! | Pass |
| 12. | Users enters valid email and password and click register button | System create new account in firebase and go to search activity | User’s email and password created in firebase and go to search activity | Pass |
| 13. | Users enters existed email and password and click register button | System show error message | System show toast: Sign up error! | Pass |
| 14. | Users enters correct email and password and click register button | System go to search activity | System go to search activity | Pass |

### 5.1.3 DriverLoginActivity.java

<table>
<tr><th colspan="5">Name of Test Case</th></tr>
<tr><td colspan="2"><b>Test Case Filename:</b> DriverLoginActivity.java</td><td colspan="3"><b>Date Created:</b> 10/5/2020</td></tr>
<tr><td colspan="2"><b>Tester(s):</b> Wong Jun Wei</td><td colspan="3"><b>Access Path:</b><br>C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1</td></tr>
<tr><td colspan="5"><b>Transactions:</b> DriverMapNew.java / Admin.java</td></tr>
<tr><td colspan="5"><b>Purpose of Test Case:</b> To test driver register new account and login with registered account and admin can login to admin page.</td></tr>
<tr><th>Step No.</th><th>Input Values to Execute Test Case</th><th>Expected Results</th><th>Actual Results</th><th>Pass / Fail - Comments</th></tr>
<tr><td>1.</td><td>Admin enters correct email and password of admin account</td><td>System go to admin page.</td><td>System go to admin page.</td><td>Pass</td></tr>
<tr><td>2.</td><td>Driver empty all fields and click register button</td><td>System show error message</td><td>System show toast: Please enter email or password to sign up!</td><td>Pass</td></tr>
<tr><td>3.</td><td>Driver empty all fields and click login button</td><td>System show error message</td><td>System show toast: Please enter email or password to login!</td><td>Pass</td></tr>
<tr><td>4.</td><td>Driver empty email and click register button</td><td>System show error message</td><td>System show toast: Please enter email or password to sign up!</td><td>Pass</td></tr>
<tr><td>5.</td><td>Driver empty password and click register button</td><td>System show error message</td><td>System show toast: Please enter email or password to sign up!</td><td>Pass</td></tr>
<tr><td>6.</td><td>Driver empty email and click login button</td><td>System show error message</td><td>System show toast: Please enter email or password to login!</td><td>Pass</td></tr>
<tr><td>7.</td><td>Driver empty password and click login button</td><td>System show error message</td><td>System show toast: Please enter email or password to login!</td><td>Pass</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| 8. | Driver enters invalid format of email and click register button | System show error message | System show toast: Sign up error! | Pass |
| 9. | Driver enters invalid format of password and click register button | System show error message | System show toast: Sign up error! | Pass |
| 10. | Driver enters wrong email and click login button | System show error message | System show toast: Sign in error! | Pass |
| 11. | Driver enters wrong password and click login button | System show error message | System show toast: Sign in error! | Pass |
| 12. | Driver enters valid email and password and click register button | System create new account in firebase and go to driver map activity | Driver's email and password created in firebase and go to driver map activity | Pass |
| 13. | Driver enters existed email and password and click register button | System show error message | System show toast: Sign up error! | Pass |
| 14. | Driver enters correct email and password and click register button | System go to driver map activity | System go to driver map activity | Pass |

### 5.1.4 Admin.java

| Name of Test Case | |
|---|---|
| **Test Case Filename:** Admin.java | **Date Created:** 15/5/2020 |
| **Tester(s):** Wong Jun Wei | **Access Path:** C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1 |
| **Transactions:** MainActivity.java | |
| **Purpose of Test Case:** To test admin update the courier price in firebase database and logout function. | |

| Step No. | Input Values to Execute Test Case | Expected Results | Actual Results | Pass / Fail - Comments |
|---|---|---|---|---|
| 1. | Admin enters all couriers' price and click submit button | System update couriers' price at firebase database | System update couriers' price at firebase database | Pass |
| 2. | Admin empty any courier's price and click submit button | System show error message | System show toast: Please fill up all price! | Pass |
| 3. | Admin presses logout button | System goes to main page | System goes to main page | Pass |

### 5.1.5 Search.java

| **Name of Test Case** | | | | |
|---|---|---|---|---|
| **Test Case Filename:** Search.java | | **Date Created:** 15/5/2020 | | |
| **Tester(s):** Wong Jun Wei | | **Access Path:** C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1 | | |
| **Transactions:** NearbyCourier.java | | | | |
| **Purpose of Test Case:** To test the values pass to next activity is correct and all button is functionable. | | | | |
| **Step No.** | **Input Values to Execute Test Case** | **Expected Results** | **Actual Results** | **Pass / Fail - Comments** |
| 1. | User continuously click minus button | Lowest value is 0.5 | Lowest weight is 0.5 | Pass |
| 2. | User continuously click plus button | Highest value is 10.0 | Highest value is 10.0 | Pass |
| 3. | User drags seek bar | System display seek bar value correctly | System display seek bar value correctly | Pass |
| 4. | User clicks search button after choose the weight and search radius | System pass weight and search radius value to next activity | System pass weight and search radius value to next activity | Pass |
| 5. | User clicks search button without choose weight and search radius | System pass default value of weight and search radius to next activity | System pass default value of weight and search radius to next activity | Pass |
| 6. | User choose weight value and click search button | System pass the choose weight value and default search radius to next activity | System pass the choose weight value and default search radius to next activity | Pass |

| | | | | |
|---|---|---|---|---|
| 7. | User choose search radius and click search button | System pass the choose search radius value and default weight to next activity | System pass the choose search radius value and default weight to next activity | Pass |
| 8. | User clicks profile icon | System goes to user profile page | System goes to user profile page | Pass |

### 5.1.6 NearbyCourier.java

| **Name of Test Case** | | | | |
|---|---|---|---|---|
| **Test Case Filename:** NearbyCourier.java | | **Date Created:** 15/5/2020 | | |
| **Tester(s):** Wong Jun Wei | | **Access Path:** C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1 | | |
| **Transactions:** Parcelinfo.java | | | | |
| **Purpose of Test Case:** To test search result of courier companies details and user current location successfully show in map. | | | | |
| **Step No.** | **Input Values to Execute Test Case** | **Expected Results** | **Actual Results** | **Pass / Fail - Comments** |
| 1. | Get user current location | User current location show on map | System displays blue circle that represent current location | Pass |
| 2. | Get all supported courier companies details in the application within the search radius | Get Places API result and display in marker | System displays multiple red markers represent supported courier companies | Pass |
| 3. | Get all supported courier companies’ location | Get Location API result and display below marker | System displays car icon below courier companies’ marker | Pass |
| 4. | User want to know the information of courier company and click marker | System shows information window of the selected courier company | System shows information window of the selected courier company contains its name, address, price, rating and open status. | Pass |
| 5. | User want to know the distance between courier company and current location and click car icon | System shows information window of the selected courier company | System shows information window of the selected courier company contains distance and estimate time to reach there. | Pass |

| | | | | |
|---|---|---|---|---|
| 6. | User want to send parcel to the selected courier company and click information window | System display message ask for confirmation | System show alert dialog: You want send the parcel to “selected courier company” ? | Pass |
| 7. | User regret the decision and press “No” at alert dialog | System return to map | System return to map | Pass |
| 8. | User confirm the decision and press “Yes” at alert dialog | System pass the information of selected courier company to next activity | System pass the information of selected courier company to next activity | Pass |

### 5.1.7 Parcelinfo.java

| Name of Test Case | |
|---|---|
| **Test Case Filename:** Parcelinfo.java | **Date Created:** 25/5/2020 |
| **Tester(s):** Wong Jun Wei | **Access Path:** C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1 |
| **Transactions:** CustomerMapActivity.java | |
| **Purpose of Test Case:** To test user completely fills up all the fields and pass the correct data to next activity. | |

| Step No. | Input Values to Execute Test Case | Expected Results | Actual Results | Pass / Fail - Comments |
|---|---|---|---|---|
| 1. | The courier's price selected from previous activity | System shows correct price | System shows correct price | Pass |
| 2. | User fills up all the fields and click confirm button | System goes to customer map activity | System goes to customer map activity | Pass |
| 3. | User only enter receiver's name and click confirm button | System show error message | System show toast: Please fill up all the fields! | Pass |
| 4. | User only enter receiver's phone number and click confirm button | System show error message | System show toast: Please fill up all the fields! | Pass |
| 5. | User only enter receiver's address and click confirm button | System show error message | System show toast: Please fill up all the fields! | Pass |
| 6. | User click the clear button | System clear all the fields | System clear all the fields | Pass |

### 5.1.8 CustomerMapActivity.java

<table>
<tr><th colspan="5">Name of Test Case</th></tr>
<tr><td colspan="2"><b>Test Case Filename:</b> CustomerMapActivity.java</td><td colspan="3"><b>Date Created:</b> 20/6/2020</td></tr>
<tr><td colspan="2"><b>Tester(s):</b> Wong Jun Wei</td><td colspan="3"><b>Access Path:</b><br>C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1</td></tr>
<tr><td colspan="5"><b>Transactions:</b> MainActivity.java / Search.java / HistoryActivity.java / CustomerSettingsActivity.java</td></tr>
<tr><td colspan="5"><b>Purpose of Test Case:</b> To test the data pass from previous activity is correct and find the correct driver type and go to correct page.</td></tr>
<tr><th>Step No.</th><th>Input Values to Execute Test Case</th><th>Expected Results</th><th>Actual Results</th><th>Pass / Fail - Comments</th></tr>
<tr><td>1.</td><td>User clicks home button</td><td>System goes to the search page</td><td>System goes to the search page</td><td>Pass</td></tr>
<tr><td>2.</td><td>User clicks history button</td><td>System goes to the history page</td><td>System goes to the history page</td><td>Pass</td></tr>
<tr><td>3.</td><td>User clicks logout button</td><td>System goes to main page</td><td>System goes to main page</td><td>Pass</td></tr>
<tr><td>4.</td><td>User clicks settings button</td><td>System goes to user settings page</td><td>System goes to user settings page</td><td>Pass</td></tr>
<tr><td>5.</td><td>Get user current location</td><td>User current location show on map</td><td>System displays blue circle that represent current location</td><td>Pass</td></tr>
<tr><td>6.</td><td>User selects car and click find driver button</td><td>Found a car driver and show distance between driver</td><td>Found a car driver and show distance between driver</td><td>Pass</td></tr>
<tr><td>7.</td><td>User selects motorcycle and click find driver button</td><td>Found a motorcycler driver and show distance between driver</td><td>Found a motorcycler driver and show distance between driver</td><td>Pass</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| 8. | Send user location to firebase database | Firebase add the user location to database | Firebase add the user location to database | Pass |
| 9. | Send information of the selected courier company and parcel information to firebase database when found driver | Firebase add the information to database | Firebase add the information to database | Pass |
| 10. | User clicks driver found button to cancel the request | Firebase delete the request data | Firebase delete the request data | Pass |

### 5.1.9 DriverMapNew.java

<table>
<tr><th colspan="5">Name of Test Case</th></tr>
<tr><td colspan="2"><b>Test Case Filename:</b> DriverMapNew.java</td><td colspan="3"><b>Date Created:</b> 28/6/2020</td></tr>
<tr><td colspan="2"><b>Tester(s):</b> Wong Jun Wei</td><td colspan="3"><b>Access Path:</b><br>C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1</td></tr>
<tr><td colspan="5"><b>Transactions:</b> MainActivity.java / HistoryActivity.java / DriverSettingsActivity.java</td></tr>
<tr><td colspan="5"><b>Purpose of Test Case:</b> To test driver get correct information of user and parcel and go to correct page.</td></tr>
<tr><th>Step No.</th><th>Input Values to Execute Test Case</th><th>Expected Results</th><th>Actual Results</th><th>Pass / Fail - Comments</th></tr>
<tr><td>1.</td><td>Driver clicks history button</td><td>System goes to the history page</td><td>System goes to the history page</td><td>Pass</td></tr>
<tr><td>2.</td><td>Driver clicks logout button</td><td>System goes to main page</td><td>System goes to main page</td><td>Pass</td></tr>
<tr><td>3.</td><td>Driver clicks settings button</td><td>System goes to driver settings page</td><td>System goes to driver settings page</td><td>Pass</td></tr>
<tr><td>4.</td><td>Tap on working switch</td><td>System shows driver current location and continuously update to database</td><td>System displays blue circle that represent current location and firebase continuously update the latitude and longitude</td><td>Pass</td></tr>
<tr><td>5.</td><td>Tap off working switch</td><td>System stops update the location of driver and delete the data in database.</td><td>System stops update the location of driver and delete the data in database</td><td>Pass</td></tr>
<tr><td>6.</td><td>User request data add into database while working switch is on</td><td>System get the request data from database and show on map</td><td>System shows user request information which include user and parcel information and draw route on map</td><td>Pass</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| 7. | Driver clicks picked parcel button | System shows selected courier company’s location and draw route on map | System shows selected courier company’s location and draw route on map | Pass |
| 8. | Driver clicks drive completed | System create history of request information and remove all the marker on map | History of request information created and remove all the marker on map | Pass |

### 5.1.10 HistoryActivity.java

| Name of Test Case | | | | |
|---|---|---|---|---|
| **Test Case Filename:** HistoryActivity.java | | **Date Created:** 2/7/2020 | | |
| **Tester(s):** Wong Jun Wei | | **Access Path:** C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1 | | |
| **Transactions:** HistorySingleActivity.java | | | | |
| **Purpose of Test Case:** To test user and driver able view all the request history and driver able to payout the balance. | | | | |
| **Step No.** | **Input Values to Execute Test Case** | **Expected Results** | **Actual Results** | **Pass / Fail - Comments** |
| 1. | User completed to make a driver request | System shows request ID, date and time at history page | System shows request ID, date and time at history page | Pass |
| 2. | Driver completed a user request | System shows request ID, date and time at history page | System shows request ID, date and time at history page | Pass |
| 3. | Current user is a User | System shows request ID, date and time at history page | System shows request ID, date and time at history page | Pass |
| 4. | Current user is a Driver | System show current driver's balance, payout button | System show current driver's balance, payout button | Pass |
| 5. | Driver enter a valid Paypal email and click payout | System goes to payment gateway | System goes to payment gateway | Pass |
| 6. | Driver enter not valid Paypal email and click payout | System show error message | System show toast: Error: Could not complete Payout | Pass |
| 7. | User clicks the request ID | System show the details of the request ID | System show the details of the request ID | Pass |

| | | | | |
|---|---|---|---|---|
| 8. | Driver clicks the request ID | System show the details of the request ID | System show the details of the request ID | Pass |

### 5.1.11 HistorySingleActivity.java

<table>
<tr><th colspan="5">Name of Test Case</th></tr>
<tr><td colspan="2"><b>Test Case Filename:</b> HistorySingleActivity.java</td><td colspan="3"><b>Date Created:</b> 20/7/2020</td></tr>
<tr><td colspan="2"><b>Tester(s):</b> Wong Jun Wei</td><td colspan="3"><b>Access Path:</b><br>C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1</td></tr>
<tr><td colspan="5"><b>Transactions:</b> PayPalConfig.java</td></tr>
<tr><td colspan="5"><b>Purpose of Test Case:</b> To test display correct information to current user and update the data to the database.</td></tr>
<tr><th>Step No.</th><th>Input Values to Execute Test Case</th><th>Expected Results</th><th>Actual Results</th><th>Pass / Fail - Comments</th></tr>
<tr><td>1.</td><td>Current user is User</td><td>System shows correct user layout</td><td>System shows correct user layout</td><td>Pass</td></tr>
<tr><td>2.</td><td>Current user is Driver</td><td>System shows correct driver layout</td><td>System shows correct driver layout</td><td>Pass</td></tr>
<tr><td>3.</td><td>User clicks the rating bar</td><td>System add the rating bar value to driver rating in database</td><td>System add the rating bar value to driver rating in database</td><td>Pass</td></tr>
<tr><td>4.</td><td>User clicks pay here button</td><td>System goes to payment gateway</td><td>System goes to payment gateway</td><td>Pass</td></tr>
<tr><td>5.</td><td>User successfully pay the ride price</td><td>System add paid result to request ID in database</td><td>Firebase database of request ID value successfully updated</td><td>Pass</td></tr>
<tr><td>6.</td><td>User paid the request ID</td><td>System get the parcel information from database and show to user</td><td>System get the parcel information from database and show to user</td><td>Pass</td></tr>
<tr><td>7.</td><td>User does not pay the request ID</td><td>System show pay here button</td><td>System show pay here button</td><td>Pass</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| 8. | Driver add image, enter tracking number and click submit button | System update the data to firebase | System show toast: Info submitted! and update the database data. | Pass |
| 9. | Driver empty tracking number and click submit button | System show error message | System show toast: Please add tracking number! | Pass |

### 5.1.12 CustomerSettingsActivity.java

| **Name of Test Case** | | | | |
|---|---|---|---|---|
| **Test Case Filename:** CustomerSettingsActivity.java | | **Date Created:** 25/7/2020 | | |
| **Tester(s):** Wong Jun Wei | | **Access Path:** C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1 | | |
| **Transactions:** null | | | | |
| **Purpose of Test Case:** To test user enter valid data and successfully update data in database. | | | | |
| **Step No.** | **Input Values to Execute Test Case** | **Expected Results** | **Actual Results** | **Pass / Fail - Comments** |
| 1. | User empty name field and click confirm button | System show error message | System show toast: Please fill up your name and phone! | Pass |
| 2. | User empty phone field and click confirm button | System show error message | System show toast: Please fill up your name and phone! | Pass |
| 3. | User empty all field and click confirm button | System show error message | System show toast: Please fill up your name and phone! | Pass |
| 4. | User fills up all field and click confirm button | System update the value to database | System show toast: Profile Updated! | Pass |

### 5.1.13 DriverSettingsActivity.java

| Name of Test Case | |
|---|---|
| **Test Case Filename:** DriverSettingsActivity.java | **Date Created:** 29/7/2020 |
| **Tester(s):** Wong Jun Wei | **Access Path:** C:\Users\User\AndroidStudioProjects\parcel1\app\src\main\java\com\example\parcel1 |
| **Transactions:** null | |
| **Purpose of Test Case:** To test driver enter valid data and successfully update data in database. | |

| Step No. | Input Values to Execute Test Case | Expected Results | Actual Results | Pass / Fail - Comments |
|---|---|---|---|---|
| 1. | Driver empty name field and click confirm button | System show error message | System show toast: Please fill up all the fields! | Pass |
| 2. | Driver empty phone field and click confirm button | System show error message | System show toast: Please fill up all the fields! | Pass |
| 3. | Driver empty vehicle number field and click confirm button | System show error message | System show toast: Please fill up all the fields! | Pass |
| 3. | Driver empty all field and click confirm button | System show error message | System show toast: Please fill up all the fields! | Pass |
| 4. | Driver fills up all field and click confirm button | System update the value to database | System show toast: Profile Updated! | Pass |

## 5.2 Usability Testing

Usability testing is used to measure the ease of use and also identifying the problems in the design of the application. It is able to help the application uncovering the opportunity to make improvements in resulting provide target the users with better user experience. The whole application designed will be taken for testing.

### 5.2.1 Visibility of System Status

| Visibility of System Status | | | |
|---|---|---|---|
| **Purpose:** The system should always keep user informed about what is going on, through appropriate feedback within reasonable time | | | |
| **Step No.** | **Review Checklist** | **Yes / No / Not Applicable** | **Comments** |
| 1.1 | Is there a consistent font design and stylist treatment across the system? | Yes | - |
| 1.2 | Is there feedback when function keys are pressed? | Yes | - |
| 1.3 | Does every display begin with a title or header that describes screen contents? | Yes | - |
| 1.4 | Is there a consistent icon design scheme and stylistic treatment across the system? | Yes | - |
| 1.5 | Do menu instructions, prompts, and error messages appear in the same place(s) on each menu? | Yes | - |
| 1.6 | If overtype and insert mode are both available, is there a visible indication of which one the user is in? | Yes | - |
| 1.7 | If pop-up windows are used to display error messages, do they allow the user to see the field in error? | Yes | - |
| 1.8 | Is there some form of system feedback for every operator action? | Yes | - |
| 1.9 | Is there visual feedback in menus or dialog boxes about which choices are selectable? | Yes | - |
| 1.10 | Is there visual feedback when objects are selected or moved? | Yes | - |
| 1.11 | Is the current status of an icon clearly indicated? | Yes | - |
| 1.12 | Are response times appropriate to the user's cognitive processing? | Yes | - |

### 5.2.2 Match Between System and the Real World

| Match Between System and the Real World | | | |
|---|---|---|---|
| **Purpose:** The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system oriented terms. Follow real-world conventions, making information appear in a natural and logical order. | | | |
| **Step No.** | **Review Checklist** | **Yes / No / Not Applicable** | **Comments** |
| 2.1 | Are icons concrete and familiar? | Yes | - |
| 2.2 | Are menu choices ordered in the most logical way, given the user, the item names, and the task variables? | Yes | - |
| 2.3 | If shape is used as a visual cue, does it match cultural conventions? | Yes | - |
| 2.4 | Do the selected colors correspond to common expectations about color codes? | Yes | - |
| 2.5 | When prompts imply a necessary action, are the words in the message consistent with that action? | Yes | - |
| 2.6 | Do keystroke references in prompts match actual key names? | Yes | - |
| 2.7 | On data entry screens, are tasks described in terminology familiar to users? | Yes | - |
| 2.8 | Are field-level prompts provided for data entry screens? | Yes | - |
| 2.9 | For question and answer interfaces, are questions stated in clear, simple language? | Yes | - |
| 2.10 | Does the command language employ user jargon and avoid computer jargon? | Yes | - |
| 2.11 | Are input data codes meaningful? | Yes | - |
| 2.12 | Have uncommon letter sequences been avoided whenever possible? | Yes | - |
| 2.13 | Are function keys labeled clearly and distinctively, even if this means breaking consistency rules? | Yes | - |

### 5.2.3 Consistency and Standards

| Consistency and Standards | | | |
|---|---|---|---|
| **Purpose:** Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions. | | | |
| **Step No.** | **Review Checklist** | **Yes / No / Not Applicable** | **Comments** |
| 3.1 | Have industry or company formatting standards been followed consistently in all screens within a system? | Yes | - |
| 3.2 | Has a heavy use of all uppercase letters on a screen been avoided? | Yes | - |
| 3.3 | Do abbreviations not include punctuation? | Yes | - |
| 3.4 | Are icons labeled? | Yes | - |
| 3.5 | Are there no more than twelve to twenty icon types? | Yes | - |
| 3.6 | Are there salient visual cues to identify the active window? | Yes | - |
| 3.7 | Does each window have a title? | Yes | - |
| 3.8 | Does the menu structure match the task structure? | Yes | - |
| 3.9 | Have industry or company standards been established for menu design, and are they applied consistently on all menu screens in the system? | Yes | - |
| 3.10 | Do on-line instructions appear in a consistent location across screens? | Yes | - |
| 3.11 | Are field labels and fields distinguished typographically? | Yes | - |
| 3.12 | Are field labels consistent from one data entry screen to another? | Yes | - |
| 3.13 | Are fields and labels left-justified for alpha lists and right-justified for numeric lists? | Yes | - |
| 3.14 | Do field labels appear to the left of single fields and above list fields? | Yes | - |
| 3.15 | Are attention-getting techniques used with care? | Yes | - |
| 3.16 | Are there no more than four to seven colors, and are they far apart along the visible spectrum? | Yes | - |
| 3.17 | Have pairings of high-chroma, spectrally extreme colors been avoided? | Yes | - |
| 3.18 | Is the most important information placed at the beginning of the prompt? | Yes | - |
| 3.19 | Are user actions named consistently across all prompts in the system? | Yes | - |
| 3.20 | Are high-value, high-chroma colors used to attract attention? | Yes | - |

### 5.2.4 Help Users Recognize, Diagnose, and Recover from Errors

| **Help Users Recognize, Diagnose, and Recover from Errors** | | | |
|---|---|---|---|
| **Purpose:** Error messages should be expressed in plain language. | | | |
| **Step No.** | **Review Checklist** | **Yes / No / Not Applicable** | **Comments** |
| 4.1 | Are prompts stated constructively, without overt or implied criticism of the user? | Yes | - |
| 4.2 | Do prompts imply that the user is in control? | Yes | - |
| 4.3 | Are prompts brief and unambiguous. | Yes | - |
| 4.4 | Are error messages worded so that the system, not the user, takes the blame? | Yes | - |
| 4.5 | If humorous error messages are used, are they appropriate and inoffensive to the user population? | Yes | - |
| 4.6 | Are error messages grammatically correct? | Yes | - |
| 4.7 | Do error messages avoid the use of violent or hostile words? | Yes | - |
| 4.8 | Do error messages avoid an anthropomorphic tone? | Yes | - |
| 4.9 | Do all error messages in the system use consistent grammatical style, form, terminology, and abbreviations? | Yes | - |
| 4.10 | Do messages place users in control of the system? | Yes | - |
| 4.11 | Do error messages inform the user of the error's severity? | Yes | - |
| 4.12 | Do error messages suggest the cause of the problem? | Yes | - |
| 4.13 | Do error messages provide appropriate semantic information? | Yes | - |
| 4.14 | Do error messages provide appropriate syntactic information? | Yes | - |
| 4.15 | Do error messages indicate what action the user needs to take to correct the error? | Yes | - |

### 5.2.5 Error Prevention

| Error Prevention | | | |
|---|---|---|---|
| **Purpose:** Even better than good error messages is a careful design which prevents a problem from occurring in the first place. | | | |
| **Step No.** | **Review Checklist** | **Yes / No / Not Applicable** | **Comments** |
| 5.1 | If the database includes groups of data, can users enter more than one group on a single screen? | Yes | - |
| 5.2 | Are menu choices logical, distinctive, and mutually exclusive? | Yes | - |
| 5.3 | Are data inputs case-blind whenever possible? | Yes | - |
| 5.4 | Are error messages worded so that the system, not the user, takes the blame? | Yes | - |
| 5.5 | If the system displays multiple windows, is navigation between windows simple and visible? | Yes | - |
| 5.6 | Are the function keys that can cause the most serious consequences in hard-to-reach positions? | Yes | - |
| 5.7 | Are the function keys that can cause the most serious consequences located far away from low consequence and high-use keys? | Yes | - |
| 5.8 | Has the use of qualifier keys been minimized? | Yes | - |
| 5.9 | If the system uses qualifier keys, are they used consistently throughout the system? | Yes | - |
| 5.10 | Does the system prevent users from making errors whenever possible? | Yes | - |
| 5.11 | Does the system warn users if they are about to make a potentially serious error? | Yes | - |
| 5.12 | Do fields in data entry screens and dialog boxes contain default values when appropriate? | Yes | - |

### 5.2.6 Recognition Rather Than Recall

| **Recognition Rather Than Recall** | | | |
|---|---|---|---|
| **Purpose:** Make objects, actions, and options visible. | | | |
| **Step No.** | **Review Checklist** | **Yes / No / Not Applicable** | **Comments** |
| 6.1 | Does the data display start in the upper-left corner of the screen? | Yes | - |
| 6.2 | Are multiword field labels placed horizontally? | Yes | - |
| 6.3 | Are prompts, cues, and messages placed where the eye is likely to be looking on the screen? | Yes | - |
| 6.4 | Have prompts been formatted using white space, justification, and visual cues for easy scanning? | Yes | - |
| 6.5 | Do text areas have "breathing space" around them? | Yes | - |
| 6.6 | Is white space used to create symmetry and lead the eye in the appropriate direction? | Yes | - |
| 6.7 | Have items been grouped into logical zones, and have headings been used to distinguish between zones? | Yes | - |
| 6.8 | Are zones no more than twelve to fourteen characters wide and six to seven lines high? | Yes | - |
| 6.9 | Have zones been separated by spaces, lines, color, letters, bold titles, rules lines, or shaded areas? | Yes | - |
| 6.10 | Are field labels close to fields, but separated by at least one space? | Yes | - |
| 6.11 | Are size, boldface, underlining, color, shading, or typography used to show the relative quantity or importance of different screen items? | Yes | - |
| 6.12 | Are borders used to identify meaningful groups? | Yes | - |
| 6.13 | Has the same color been used to group related elements? | Yes | - |
| 6.14 | Is color coding consistent throughout the system? | Yes | - |
| 6.15 | Is color used in conjunction with some other redundant cue? | Yes | - |
| 6.16 | Is there good color and brightness contrast between image and background colors? | Yes | - |

### 5.2.7 Aesthetic and Minimalist Design

| Aesthetic and Minimalist Design | | | |
|---|---|---|---|
| **Purpose:** Dialogues should not contain information which is irrelevant or rarely needed. | | | |
| **Step No.** | **Review Checklist** | **Yes / No / Not Applicable** | **Comments** |
| 7.1 | Is only information essential to decision making displayed on the screen? | Yes | - |
| 7.2 | Are all icons in a set visually and conceptually distinct? | Yes | Each icon has own use. |
| 7.3 | Have large objects, bold lines, and simple areas been used to distinguish icons? | Yes | - |
| 7.4 | Does each icon stand out from its background? | Yes | High visible icon used. |
| 7.5 | Are meaningful groups of items separated by white space? | Yes | - |
| 7.6 | Does each data entry screen have a short, simple, clear, distinctive title? | Yes | - |
| 7.7 | Are field labels brief, familiar, and descriptive? | Yes | - |

### 5.2.8 Privacy

| Step No. | Review Checklist | Yes / No / Not Applicable | Comments |
|---|---|---|---|
| **Privacy** | | | |
| **Purpose:** The system should help the user to protect personal or private information. | | | |
| 6.1 | Are protected areas completely inaccessible? | Yes | User cannot access the protected areas without password. |
| 6.2 | Can protected or confidential areas be accessed with certain passwords. | Yes | - |
| 6.3 | Is this feature effective and successful? | Yes | System safer. |

# Chapter 6 Conclusion

## 6.1 Project Review, Discussions and Conclusion

The raising of technology and business makes E-commerce business becoming more and more complex. There are multiple ways to run an e-commerce business. Using online shopping platforms, social media applications even using messaging apps also can start an online business. In order to ship out the product to the buyer, the online seller needs to find out the courier company that can provide lower shipping fees to reduce the business costs. At the same time, they also want to satisfy the need of the customer. What the customer wants is to buy the product at a lower cost and get the product as fast as possible.

The comparison of courier company shipping fees and the time of seller send the parcel to the courier centre is causing the waste of so much of time. Therefore, this project will develop an ecommerce shipping solution for mobile platform that can benefit not only the seller but their customers as well in reducing the shipping fee charge and can receive their parcel faster. Through this application, seller can have more money gain because this application helps them to increase the amount of parcel ship and achieve effective shipping. Besides, seller also can always find a courier which best suit their requirement.

This application is developed to solve the problems of seller such as bring a loss of money when keep using the same courier delivery company, facing low efficiency of shipping process while their areas do not provide pickup service from courier delivery company and forced to increase of the business costs in order to meet the customer expectation.

To solve these problems stated, this application is planned to develop with solutions such as provide GPS search function to find out nearby courier delivery companies and get to know the details of them. Secondly, the application also will provide a delivery service to the vendors if they wish to ship their parcel to the selected courier delivery company effectively. Last but not least, the application provides the price and distance estimation framework to help the seller determine which courier delivery company is their best suit choice.

The main challenge of this project is the price listed will not be so accurate because it is an estimate price. The price of courier company offer will be changed sometimes. The administrator has to update the price frequently.

Another challenge of this project is when there is no active driver around, the seller could not found a driver to help send the parcel to the selected courier company. The seller has to send the parcel with themselves if happen this problem.

Overall, the ecommerce shipping solution for mobile platform gives many benefits to not only sellers in e-commerce business but the buyers also will increase the satisfaction of experiencing online buying because it is very cost-effective and convenient.

## 6.2 Novelties and Contributions

The development of this project will contribute to the e-commerce sellers, buyers, courier companies and also driver of this application. Through using this application, the e-commerce sellers can ship out their parcels effectively and conveniently. The time for sellers operates their online business will be increased. So that, they will be found more buyers and resulting in increased revenue. With decreasing the burden of the seller, the buyer can get the product they want to buy at a lower price and the thing also will be reached faster. Other than that, the seller will not just be using one specific courier company because they can found other courier companies that offer a lower shipping fee. As result, the medium size of the courier company can increase the productivity of the business. Lastly, the application also opens a vacancy to the people who know how to drive and give them gain extra income.

## 6.3 Future Work

In the coming future, the system can cooperate with more and more courier company so that the seller will have more options to ship out their parcel. The price estimated will be more accurate and it can be set by the courier company staff. The type of transport also will add more such as bicycle and lorry.

## Bibliography

Allen, K. (2016). *THE 21ST CENTURY SPICE TRADE*. [ebook] DHL Express, pp.3-19. Available at: http://www.dhl.com/content/dam/downloads/g0/press/publication/g0_dhl_express_cross_border_ecommerce_21st_century_spice_trade.pdf [Accessed 29 Mar. 2019].

Battan, S. and Choo, J. (2016). *E-Commerce Trends and Challenges: A Logistics and Supply Chain Perspective*. [ebook] The Logistics Institute, pp.7-54. Available at: http://www.tliap.nus.edu.sg/pdf/whitepapers/vol16-Nov-ti.pdf [Accessed 29 Mar. 2019].

Dhl.de. (2019). *Premium*. [online] Available at: https://www.dhl.de/en/paket/information/geschaeftskunden/premium.html [Accessed 29 Mar. 2019].

Frey, C., Bilerman, M., Guy, A., Chua, J. and Kate, M. (2017). *Citi GPS: Global Perspectives & Solutions*. [ebook] Citigroup, pp.3-55. Available at: https://www.oxfordmartin.ox.ac.uk/downloads/CITI%20REPORT%20ADR0N.pdf [Accessed 29 Mar. 2019].

Garg, R. (2019). *Next Generation E-Commerce Marketplace is Here and Technology is Fuelling its Growth*. [online] Entrepreneur. Available at: https://www.entrepreneur.com/article/317179 [Accessed 29 Mar. 2019].

Hasnan, Norlena, et al. “SIX MAIN INNOVATION ISSUES: A CASE OF SERVICE INNOVATION OF POSTAL AND COURIER SERVICES IN MALAYSIA.” Journal of Technology Management and Business, vol. 1, no. 1, 20 July 2014, publisher.uthm.edu.my/ojs/index.php/jtmb/article/view/910. [Accessed 1 Apr. 2020].

Kraus, T. (2018). *Zone Skipping Strategies to Reduce E-Commerce Shipping Costs*. [ebook] Honeywell Intelligrated, pp.3-4. Available at: https://www.intelligrated.com/sites/default/files/resources/10081-%28ZSSWP%29-ZoneSkipping_WhitePaper_FINAL_LR3_0.pdf [Accessed 29 Mar. 2019].

Moreno, K., Moreno, H., Brendish, L. and Pasternak, Z. (2017). *SHIPPING AS STRATEGY: How Small and Midsize Retailers Can Best Meet Customers' Delivery Expectations in the Age of Amazon*. [ebook] FORBES INSIGHTS, pp.4-14. Available at: https://i.forbesimg.com/2017/10/6/PB_ecommerce.pdf [Accessed 29 Mar. 2019].

Sellercentral.amazon.com. (2019). *Premium Shipping - Amazon Seller Central*. [online] Available at: https://sellercentral.amazon.com/gp/help/external/201503640?language=en-US&ref=mpbc_201616540_cont_201503640 [Accessed 29 Mar. 2019].

**Appendix**

# FINAL YEAR PROJECT WEEKLY REPORT

*(Project II)*

| | |
|---|---|
| **Trimester, Year: Y4S3** | **Study week no.: 2** |
| **Student Name & ID: Wong Jun Wei** | |
| **Supervisor: Mr. Yeck Yin Ping** | |
| **Project Title: ECOMMERCE SHIPPING SOLUTION FOR MOBILE PLATFORM** | |

**1. WORK DONE**

Previous work from Project I.

**2. WORK TO BE DONE**

Research on driver module and payment module.
Finish the development of whole application.

**3. PROBLEMS ENCOUNTERED**

No

**4. SELF EVALUATION OF THE PROGRESS**

Working progress is quite smooth.

Supervisor's signature

Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

*(Project II)*

| | |
|---|---|
| **Trimester, Year: Y4S3** | **Study week no.: 4** |
| **Student Name & ID: Wong Jun Wei 15ACB06127** | |
| **Supervisor: Mr. Yeck Yin Ping** | |
| **Project Title: ECOMMERCE SHIPPING SOLUTION FOR MOBILE PLATFORM** | |

**1. WORK DONE**

Study the driver module.

**2. WORK TO BE DONE**

Modify the driver module sample.

**3. PROBLEMS ENCOUNTERED**

The free credit given by Google Cloud Platform has expired, have to link the project to another account or upgrade the current account.

**4. SELF EVALUATION OF THE PROGRESS**

Slow, have to speed up the progress.

Supervisor's signature

Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

*(Project II)*

| | |
|---|---|
| **Trimester, Year: Y4S3** | **Study week no.: 6** |
| **Student Name & ID: Wong Jun Wei 15ACB06127** | |
| **Supervisor: Mr. Yeck Yin Ping** | |
| **Project Title: ECOMMERCE SHIPPING SOLUTION FOR MOBILE PLATFORM** | |

**1. WORK DONE**

Driver module completed.

**2. WORK TO BE DONE**

Add payment module (Stripe platform) into the application.

**3. PROBLEMS ENCOUNTERED**

No.

**4. SELF EVALUATION OF THE PROGRESS**

Slow, have to speed up the progress.

Supervisor's signature

Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

*(Project II)*

| | |
|---|---|
| **Trimester, Year: Y4S3** | **Study week no.: 8** |
| **Student Name & ID: Wong Jun Wei 15ACB06127** | |
| **Supervisor: Mr. Yeck Yin Ping** | |
| **Project Title: ECOMMERCE SHIPPING SOLUTION FOR MOBILE PLATFORM** | |

**1. WORK DONE**

Half of the new payment module (paypal).

**2. WORK TO BE DONE**

Full payment module (paypal).
Link all modules and redesign better UI.

**3. PROBLEMS ENCOUNTERED**

Previous payment module (Stripe) not supported in Malaysia.

**4. SELF EVALUATION OF THE PROGRESS**

Slow, have to speed up the progress.

Supervisor's signature

Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

*(Project II)*

| | |
|---|---|
| **Trimester, Year: Y4S3** | **Study week no.: 10** |
| **Student Name & ID: Wong Jun Wei 15ACB06127** | |
| **Supervisor: Mr. Yeck Yin Ping** | |
| **Project Title: ECOMMERCE SHIPPING SOLUTION FOR MOBILE PLATFORM** | |

**1. WORK DONE**

Driver module (ride-hailing, history, payment)

**2. WORK TO BE DONE**

Admin module (add and update courier price rate).
Horizontal card view at courier company searching map.
Parcel information form.
Concurrently start writing the FYP2 report.

**3. PROBLEMS ENCOUNTERED**

No.

**4. SELF EVALUATION OF THE PROGRESS**

Very slow.

Supervisor's signature

Student's signature

# Poster

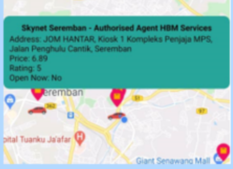
UTAR UNIVERSITI TUNKU ABDUL RAHMAN

## ECOMMERCE SHIPPING SOLUTION FOR MOBILE PLATFORM

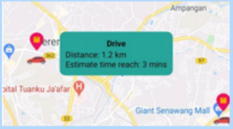
Student: Wong Jun Wei

Supervisor: Yeck Yin Ping

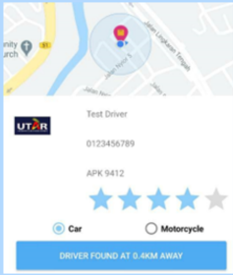
### RESULTS



Skynet Seremban - Authorised Agent HBM Services
Address: JOM HANTAR, Kiosk 1 Kompleks Penjaja MPS, Jalan Penghulu Cantik, Seremban
Price: 6.89
Rating: 5
Open Now: No



Drive
Distance: 1.2 km
Estimate time reach: 3 mins



Test Driver

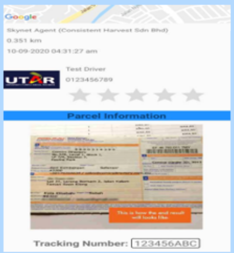
0123456789

APK 9412

★★★★☆

◉ Car ○ Motorcycle

DRIVER FOUND AT 0.4KM AWAY



Skynet Agent (Consistent Harvest Sdn Bhd)
0.351 km
10-09-2020 04:31:27 am

Test Driver
0123456789

☆☆☆☆☆

Parcel Information

This is how the end result will looks like

Tracking Number: 123456ABC

### CONCLUSION

The ecommerce shipping solution for mobile platform gives many benefits to not only sellers in e-commerce business but the buyers also will increase the satisfaction of experiencing online buying because it is very cost-effective and convenient.

### INTRODUCTION

This project is about an ecommerce shipping solution for mobile platform that aims to reduce the shipping fee charge and also decrease the time waiting for the buyer to receive their parcel.

### PROBLEM STATEMENT

- Carrier-specific systems bring a loss of the money to vendor and customer
- Low efficiency of shipping process increases the waiting time of customer
- Increase true business costs within the whole supply chain to meet customer expectation

### OBJECTIVE

- To provide a GPS search function to find out the nearby courier delivery companies around the vendor location and suggest it to the vendor.
- To develop an ecommerce shipping solution for mobile platform that allows vendor can have multiple options on choosing a courier delivery company and request a driver sends their parcel to the selected courier delivery company.
- To provide a price and distance estimation framework to help vendor evaluate which courier delivery company is the best choice compare with other courier delivery companies to ship the parcel.

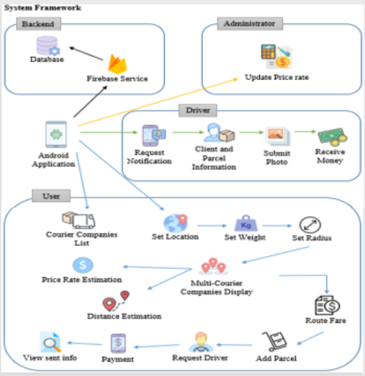
### PROJECT SCOPE

- Develop an ecommerce shipping solution for mobile platform to benefit the vendor and customer.
- This application will be built in a global positioning system (GPS) function to find out all the nearby courier delivery companies around the vendor location.
- Help vendors to find a driver to send their parcel to the selected courier delivery company with extremely low route fare charging to reduce the burden of the vendor.

### DISCUSSION

| Author | Characteristic | | | | | | |
|---|---|---|---|---|---|---|---|
| | Optimum Asset Utilization | Time Saving | Cost-effectively | Flexibility | Reliability | Heterogeneous Shipping | Availability |
| Moreno. H. S. et al. | | | ✓ | ✓ | | ✓ | ✓ |
| Allen. K | | | | ✓ | ✓ | ✓ | ✓ |
| Joerss. M. et al. | ✓ | ✓ | ✓ | | ✓ | | |
| Battan. S. et al. | ✓ | ✓ | ✓ | | ✓ | ✓ | |
| Frey. C. B. et al. | | ✓ | ✓ | ✓ | | | ✓ |
| Kraus. T | ✓ | ✓ | ✓ | | | | ✓ |
| eCommerce Shipping Solution for Mobile Platform | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

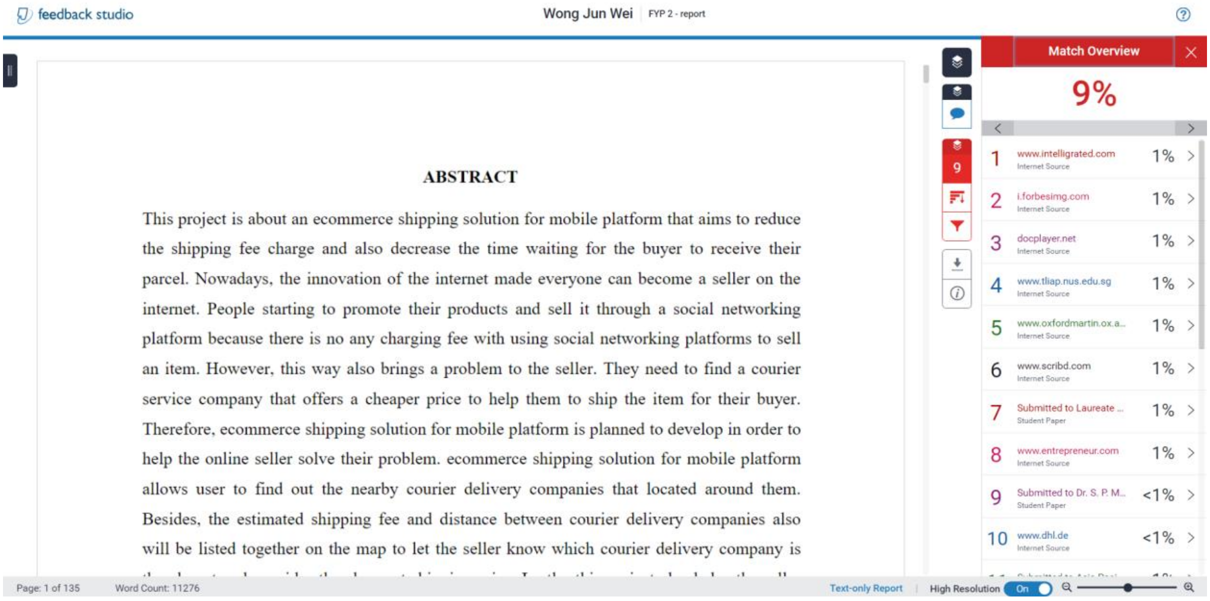
### FRAMEWORK



### NOVELTIES AND CONTRIBUTIONS

1. This project will contribute to the e-commerce sellers, buyers, courier companies and also driver of this application. Through using this application, the e-commerce sellers can ship out their parcels effectively and conveniently.
2. The time for sellers operates their online business will be increased. So that, they will be found more buyers and resulting in increase revenue. With decreasing the burden of the seller, the buyer can get the product they want to buy at a lower price and the thing also will be reached faster.
3. Other than that, the seller will not just be using one specific courier company because they can found other courier companies that offer a lower shipping fee.
4. As result, the medium size of the courier company can increase the productivity of the business. Lastly, the application also opens a vacancy to the people who know how to drive and give them gain extra income.

# Plagiarism Check Result



Wong Jun Wei | FYP 2 - report

**ABSTRACT**

This project is about an ecommerce shipping solution for mobile platform that aims to reduce the shipping fee charge and also decrease the time waiting for the buyer to receive their parcel. Nowadays, the innovation of the internet made everyone can become a seller on the internet. People starting to promote their products and sell it through a social networking platform because there is no any charging fee with using social networking platforms to sell an item. However, this way also brings a problem to the seller. They need to find a courier service company that offers a cheaper price to help them to ship the item for their buyer. Therefore, ecommerce shipping solution for mobile platform is planned to develop in order to help the online seller solve their problem. ecommerce shipping solution for mobile platform allows user to find out the nearby courier delivery companies that located around them. Besides, the estimated shipping fee and distance between courier delivery companies also will be listed together on the map to let the seller know which courier delivery company is

Match Overview

9%

| | | |
|---|---|---|
| 1 | www.intelligrated.com<br>Internet Source | 1% |
| 2 | i.forbesimg.com<br>Internet Source | 1% |
| 3 | docplayer.net<br>Internet Source | 1% |
| 4 | www.tliap.nus.edu.sg<br>Internet Source | 1% |
| 5 | www.oxfordmartin.ox.a...<br>Internet Source | 1% |
| 6 | www.scribd.com<br>Internet Source | 1% |
| 7 | Submitted to Laureate ...<br>Student Paper | 1% |
| 8 | www.entrepreneur.com<br>Internet Source | 1% |
| 9 | Submitted to Dr. S. P. M...<br>Student Paper | <1% |
| 10 | www.dhl.de<br>Internet Source | <1% |

Page: 1 of 135    Word Count: 11276

## FYP 2 - report

ORIGINALITY REPORT

| 9% | 7% | 0% | 4% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | www.intelligrated.com<br>Internet Source | 1% |
| 2 | i.forbesimg.com<br>Internet Source | 1% |
| 3 | docplayer.net<br>Internet Source | 1% |
| 4 | www.tliap.nus.edu.sg<br>Internet Source | 1% |
| 5 | www.oxfordmartin.ox.ac.uk<br>Internet Source | 1% |
| 6 | www.scribd.com<br>Internet Source | 1% |
| 7 | Submitted to Laureate Education Inc.<br>Student Paper | 1% |
| 8 | www.entrepreneur.com<br>Internet Source | 1% |
| 9 | Submitted to Dr. S. P. Mukherjee International Institute of Information Technology (IIIT-NR) | <1% |

|  |  |  |
|---|---|---|
|  | Student Paper |  |
| 10 | www.dhl.de<br>Internet Source | <1 % |
| 11 | Submitted to Asia Pacific University College of Technology and Innovation (UCTI)<br>Student Paper | <1 % |
| 12 | developer.android.com<br>Internet Source | <1 % |
| 13 | Submitted to CSU, San Jose State University<br>Student Paper | <1 % |
| 14 | Submitted to Canterbury Christ Church University<br>Student Paper | <1 % |
| 15 | www.coursehero.com<br>Internet Source | <1 % |
| 16 | Submitted to University of Greenwich<br>Student Paper | <1 % |
| 17 | assets.kpmg<br>Internet Source | <1 % |
| 18 | Submitted to Taylor’s Education Group<br>Student Paper | <1 % |
| 19 | Submitted to University of London External System<br>Student Paper | <1 % |

|  |  |  |
|---|---|---|
| 20 | Submitted to University of Information Technology, Yangon<br>Student Paper | <1% |

|  |  |  |  |
|---|---|---|---|
| Exclude quotes | On | Exclude matches | < 2 words |
| Exclude bibliography | On |  |  |

| | | | |
|---|---|---|---|
| **Universiti Tunku Abdul Rahman** | | | |
| **Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)** | | | |
| Form Number: FM-IAD-005 | Rev No.: 0 | Effective Date: 01/10/2013 | Page No.: 1of 1 |

## FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

| | |
|---|---|
| **Full Name(s) of Candidate(s)** | Wong Jun Wei |
| **ID Number(s)** | 15ACB06127 |
| **Programme / Course** | BACHELOR OF INFORMATION SYSTEMS (HONS) INFORMATION SYSTEMS ENGINEERING |
| **Title of Final Year Project** | eCommerce Shipping Solution for Mobile Platform |

| **Similarity** | **Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)** |
|---|---|
| **Overall similarity index:** 9 **%**<br>**Similarity by source**<br>Internet Sources: 7 %<br>Publications: 0 %<br>Student Papers: 4 % | |
| **Number of individual sources listed** of more than 3% similarity: 0 | These similarities are titles and captions which are standardized across reports. |

**Parameters of originality required and limits approved by UTAR are as Follows:**

**(i) Overall similarity index is 20% and below, and**

**(ii) Matching of individual sources listed must be less than 3% each, and**

**(iii) Matching texts in continuous block must not exceed 8 words**

***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

***Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.***

Signature of Supervisor

Name: Yeck Yin Ping

Date: 10/9/2020

Signature of Co-Supervisor

Name: ______________

Date: ______________

# UNIVERSITI TUNKU ABDUL RAHMAN

## FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

### CHECKLIST FOR FYP2 THESIS SUBMISSION

| | |
|---|---|
| Student Id | 15ACB06127 |
| Student Name | Wong Jun Wei |
| Supervisor Name | Yeck Yin Ping |

| TICK (√) | **DOCUMENT ITEMS**<br>Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item. |
|---|---|
| ✓ | Front Cover |
| ✓ | Signed Report Status Declaration Form |
| ✓ | Title Page |
| ✓ | Signed form of the Declaration of Originality |
| ✓ | Acknowledgement |
| ✓ | Abstract |
| ✓ | Table of Contents |
| ✓ | List of Figures (if applicable) |
| ✓ | List of Tables (if applicable) |
| ✓ | List of Symbols (if applicable) |
| ✓ | List of Abbreviations (if applicable) |
| ✓ | Chapters / Content |
| ✓ | Bibliography (or References) |
| ✓ | All references in bibliography are cited in the thesis, especially in the chapter of literature review |
| ✓ | Appendices (if applicable) |
| ✓ | Poster |
| ✓ | Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005) |

*Include this form (checklist) in the thesis (Bind together as the last page)

| | |
|---|---|
| I, the author, have checked and confirmed all the items listed in the table are included in my report.<br>________________________<br>(Signature of Student)<br>Date:10/9/2020 | Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.<br>________________________<br>(Signature of Supervisor)<br>Date: 10/9/2020 |