APPOINTMENT SCHEDULING SYSTEM FOR CAR WASH SERVICE

BY

LEE YING FOONG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

MAY 2021

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: APPOINTMENT SCHEDULING SYSTEM FOR CAR WASH SERVICE

Academic Session: MAY 2021

I LEE YING FOONG
(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,
ying	[illegible]
(Author's signature)	(Supervisor's signature)
Address:	
81, Tingkat Zarib 6,	
Taman Pinji Mewah,	Yeck Yin Ping
31500 Ipoh, Perak.	Supervisor's name
Date: 3/9/2021	Date: 3/9/2021

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Date: ___3/9/2021___

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that _____LEE YING FOONG_______ (ID No: __17ACB02770___) has completed this final year project/ dissertation/ thesis* entitled "Appointment Scheduling System for Car Wash Service" under the supervision of _Mr. Yeck Yin Ping__(Supervisor) from the Department of Computer Science__, Faculty of __Information and Communication Technology__.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

ying

____
LEE YING FOONG

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled “APPOINTMENT SCHEDULING SYSTEM FOR CAR WASH SERVICE” 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	:	ying
Name	:	LEE YING FOONG
Date	:	3/9/2021

ACKNOWLEDGEMENTS

I would like to express my appreciation to my supervisors, Mr. Yeck Yin Ping who has given me this learning opportunity and work on this project. I offer my sincere appreciation to Mr. Yeck for encouragement and guidance throughout Final Year Project.

Besides, I would also like to thank my family and friends for supporting me throughout the project.

ABSTRACT

This project is a mobile application for appointment scheduling system for car wash service. The demand of car wash service nowadays is increasing as most of the people are too busy to wash car themselves. Car wash center has provided various car wash or car detailing services such as foam wash, waxing, polishing which cause convenient to customers. However, there are some limitations found which customers has to queue up for a long time until their turn. During the peak hour, customers might have to wait for a few hours which actually waste customers' time. Next, customers who are not familiar to the area are difficult to get a car wash service. They do not know which is the nearest car wash center from their own location. Another problem faced is the attendance of car wash appointment which people think that booking an appointment is troublesome as they have to go to the car wash center to book appointment or they have to make calls to discuss with staff for available time slot. These issues and limitations need to solved to improve the performance of appointment scheduling service. Hence, the proposed solutions in this application are able to solve the problems by integrating appointment scheduling system in car wash service mobile application. Customers are able to make appointment by using the application. Besides, GPS location tracking is used in order to solve the problem that users wanted to get car wash service at car wash center nearby them. The location of the car wash center is stored in the database and it will be retrieve and display at Google Map when it is nearby user's current location. In addition, Improved Least Laxity First scheduling algorithm is also used to schedule the appointment on real-time basis. Customers can manage their appointment at anytime and anywhere. These proposed solutions are to improve the performance and customers' user experience of the application.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 Problem Statement

Car wash service had become common and widely spread among all around the world nowadays. However, there are some problems faced by car wash industry. Some customers found out that long queue for car wash service is wasting their time and they are not familiar with the location of car wash center nearby their location. Besides, car wash center would also like to increase their customers' attendance.

i. **Long queue for car wash is time wasting**

   When customers want to obtain car wash service or car grooming service, they have to proceed to car wash center. Besides, they have to queue up if all the car wash center's cleaner is serving other customers. The minimum duration to complete a car wash service is 20 minutes. Thus, customers may have to wait at the car wash center for a few hours only they can be served during the peak hour. However, the society now is living in fast pace of life. Most of the customers are busy as they have to go to work and they are not able to wait for the queue and wash their car. Thus, it causes inconvenient to the customers as they are wasting their time waiting at the car wash center.

ii. **Poor attendance of car wash appointment**

   Customers of car wash come from different age group and background. In the old days, people make appointment by making phone call and discuss with the staff of the car wash center for available time slot. Some of them might also purposely go for miles to the car wash center to make appointment. It is difficult and inconvenient for people who are busy. Thus, most of the people will not make appointment as they felt troublesome to do so.

iii. **Unfamiliar with the location of car wash center near them**

   Customers who went for work at outstation or went to other places may wish to have a car wash service. However, they did not know if there are any car wash center near them as they cannot browse any car wash center near them. Customers also did not know the location of the car wash center around them and they need to be guided. Besides, there are some customers who want to go to the nearest car wash center but they cannot do so. Thus, it is difficult for customers to get a car wash service as they are not familiar with that area.

1.2 Background and Motivation

Car wash is a service provided which used to clean the exterior or the interior of the vehicles. There are many type of car wash service such as self-serve, fully automated or full-service. Usually people have to pay for car wash service and their vehicles is washed by the workers at car wash center. For self-service car wash, after customers reach the station, they have to select various setting on the self-service station. Then, customers will be able to clean their vehicles themselves. Automated car wash such as exterior rollover car wash is where a car wash tools moves over customers' car to clean it. Next, for full-service, the vehicles are cleaned manually by the workers in car wash center. Car wash is used to get rid of the dirt of vehicles. Cleaning the car from time to time will protect the value of the vehicles as the vehicles will not spoilt easily.

The needs for car wash service increased every year thus the car wash industry are growing in short-term and long-term. According to Paul Fazio CEO, over 19.2% increase in the number of customers that getting a professional car wash service in 2014 compared to 1996 (Paul Fazio CEO, 2015). Besides, a survey also shown that there is a growth in car wash industry which the number of car washed in 2012 is 2.1% higher than the previous year (Gaille, 2017).

Nowadays, car wash service can be found in anywhere such as stand-alone car wash center, gas station, petrol station and elsewhere. Car wash service now are evolving and using more advanced, eco-friendly and convenient equipment and the industry is becoming successful today.

The main motivation of developing this project is to allow users make appointment of the car wash center online. The user just has to select and fill in some details in order to make an appointment successfully. It saves many time as user does not need to call to the car wash center to confirm the appointment or went to the car wash center to make an appointment.

1.3 Project Objectives

In this project, the objectives are discussed as follows. The main objective is to develop a mobile application which is appointment scheduling system for car wash service.

i. **To apply Least Laxity First (LLF) scheduling algorithm for scheduling appointments on real-time basis.**

   Least Laxity First (LLF) scheduling algorithm will be applied in this project in order to schedule the appointments. Users can view the time-slot available and choose the time-slot they preferred on real-time basis. Real-time basis scheduling algorithm brings convenient to customers and increases customers' satisfaction.

ii. **To implement GPS location tracking for real-time status monitoring.**

   GPS location tracking is used to track the location of car wash nearby customers' location. Customers able to view the distance and details of each car wash center. Furthermore, customers can also check on the current status and availability of the car wash center. Hence, customers are able to decide which car wash center they preferred.

iii. **To integrate appointment scheduling system in the car wash service mobile application.**

   Customers is important as it is the foundation of a success business. Users just have to download the mobile application and they can make appointment for car wash service. They do not have to search for the car wash center and call to book for appointment. Appointment scheduling system can enhance the user experience while using this mobile application. Hence, this application can increase the usage of users and also provide convenient to users.

1.4 Project Scope

The focus of the project is to develop a mobile application which is an appointment scheduling system for car wash service. The mobile application is in Android platform. Users are able to make appointment for car wash service using the mobile application at anywhere and anytime. Besides, users also able to browse the car wash center that are located near their locations.

i. **Real-time scheduling**

   Real-time scheduling is included in the scheduling system for car wash service. Scheduling algorithm which is Least Laxity First (LLF) is applied in order to schedule and make arrangements for the appointments made by users. LLF scheduling algorithm plan the appointments according to the laxity of the appointments. Users are able to select and reserve a time slot for car wash service online by using the mobile application developed.

ii. **GPS location tracking**

   It is a feature that use GPS service to track the location of the car wash center. The users are able to browse car wash center nearby their location in the GPS. Users just have to select the car wash center and GPS will guide them to the destination. Besides, users can also track the status of the car wash center for example users can get to know how many cars are queuing for car wash service and the estimated time.

iii. **Automated notification**

   When users confirm or cancel the appointment, notification is sent automatically for confirmation. Sending reminders and confirmations can directly increase the customers to attend the appointment (Anastasia Masters, 2019). This feature benefit user which is busy or forgetful.

1.5 System Approach / Study

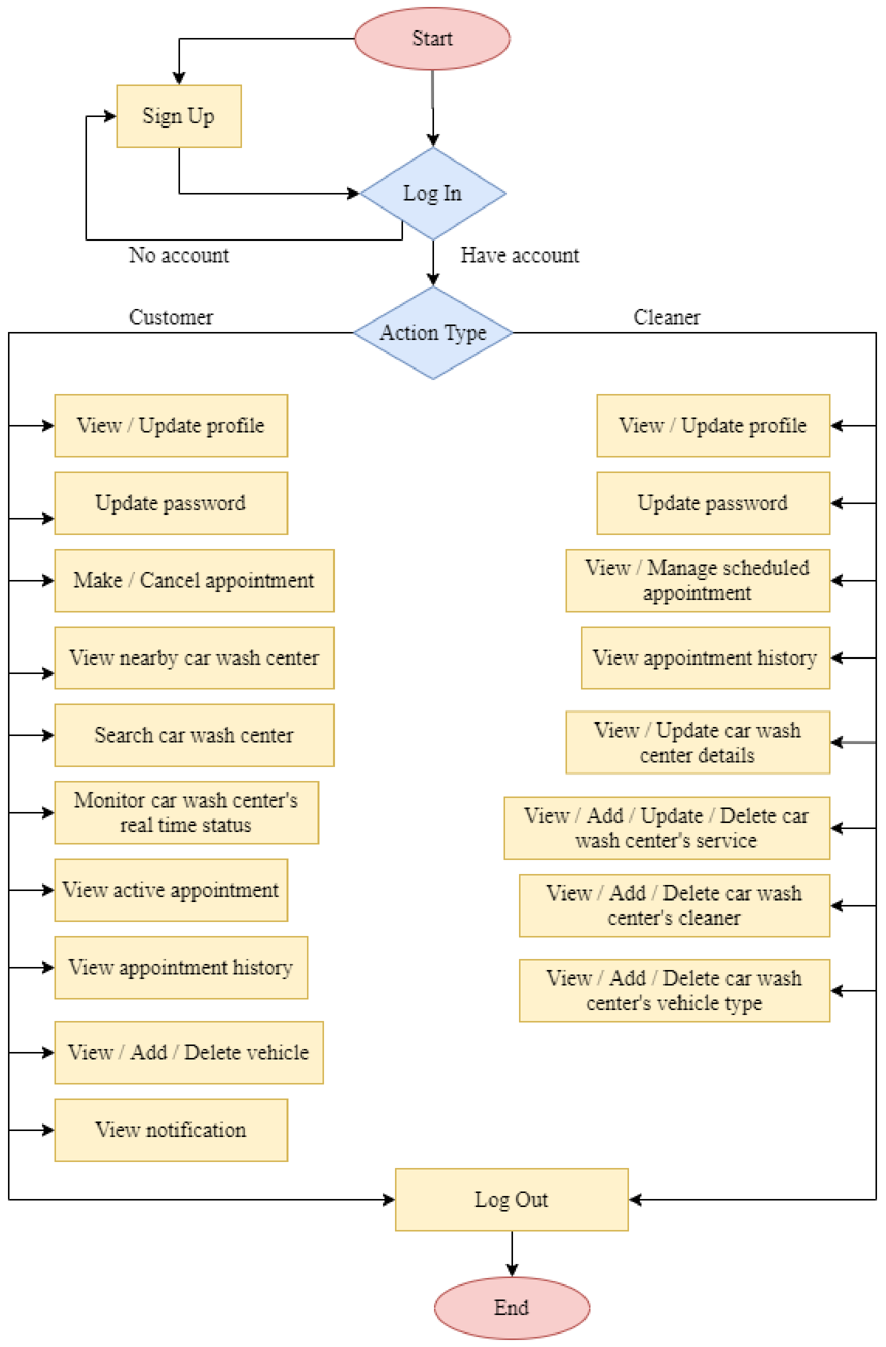

Start
Sign Up
Log In
No account
Have account
Customer
Action Type
Cleaner
View / Update profile
Update password
Make / Cancel appointment
View nearby car wash center
Search car wash center
Monitor car wash center's real time status
View active appointment
View appointment history
View / Add / Delete vehicle
View notification
View / Update profile
Update password
View / Manage scheduled appointment
View appointment history
View / Update car wash center details
View / Add / Update / Delete car wash center's service
View / Add / Delete car wash center's cleaner
View / Add / Delete car wash center's vehicle type
Log Out
End

Figure 1-5-1: Flowchart of Appointment Scheduling System for Car Wash Service

Firstly, user must register an account to access the application. If the user already has an account, user can just sign in the account. When user signed in successfully, the user role is identified. Customer and car wash center's cleaner are able to perform different actions in the application. Customer and cleaner both are able to manage their own profile and also manage their password. Customer is able to make or cancel the appointment. Customer can view the car wash center nearby customer's location. The location of the car wash center nearby is represented by the marker and the marker will be pinned on the map. Besides, customer also able to search the car wash center by name. The status of car wash center can also monitor by customer. Customer can select their preferable car wash center. Customer need to select the vehicle, time slot and service in order to book an appointment successfully. Customer can also cancel an appointment within an hour after made the appointment. Notification is received by customer when they successfully made an appointment or cancelled an appointment. In addition, customer can view the active appointment and appointment history. Customer able to manage the vehicle. They can view, add or delete the vehicle.

Next, cleaner is able to manage the scheduled appointment list. Cleaner can view the scheduled appointment and manage the status of the appointment. Cleaner can change the status of the appointment and notify the customer. Cleaner can manage the car wash center profile, car wash center’s service, vehicle type and cleaner. Cleaner can view, edit or delete car wash center service while they can view, add and delete cleaner and vehicle type. Moreover, any updates will also be updated in the database. Lastly, customer and car wash center’s cleaner can log out the application.

1.6 Highlight of What Have Been Achieved

Appointment Scheduling System for Car Wash Service mobile application is developed to provide an online platform to make an appointment through mobile application. It is to improve the quality and efficiency of customer making an appointment and also improve the service and increase profit of the car wash center. In this mobile application, the appointment scheduling system is integrated into the car wash service. Customer can just download the application and mobile application and register for an account then they are able to enjoy the service such as make appointment and search for car wash center nearby. Hence, the attractive function in the application can increase the usage of customers in the same time increase the profit of the car wash center.

For customer, they can view the car wash center nearby, their distance, and also the details and real time status of the car wash center. They can also search for car wash center that by name if the car wash center is not nearby user's location. They can see all the available service, and available time slot for the selected service. Next, when user confirm the appointment, a notification is sent to customer in order to inform the confirmation of appointment. Customer can view their active appointment in the home page of the application. Besides, they can view their appointment history and also notification received. All the appointment details and notification details is display for customer to have a better view. When they clicked on the car wash center's marker on the map in the appointment details page, the application will direct customer to the Google Map application and direct user to the car wash center. Customer can also manage their own account and change their password. they can change the profile details and password as they preferred. Customer can also manage their own vehicle. They can add or delete their vehicle. Hence, they can just select the vehicle when they make appointment, so they do not have to fill in the vehicle details every time they make appointment.

For car wash center cleaner, they are able to view all the scheduled appointment of the car wash center. They can also manage the status of the car wash center. For example, when they start the car wash center, they can change the appointment status from waiting status to cleaning in progress status. Next, they can also view all the appointment history of the car wash center. In addition, the cleaners are able to manage the car wash center's profile, service, vehicle type and cleaner. For the car wash center's profile, the cleaner can view and update the details of the profile. Cleaner can also view, add, update or delete the car wash center's service. Besides, cleaner are able to view, add or delete the vehicle type and cleaner.

1.7 Report Organization

This report has 6 chapters. Each of the chapter has different content and information. In Chapter 1, problem statement of this project is determined. The problem statement and scope is set according to the problem statement defined. The achievement achieved is also stated in this chapter.

In Chapter 2, literature review of similar application is listed. The strengths, weaknesses and also recommendations to the application is stated in this chapter. The applications are also compared and discussed. In addition, a few scheduling algorithm is also reviewed. Comparison between the scheduling algorithms is discussed in this chapter too.

In Chapter 3, it consists of system design and system overview. Framework of this project is explained here. Besides, use case description and activity diagram is listed and drawn. Explanation of code of the application development is stated in this chapter.

In Chapter 4, it includes methodology, tools and requirements used to complete this project. Timeline of this project is added in this chapter.

In Chapter 5, the implementation of application is listed here. Besides, the testing of the performance of scheduling algorithm is also explained in this chapter.

In Chapter 6, project review, novelties, contribution made in this project is stated here. The improvements and future work is also added in this chapter.

CHAPTER 2 LITERATURE REVIEW

2.1 Introduction

There is various car wash application launch in the market nowadays. Car wash application that have been reviewed is using appointment scheduling method to schedule customers' appointment. However, every application has their own strengths or weaknesses.

2.2 VIPBOX

2.2.1 Brief

VIPBOX is an application that provide car wash service to customers. VIPBOX application was launched in Google Play Store since 2019. This application allows customers to book for car wash service. (NEWPAGES.com.my) VIPBOX manage appointments by using scheduling method which provide convenient to customers.

2.2.2 Strengths

VIPBOX is using appointment scheduling method for customers to make appointment. Customers have the choice to choose the time slots when they are making an appointment. Thus, it actually saves customers' time as they do not have to wait for their turn to wash their car (Harvsey, 2019). Besides, appointment scheduling method also utilize manpower in car wash center. It will prevent crowded situation during peak hour.

Real-time scheduling. Appointment request that has been sent by customers can be confirm or cancel. Customers can manage their appointment online anytime and anywhere thus increase customer satisfaction. Real-time scheduling saves time as customers do not have to make calls for reservation and the reservation need not be processed (User, 2016). Real-time booking has overcome a problem which car wash service center agent might missed some customers when customers make call after business hour. Reservation of car wash service is available 24 hours every day.

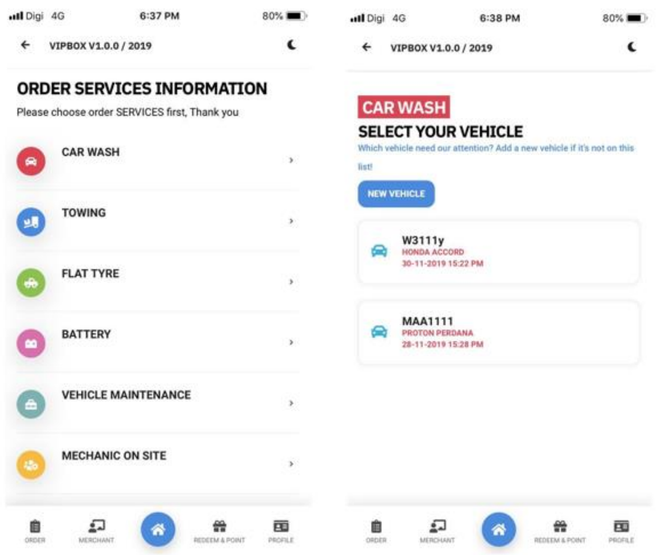


Figure 2-2-2-1: Screenshots of steps making an appointment (1)

(NEWPAGES.com.my)

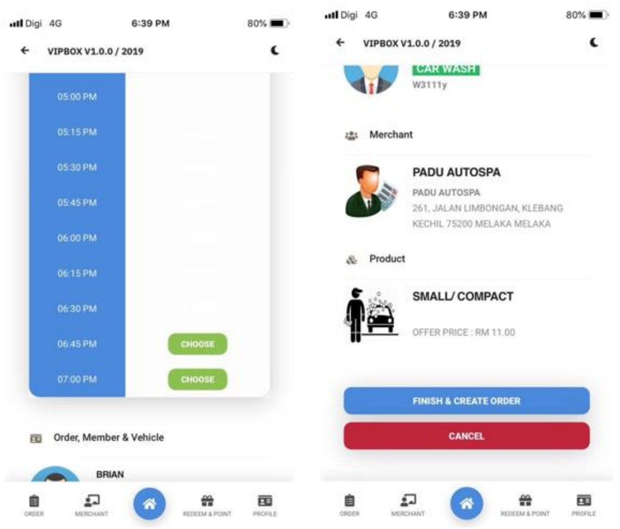


Figure 2-2-2-2: Screenshots of steps making an appointment (2)

(NEWPAGES.com.my)

Automatic message notification. Notification and confirmation email is send to the customers when they confirmed, rescheduled or cancelled an appointment. Besides, notification also displayed incoming appointment's information (McCabe, 2019). Customers are being remind of the appointment made hence the chances of customers cancel their appointment are reduced.

Account managing. Customers are able to manage their account after they registered for an account. They can save or edit their personal information such as name and email. Furthermore, customers can manage their car's details, payment information and their password. It is more convenient because when customers make an appointment, they can straight away select the details they added in their profile rather than typing in one by one.

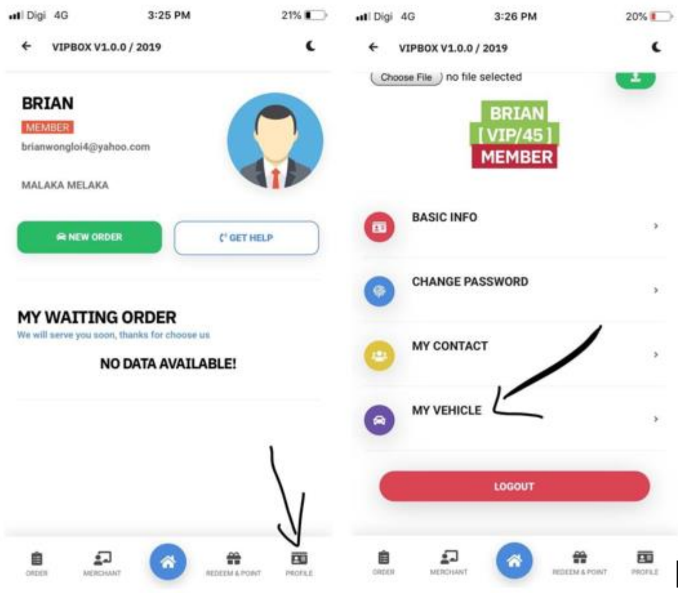


Figure 2-2-2-3: Screenshots of profile page of VIPBOX (NEWPAGES.com.my)

2.2.3 Weaknesses

GPS solution. It is hard for customers to search car wash center nearby. Customers has to filter car wash center by address one by one. Furthermore, some of the customers may not know the location of the car wash center thus it causes inconvenient to customers.

Online payment method. Customers have to pay cash instead of different online payment method. This causes inconvenience to customers as cashless society is now growing up and customers may not have that much cash on hand. Besides, store cash at car wash center increase risk for theft (Advantages and disadvantages of online payments, 2017).

2.2.4 Recommendations

GPS solution. GPS solution to show the direction to the car wash center selected. GPS solution may navigate customers who do not familiar with the location of the car wash center (Goodman, 2019). Besides, GPS solution can also show the car wash center that nearby customers' location. By using this feature, customers can immediately decide which car wash center they want. This feature is so customer oriented and hence result the increasing of the customer satisfaction (AF Staff, 2019).

Online payment method. VIPBOX should have providing different payment method for example online banking. Cashless transaction is on trending in the society nowadays. Online payment is convenient for customers to pay for the service as the transaction speed is fast (Advantages and disadvantages of online payments, 2017). Thus, feedback is provided straight away to customers and car wash center.

2.3 Spiffy

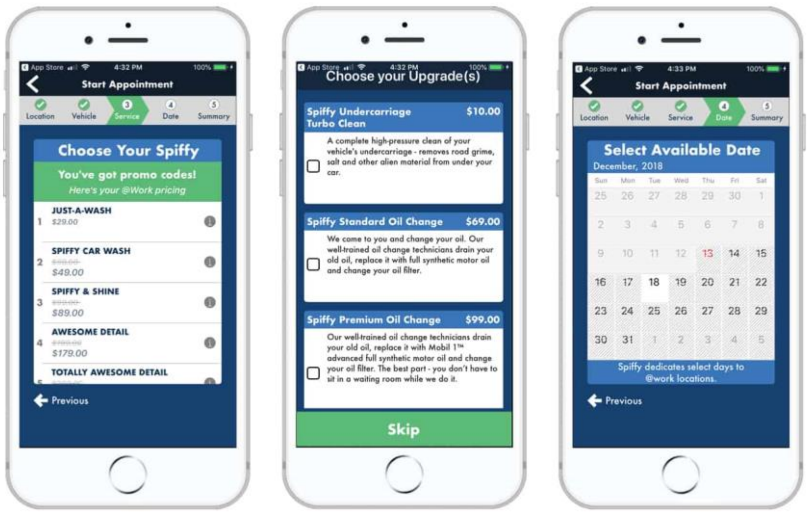
2.3.1 Brief

Spiffy is a mobile application that enables customers to schedule an appointment for car wash and car care service. Spiffy is only available in Canada and United States. Customers just have to make appointment and the technicians will come to them with tools. After perform the service, customers can rate their service using this mobile apps.

2.3.2 Strengths

Spiffy application is using scheduling method to manage customers' appointment. Scheduling method benefits the car wash center to set up task in order of priority. When the appointment is scheduled, it will prevent car wash center cleaner from too busy or procrastinating. Before that, customers have to queue up for car wash service. But now, customers can reserve a slot for car wash service, thus save a lot of time from waiting.

Real-time scheduling. After customers requested for the appointment, immediately car wash center agent received the request and process it. Customers have the flexibility to make appointment whether in home, office, or café at any time (The Biggest Benefits of Real-time Scheduling 2016). Besides, it also allows customers to alter their appointment if there are any changes.


App Store
4:32 PM
100%
Start Appointment
Location
Vehicle
Service
Date
Summary
Choose Your Spiffy
You've got promo codes!
Here's your @Work pricing
1 JUST-A-WASH $29.00
2 SPIFFY CAR WASH ~~$60.00~~ $49.00
3 SPIFFY & SHINE ~~$99.00~~ $89.00
4 AWESOME DETAIL ~~$199.00~~ $179.00
TOTALLY AWESOME DETAIL
Previous
Choose your Upgrade(s)
Spiffy Undercarriage Turbo Clean $10.00
Spiffy Standard Oil Change $69.00
Spiffy Premium Oil Change $99.00
Skip
4:33 PM
Select Available Date
December, 2018
Spiffy dedicates select days to @work locations.
Previous

A complete high-pressure clean of your vehicle's undercarriage - removes road grime, salt and other alien material from under your car.

We come to you and change your oil. Our well-trained oil change technicians drain your old oil, replace it with full synthetic motor oil and change your oil filter.

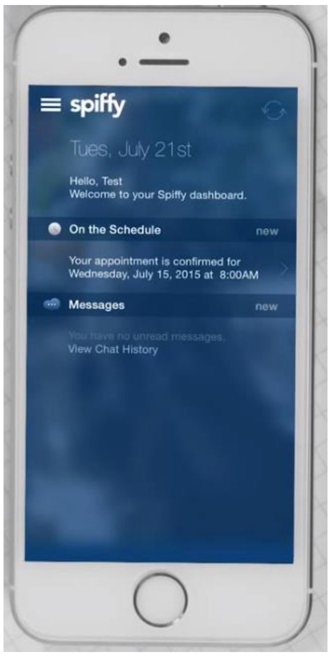
Our well-trained oil change technicians drain your old oil, replace it with Mobil 1™ advanced full synthetic motor oil and change your oil filter. The best part - you don't have to sit in a waiting room while we do it.

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Figure 2-3-2-1: Screenshots of steps making an appointment (Baxter, 2020)

Real-time chat. Spiffy provides functions like online chatting which enables communication between car wash agent and customers. It is efficient for customers and agents. Customer's satisfaction is related to agent's response time. Thus with online customer service, agents can serve a few customers at one time (Alina, 2019).

Automatic message notification. Automatic message notification is also provided by Spiffy. Notification is received when appointment is confirmed or cancelled, cleaner is on the way and when progress is done. Customers are satisfied as they can track the progress of the car wash. Notification will also help to remind the customers so that they will not forget about their appointment (Anastasia Z., 2018).



Tues, July 21st

Hello, Test
Welcome to your Spiffy dashboard.

On the Schedule new

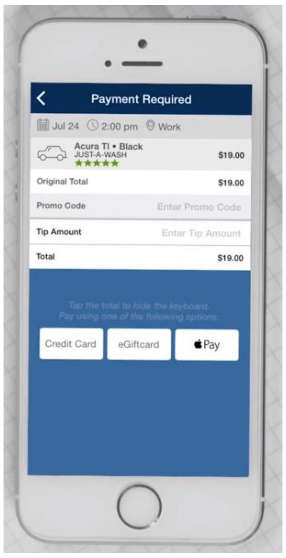
Your appointment is confirmed for
Wednesday, July 15, 2015 at 8:00AM

Messages new

You have no unread messages.
View Chat History

Figure 2-3-2-2: Screenshot of notification received after the appointment is confirmed

Online payment. Spiffy provides various online payment method as they accept eGift Cards, Apple Pay and credit cards. Customers can proceed to payment after their service is completed. Online payment has a lot of advantages as it keeps customers' and cleaners' safety and it is also convenient (Advantages and disadvantages of online payments, 2017).


Payment Required

Jul 24 2:00 pm Work

Acura TI • Black
JUST-A-WASH $19.00

Original Total	$19.00
Promo Code	Enter Promo Code
Tip Amount	Enter Tip Amount
Total	$19.00

Tap the total to hide the keyboard.
Pay using one of the following options.

Figure 2-3-2-3: Screenshot of various online payment method

2.3.3 Weaknesses

GPS route tracking solution. Car wash center will send their cleaner to the location set by the customers to provide car wash service. However, customers are not able to track the location of cleaner. It is more convenient when the customers can track their status and location so that they can be ready on the estimated arrival time.

2.3.4 Recommendations

GPS route tracking solution. GPS route tracking solution will track the location of the cleaner and customers can see route, idle time and estimated arrival time (Haq, 2016). Thus, it can provide better customer service.

2.4 BR CAR WASH

2.4.1 Brief

BR car wash is a car cleaning application which include many different functional applications developed by BR Softtech expertise. This application had used appointment scheduling method and allows customers to make appointment. BR car wash is offered in Android and iPhone platforms.

2.4.2 Strengths

BR car wash has make use of appointment scheduling method to schedule the appointment requested by customers. During the old days, car wash center hire cleaner to overcome the problem of insufficient manpower during peak hour. However by using appointment scheduling, manpower is utilized thus it save money from hiring cleaner.

GPS location tracking solution. Customers can view the location of car wash center which is near by their location in the map of GPS. This feature benefits customers which is unfamiliar with the area but they need a car wash service. Furthermore, it is easy to use as customers only need to select the car wash center by clicking on the GPS map.

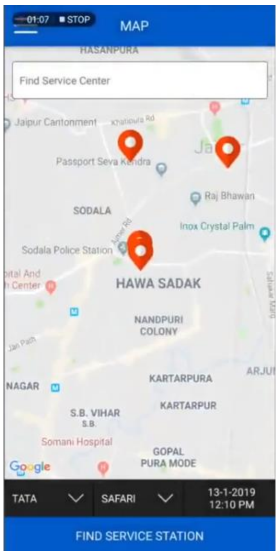


Figure 2-4-2-1: Screenshot of map that display the location of nearby car wash center

Real-time scheduling. Appointment can be made by customers by just entering the details for example date, time, location and etc. Appointment can be made online 24 hours every day. Thus, car wash center also no need to hire receptionist to manage the appointments. Besides, real-time scheduling is error free thus there will be no error while managing appointment (The Biggest Benefits of Real-time Scheduling 2016).

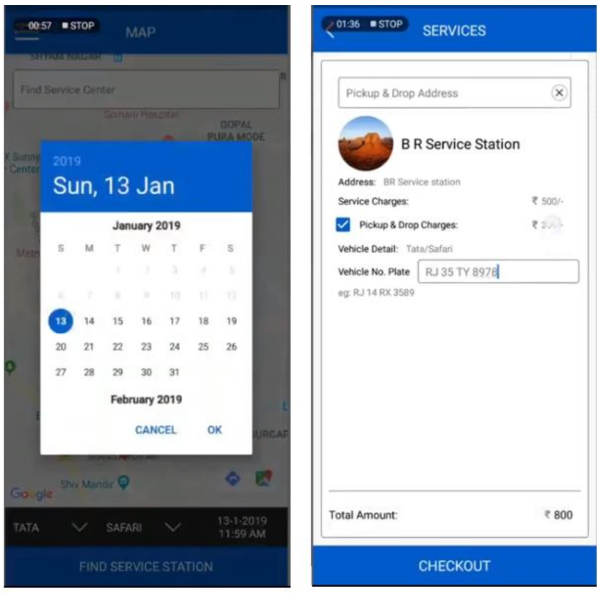
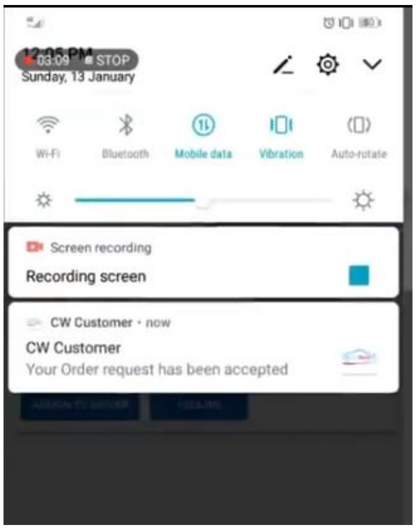


Figure 2-4-2-2: Screenshots of scheduling an appointment

Automatic message notification. Notification is sent to the customers when they confirmed or cancelled an appointment as conformation. Besides, it is important to update customers when every procedure is completed by cleaner (McCabe, 2019). Customer satisfaction increases when there are clear with every details of procedure and service.

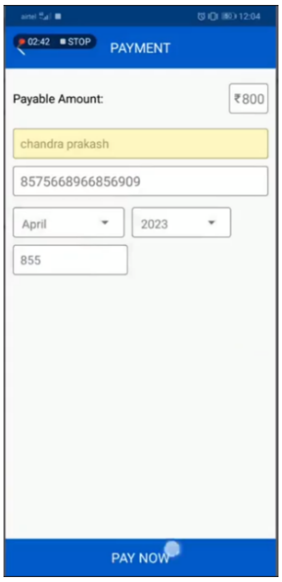


Screen recording
Recording screen

CW Customer • now
CW Customer
Your Order request has been accepted

Figure 2-4-2-3: Screenshot of notification received after order request is accepted

Online payment. BR CAR WASH provides online payment as they accept credit cards. Customers can pay with credit cards during the time when they make appointment. Cashless transaction is convenient to customers (Advantages and disadvantages of online payments, 2017) and company as the report is generated automatically when the transaction is completed. Besides, customers also no need to have so much money on their hand.


PAYMENT

Payable Amount: ₹800

chandra prakash

8575668966856909

April 2023

855

PAY NOW

Figure 2-4-2-4: Screenshot of online payment feature

2.4.3 Weaknesses

GPS route tracking solution. After customer’s appointment is confirmed, a driver is assigned to the customer for car wash service. Nevertheless, customers cannot track the location of driver. Customers can prepare and get ready before the arrival time if they can view their location and status.

2.4.4 Recommendations

GPS route tracking solution. The location of the driver will be tracked by using GPS route tracking solution so that customers can see route, idle time and estimated arrival time. Thus, it can provide better customer service (Haq, 2016).

2.5 WASHOS

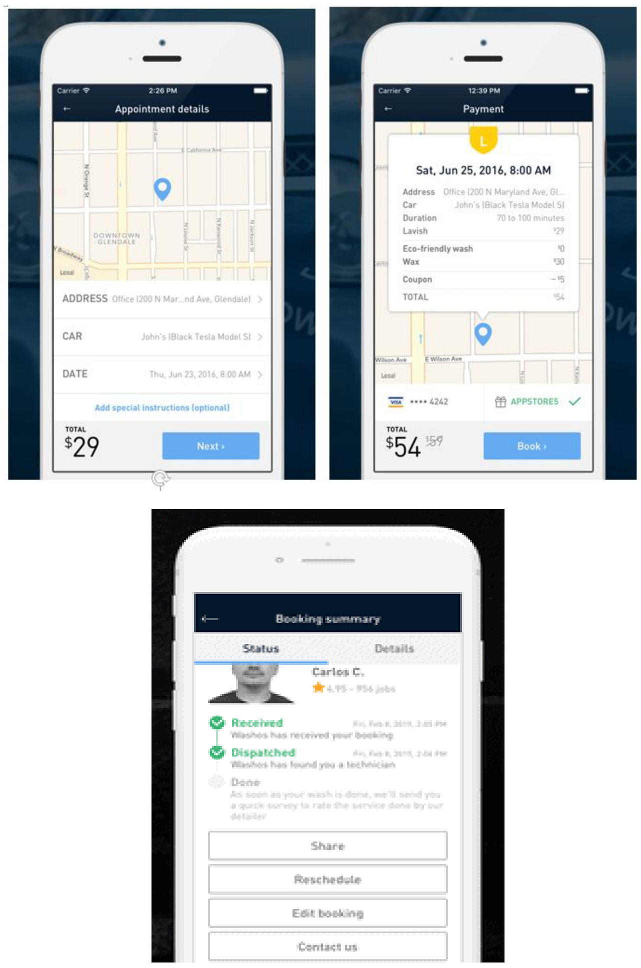
2.5.1 Brief

Washos is a mobile car wash application that using appointment scheduling method to provide customers car wash services after they had make their appointment. Washos delivers car services to customer’s doorstep and all the employees are committed to provide satisfying and quality service. Their car wash service areas are Los Angeles and Orange Country. Washos mobile car wash application has been launch officially in Apple App Store in 2015.

2.5.2 Strengths

Appointment scheduling method is applied by WASHOS in their application. By using appointment scheduling method, time-slot options are available to customers so that they can choose the slot that they preferred. Furthermore, appointment scheduling method may also increase the appointment and sales of the company.

Real-time scheduling. By using real-time scheduling, customers can make appointment 24/7 at anywhere (The Biggest Benefits of Real-time Scheduling 2016). Customers do not have to make phone calls or went to the car wash center to make reservations. Besides, it is more to customers self-service hence customers enquiries with be lesser. Making an appointment online is easy as customers just have to fill in all the details such as phone number, car plate, and locations.



Carrier 2:26 PM
Appointment details

ADDRESS Office (200 N Mar...nd Ave, Glendale)
CAR John's (Black Tesla Model S)
DATE Thu, Jun 23, 2016, 8:00 AM
Add special instructions (optional)
TOTAL $29 Next ›

Carrier 12:39 PM
Payment

Sat, Jun 25, 2016, 8:00 AM

Address	Office (200 N Maryland Ave, Gl...
Car	John's (Black Tesla Model S)
Duration	70 to 100 minutes
Lavish	$29
Eco-friendly wash	$0
Wax	$30
Coupon	− $5
TOTAL	$54

•••• 4242 APPSTORES ✓
TOTAL $54 ~~$59~~ Book ›

Booking summary
Status Details
Carlos C.
★ 4.95 – 956 jobs
Received — Washos has received your booking
Dispatched — Washos has found you a technician
Done — As soon as your wash is done, we'll send you a quick survey to rate the service done by our detailer
Share
Reschedule
Edit booking
Contact us

Figure 2-5-2-1: Screenshots of booking details (Mobile Car Wash and Detailing)

Automatic email notification. Email is sent to the customers for reminders automatically. It helps those customers who always forget things as they are busy doing works. Hence, the attendance of customers in appointments will increase. Furthermore, automatic email notification is user friendly as reminders are sent without disrupting customers' day (Anastasia Z., 2018).

Account managing. Customers are able to manage their account after they registered for an account. They can save or edit their personal information such as name and email. Furthermore, customers can manage their car's details, card's details and also edit their locations. It is more convenient because when customers make an appointment, they can straight away select the details they added in their profile rather than typing in one by one.

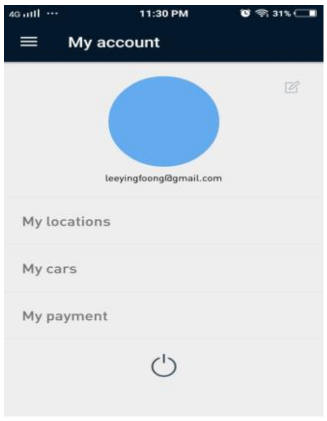


Figure 2-5-2-2: Screenshot of user's account

Online payment. WASHOS is a 100% cashless service which they do not accept cash during transaction. Cashless society is on trending now and more and more business are performing cashless transaction. WASHOS benefits customers with online payment as the payment can be automatic and the transaction speed is fast (Advantages and disadvantages of online payments, 2017). Besides, online payment lower the risk of theft.

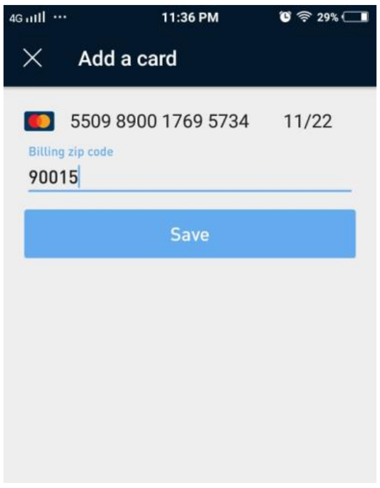


Figure 2-5-2-3: Screenshot of adding a card for payment

2.5.3 Weakness

GPS route tracking solution. Cleaner is assigned to the customers to clean their car after the appointment is successfully booked. Cleaner will go to the location provided by customers. Nevertheless, customers could not know the location of cleaner when they are on the way to the location provided. It benefits customers more if customers can check their route and status so that it saves customers' time as they can get prepare themselves on the estimated arrival time.

2.5.4 Recommendation

GPS route tracking solution. GPS route tracking solution will track the location and status of the cleaner. Thus, customers are able to view the route, idle time and estimated arrival time (Haq, 2016). Furthermore, GPS route tracking solution also secured cleaner's safety, assets and also operational (Haq, 2016).

2.6 Alphasphinx

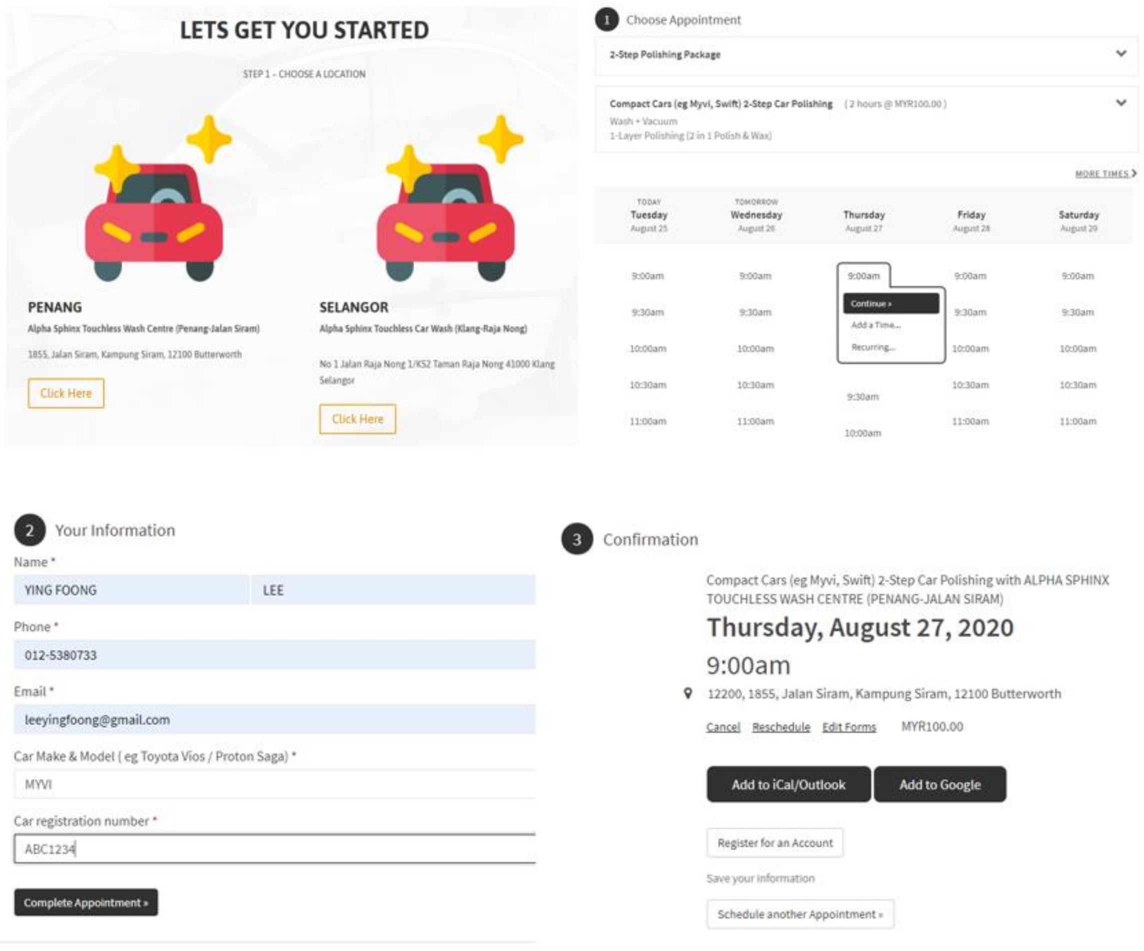
2.6.1 Brief

Alphasphinx is a website (https://alphasphinx.com/) owned by Alpha Sphinx (M) Sdn Bhd which provides excellent car wash services. Their service areas are Penang and Selangor, Malaysia. Appointment scheduling is available for customers to select their preferable time slot for car wash service in Alphasphinx. Their mission is to ensure their customers get the best services in vehicles washing and detailing industry to meet customers' satisfaction.

2.6.2 Strengths

In Alphasphinx website, they have provide booking appointment or reservation features by using appointment scheduling method. By using this method, it can keep every employee on the same page as the whole team will follow the scheduled appointment to provide services (PalmerScheduleSaturday, 2019). Hence, manpower is fully utilize.

Real-time scheduling. Alphasphinx provides online scheduling when customers make an appointment. Customers are able to edit or cancel appointment if they wish to make changes. Besides, customers can also reserve their preferable time slot for car wash service whenever they want. Furthermore, the company may also increase their production efficiencies and lower down their production cost. Company does not have to hire a clerk to answer the phone for reservation and customers can make appointment although it is not business hour (The Biggest Benefits of Real-time Scheduling 2016).


LETS GET YOU STARTED

STEP 1 - CHOOSE A LOCATION

PENANG
Alpha Sphinx Touchless Wash Centre (Penang-Jalan Siram)
1855, Jalan Siram, Kampung Siram, 12100 Butterworth

SELANGOR
Alpha Sphinx Touchless Car Wash (Klang-Raja Nong)
No 1 Jalan Raja Nong 1/KS2 Taman Raja Nong 41000 Klang Selangor

1 Choose Appointment

2-Step Polishing Package

Compact Cars (eg Myvi, Swift) 2-Step Car Polishing (2 hours @ MYR100.00)
Wash + Vacuum
1-Layer Polishing (2 in 1 Polish & Wax)

TODAY Tuesday August 25	TOMORROW Wednesday August 26	Thursday August 27	Friday August 28	Saturday August 29
9:00am	9:00am	9:00am	9:00am	9:00am
9:30am	9:30am	9:30am	9:30am	9:30am
10:00am	10:00am	10:00am	10:00am	10:00am
10:30am	10:30am		10:30am	10:30am
11:00am	11:00am		11:00am	11:00am

2 Your Information

Name *: YING FOONG LEE
Phone *: 012-5380733
Email *: leeyingfoong@gmail.com
Car Make & Model (eg Toyota Vios / Proton Saga) *: MYVI
Car registration number *: ABC1234

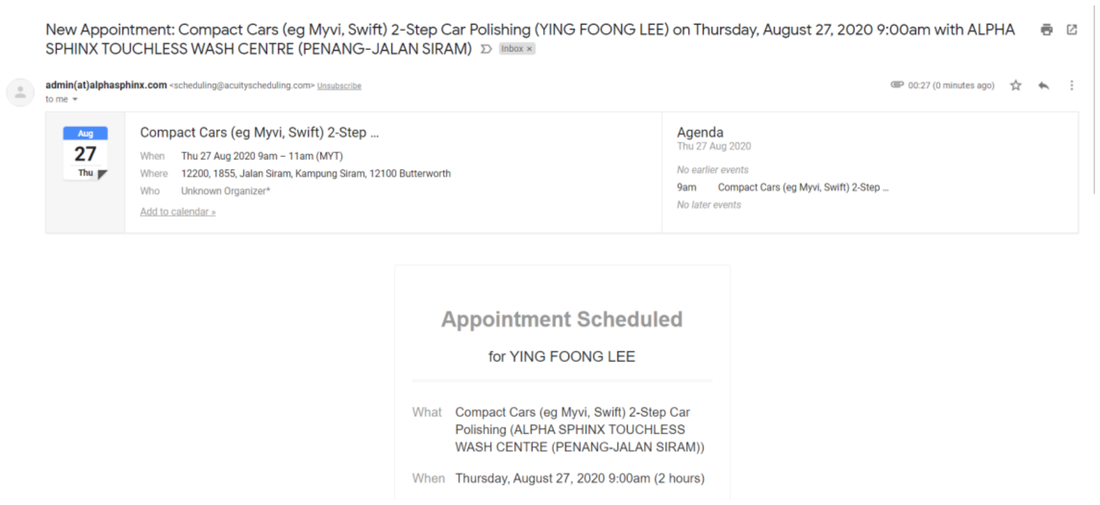
3 Confirmation

Compact Cars (eg Myvi, Swift) 2-Step Car Polishing with ALPHA SPHINX TOUCHLESS WASH CENTRE (PENANG-JALAN SIRAM)
Thursday, August 27, 2020
9:00am
12200, 1855, Jalan Siram, Kampung Siram, 12100 Butterworth
MYR100.00

Save your information

Figure 2-6-2-1: Screenshots of making an appointment

Automatic email notification. Email is sent when customers had successfully booked an appointment. It is a great way to acknowledge customers that the appointment is confirmed. Next, when customers edit the details of appointment or cancel the appointment, customers will also receive email as confirmation. In a nut shell, automatic email notification is important as customers can receive information and increase the efficiency (McCabe, 2019).


New Appointment: Compact Cars (eg Myvi, Swift) 2-Step Car Polishing (YING FOONG LEE) on Thursday, August 27, 2020 9:00am with ALPHA SPHINX TOUCHLESS WASH CENTRE (PENANG-JALAN SIRAM) Inbox ×

admin(at)alphasphinx.com <scheduling@acuityscheduling.com> Unsubscribe
to me

00:27 (0 minutes ago)

Aug 27 Thu

Compact Cars (eg Myvi, Swift) 2-Step ...

When Thu 27 Aug 2020 9am – 11am (MYT)
Where 12200, 1855, Jalan Siram, Kampung Siram, 12100 Butterworth
Who Unknown Organizer*
Add to calendar »

Agenda
Thu 27 Aug 2020
No earlier events
9am Compact Cars (eg Myvi, Swift) 2-Step ...
No later events

Appointment Scheduled
for YING FOONG LEE

What Compact Cars (eg Myvi, Swift) 2-Step Car Polishing (ALPHA SPHINX TOUCHLESS WASH CENTRE (PENANG-JALAN SIRAM))
When Thursday, August 27, 2020 9:00am (2 hours)

Figure 2-6-2-2: Screenshot of appointment confirmation email

2.6.3 Weakness

Online Payment. In Alphasphinx, there are no payment method available in their website. Customers have to pay at the car wash center. Cashless society is growing up nowadays and people now usually pay with e-wallet or credit card. Thus, it causes inconvenient to some customers as they prefer to pay online.

2.6.4 Recommendation

Online Payment. Alphasphinx should provide various payment method for example online banking, e-wallet, credit cards etc. as cashless transaction is on trending in the society. Customers are not necessary to have a lot of money on hand. Cashless transaction increases the safety and also decreases the crime rate of the society. When online transaction is done, the transaction details is recorded and it is easier for the company to review.

2.7 Comparison of Functions and Proposed Solution in table form

	VIPBOX	**Spiffy**	**BR CAR WASH**	**WASHOS**	**Alphasphinx**	**Proposed Solution**
Profile Management						
Registered account/ Sign in account with email	√	√	√	√		√
Registered account/ Sign in account with social media		√				
Edit profile	√	√	√	√		√
View transaction history	√					
View account balance						
Car Wash Center Feature						
View car wash center nearby			√			√
Search car wash center by name/ location	√					√
View car wash center's details	√	√	√	√	√	√
Google Map navigation			√		√	√

Table 2-7-1: Comparison of Functions and Proposed solution (1)

	VIPBOX	**Spiffy**	**BR CAR WASH**	**WASHOS**	**Alphasphinx**	**Proposed Solution**
Appointment Feature						
Make appointment	√	√	√	√	√	√
Cancel appointment	√	√	√	√	√	√
Alter appointment	√	√	√	√	√	
Preview appointment summary	√	√	√	√	√	√
View appointment progress			√	√		√
View appointment history	√	√	√	√		√
Vehicle feature						
Add vehicle	√	√	√	√		√
Delete vehicle	√	√	√	√		√
Edit vehicle	√	√	√	√		
View vehicle	√	√	√	√		√
Reward feature						
Gift cards		√		√		
Coupon				√		
Redeem point	√					

Table 2-7-2: Comparison of Functions and Proposed solution (2)

	VIPBOX	**Spiffy**	**BR CAR WASH**	**WASHOS**	**Alphasphinx**	**Proposed Solution**
Notification						
Appointment confirmation email		√		√	√	
Application notification			√			√
Payment Method						
Cash					√	√
E-payment	√	√	√	√		
E-wallet						
Additional Feature						
Feedback					√	
Help center	√	√	√	√		
Invite friends		√		√		
Frequently asked questions		√		√		

Table 2-7-3: Comparison of Functions and Proposed solution (3)

2.8 Critical Remarks

The following table is to benchmark the features in car wash applications that using appointment scheduling method.

Features	VIPBOX	Spiffy	BR Car Wash	WASHOS	Alphasphinx
Real-time scheduling	√	√	√	√	√
Automatic message notification	√	√	√	√	√
GPS location tracking solution			√		
Online payment method		√	√	√	
Real-time chat		√			
Account managing	√			√	

√: Yes   Blank: No

Table 2-8-1: Benchmark of the features in car wash applications

2.9 Scheduling Algorithm

Scheduling algorithm is an algorithm used to schedule different tasks in CPU. The scheduling algorithm will determine which tasks to be execute or which tasks to be hold. There is a few type of scheduling algorithm such as First Come First Serve, Shortest Time Remaining, Priority Scheduling and many more. Nowadays the implementation of scheduling algorithm is widespread and common in our lives. Scheduling algorithm not only solves problems in our daily lives, it also improves efficiency.

2.9.1 First Come First Serve (FCFS)

2.9.1.1 Brief

First Come First Serve (FCFS) scheduling algorithm is one of the simplest algorithm. FCFS schedules tasks according to the task's arrival time, where the first come task will be the first serve task. It is same as First In First Out (FIFO), which the task that added into the queue first will be the task that leave the queue first (FCFS scheduling Algorithm: What is, example program). In addition, FCFS is also a non-preemptive process. For example, in real-life situation, FCFS is applied on the ticket counter. Customer is served based on the queue. The first customer who arrives the queue will be the first one to be served and the first one to buy the ticket.

2.9.1.2 Advantages

FCFS is easy to understand. It is also simple to implement as it only includes simple logic. Moreover, starvation will not occur since the tasks are executed according to the arrival time.

2.9.1.3 Disadvantages

FCFS is a non-preemptive process, where the task will be executed to the completion then only the next task will be executed. Hence, the average waiting time of task is long. Task that has short execution time but arrive late will undergo long waiting time and result in Convoy effect which the whole process is delayed because of the slow processes (Difference between FCFS and SJF CPU scheduling algorithms 2020). Hence, FCFS causes lower CPU utilization.

2.9.2 Shortest Job First (SJF)

2.9.2.1 Brief

Shortest Job First (SJF) scheduling algorithm schedules tasks according to the execution time of the task. The task that has the smallest execution time will have highest priority to be execute next. Hence, it is a greedy algorithm as it always executes the shortest execution time task. SJF is non-preemptive while its preemptive version is called as Shortest Remaining Time First (SRTF).

2.9.2.2 Advantages

It has low average waiting time compared to First Come First Serve (FCFS). Besides, SJF increase the throughput as more tasks are executed in a short time. In short, SJF has high effectiveness and high utilization due to the low average waiting time (Techopedia, 2014).

2.9.2.3 Disadvantages

The disadvantages of implementing SJF is if there are many task with short execution time coming in, those task with longer execution time will undergo longer waiting time. Thus, it will lead to starvation where starvation is an unlimited postponement of a task due to the failure to allocate resource to a task. In addition, SJF is not suitable to be applied on a short term CPU scheduling.

2.9.3 Least Laxity First (LLF)

2.9.3.1 Brief

Least Laxity First (LLF) is also known as Least Slack Time scheduling algorithm. It is a real-time priority-driven scheduling algorithm. LLF algorithm schedules task according to the laxity time. Laxity time defined as the urgency of a task to be execute. It is the time left if the task was executed now. Thus, the task that has the least laxity time will have the highest priority to be execute next.

2.9.3.2 Advantages

Least Laxity First (LLF) is an optimal algorithm when preemption is allowed. LLF will come out with a feasible schedule if the feasible schedule exists. (ShivamKumar, 2020) Besides, LLF also able to predict which task is going to miss its deadline. Hence, it needs knowledge such as execution time and deadline of all the tasks.

2.9.3.3 Disadvantages

Least Laxity First (LLF) is a complex algorithm compared to other algorithm like First Come First Serve (FCFS). LLF required additional information such as execution time and deadlines. However, the execution time of the task in real-life is hard to predict.

2.10 Comparison of scheduling algorithm in table form

	First Come First Serve (FCFS)	Shortest Job First (SJF)	Least Laxity First (LLF)
Scheduled by	Task has shortest arrival time scheduled first in it	Task has shortest execution time scheduled first in it	Task has minimum laxity time scheduled first in it
Average waiting time	Long	Shortest	Short
Complexity	Simple	Moderate	Complex
Starvation	No	Yes, it may cause starvation	No
Predict which task is going to miss its deadline	No	No	Yes

Table 2-10-1: Comparison of scheduling algorithm in table form

2.11 Discussion

In Table 2-8-1, all of the car wash application reviewed provides two features which is real-time scheduling and automatic message notification. Both of these features are essential to be include in car applications that using appointment scheduling method according to the applications reviewed. Besides, there are also several applications provide online payment method. However, only one applications provide GPS route tracking solution which is BR Car Wash. Thus, GPS route tracking solution is not popular to be apply in car wash application currently.

The feature which is real-time scheduling will be included in this project. It provides advantages to customers as they are just a click away to make an appointment. Customers can book at anywhere, anytime they like. Furthermore, customers are able to select their preferable time slot for car wash service based on real-time basis. It is convenient and time saving as it is faster and more reliability when making an appointment.

Next, automatic notification is also sent to customers after they confirmed or cancelled the appointment. Customers are able to receive notification with the details of the appointment. Besides, on the day of the appointment, notification is also sent to the customers in order to remind them that they are having an appointment today. Automatic notification or reminders helps those customers which is too busy or forgetful. It prevents the no-show of customers and decrease the loss of the car wash company.

Last but not least, GPS location tracking is an important feature to be added in this project. Users can track the status of the car wash center in order to predict the arrival time. Besides, customers are able to browse the car wash center nearby their location and get to know their distances. GPS may also lead customers to the selected car wash center in case customers are not familiar with the location of car wash center.

In Table 2-10-1, it shows the comparison between 3 scheduling algorithm which is First Come First Serve (FCFS), Shortest Job First (SJF), Least Laxity First (LLF) scheduling algorithm. FCFS scheduling algorithm has the highest average waiting time among all the scheduling algorithm. Average waiting time is one of the most important feature to determine whether the algorithm is efficient or not. Long waiting time may decrease the level of customer satisfaction and lead to bad reputation of the car wash center. In worst case, car wash centers will lose their customer because of long waiting time.

Hence, it is important to have high efficiency and short waiting time for customers. Next, the situation which is starvation should be avoided. Starvation will cause some of the tasks not going to be executed or the tasks have to wait indefinitely although the tasks arrived early. Customer who made appointment at a particular time slot would wish their appointment finished on time. However, if starvation happens, the appointment will be delayed and missed the deadline. Hence, scheduling algorithm should prevent the happen of starvation.

CHAPTER 3 SYSTEM DESIGN

3.1 Overview

The use case diagram, use case diagram descriptions and activity diagram are done in this chapter to show the overview of the system. Besides, the development process is briefly explained through the code of the project.

3.2 System Design/ Overview

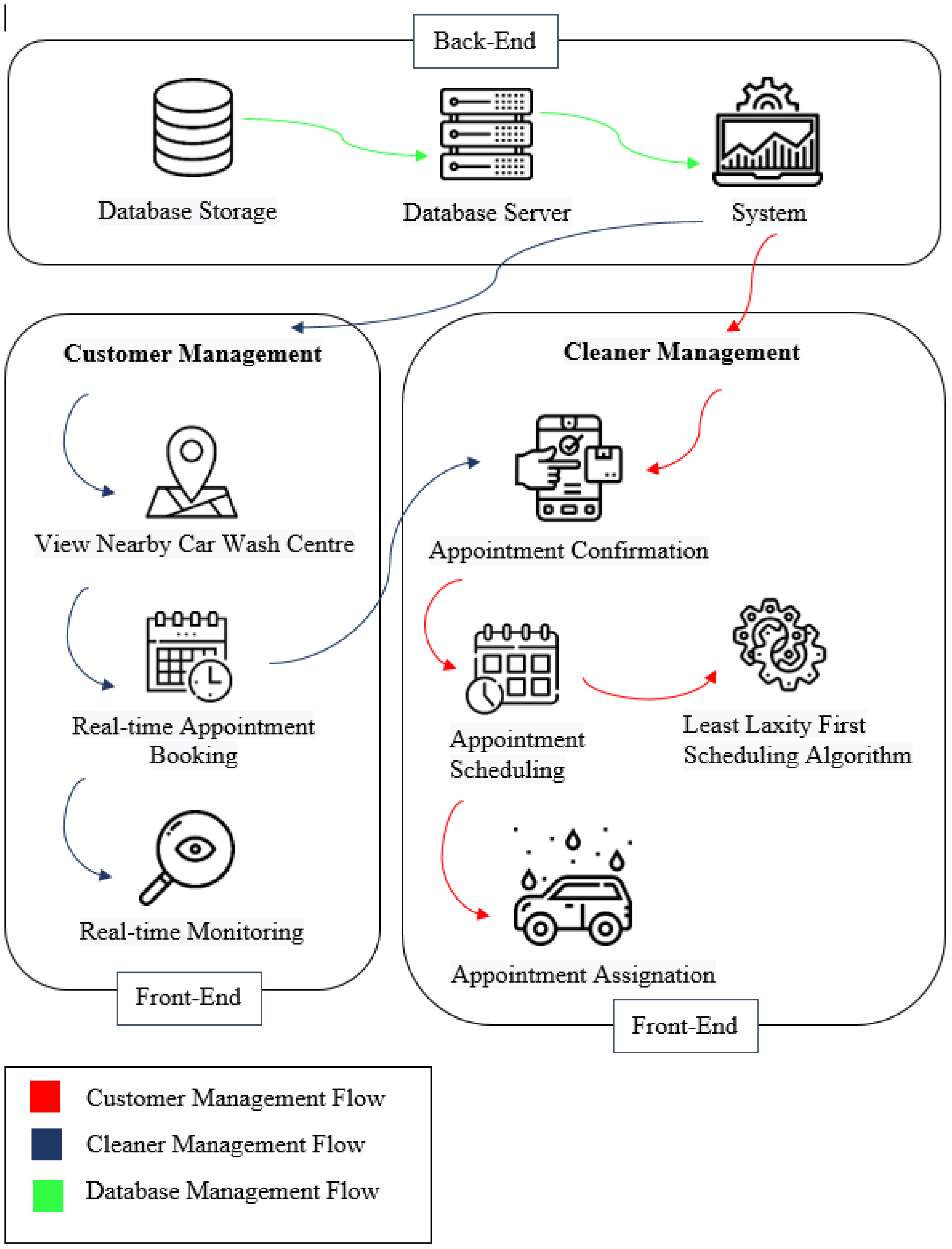


Figure 3-2-1: System Framework

The system is separated into front-end and back-end. Besides, there are 3 different flow in the system framework which is customer management flow, cleaner management flow and database management flow. Different flow is represented by different color of arrows.

3.2.1 Customers Management Flow

Customers management flow starts when users view nearby car wash centre on GPS. The location car wash centre that is nearby users' location is shown in the map. The users can make appointment on real-time basis. In addition, users can view the details and real-time status of the car wash centre when they clicked into the car wash centre's icon. When users sent their car to car wash centre, they also may monitor the cleaning progress of the car.

3.2.2 Cleaner Management Flow

Cleaner management flow starts when the appointment is booked by the users. The details of the appointment are received and the system will schedule the appointment made. Least Laxity First scheduling algorithm is used in this system to schedule all the appointments. The appointment is scheduled and then the appointment is assigned to the cleaner for car wash service.

3.2.3 Database Management Flow

Database management flow starts when users register an account in this application. The users must use their email to registered for an account and user's details filled in is stored in the database. The appointment made by users is also stored in the database so that the system can schedule appointment in real time. Car wash center's location is also stored in the database so that users can track nearby car wash center. Moreover, car wash center's details and the services provide is also stored in the database. In this system, the database server has provided service to customers and cleaners. Thus, all the data stored can be tracked.

3.3 Use Case

A use case represents a scenario which describe how users interact with the system to perform a task. This section has included use case diagram and use case description. Besides, the relationship between the user in achieving a goal is also described in the use case description. Use case has provided a guideline that how the system should work.

3.3.1 Use Case Diagram

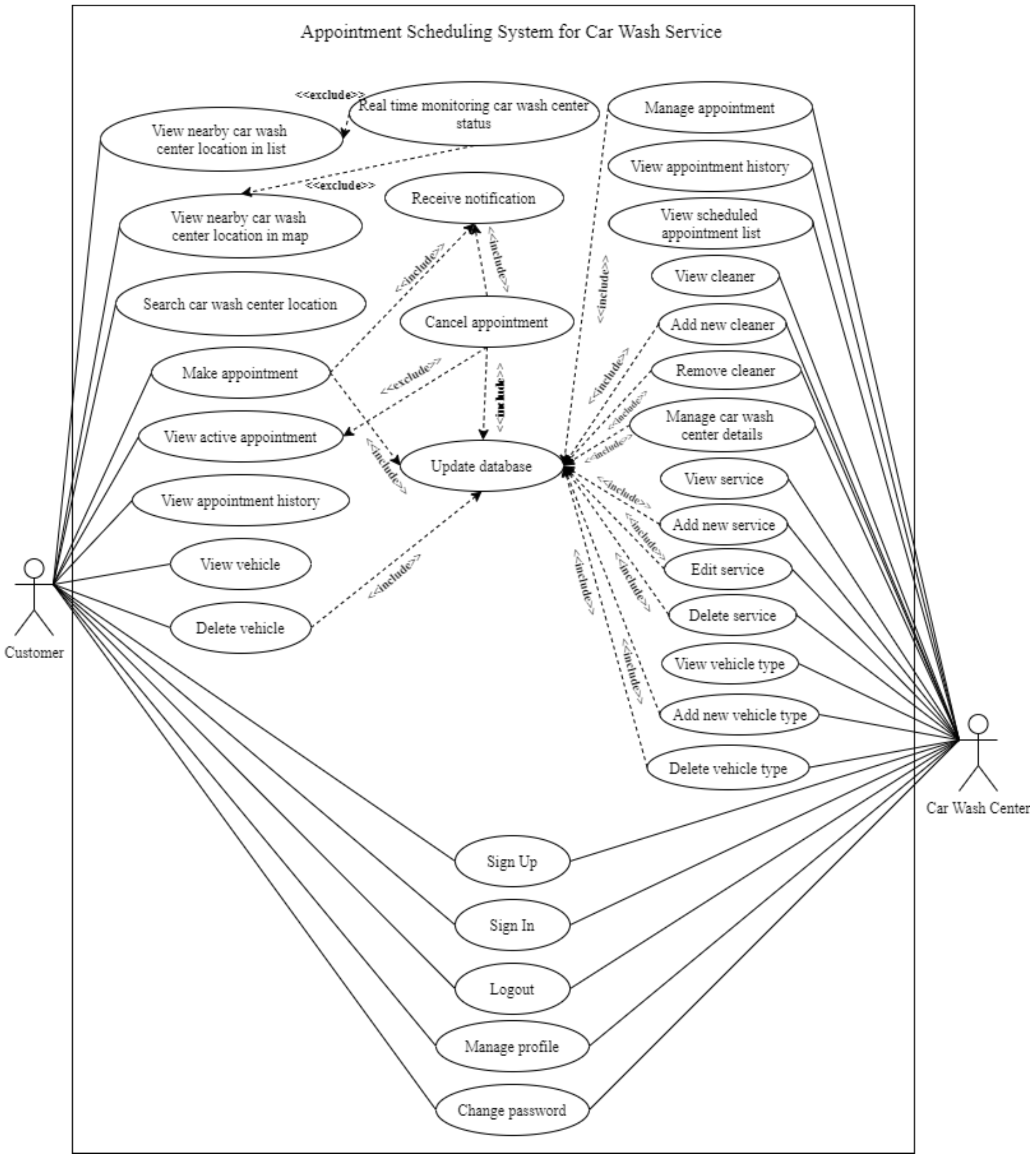


Figure 3-3-1-1: Use Case Diagram

Figure 3-3-1-1 above has shown the use case diagram of the appointment scheduling system for car wash service. There are two type of users in this use case diagram which is customer and cleaner of car wash center. Customer and cleaner are able to sign up, sign in, sign out an account in this application. They also able to manage their profile and edit their profile details such as username, phone number and manage their password.

Next, customer is able to view nearby car wash center's location or search for the location of car wash center. They are also allowed to monitor the status of the car wash center in real-time. Customer can make appointment for car wash service. Customers can also view active appointment and their real-time progress on the home page. Appointment can be cancel within an hour after the appointment was made. Notification will be sent to customer when they made an appointment or cancelled an appointment successfully. All the changes made to the appointment will be updated in the database. They can also check their appointment history to see their previous appointment. In addition, customers can view their vehicles in a list. They can add or delete vehicle themselves.

For cleaner of car wash center, they are able to view all the appointments' details and also the scheduled appointment list. Cleaner are able to manage the appointment such as they can update the status of the appointment when they start cleaning process or they had already finished the cleaning process. In addition, cleaner is allowed to manage the car wash center's details. Any changes made in the car wash center details is updated in the database. Cleaner is able to add new cleaner or remove cleaner. Next, they are also allowed to add new service to the car wash center, edit the service details and delete the service. Last but not least, they are able to manage the vehicle type such as add new vehicle type and delete vehicle type.

3.3.2 Use Case Description

3.3.2.1 Sign Up

Use Case ID	U001	
Use Case Name	Sign Up	
Purpose	To allow user to sign up for an account in the application	
Actors	Customer, Cleaner	
Trigger	User clicks "sign up" button	
Precondition	1. Application is at sign up page 2. User has internet connection	
Normal Flow	Step	Action
	1	User enters email, password and confirm password
	2	User clicks "sign up" button
	3	System authenticates the user's identity
	4	Database is updated if user does not exists in database
	5	System direct user to create profile page
	6	User enters profile information
	7	User clicks "complete" button
	8	System direct user to home page
Alternative Flow	1.1 1.2	If email, password or confirm password is not complete, the system will display message to inform user. If password and confirm password is not same, the system will display message to inform user.
	2.1	If user clicks sign in button, the system will direct user to sign in page.
	4.1	If email have been used before, the system will back to normal flow step 1.
	6.1	If profile information is incomplete, the system will display message to inform user.

Table 3-3-2-1-1: Sign Up Use Case Description

3.3.2.2 Sign In

Use Case ID	U002	
Use Case Name	Sign In	
Purpose	To allow user to sign in to the application	
Actors	Customer, Cleaner	
Trigger	User clicks "sign in" button in sign in page	
Precondition	1. User is at sign in page 2. User has internet connection	
Normal Flow	Step	Action
	1	User enters email and password
	2	User clicks "sign in" button
	3	System authenticates the user's identity
	4	User is signed in to the application and directed to home page if the user exists in database
Alternative Flow	1.1 1.2 1.3	If the information needed is not complete, the system will display message to inform user. If the email is incorrect, the system will display message to inform user. If the password is incorrect, the system will display message to inform user.
	2.1	If user clicks sign up button, the system will direct user to sign up page.
	3.1	If user is not exist, the system will direct user to sign up page

Table 3-3-2-2-1: Sign In Use Case Description

3.3.2.3 Manage Profile

Use Case ID	U003	
Use Case Name	Manage Profile	
Purpose	To allow user to manage their profile in the application	
Actors	Customer, Cleaner	
Trigger	User clicks on profile feature in the navigation bar in home page	
Precondition	1. User is signed in to the system 2. User is in home page 3. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on profile feature in the navigation bar in home page
	2	System extracts the profile of current user from database
	3	System display profile to user
	4	User edit their profile information
	5	User clicks “save” button
	6	User’s input is retrieved and system will update the database
	7	System display message to inform user that profile update successfully
Alternative Flow	5.1 5.2	If the fields required is not completed, the system will display message to inform user. If user clicks “change password” button, the system will direct user to manage password page.

Table 3-3-2-3-1: Manage Profile Use Case Description

3.3.2.4 Manage Password

Use Case ID	U004	
Use Case Name	Manage Password	
Purpose	To allow user to manage their password in the application	
Actors	Customer, Cleaner	
Trigger	User clicks on "change password" button	
Precondition	1. User is signed in to the system 2. User is in manage password page 3. User has internet connection	
Normal Flow	Step	Action
	1	User enters current password, new password and confirm new password
	2	User clicks "change" button
	3	User's input is retrieved and system will update the database
	4	System display message to inform user that password change successfully
Alternative Flow	2.1 2.2 2.3	If fields required is not completed, the system will display message to inform user. If current password is incorrect, the system will display message to inform user. If new password and confirm new password is not the same, the system will display message to inform user.

Table 3-3-2-4-1: Manage Password Use Case Description

3.3.2.5 View Nearby Car Wash Center's Location in Map

Use Case ID	U005	
Use Case Name	View Nearby Car Wash Center's Location in Map	
Purpose	To allow user to view nearby car wash center's location in map	
Actors	Customer	
Trigger	User clicks on "book now" button in home page	
Precondition	1. User is signed in to the system 2. User is at home page 3. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on "book now" button in home page
	2	System ask for permission to access user's location
	3	User clicks "allow" button
	4	System get current location from user
	5	System extracts car wash center details which its location is nearby user's location from database
	6	System will put a marker on the location of car wash center if there are any car wash center nearby
Alternative Flow	4.1 5.1	If system failed to get current location, the system will display message to inform user and direct user back to home page If there is no car wash center nearby, the system will display message to inform user.

Table 3-3-2-5-1: View Nearby Car Wash Center's Location in Map Use Case Description

3.3.2.6 View Nearby Car Wash Center's Location in List

Use Case ID	U006	
Use Case Name	View Nearby Car Wash Center's Location in List	
Purpose	To allow user to view nearby car wash center's location in list	
Actors	Customer	
Trigger	User clicks on "book now" button in home page	
Precondition	1. User is signed in to the system 2. User is at home page 3. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on "book now" button in home page
	2	System ask for permission to access user's location
	3	User clicks "allow" button
	4	System get current location from user
	5	System extracts car wash center details which its location is nearby user's location from database
	6	User clicks "view list" button
	7	System direct user to view nearby car wash center in list page
	8	System display nearby car wash center in list
Alternative Flow	4.1 5.1	If system failed to get current location, the system will display message to inform user and direct them back to home page. If there is no car wash center nearby, the system will display message to inform user.

Table 3-3-2-6-1: View Nearby Car Wash Center's Location in List Use Case Description

3.3.2.7 Search Car Wash Center’s Location

Use Case ID	U007	
Use Case Name	Search Car Wash Center’s Location	
Purpose	To allow user to search for car wash center’s location	
Actors	Customer	
Trigger	User clicks on search bar in view nearby car wash center in map page	
Precondition	1. User is signed in to the system 2. User has internet connection 3. User allows system to access to the device’s location	
Normal Flow	Step	Action
	1	User clicks on search bar in view nearby car wash center in map page
	2	System direct user to search car wash center page
	3	User can search car wash center by name
	4	System extracts car wash center which match with the input from database
	5	System extracts car wash center’s details which match with the input from database and put in a list
	6	User clicks on the car wash center
	7	System will direct user to a map and put a pin icon on the location of car wash center
Alternative Flow	5.1	If there is no car wash center found, the system will display message to inform user

Table 3-3-2-7-1: Search Car Wash Center’s Location Use Case Description

3.3.2.8 Real-time Monitoring Car Wash Center Status

Use Case ID	U008	
Use Case Name	Real-time Monitoring Car Wash Center Status	
Purpose	To allow user to monitor car wash center's status in real-time	
Actors	Customer	
Trigger	User clicks on car wash center's location marker in Google Map	
Precondition	1. User is signed in to the system as Customer 2. User is at Google Map page 3. User has internet connection 4. User allows system to access to the device's location	
Normal Flow	Step	Action
	1	User clicks on car wash center's location marker in Google Map
	2	System extracts the real-time status details of the selected car wash center from database
	3	System will pop out window displaying the details of car wash center
Alternative Flow	N/A	

Table 3-4-2-8-1: Real-time Monitoring Car Wash Center Status Use Case Description

3.3.2.9 Make appointment

Use Case ID	U009	
Use Case Name	Make Appointment	
Purpose	To allow user to make appointment for car wash service	
Actors	Customer	
Trigger	User clicks on "book" button	
Precondition	1. User is signed in to the system 2. User has internet connection 3. User is at home page	
Normal Flow	Step	Action

	1	User clicks on “book now” button
	2	User clicks on the car wash center’s marker pinned on the map
	3	System pop out a window
	4	User clicks on “book” button
	5	System direct user to select vehicle type page
	6	User clicks on preferable vehicle type
	7	System direct user to select vehicle page
	8	User selects preferable vehicle
	9	User click “continue” button
	10	System direct user to select service page
	11	User clicks on preferable service of the car wash center
	12	System direct user to service details page displaying details of the selected service
	13	User clicks “continue” button
	14	System direct user to select timeslot page
	15	User selects preferable timeslot
	16	User clicks “continue” button
	17	System direct user to appointment overview page
	18	User clicks “confirm” button
	19	System updates database
Alternative Flow	2.1 8.1	If user clicks on “view list” button, the system will direct user to view nearby car wash center list page. Then, it will back to normal flow step 4. If user clicks on “add vehicle” button, alert dialog will pop out. User enters required information and clicks “ok” button. If required fields is not complete, the system will display message to inform user.

Table 3-3-2-9-1: Make Appointment Use Case Description

3.3.2.10 Receive Notification

Use Case ID	U010	
Use Case Name	Receive Notification	
Purpose	To notify user when user make appointment and cancel appointment	
Actors	Customer	
Trigger	User clicks on "confirm" button in appointment overview page or user clicks on "cancel" button in appointment details page.	
Precondition	1. User is signed in to the system 2. User is at appointment overview page or appointment details page 3. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on "confirm" button in appointment overview page
	2	System sends notification to user
Alternative Flow	1.1	If user clicks on "cancel" button in appointment details page, go to normal flow step 2.

Table 3-3-2-10-1: Receive Notification Use Case Description

3.3.2.11 View Active Appointment

Use Case ID	U011	
Use Case Name	View Active Appointment	
Purpose	To allow user to view active appointment and its real-time progress	
Actors	Customer	
Trigger	User at home page	
Precondition	1. User is signed in to the system 2. User has internet connection 3. User is at home page	
Normal Flow	Step	Action
	1	User is at home page.
	2	System retrieve active appointment of user from database
	3	System display appointment at home page
	4	User clicks on the appointment
	5	System directs user to appointment details page
Alternative Flow	N/A	

Table 3-3-2-11-1: View Active Appointment Use Case Description

3.3.2.12 Cancel Appointment

Use Case ID	U012	
Use Case Name	Cancel Appointment	
Purpose	To allow user to cancel appointment	
Actors	Customer	
Trigger	User clicks on “cancel” button in appointment details page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User is at appointment details page

	2	User clicks on “cancel” button in appointment details page
	3	System will pop out alert dialog
	4	User clicks “yes” button
	5	System will validate the request and update the database
Alternative Flow	3.1	If user clicks “no”, then go back to normal flow step 1.

Table 3-3-2-12-1: Cancel Appointment Use Case Description

3.3.2.13 View Appointment History

Use Case ID	U013	
Use Case Name	View Appointment History	
Purpose	To allow user to view appointment history	
Actors	Customer	
Trigger	User clicks on history feature in the navigation bar in home page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on history feature in the navigation bar in home page
	2	System directs user to appointment history page
	3	System extracts all the appointments made by user from database
	4	System display all appointment in list
	5	User clicks on an appointment in appointment list
	6	System direct user to appointment details page
Alternative Flow	3.1	If no appointment is found, the system will display message to inform user.

Table 3-3-2-13-1: View Appointment History Use Case Description

3.3.2.14 View Notification

Use Case ID	U014	
Use Case Name	View Notification	
Purpose	To allow user to view all notification	
Actors	Customer	
Trigger	User clicks on notification feature in the navigation bar in home page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on notification feature in the navigation bar in home page
	2	System directs user to notification page
	3	System extracts all the notification received by user from database
	4	System display all notification in list
	5	User clicks on an notification in notification list
	6	System direct user to notification details page
Alternative Flow	3.1	If no notification is found, the system will display message to inform user.

Table 3-3-2-14-1: View Notification Use Case Description

3.3.2.15 View Vehicle

Use Case ID	U015
Use Case Name	View Vehicle
Purpose	To allow user to view all their vehicle
Actors	Customer
Trigger	User clicks on vehicle feature in the navigation bar in home page
Precondition	1. User is signed in to the system 2. User has internet connection

Normal Flow	Step	Action
	1	User clicks on notification feature in the navigation bar in home page
	2	System direct user to vehicle page
	3	System extracts all the user's vehicle from database
	4	System display all vehicle in list
Alternative Flow	3.1	If no notification is found, the system will display message to inform user.

Table 3-3-2-15-1: View Vehicle Use Case Description

3.3.2.16 Add Vehicle

Use Case ID	U016	
Use Case Name	Add Vehicle	
Purpose	To allow user to add their vehicle	
Actors	Customer	
Trigger	User clicks "add vehicle" button in user's vehicle page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks "add vehicle" button in user's vehicle page
	2	System pop out an alert dialog
	3	User enters required information and clicks "ok"
	4	System retrieve user input and updates the database
Alternative Flow	3.1 3.2	If required information is not complete, the system will display message to inform user. If user clicks "cancel", the system will close the alert dialog

Table 3-3-2-16-1: Add Vehicle Use Case Description

3.3.2.17 Delete Vehicle

Use Case ID	U017	
Use Case Name	Delete Vehicle	
Purpose	To allow user to delete their vehicle	
Actors	Customer	
Trigger	User clicks delete icon in user's vehicle page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks delete icon in user's vehicle page
	2	System pop out an alert dialog
	3	User clicks "ok"
	4	System delete the vehicle from the database
Alternative Flow	3.1	If user clicks "cancel", the system will close the alert dialog

Table 3-3-2-17-1: Delete Vehicle Use Case Description

3.3.2.18 View Scheduled Appointment List

Use Case ID	U018	
Use Case Name	View Scheduled Appointment List	
Purpose	To allow user to view all scheduled appointment list	
Actors	Cleaner	
Trigger	User clicks on washing slot button in cleaner's home page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User is at cleaner's home page
	2	System display all washing slot in list
	3	User selects and clicks on one washing slot
	4	System display scheduled appointment list of the selected washing slot

Alternative Flow	4.1	If there is no appointment found, the system will display message to inform user.

Table 3-3-2-18-1: View Scheduled Appointment List Use Case Description

3.3.2.19 Manage Appointment

Use Case ID	U019	
Use Case Name	Manage Appointment	
Purpose	To allow user to manage the appointments	
Actors	Cleaner	
Trigger	User clicks on buttons in appointment queue page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on “Start” button which indicates the car wash service is started
	2	System update the status of appointment in database
	3	User clicks “finish” button which indicates the car wash service is finished
	4	System update the status of appointment in database
Alternative Flow	1.1	If user clicks on “finish” button when the service is haven started yet, the system will display message to inform user.

Table 3-3-2-19-1: Manage Appointment Use Case Description

3.3.2.20 View Car Wash Center Details

Use Case ID	U020	
Use Case Name	View Car Wash Center Details	
Purpose	To allow user to view car wash center details	
Actors	Cleaner	
Trigger	User clicks on car wash center feature in the navigation bar in cleaner’s home page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on car wash center feature in the navigation bar in home page
	2	System directs user to car wash center details page
	3	System extracts car wash center details from database
	4	System display car wash center information
Alternative Flow	N/A	

Table 3-3-2-20-1: View Car Wash Center Details Use Case Description

3.3.2.21 Update Car Wash Center Details

Use Case ID	U021	
Use Case Name	Update Car Wash Center Details	
Purpose	To allow user to update car wash center details	
Actors	Cleaner	
Trigger	User clicks on “update” button at car wash center details page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on car wash center feature in the navigation bar in home page
	2	System directs user to car wash center details page

	3	System extracts car wash center details from database
	4	System display car wash center information
	5	User update or edit the details
	6	User clicks “update” button
	7	System retrieve input and update the database
Alternative Flow	5.1	If the inputs are not completed, the system will display message to inform user

Table 3-3-2-21-1: Update Car Wash Center Details Use Case Description

3.3.2.22 View Car Wash Center’s Service

Use Case ID	U022	
Use Case Name	View Car Wash Center’s Service	
Purpose	To allow user to view car wash center’s service	
Actors	Cleaner	
Trigger	User clicks on the “service” icon on the bottom navigation bar at car wash center details page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on car wash center feature in the navigation bar in home page
	2	User clicks on the “service” icon on the bottom navigation bar at car wash center details page
	3	System extracts car wash center’s service from database
	4	System display all car wash center’s service
Alternative Flow	3.1	If there is no service found, the system will display message to inform user.

Table 3-3-2-22-1: View Car Wash Center’s Service Use Case Description

3.3.2.23 Add Car Wash Center’s Service

Use Case ID	U023	
Use Case Name	Add Car Wash Center’s Service	
Purpose	To allow user to add car wash center’s service	
Actors	Cleaner	
Trigger	User clicks “add” button in car wash center’s service page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks “add” button in car wash center’s service page
	2	System direct user to add car wash center service page
	3	User inputs all required information
	4	System retrieve user input and updates the database
Alternative Flow	3.1	If required information is not complete, the system will display message to inform user.

Table 3-3-2-23-1: Add Car Wash Center’s Service Use Case Description

3.3.2.24 Update Car Wash Center’s Service

Use Case ID	U024	
Use Case Name	Update Car Wash Center’s Service	
Purpose	To allow user to update car wash center’s service	
Actors	Cleaner	
Trigger	User clicks “update” button in car wash center’s service details page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User is at car wash center’s service details page
	2	User update or edit service details

	3	User clicks “update” button in car wash center’s service details page
	4	System retrieve user input and updates the database
Alternative Flow	2.1	If required information is not complete, the system will display message to inform user.

Table 3-3-2-24-1: Update Car Wash Center’s Service Use Case Description

3.3.2.25 Delete Car Wash Center’s Service

Use Case ID	U025	
Use Case Name	Delete Car Wash Center’s Service	
Purpose	To allow user to delete car wash center’s service	
Actors	Cleaner	
Trigger	User clicks “delete” button in car wash center’s service details page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User is at car wash center’s service details page
	2	User clicks “delete” button in car wash center’s service details page
	3	System deletes and update the database
Alternative Flow	N/A	

Table 3-3-2-25-1: Delete Car Wash Center’s Service Use Case Description

3.3.2.26 View Car Wash Center’s Cleaner

Use Case ID	U026	
Use Case Name	View Car Wash Center’s Cleaner	
Purpose	To allow user to view car wash center’s cleaner	
Actors	Cleaner	
Trigger	User clicks on the “cleaner” icon on the bottom navigation bar at car wash center details page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on car wash center feature in the navigation bar in home page
	2	User clicks on the “cleaner” icon on the bottom navigation bar at car wash center details page
	3	System extracts car wash center’s cleaner from database
	4	System display all car wash center’s cleaner
Alternative Flow	3.1	If there is no cleaner found, the system will display message to inform user.

Table 3-3-2-26-1: View Car Wash Center’s Cleaner Use Case Description

3.3.2.27 Add Car Wash Center’s Cleaner

Use Case ID	U027	
Use Case Name	Add Car Wash Center’s Cleaner	
Purpose	To allow user to add car wash center’s cleaner	
Actors	Cleaner	
Trigger	User clicks “add cleaner” button in car wash center’s cleaner page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action

	1	User clicks “add cleaner” button in car wash center’s cleaner page
	2	System direct user to add car wash center cleaner page
	3	User inputs all required information
	4	User clicks “Add” button
	5	System retrieve user input and updates the database
Alternative Flow	3.1	If required information is not complete, the system will display message to inform user.

Table 3-3-2-27-1: Add Car Wash Center’s Cleaner Use Case Description

3.3.2.28 Delete Car Wash Center’s Cleaner

Use Case ID	U028	
Use Case Name	Delete Car Wash Center’s Cleaner	
Purpose	To allow user to delete car wash center’s cleaner	
Actors	Cleaner	
Trigger	User clicks “delete” icon in car wash center’s cleaner page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User is at car wash center’s cleaner page
	2	User clicks “delete” icon in car wash center’s cleaner page
	3	System deletes cleaner and update the database
Alternative Flow	N/A	

Table 3-3-2-28-1: Delete Car Wash Center’s Cleaner Use Case Description

3.3.2.29 View Car Wash Center’s Vehicle Type

Use Case ID	U029	
Use Case Name	View Car Wash Center’s Vehicle Type	
Purpose	To allow user to view car wash center’s vehicle type	
Actors	Cleaner	
Trigger	User clicks on the “vehicle” icon on the bottom navigation bar at car wash center details page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on car wash center feature in the navigation bar in home page
	2	User clicks on the “vehicle” icon on the bottom navigation bar at car wash center details page
	3	System extracts car wash center’s vehicle type from database
	4	System display all car wash center’s vehicle type
Alternative Flow	3.1	If there is no vehicle type found, the system will display message to inform user.

Table 3-3-2-29-1: View Car Wash Center’s Vehicle Type Use Case Description

3.3.2.30 Add Car Wash Center’s Vehicle Type

Use Case ID	U030	
Use Case Name	Add Car Wash Center’s Vehicle Type	
Purpose	To allow user to add car wash center’s vehicle type	
Actors	Cleaner	
Trigger	User clicks “add vehicle type” button in car wash center’s vehicle type page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action

	1	User clicks "add vehicle type" button in car wash center's vehicle type page
	2	System pop out an alert dialog
	3	User inputs all required information
	4	User clicks "ok" button in alert dialog
	5	System retrieve user input and updates the database
Alternative Flow	3.1 4.1	If required information is not complete, the system will display message to inform user. If user clicks "cancel" button, then the alert dialog will be closed.

Table 3-3-2-30-1: Add Car Wash Center's Vehicle Type Use Case Description

3.3.2.31 Delete Car Wash Center's Vehicle Type

Use Case ID	U031	
Use Case Name	Delete Car Wash Center's Vehicle Type	
Purpose	To allow user to delete car wash center's vehicle type	
Actors	Cleaner	
Trigger	User long clicks the vehicle type in car wash center's vehicle type page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User long clicks the vehicle type in car wash center's vehicle type page
	2	System pop out an alert dialog
	3	User clicks "ok" button in alert dialog
	4	System delete vehicle type and updates the database
Alternative Flow	3.1	If user clicks "cancel" button, the alert dialog will be closed.

Table 3-3-2-31-1: Delete Car Wash Center's Vehicle Type Use Case Description

3.3.2.32 View Appointment History

Use Case ID	U032	
Use Case Name	View Appointment History	
Purpose	To allow user to view appointment history	
Actors	Cleaner	
Trigger	User clicks on history feature in the navigation bar in cleaner's home page	
Precondition	1. User is signed in to the system 2. User has internet connection	
Normal Flow	Step	Action
	1	User clicks on history feature in the navigation bar in cleaner's home page
	2	System directs user to appointment history page
	3	System extracts all the car wash center's appointments from database
	4	System display all appointment in list
	5	User clicks on an appointment in appointment list
	6	System direct user to appointment details page
Alternative Flow	3.1	If no appointment is found, the system will display message to inform user.

Table 3-3-2-32-1: View Appointment History Use Case Description

3.4 Activity Diagram

3.4.1 Sign Up

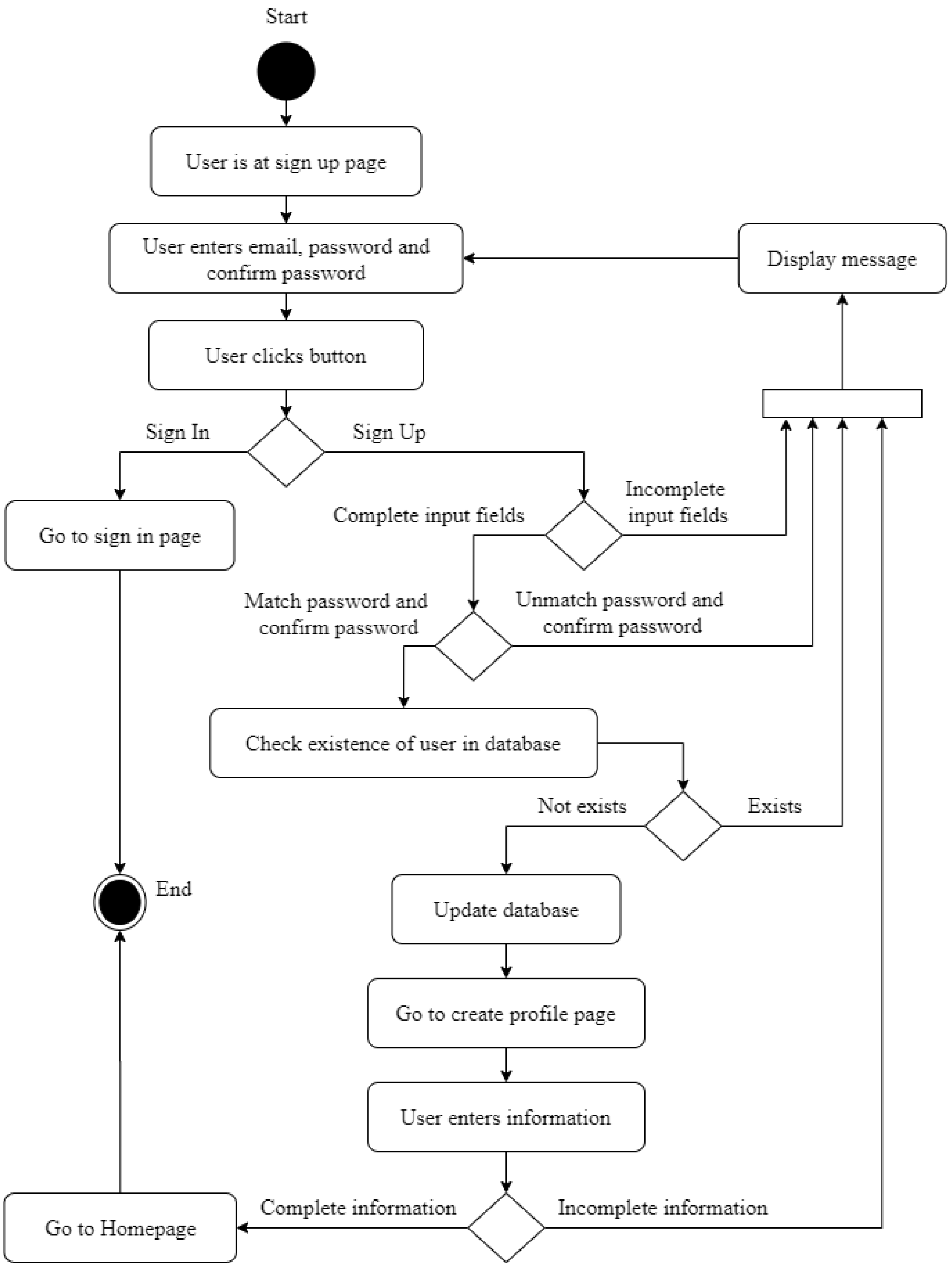


Figure 3-4-1-1: Sign Up Activity Diagram

3.4.2 Sign In

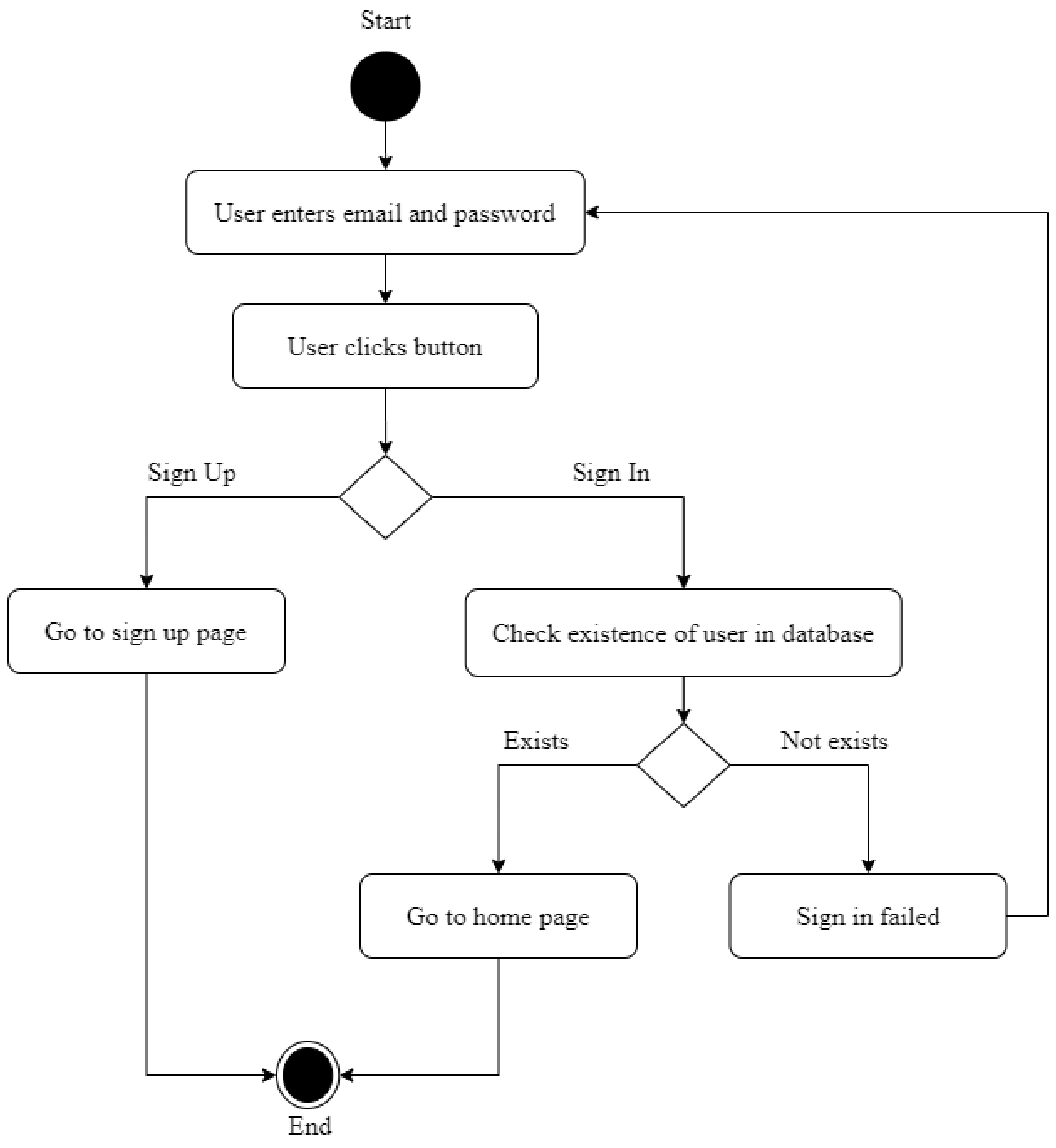


Figure 3-4-2-1: Sign In Activity Diagram

3.4.3 Manage Profile

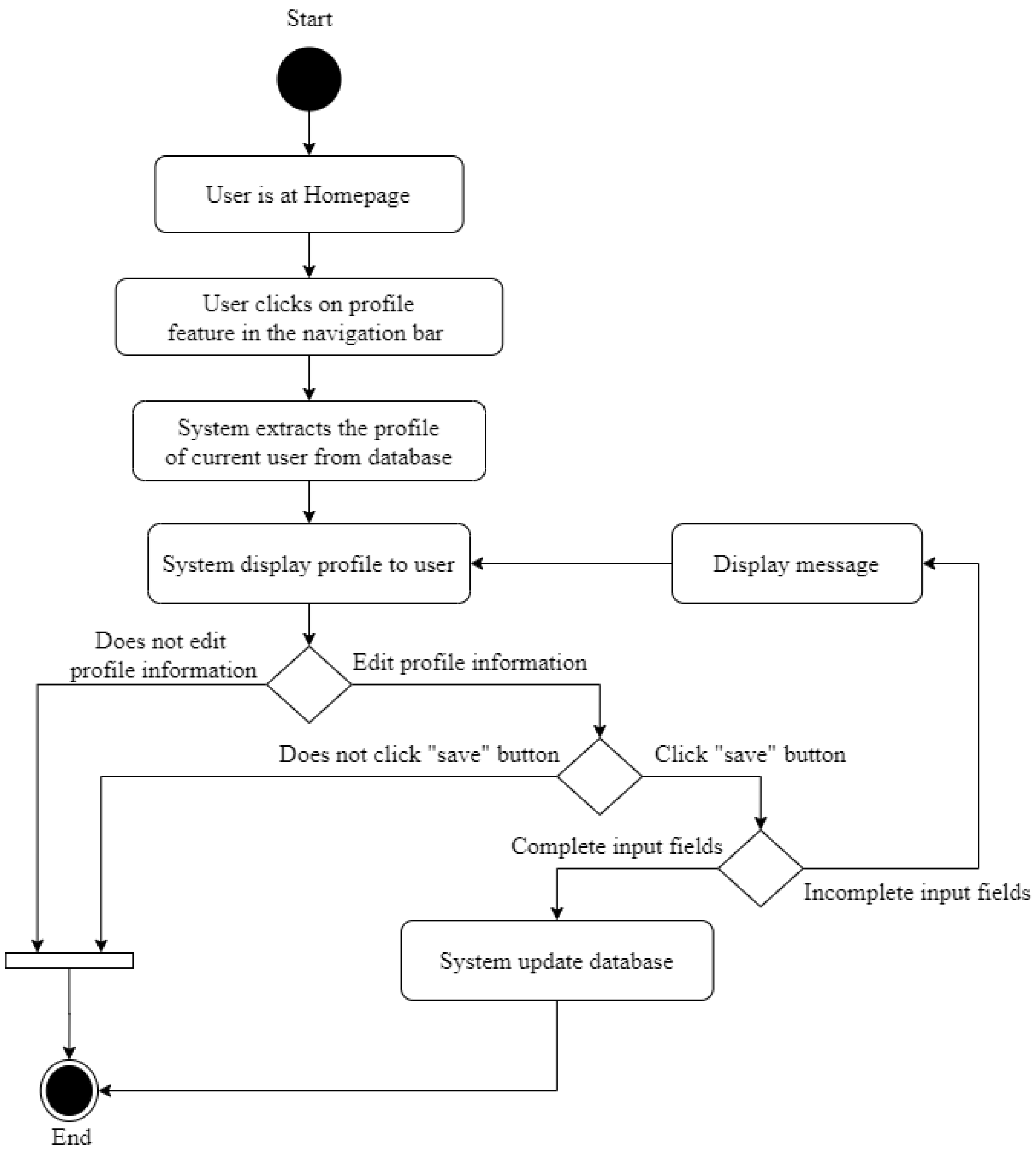

Start
User is at Homepage
User clicks on profile feature in the navigation bar
System extracts the profile of current user from database
System display profile to user
Display message
Does not edit profile information
Edit profile information
Does not click "save" button
Click "save" button
Complete input fields
Incomplete input fields
System update database
End

Figure 3-4-3-1: Manage Profile Activity Diagram

3.4.4 Manage Password

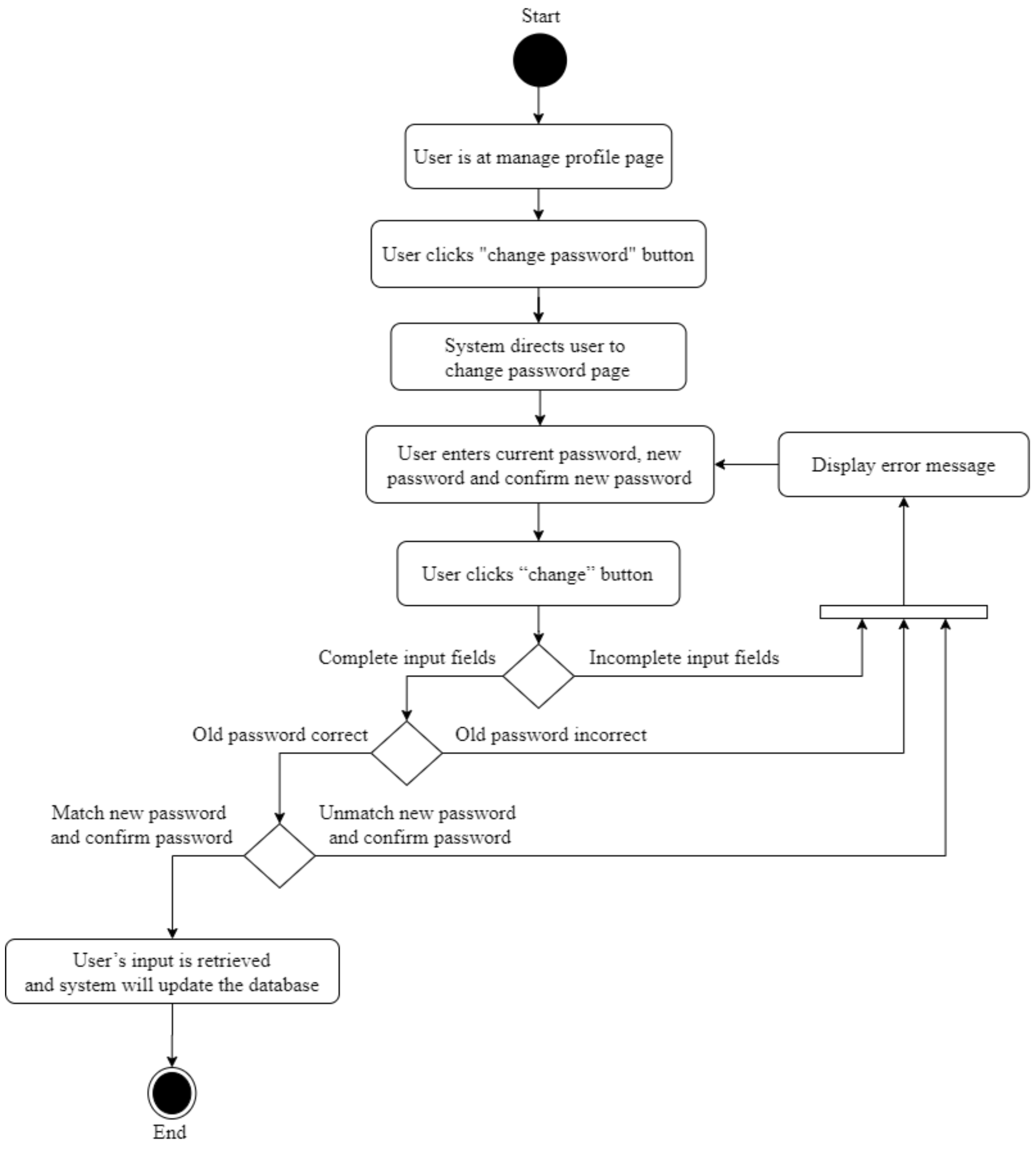

Start
User is at manage profile page
User clicks "change password" button
System directs user to change password page
User enters current password, new password and confirm new password
Display error message
User clicks "change" button
Complete input fields
Incomplete input fields
Old password correct
Old password incorrect
Match new password and confirm password
Unmatch new password and confirm password
User's input is retrieved and system will update the database
End

Figure 3-4-4-1: Manage Password Activity Diagram

3.4.5 View Nearby Car Wash Center’s Location in Map

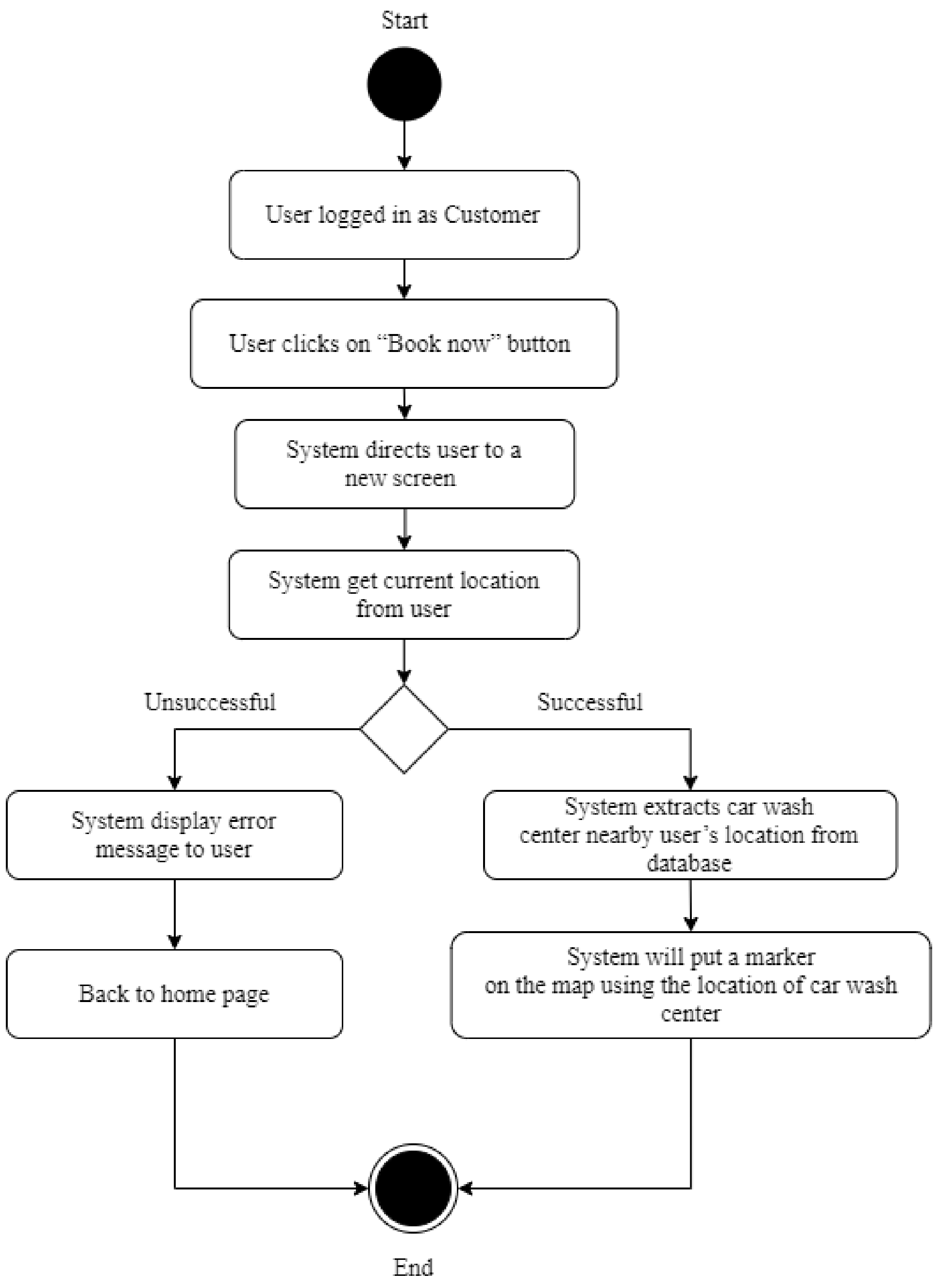

Start
User logged in as Customer
User clicks on “Book now” button
System directs user to a new screen
System get current location from user
Unsuccessful
Successful
System display error message to user
System extracts car wash center nearby user’s location from database
Back to home page
System will put a marker on the map using the location of car wash center
End

Figure 3-4-5-1: View Nearby Car Wash Center’s Location in Map Activity Diagram

3.4.6 View Nearby Car Wash Center's Location in List

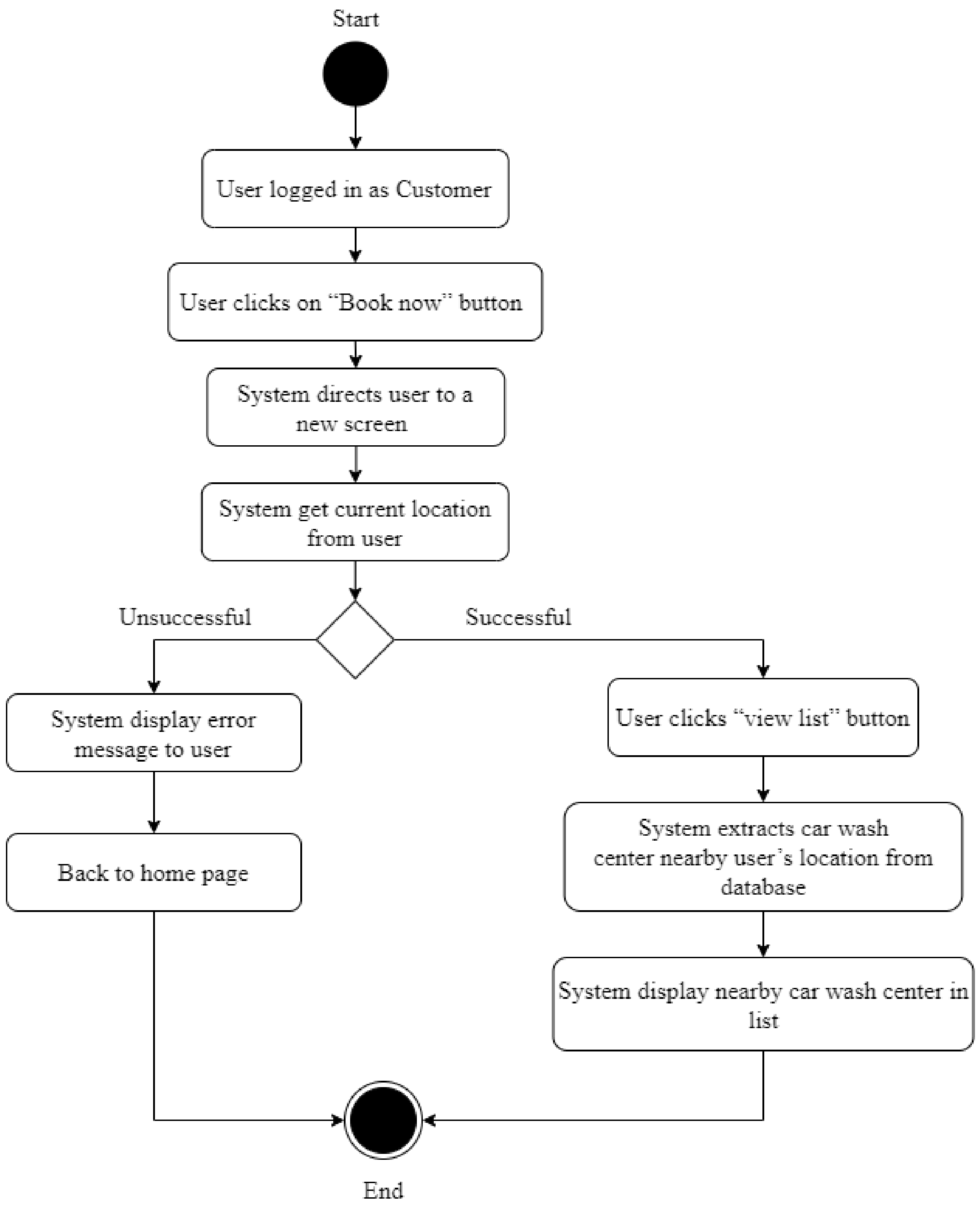

Start
User logged in as Customer
User clicks on "Book now" button
System directs user to a new screen
System get current location from user
Unsuccessful
Successful
System display error message to user
Back to home page
User clicks "view list" button
System extracts car wash center nearby user's location from database
System display nearby car wash center in list
End

Figure 3-4-6-1: View Nearby Car Wash Center's Location in List Activity Diagram

3.4.7 Search Car Wash Center's Location

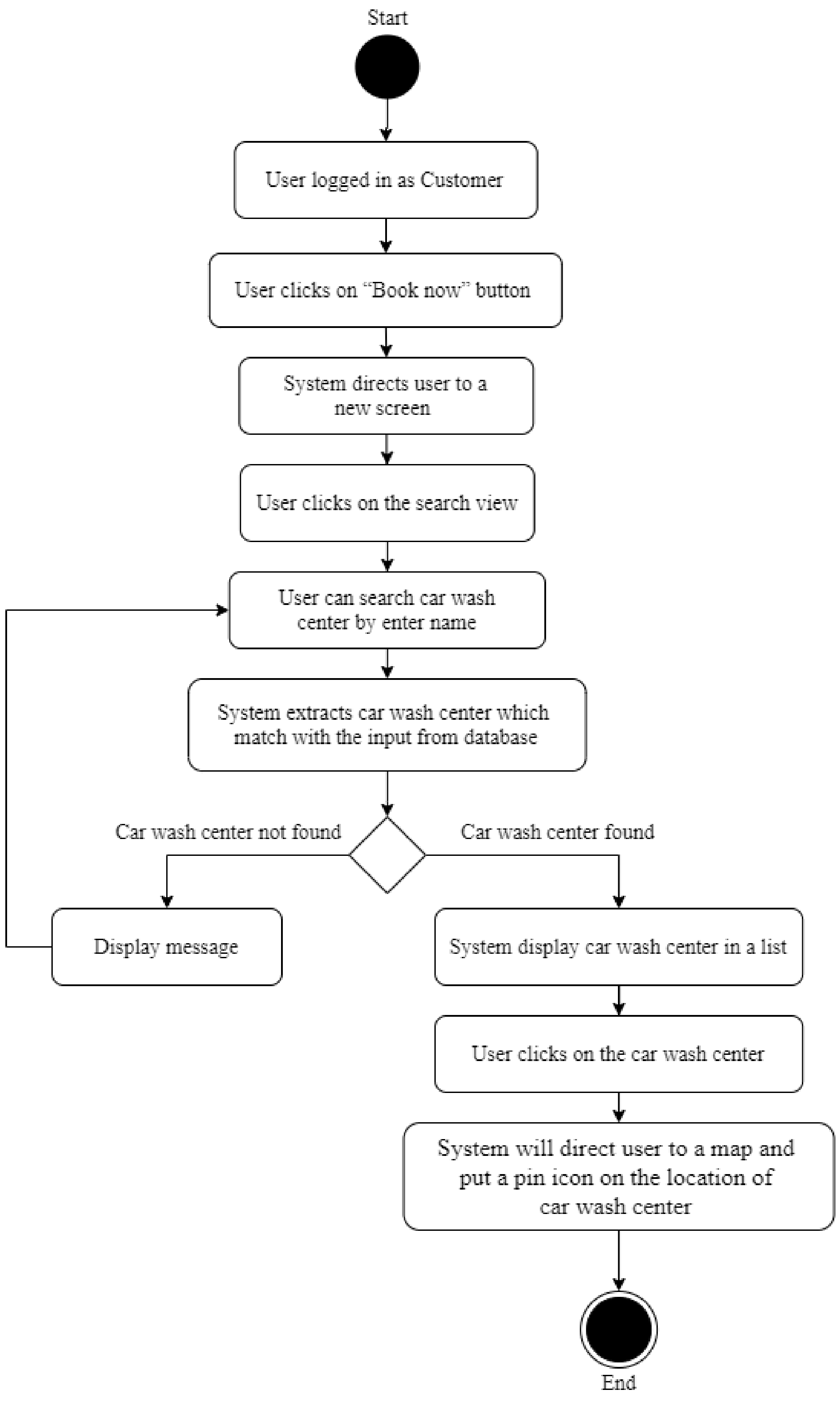

Start
User logged in as Customer
User clicks on "Book now" button
System directs user to a new screen
User clicks on the search view
User can search car wash center by enter name
System extracts car wash center which match with the input from database
Car wash center not found
Car wash center found
Display message
System display car wash center in a list
User clicks on the car wash center
System will direct user to a map and put a pin icon on the location of car wash center
End

Figure 3-4-7-1: Search Car Wash Center's Location Activity Diagram

3.4.8 Real-time Monitoring Car Wash Center Status

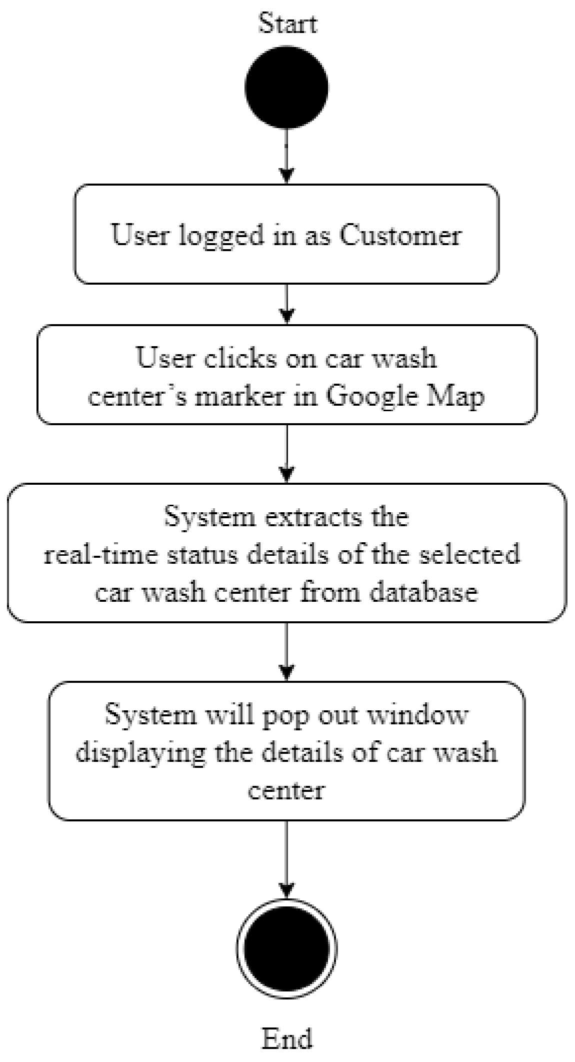

Start
User logged in as Customer
User clicks on car wash center's marker in Google Map
System extracts the real-time status details of the selected car wash center from database
System will pop out window displaying the details of car wash center
End

Figure 3-4-8-1: Real-time Monitoring Car Wash Center Status Activity Diagram

3.4.9 Make appointment

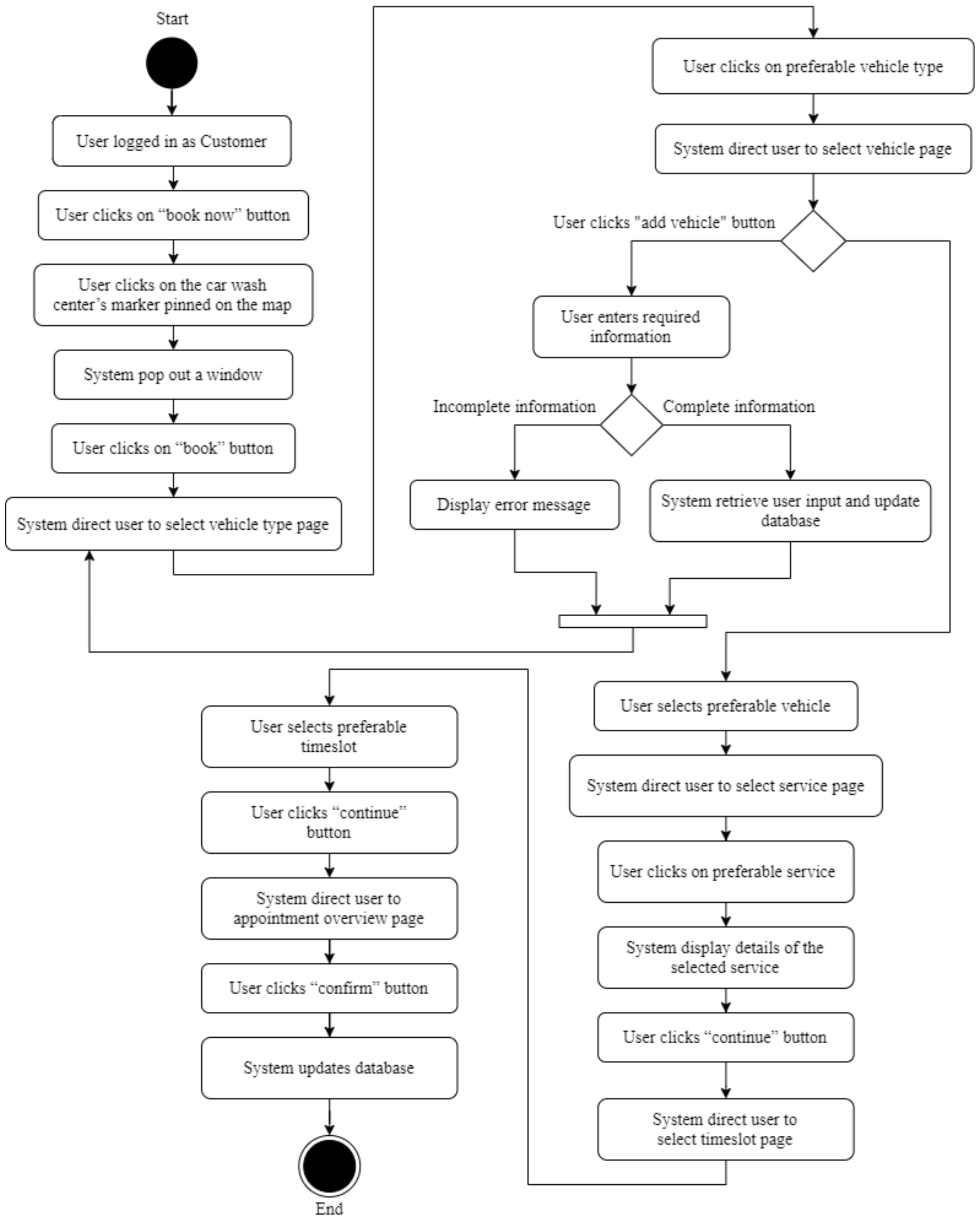

Start
User logged in as Customer
User clicks on "book now" button
User clicks on the car wash center's marker pinned on the map
System pop out a window
User clicks on "book" button
System direct user to select vehicle type page
User clicks on preferable vehicle type
System direct user to select vehicle page
User clicks "add vehicle" button
User enters required information
Incomplete information
Complete information
Display error message
System retrieve user input and update database
User selects preferable vehicle
System direct user to select service page
User clicks on preferable service
System display details of the selected service
User clicks "continue" button
System direct user to select timeslot page
User selects preferable timeslot
User clicks "continue" button
System direct user to appointment overview page
User clicks "confirm" button
System updates database
End

Figure 3-4-9-1: Make appointment Activity Diagram

3.4.10 Receive Notification

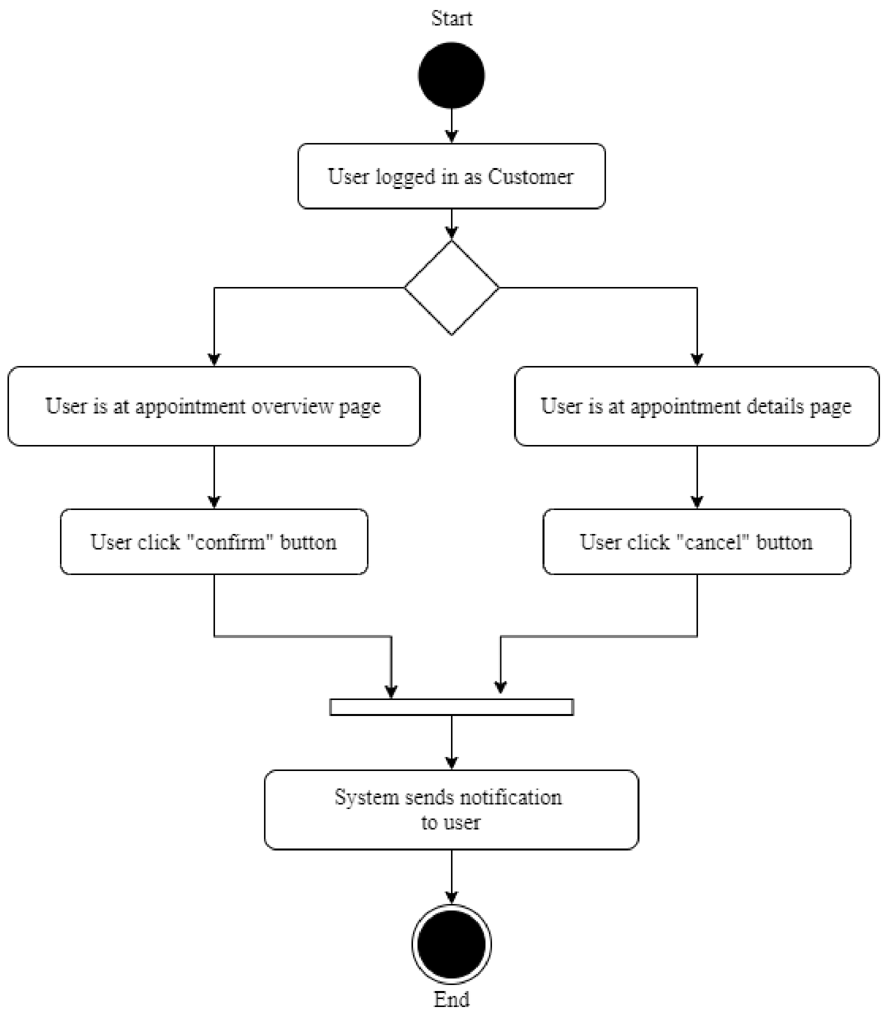

Start
User logged in as Customer
User is at appointment overview page
User is at appointment details page
User click "confirm" button
User click "cancel" button
System sends notification to user
End

Figure 3-4-10-1: Receive Notification Activity Diagram

3.4.11 View Active Appointment

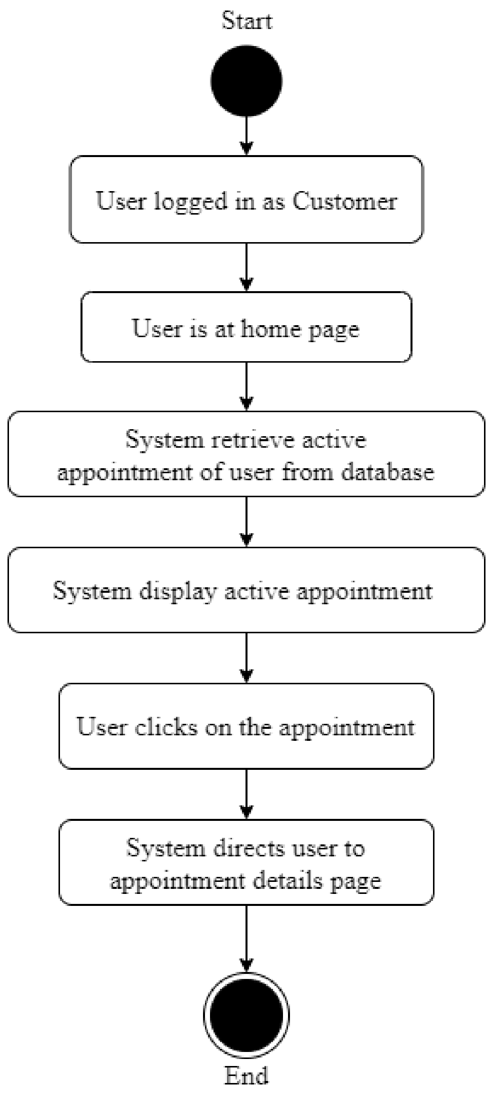

Start
User logged in as Customer
User is at home page
System retrieve active appointment of user from database
System display active appointment
User clicks on the appointment
System directs user to appointment details page
End

Figure 3-4-11-1: View Active Appointment Activity Diagram

3.4.12 Cancel Appointment

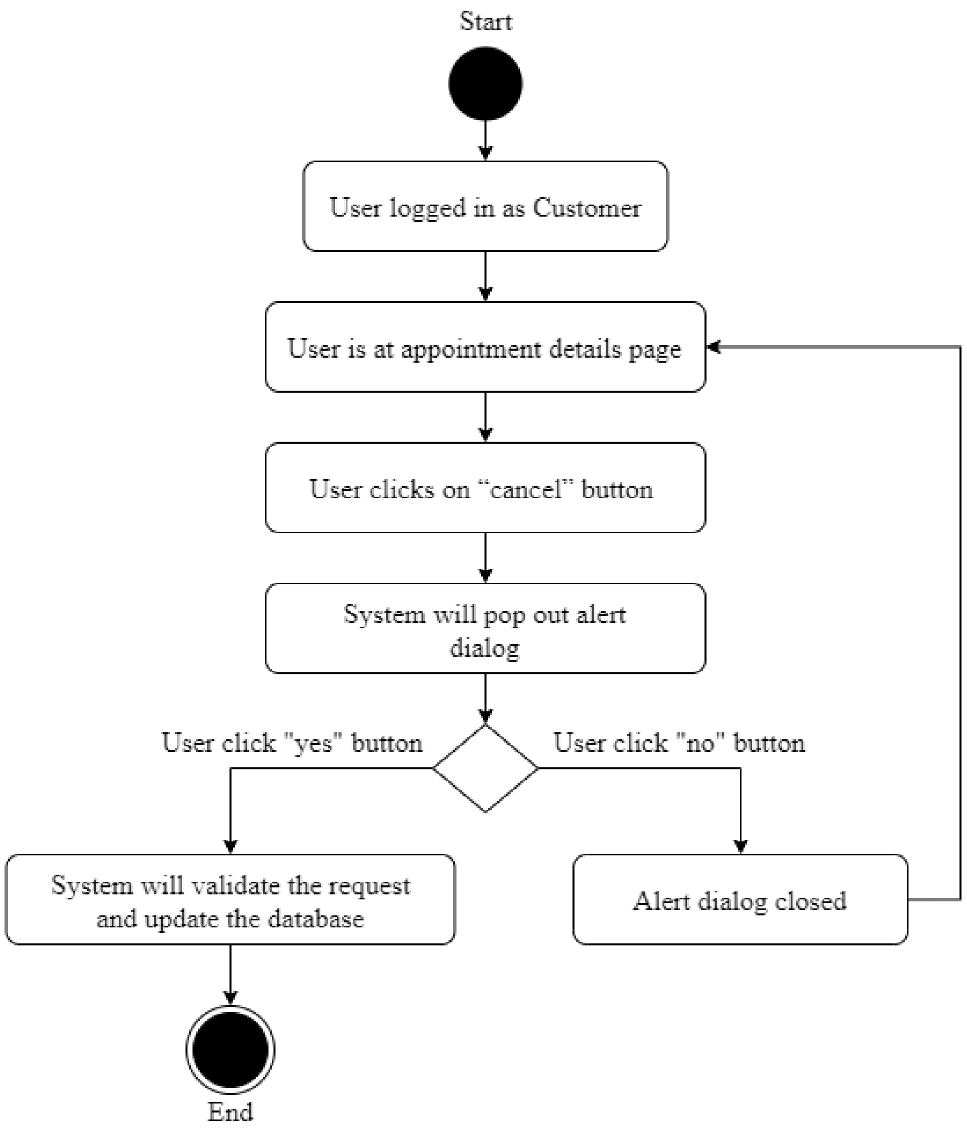

Start
User logged in as Customer
User is at appointment details page
User clicks on "cancel" button
System will pop out alert dialog
User click "yes" button
User click "no" button
System will validate the request and update the database
Alert dialog closed
End

Figure 3-4-12-1: Cancel Appointment Activity Diagram

3.4.13 View Appointment History

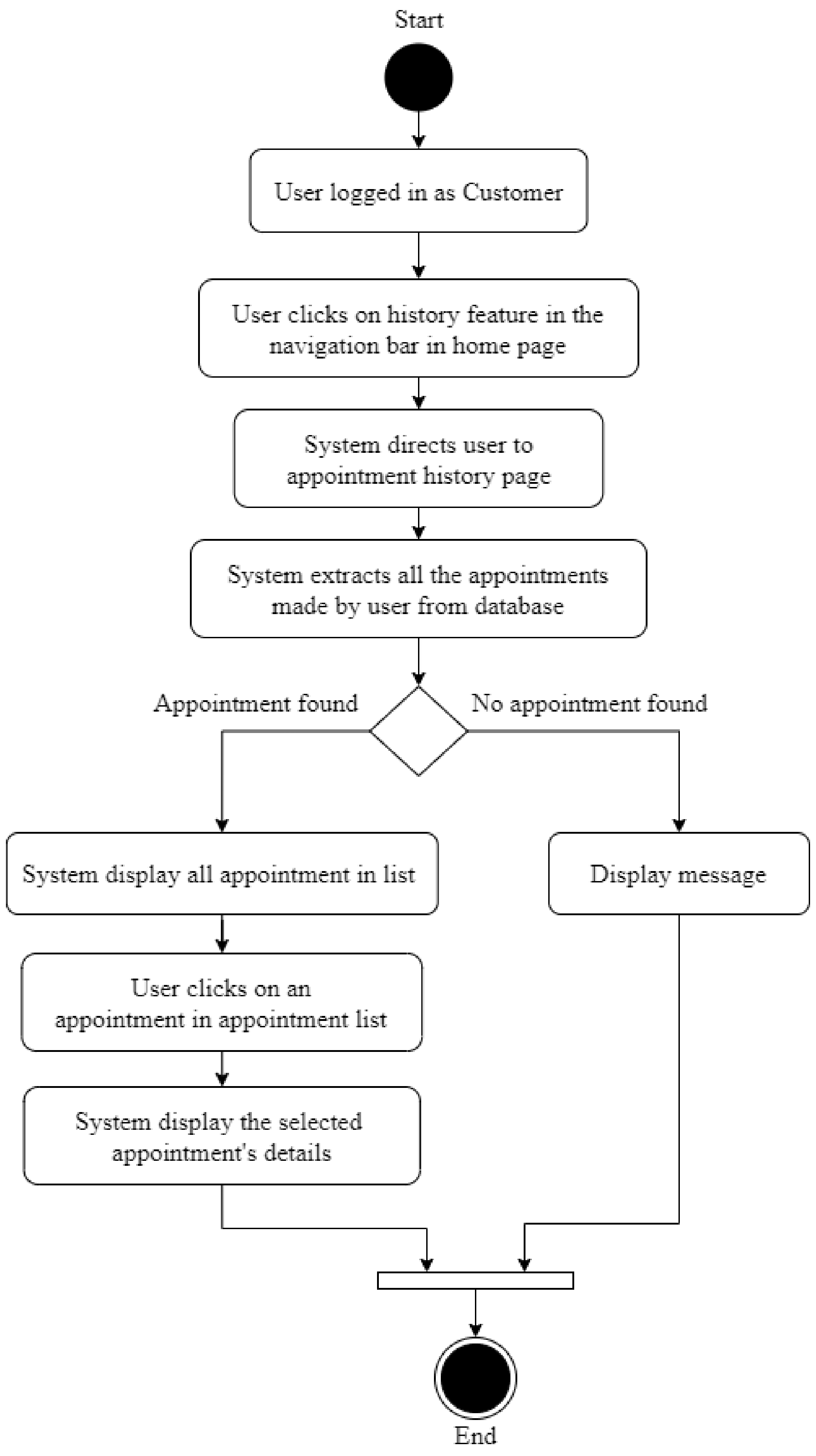

Start
User logged in as Customer
User clicks on history feature in the navigation bar in home page
System directs user to appointment history page
System extracts all the appointments made by user from database
Appointment found
No appointment found
System display all appointment in list
Display message
User clicks on an appointment in appointment list
System display the selected appointment's details
End

Figure 3-4-13-1: View Appointment History Activity Diagram

3.4.14 View Notification

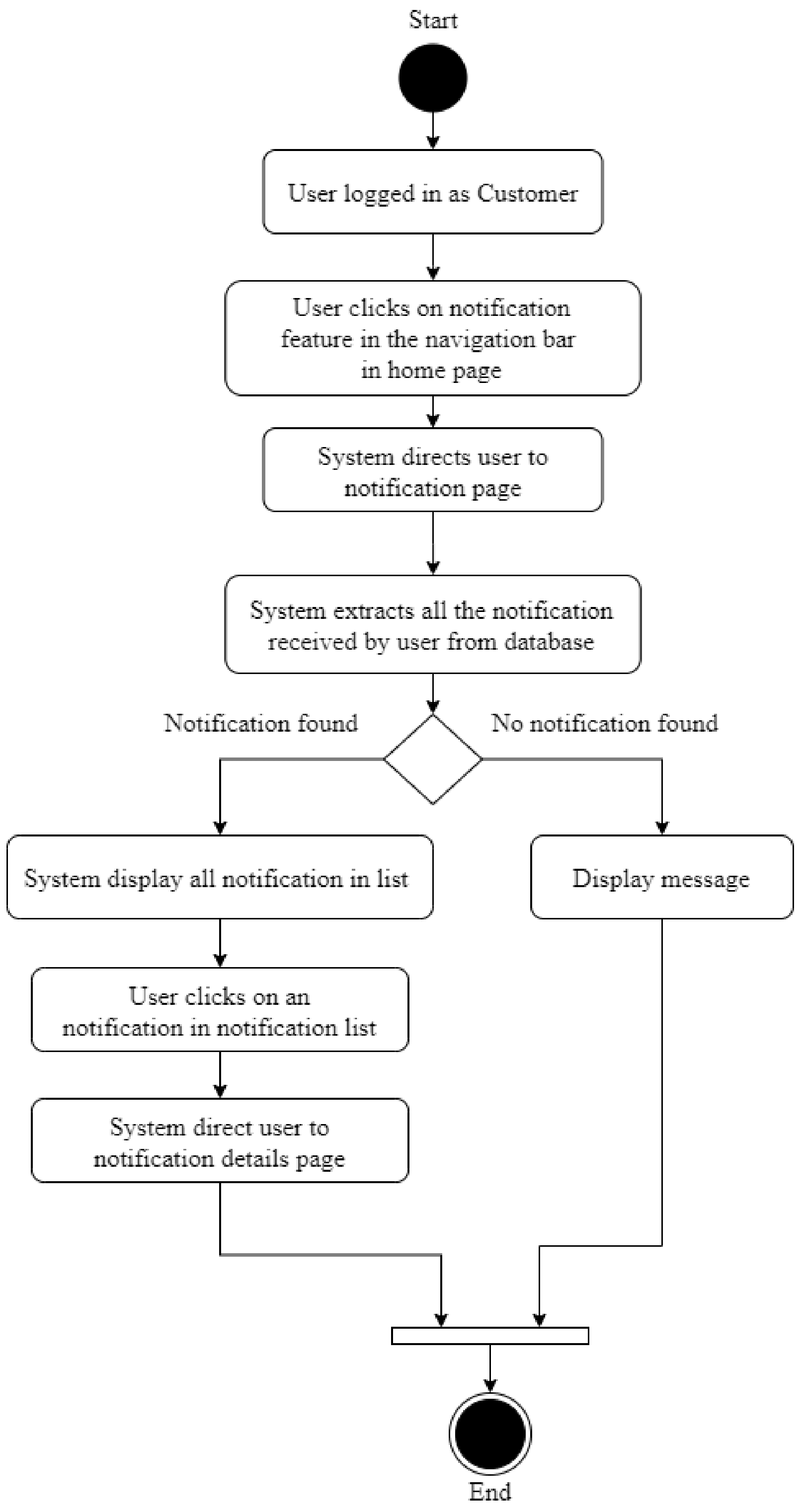

Start
User logged in as Customer
User clicks on notification feature in the navigation bar in home page
System directs user to notification page
System extracts all the notification received by user from database
Notification found
No notification found
System display all notification in list
Display message
User clicks on an notification in notification list
System direct user to notification details page
End

Figure 3-4-14-1: View Notification Activity Diagram

3.4.15 View Vehicle

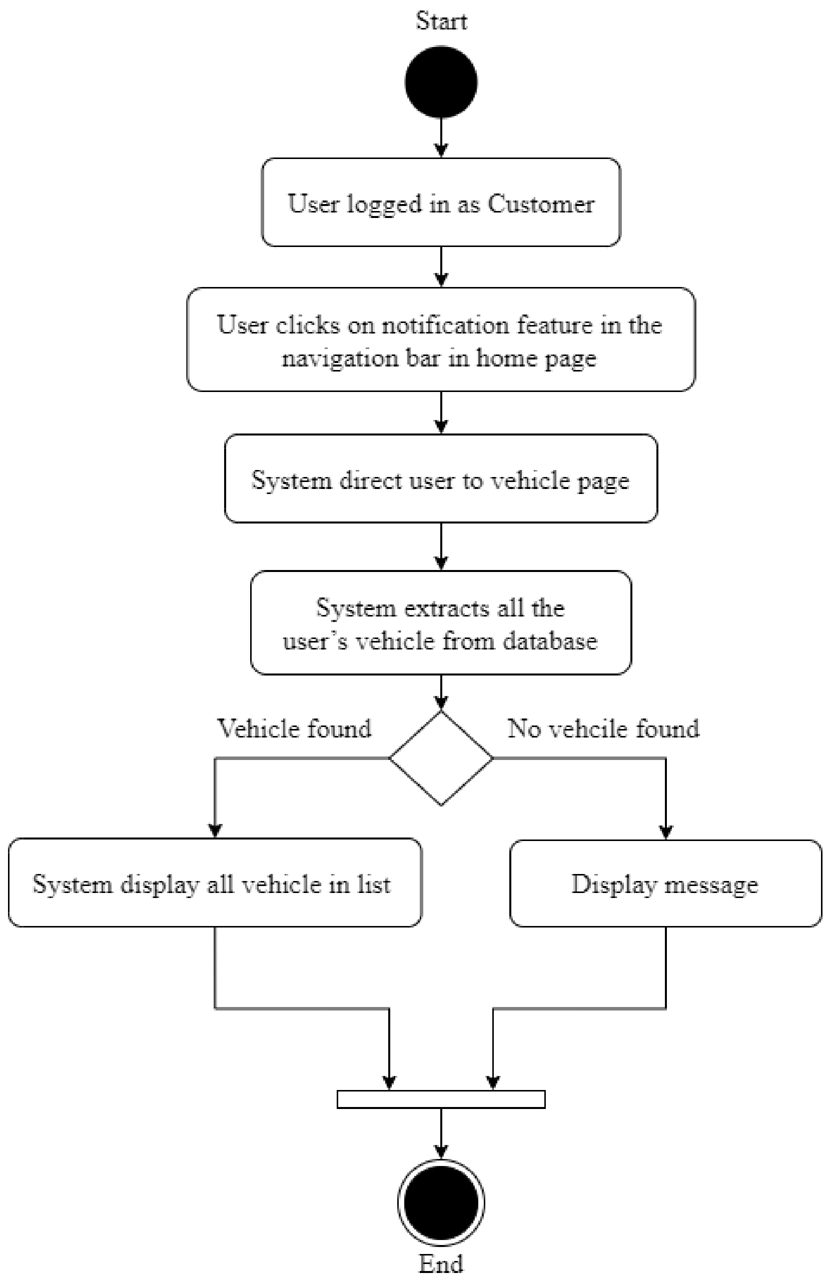

Start
User logged in as Customer
User clicks on notification feature in the navigation bar in home page
System direct user to vehicle page
System extracts all the user's vehicle from database
Vehicle found
No vehcile found
System display all vehicle in list
Display message
End

Figure 3-4-15-1: View Vehicle Activity Diagram

3.4.16 Add Vehicle

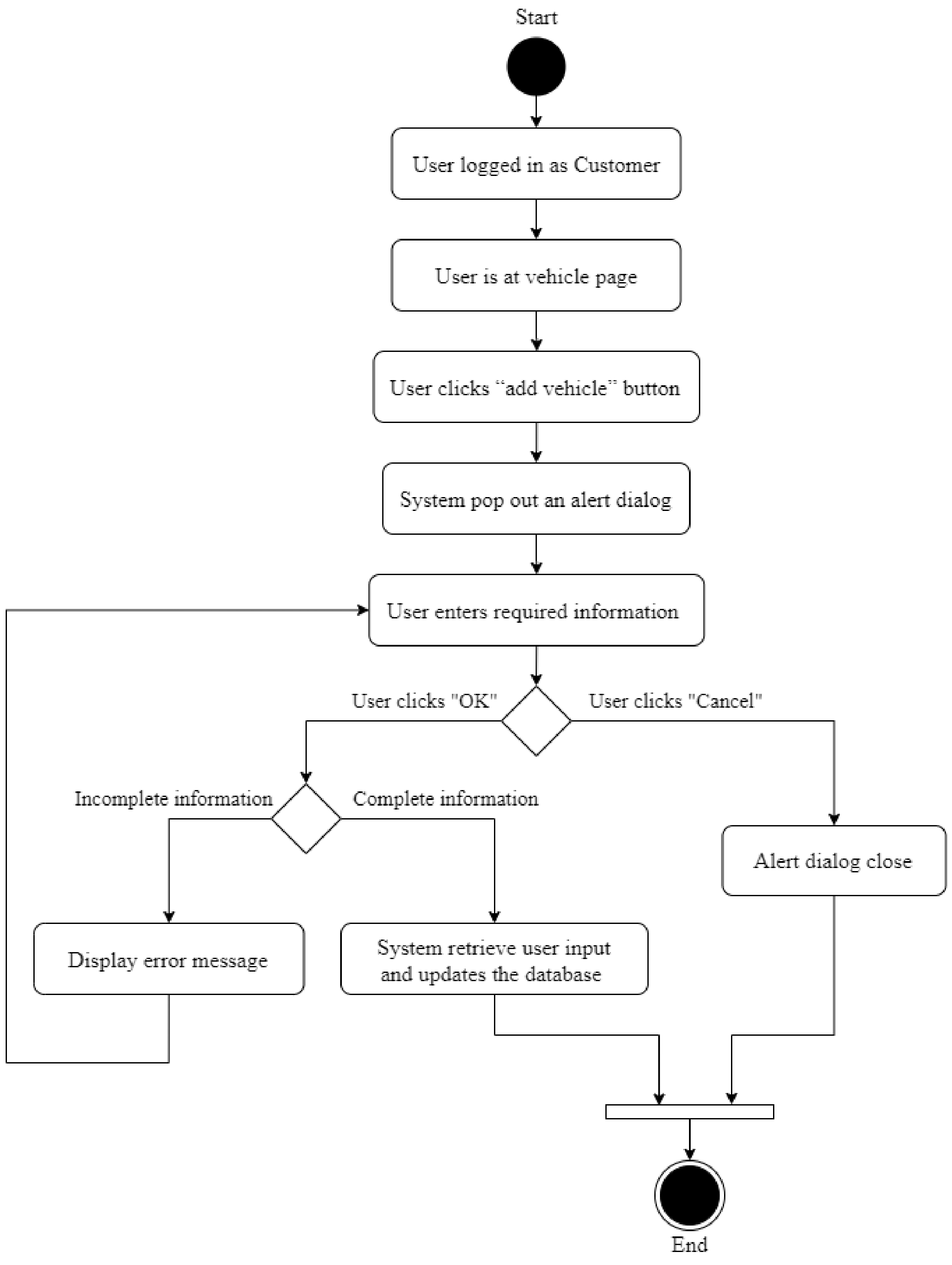

Start
User logged in as Customer
User is at vehicle page
User clicks "add vehicle" button
System pop out an alert dialog
User enters required information
User clicks "OK"
User clicks "Cancel"
Incomplete information
Complete information
Alert dialog close
Display error message
System retrieve user input and updates the database
End

Figure 3-4-16-1: Add Vehicle Activity Diagram

3.4.17 Delete Vehicle

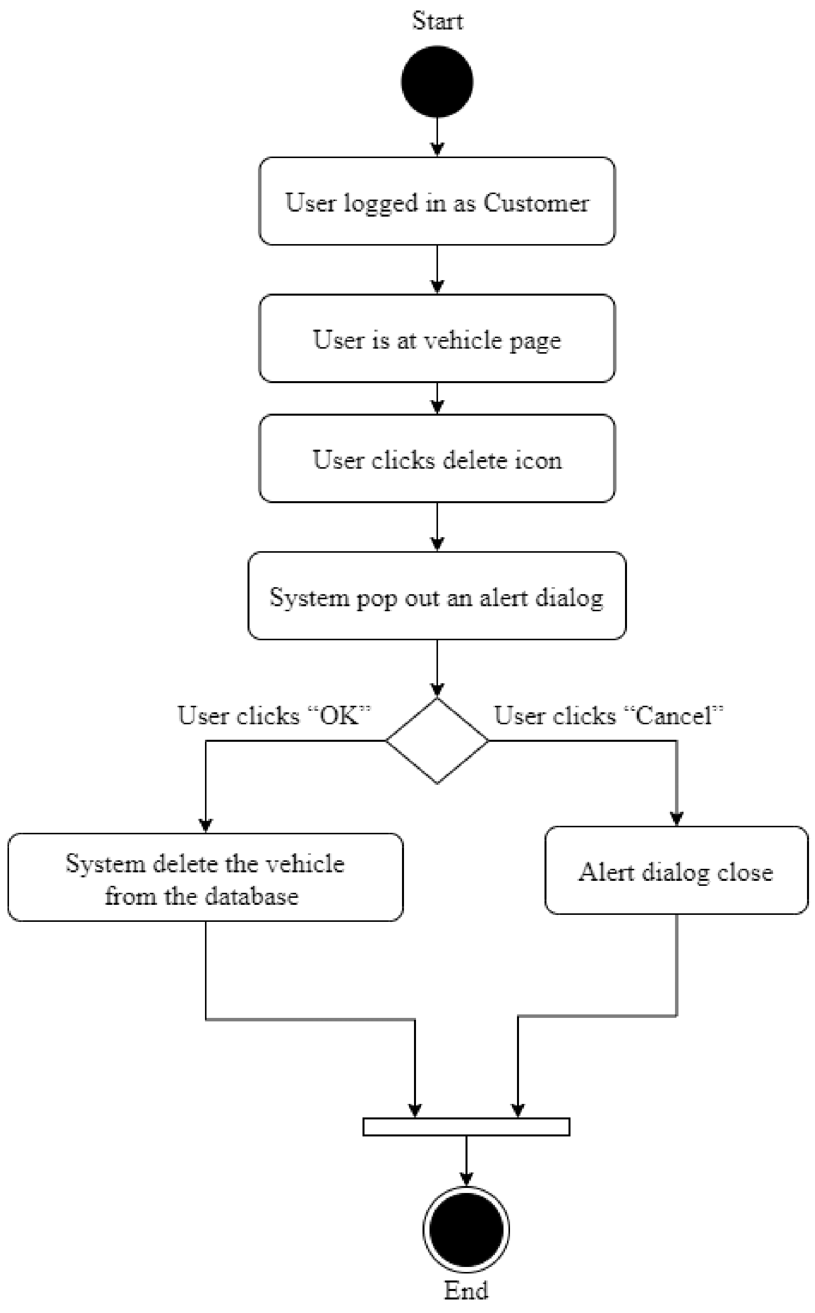


Figure 3-4-17-1: Delete Vehicle Activity Diagram

3.4.18 View Scheduled Appointment List

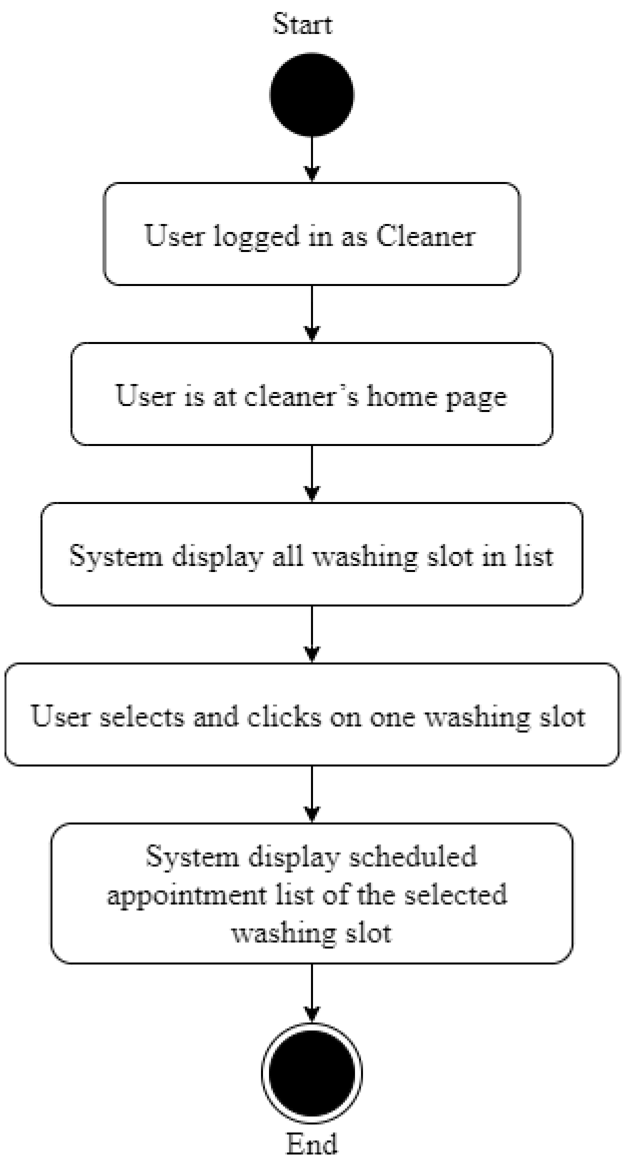

Start
User logged in as Cleaner
User is at cleaner's home page
System display all washing slot in list
User selects and clicks on one washing slot
System display scheduled appointment list of the selected washing slot
End

Figure 3-4-18-1: View Scheduled Appointment List Activity Diagram

3.4.19 Manage Appointment

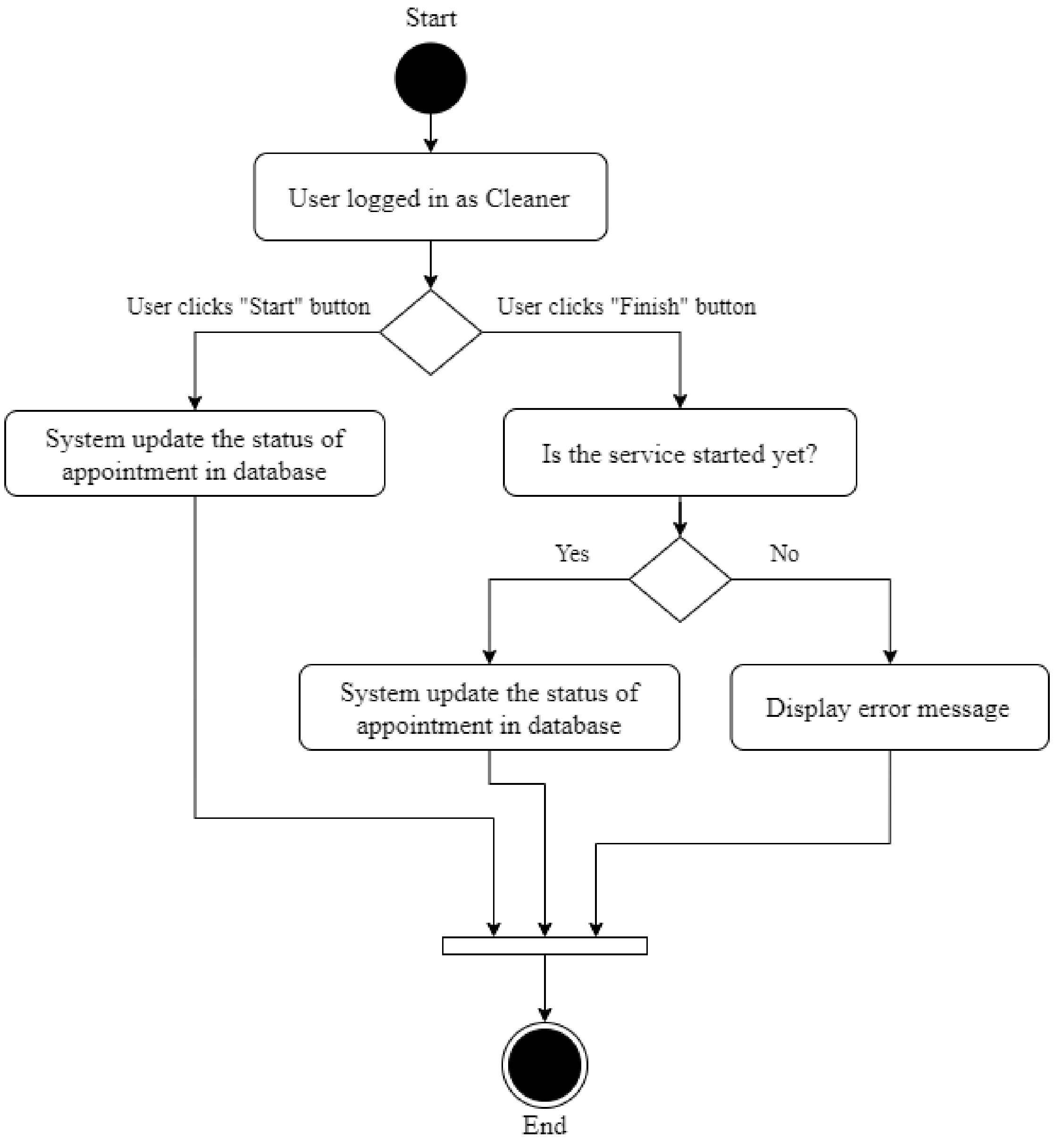

Start
User logged in as Cleaner
User clicks "Start" button
User clicks "Finish" button
System update the status of appointment in database
Is the service started yet?
Yes
No
System update the status of appointment in database
Display error message
End

Figure 3-4-19-1: Manage Appointment Activity Diagram

3.4.20 View Car Wash Center Details

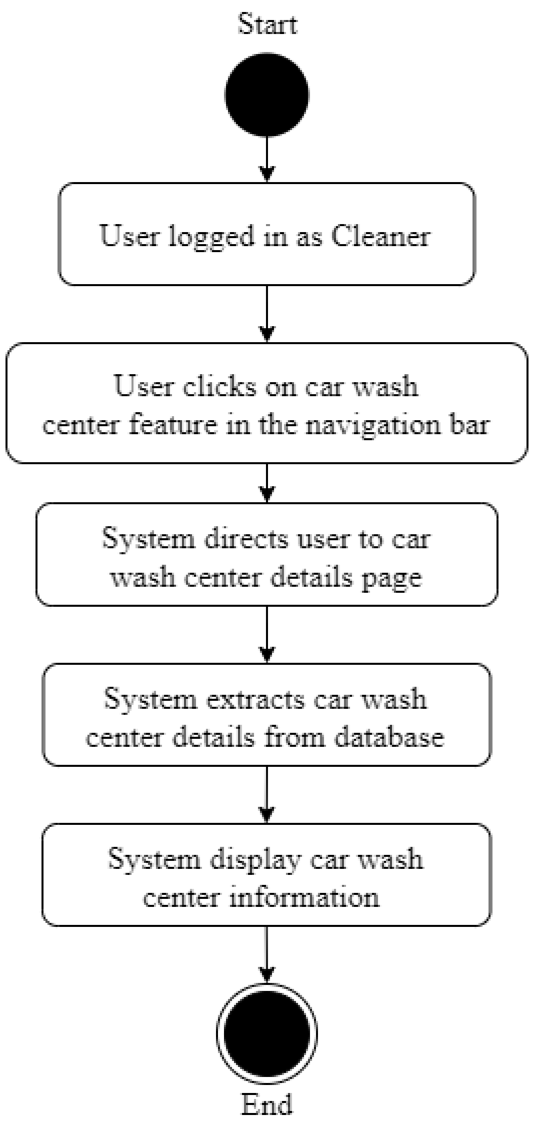

Start
User logged in as Cleaner
User clicks on car wash center feature in the navigation bar
System directs user to car wash center details page
System extracts car wash center details from database
System display car wash center information
End

Figure 3-4-20-1: View Car Wash Center Details Activity Diagram

3.4.21 Update Car Wash Center Details

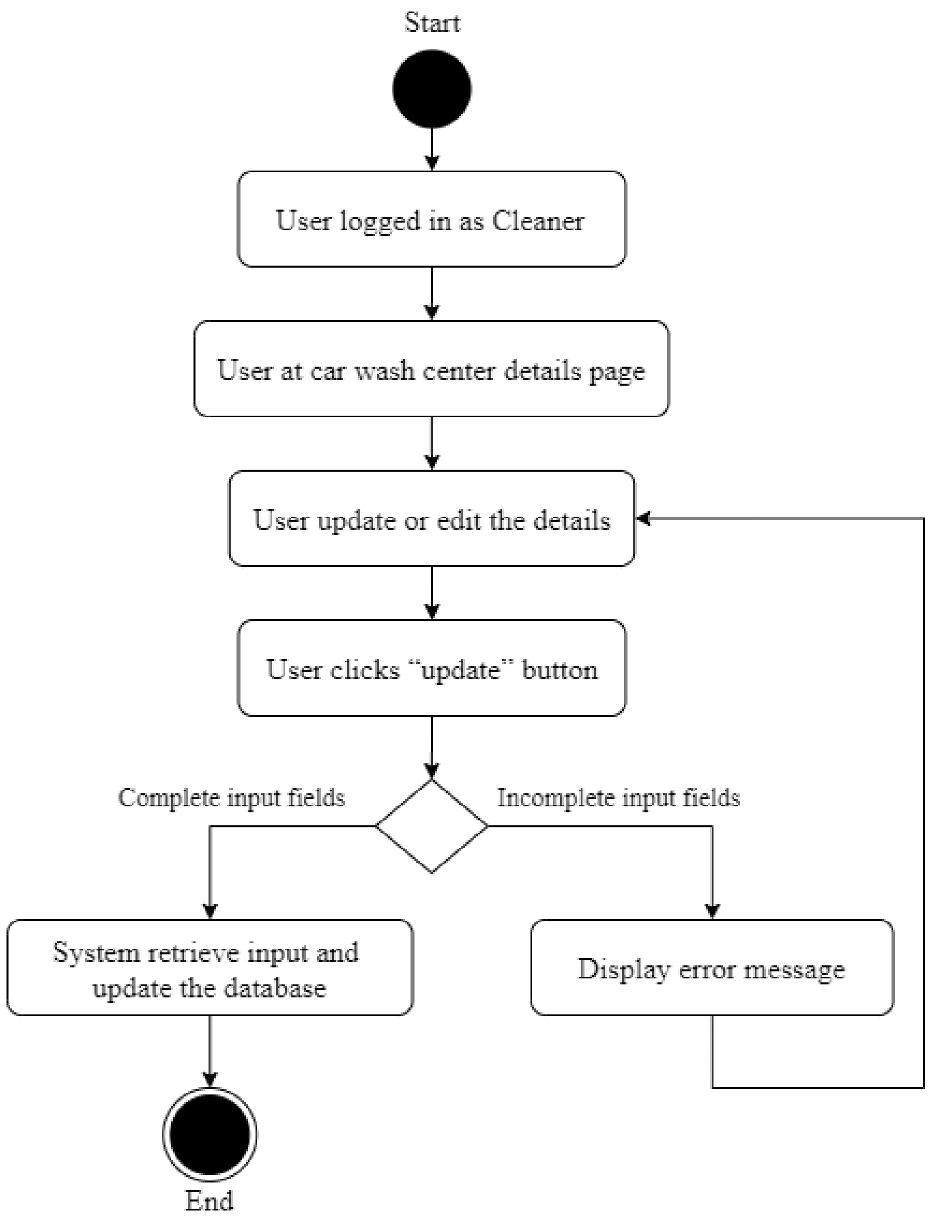

Start
User logged in as Cleaner
User at car wash center details page
User update or edit the details
User clicks "update" button
Complete input fields
Incomplete input fields
System retrieve input and update the database
Display error message
End

Figure 3-4-21-1: Update Car Wash Center Details Activity Diagram

3.4.22 View Car Wash Center’s Service

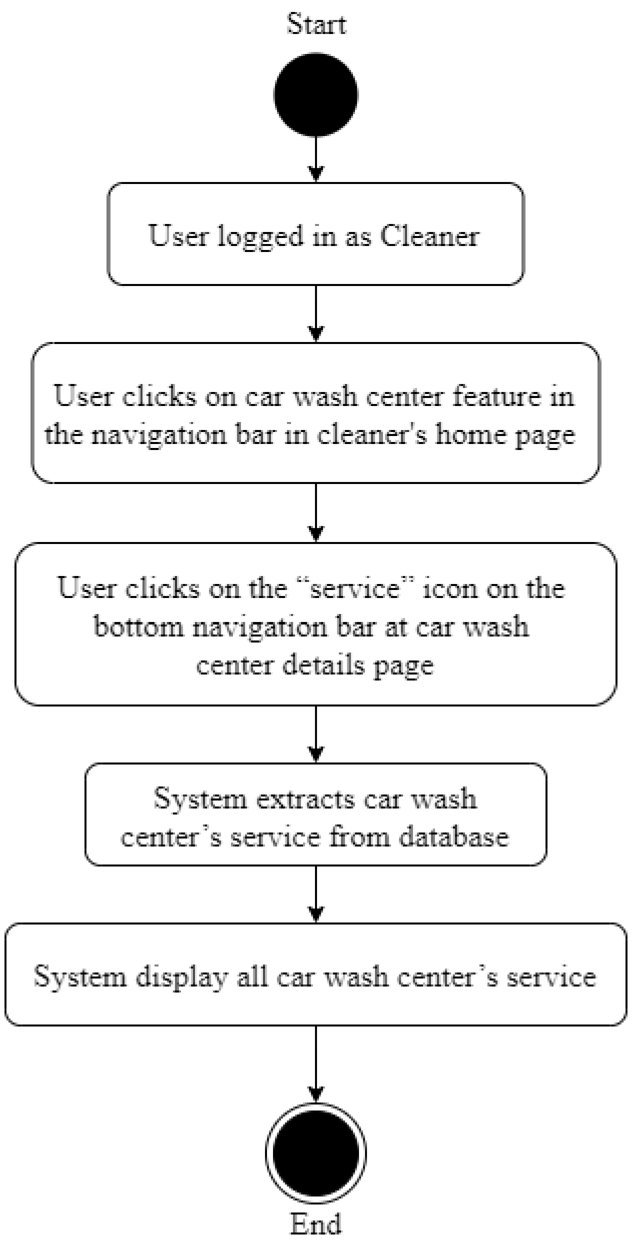

Start
User logged in as Cleaner
User clicks on car wash center feature in the navigation bar in cleaner's home page
User clicks on the “service” icon on the bottom navigation bar at car wash center details page
System extracts car wash center’s service from database
System display all car wash center’s service
End

Figure 3-4-22-1: View Car Wash Center’s Service Activity Diagram

3.4.23 Add Car Wash Center's Service

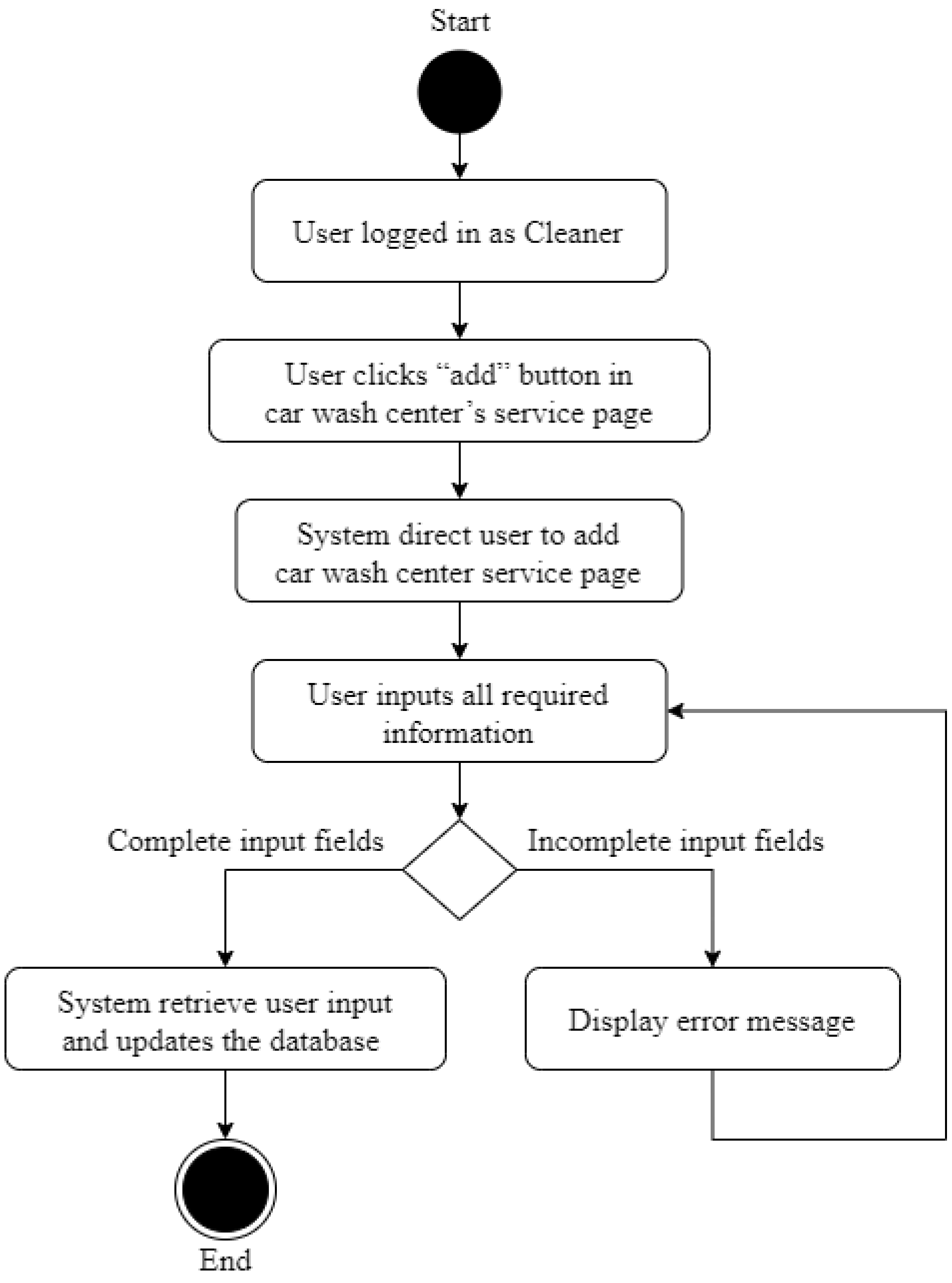

Start
User logged in as Cleaner
User clicks "add" button in car wash center's service page
System direct user to add car wash center service page
User inputs all required information
Complete input fields
Incomplete input fields
System retrieve user input and updates the database
Display error message
End

Figure 3-4-23-1: Add Car Wash Center's Service Activity Diagram

3.4.24 Update Car Wash Center’s Service

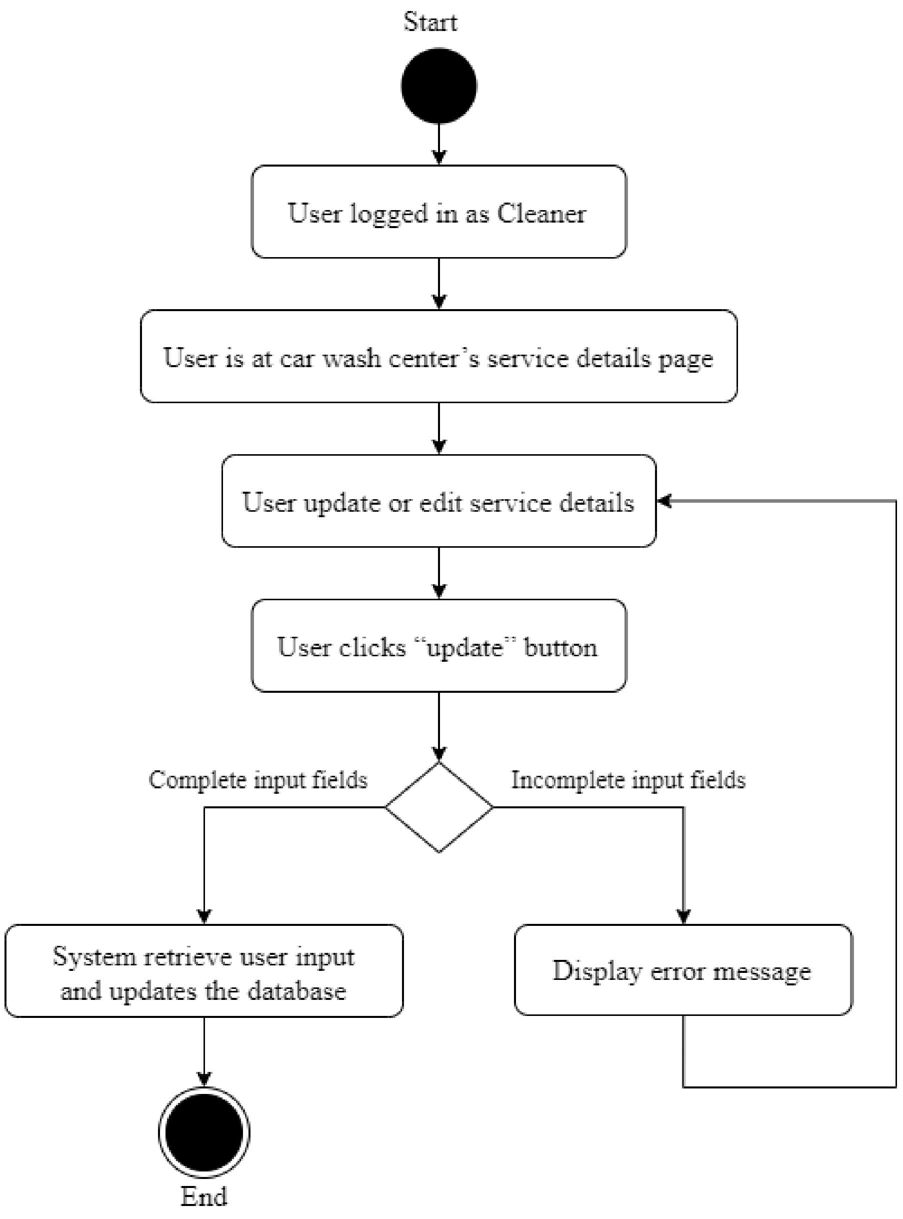

Start
User logged in as Cleaner
User is at car wash center’s service details page
User update or edit service details
User clicks “update” button
Complete input fields
Incomplete input fields
System retrieve user input and updates the database
Display error message
End

Figure 3-4-24-1: Update Car Wash Center’s Service Activity Diagram

3.4.25 Delete Car Wash Center’s Service

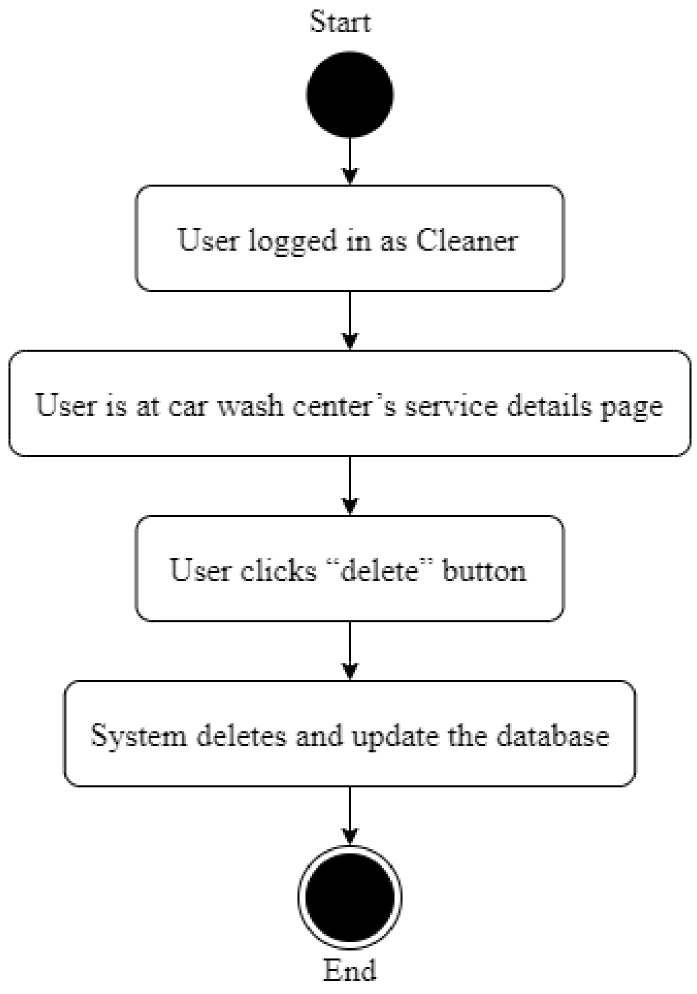

Start
User logged in as Cleaner
User is at car wash center’s service details page
User clicks “delete” button
System deletes and update the database
End

Figure 3-4-25-1: Delete Car Wash Center’s Service Activity Diagram

3.4.26 View Car Wash Center’s Cleaner

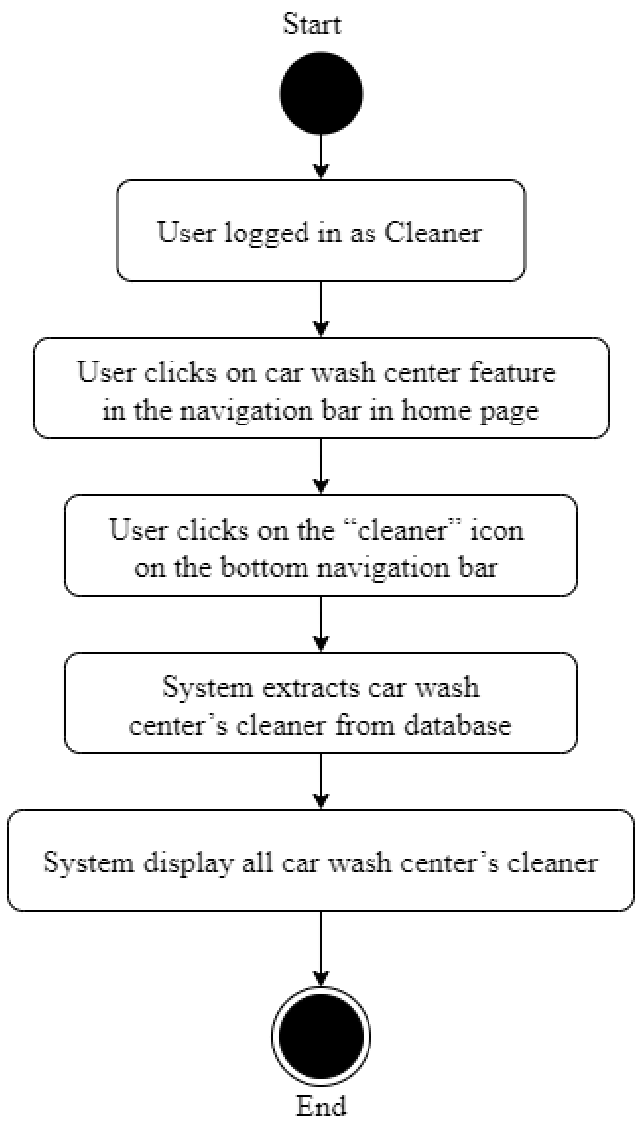

Start
User logged in as Cleaner
User clicks on car wash center feature in the navigation bar in home page
User clicks on the “cleaner” icon on the bottom navigation bar
System extracts car wash center’s cleaner from database
System display all car wash center’s cleaner
End

Figure 3-4-26-1: View Car Wash Center’s Cleaner Activity Diagram

3.4.27 Add Car Wash Center's Cleaner

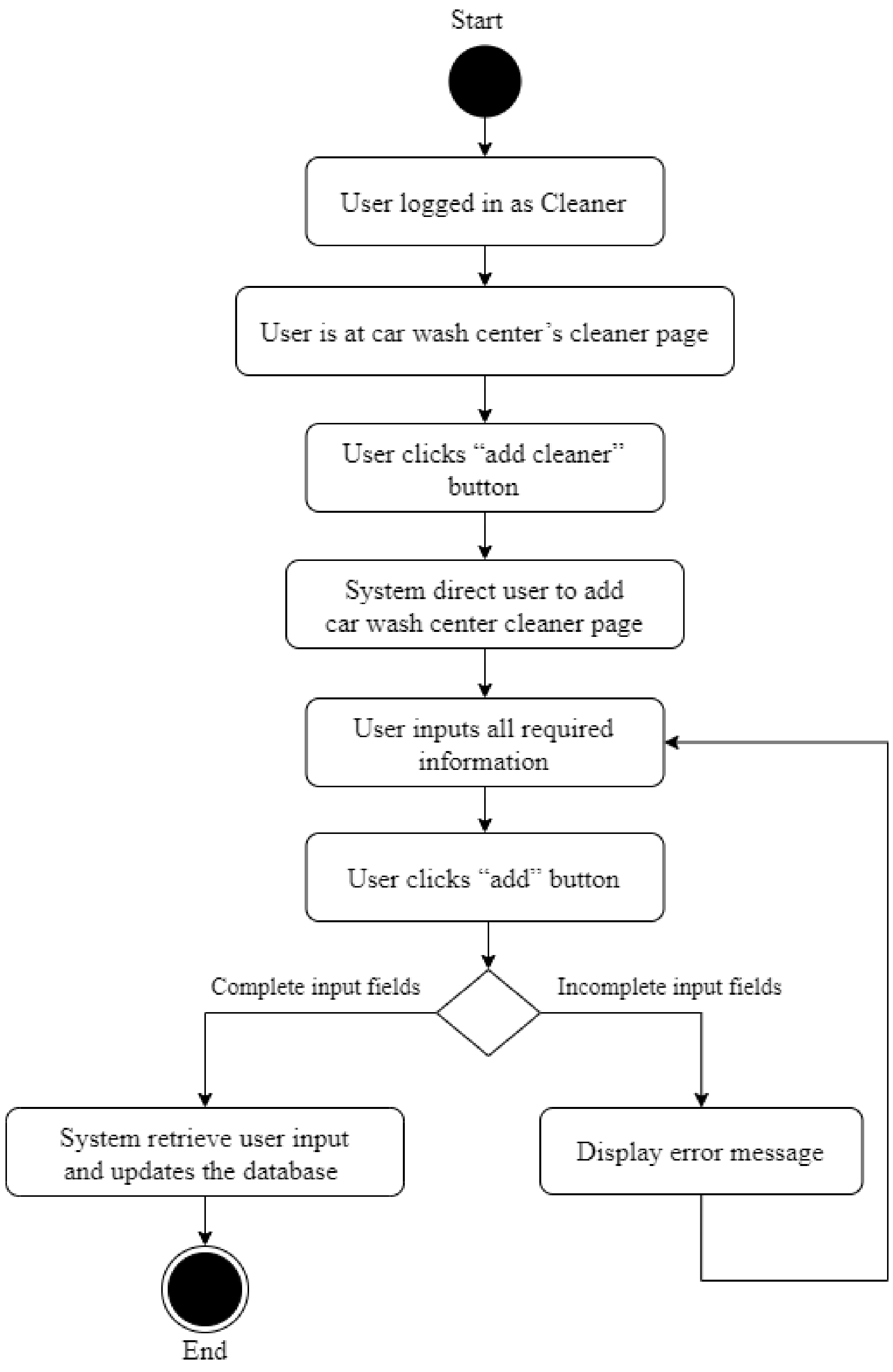

Start
User logged in as Cleaner
User is at car wash center's cleaner page
User clicks "add cleaner" button
System direct user to add car wash center cleaner page
User inputs all required information
User clicks "add" button
Complete input fields
Incomplete input fields
System retrieve user input and updates the database
Display error message
End

Figure 3-4-27-1: Add Car Wash Center's Cleaner Activity Diagram

3.4.28 Delete Car Wash Center’s Cleaner

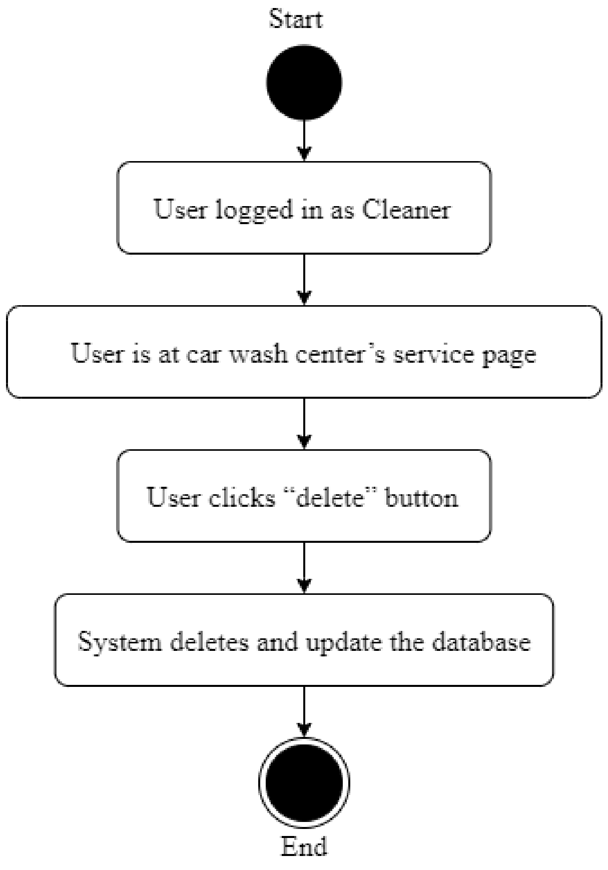

Start
User logged in as Cleaner
User is at car wash center’s service page
User clicks “delete” button
System deletes and update the database
End

Figure 3-4-28-1: Delete Car Wash Center’s Cleaner Activity Diagram

3.4.29 View Car Wash Center's Vehicle Type

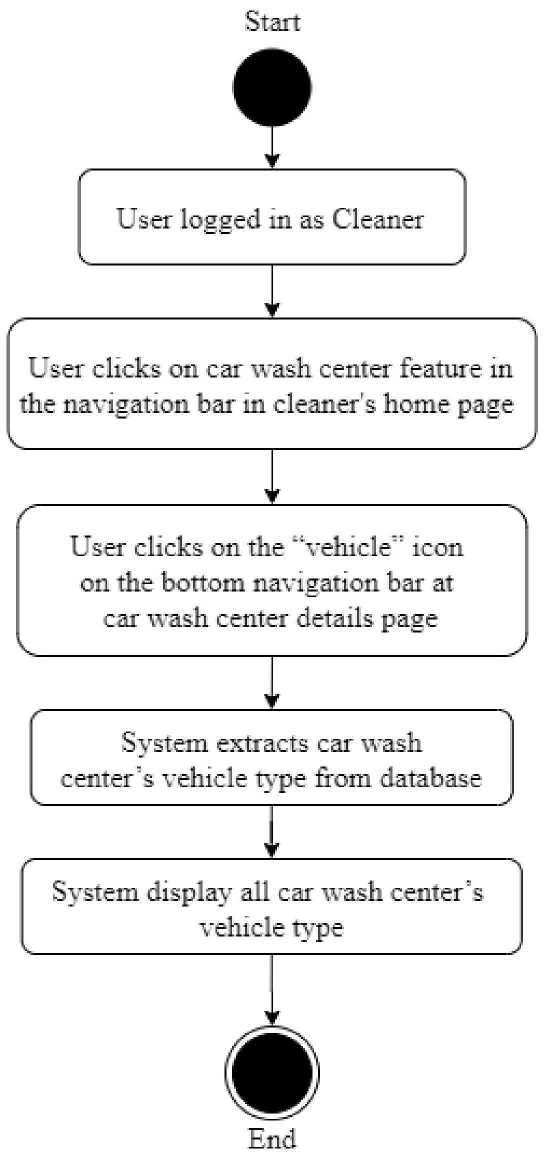

Start
User logged in as Cleaner
User clicks on car wash center feature in the navigation bar in cleaner's home page
User clicks on the "vehicle" icon on the bottom navigation bar at car wash center details page
System extracts car wash center's vehicle type from database
System display all car wash center's vehicle type
End

Figure 3-4-29-1: View Car Wash Center's Vehicle Type Activity Diagram

3.4.30 Add Car Wash Center's Vehicle Type

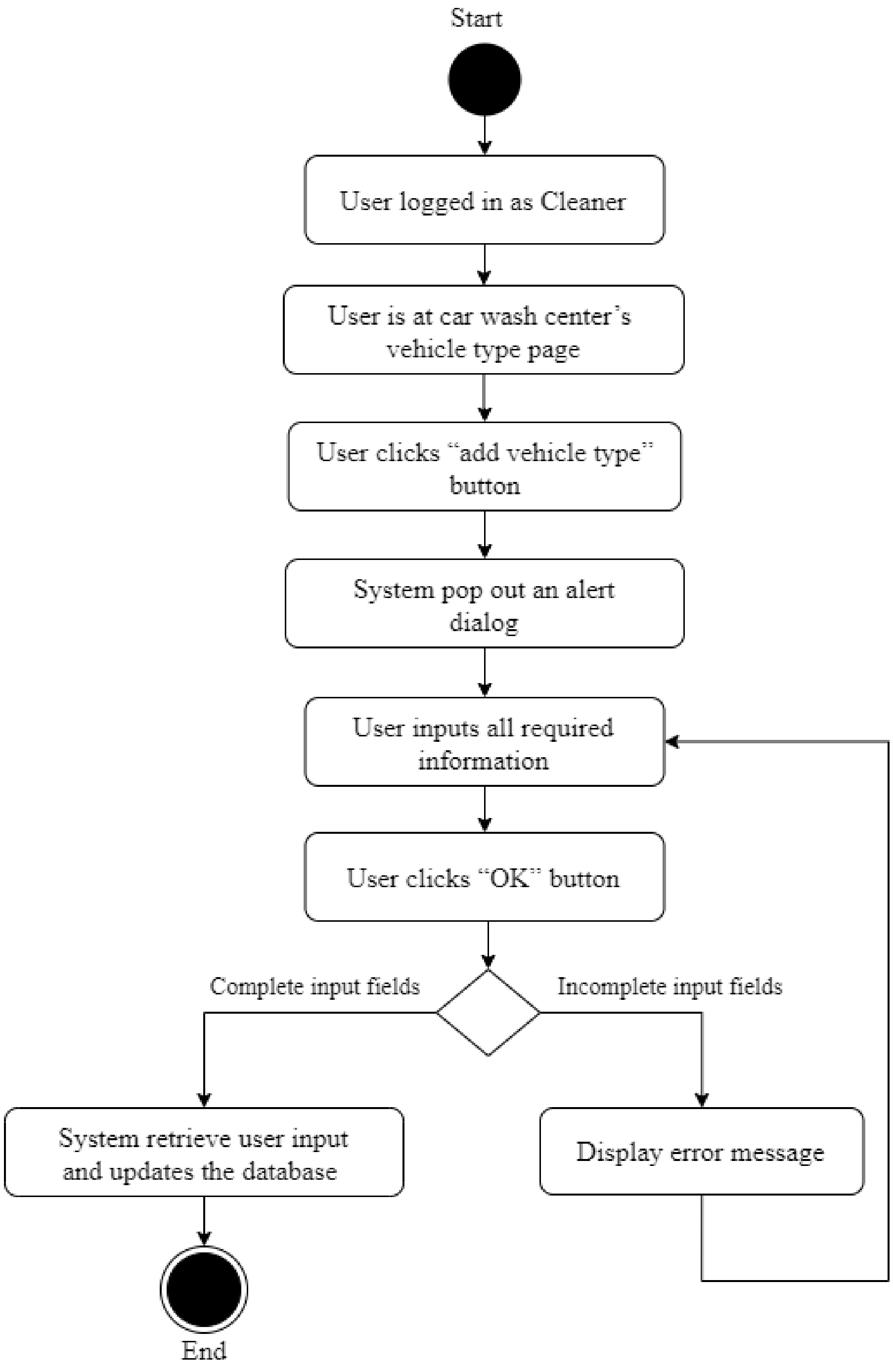

Start
User logged in as Cleaner
User is at car wash center's vehicle type page
User clicks "add vehicle type" button
System pop out an alert dialog
User inputs all required information
User clicks "OK" button
Complete input fields
Incomplete input fields
System retrieve user input and updates the database
Display error message
End

Figure 3-4-30-1: Add Car Wash Center's Vehicle Type Activity Diagram

3.4.31 Delete Car Wash Center's Vehicle Type

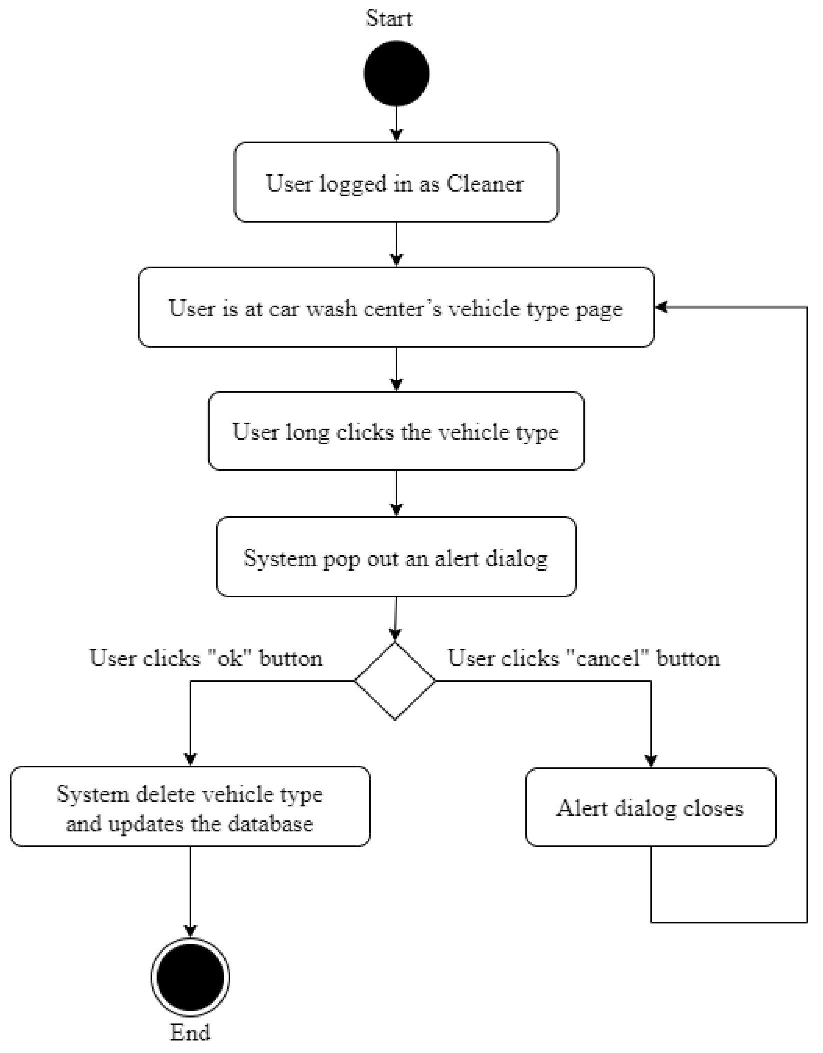

Start
User logged in as Cleaner
User is at car wash center's vehicle type page
User long clicks the vehicle type
System pop out an alert dialog
User clicks "ok" button
User clicks "cancel" button
System delete vehicle type and updates the database
Alert dialog closes
End

Figure 3-4-31-1: Delete Car Wash Center's Vehicle Type Activity Diagram

3.4.32 View Appointment History

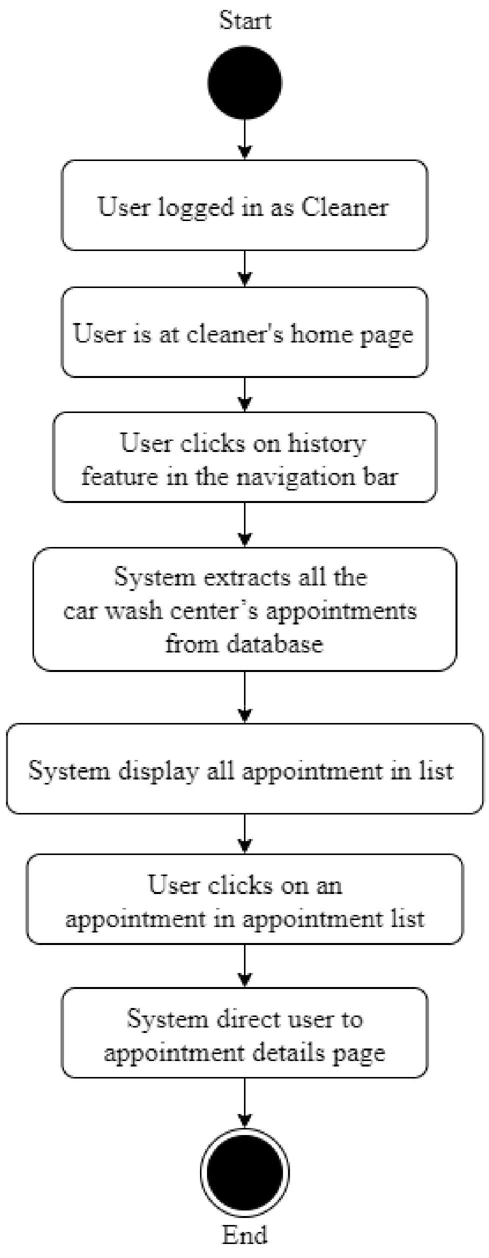

Start
User logged in as Cleaner
User is at cleaner's home page
User clicks on history feature in the navigation bar
System extracts all the car wash center's appointments from database
System display all appointment in list
User clicks on an appointment in appointment list
System direct user to appointment details page
End

Figure 3-4-32-1: View Appointment History Activity Diagram

3.5 Appointment Scheduling System Overview

Scheduling algorithm is used in this project to schedule the appointment made by users. Algorithm is used in order to minimize the waiting time of customers in car wash center. Moreover, it also utilizes the manpower and increase the performance of car wash center.

Threshold is included while scheduling appointment in this application:

i. The total number of appointment, N in a time slot cannot exceed the total number of washing slot available in car wash center. N cannot exceed the number of washing slot available in car wash center in order to prevent too many appointments made on a particular time slot and lead overbooked situation. Overbooked situation will cause much delay on the appointment made in the next time slot.

3.5.1 Least Laxity First (LLF) Scheduling Algorithm

Least Laxity First (LLF) scheduling algorithm assigns the priority of being executed according to the laxity value of the task. The task that has the lowest laxity value will have the highest priority. Although LLF scheduling algorithm is an optimal algorithm for uniprocessor system, however it leads to frequent switches among tasks, which effects the performance of cleaning process. Hence, non pre-emptive Least Laxity First scheduling algorithm is chosen to be applied in appointment scheduling. Figure 3-5-1-1 has shown the block diagram of LLF scheduling algorithm in appointment scheduling system.

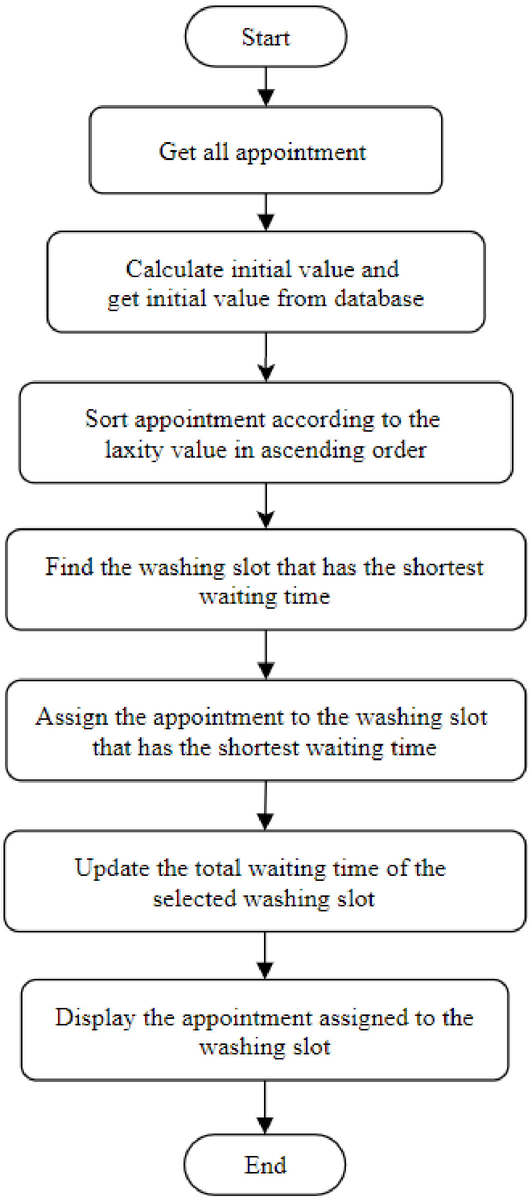

Start
Get all appointment
Calculate initial value and get initial value from database
Sort appointment according to the laxity value in ascending order
Find the washing slot that has the shortest waiting time
Assign the appointment to the washing slot that has the shortest waiting time
Update the total waiting time of the selected washing slot
Display the appointment assigned to the washing slot
End

Figure 3-5-1-1: Block Diagram of Least Laxity First Scheduling Algorithm

3.5.1.1 Get all appointment

Appointments made by customers is stored in the database. The details of the appointment such as the time slot, type of service, duration of the service is also stored in the database. The appointments are retrieved from database to act as the input of the scheduling. All the appointments on that day will be inserted into an appointment array list.

3.5.1.2 Calculate initial value and get initial value

There is some initial value needed to be calculate before proceed to scheduling phase:

i. Execution time

   Execution time is the time needed to finish the task. Its value is set by the car wash center and stored in the database. The unit of execution time is in millisecond.

ii. Worst case execution time

   Worst case execution time is the maximum time a task could take in order to finish the task. Its value is set by the car wash center and stored in the database. The unit of worst case execution time is in millisecond.

iii. Deadline

   Deadline is the time at which the task has to be completed.

iv. Laxity time

   Laxity time is the time left for the task if the task is executed now. The unit of laxity time is in millisecond.
   It is calculated using the formula below:
   Laxity time = Deadline – (Current time – Execution time)

v. Number of washing slot of car wash center

   The total number of washing slot of the car wash center is retrieved from the database. The total number of washing slot is also set by the car wash center.

3.5.1.3 Sort appointment

Least Laxity First (LLF) scheduling algorithm is used to apply in scheduling appointments on real-time basis. LLF schedules tasks and assign priority according to the laxity value. Every task has its own laxity time which is its weight to execute. The higher the weight, the smaller the laxity time which means that task that has the smallest laxity time will have the highest priority to execute first. All the tasks in appointment array list is sorted according its laxity value. If the laxity value of the task is the same, the task that has smaller execution time will have higher priority.

Non-preemptive scheduling is applied in this scheduling process. When the cleaning process started to execute, the process will continue until it is finished. Then only the next cleaning process can be executed. In this stage, the first task of ready queue will be chosen to execute. When there are new tasks added into ready queue, the tasks are sorted again according to the laxity value.

3.5.1.4 Find washing slot that has the shortest waiting time

Washing slot that has the shortest waiting time is chosen.

3.5.1.5 Assign appointment

The first appointment in the appointment array list is then assign to the washing slot that has the shortest waiting time.

3.5.1.6 Update total waiting time of washing slot

The total waiting time of washing slot is updated.

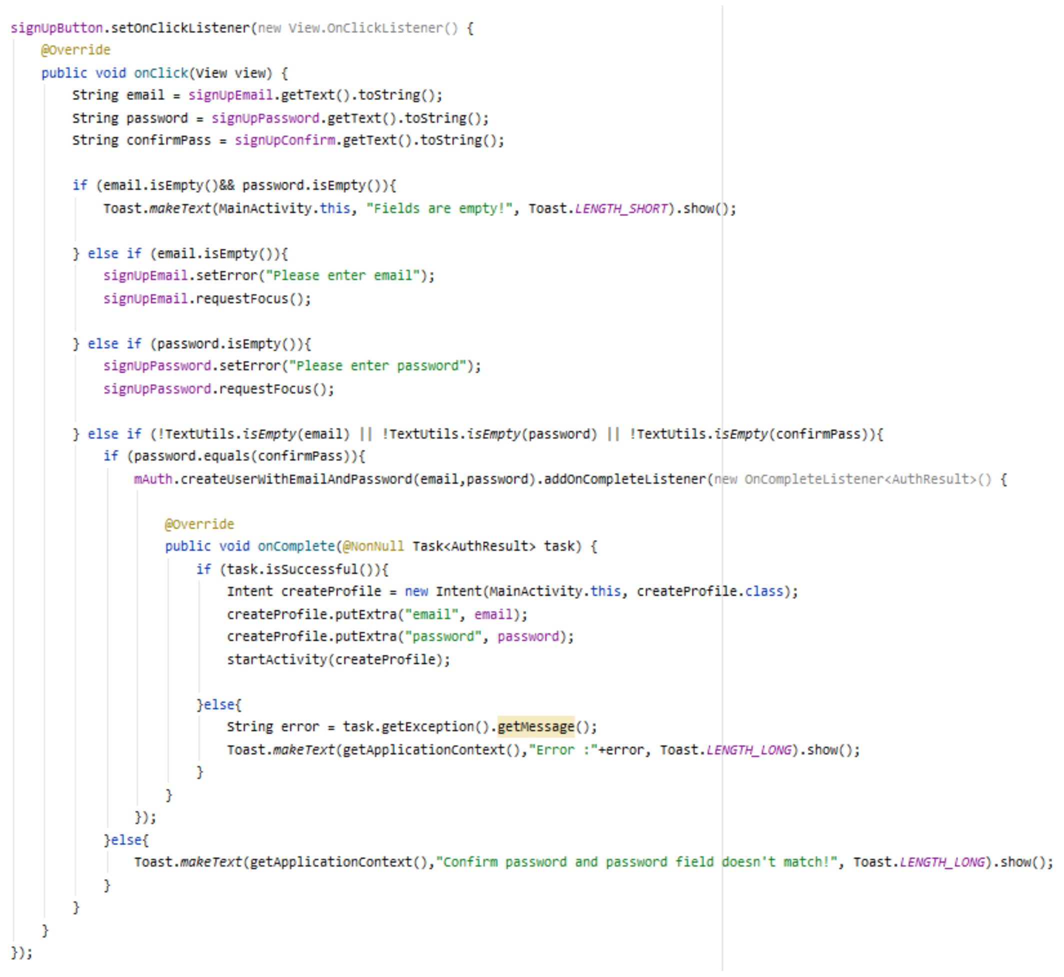
3.5.1.7 Scheduled queue

The cleaners of the car wash center will execute the tasks according to the sequence scheduled.

3.6 Coding Explanation

To start a project development, a new Android project in Android Studio and Firebase account is needed. Next, the project needs to be connected to the Firebase to exploit the authentication, real time database and cloud storage service. Dependencies is added to include the external library in Android Studio in order to implement some extra features.

3.6.1 Sign up

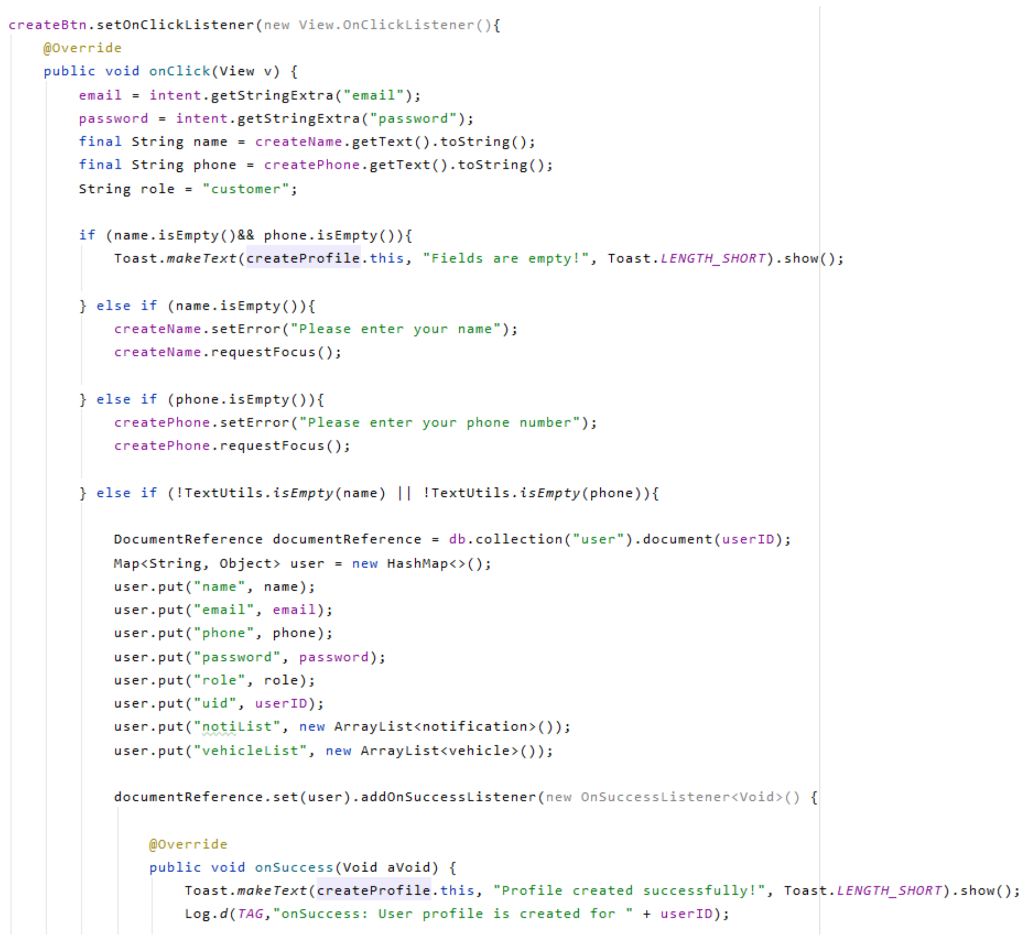


```
signUpButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String email = signUpEmail.getText().toString();
        String password = signUpPassword.getText().toString();
        String confirmPass = signUpConfirm.getText().toString();

        if (email.isEmpty()&& password.isEmpty()){
            Toast.makeText(MainActivity.this, "Fields are empty!", Toast.LENGTH_SHORT).show();

        } else if (email.isEmpty()){
            signUpEmail.setError("Please enter email");
            signUpEmail.requestFocus();

        } else if (password.isEmpty()){
            signUpPassword.setError("Please enter password");
            signUpPassword.requestFocus();

        } else if (!TextUtils.isEmpty(email) || !TextUtils.isEmpty(password) || !TextUtils.isEmpty(confirmPass)){
            if (password.equals(confirmPass)){
                mAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {

                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()){
                            Intent createProfile = new Intent(MainActivity.this, createProfile.class);
                            createProfile.putExtra("email", email);
                            createProfile.putExtra("password", password);
                            startActivity(createProfile);

                        }else{
                            String error = task.getException().getMessage();
                            Toast.makeText(getApplicationContext(),"Error :"+error, Toast.LENGTH_LONG).show();
                        }
                    }
                });
            }else{
                Toast.makeText(getApplicationContext(),"Confirm password and password field doesn't match!", Toast.LENGTH_LONG).show();
            }
        }
    }
});
```

Figure 3-6-1-1: Sign Up

Figure 3-6-1-1 shows the code for sign up function. Firstly, when the sign up button is pressed, user inputs such as email, password and confirmation password is retrieved by the system. The system will check whether there are any missing inputs. If user input is not complete, system will display error message to user. System will also check the similarity of password and confirmation password. If the password and confirmation does not match, error message will be displayed to user. Next, if there is no missing inputs and password and confirmation password is the same, the system will create user with createUserWithEmailAndPassword() function. If the user is created successfully, system will direct user to create profile page.

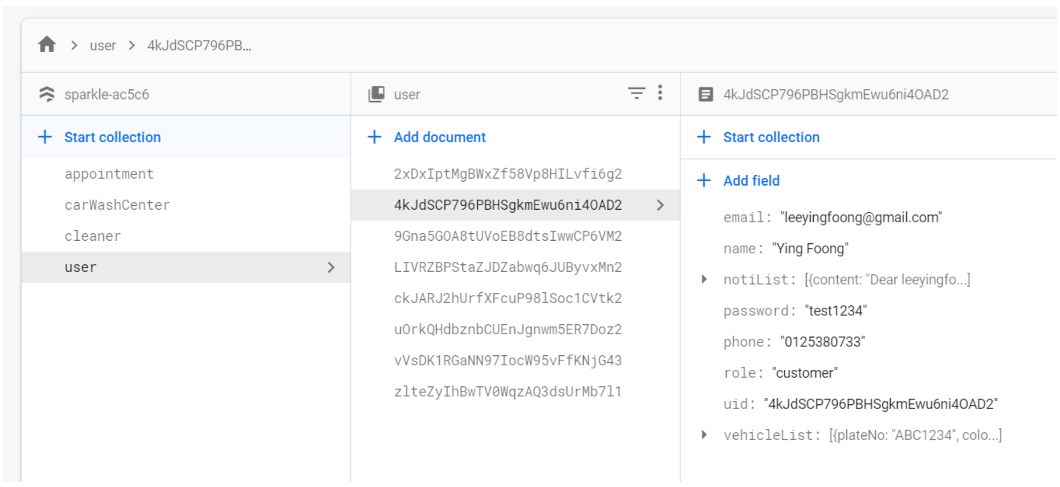
3.6.2 Create profile



```
createBtn.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        email = intent.getStringExtra("email");
        password = intent.getStringExtra("password");
        final String name = createName.getText().toString();
        final String phone = createPhone.getText().toString();
        String role = "customer";

        if (name.isEmpty()&& phone.isEmpty()){
            Toast.makeText(createProfile.this, "Fields are empty!", Toast.LENGTH_SHORT).show();

        } else if (name.isEmpty()){
            createName.setError("Please enter your name");
            createName.requestFocus();

        } else if (phone.isEmpty()){
            createPhone.setError("Please enter your phone number");
            createPhone.requestFocus();

        } else if (!TextUtils.isEmpty(name) || !TextUtils.isEmpty(phone)){

            DocumentReference documentReference = db.collection("user").document(userID);
            Map<String, Object> user = new HashMap<>();
            user.put("name", name);
            user.put("email", email);
            user.put("phone", phone);
            user.put("password", password);
            user.put("role", role);
            user.put("uid", userID);
            user.put("notiList", new ArrayList<notification>());
            user.put("vehicleList", new ArrayList<vehicle>());

            documentReference.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {

                @Override
                public void onSuccess(Void aVoid) {
                    Toast.makeText(createProfile.this, "Profile created successfully!", Toast.LENGTH_SHORT).show();
                    Log.d(TAG,"onSuccess: User profile is created for " + userID);
```

Figure 3-6-2-1: Create Profile

Figure 3-6-2-1 shows the code for create profile function. When the create button is pressed, it will trigger this create profile function. The input fields are retrieved by system and do the checking. If the input fields are not complete, they system will display error message to user. Else, system will store the input fields such as email, name and phone into HashMap. Then, all the input fields stored in the HashMap will add to “user” collection in Cloud Firestore.



```
appointment
carWashCenter
cleaner
user
```

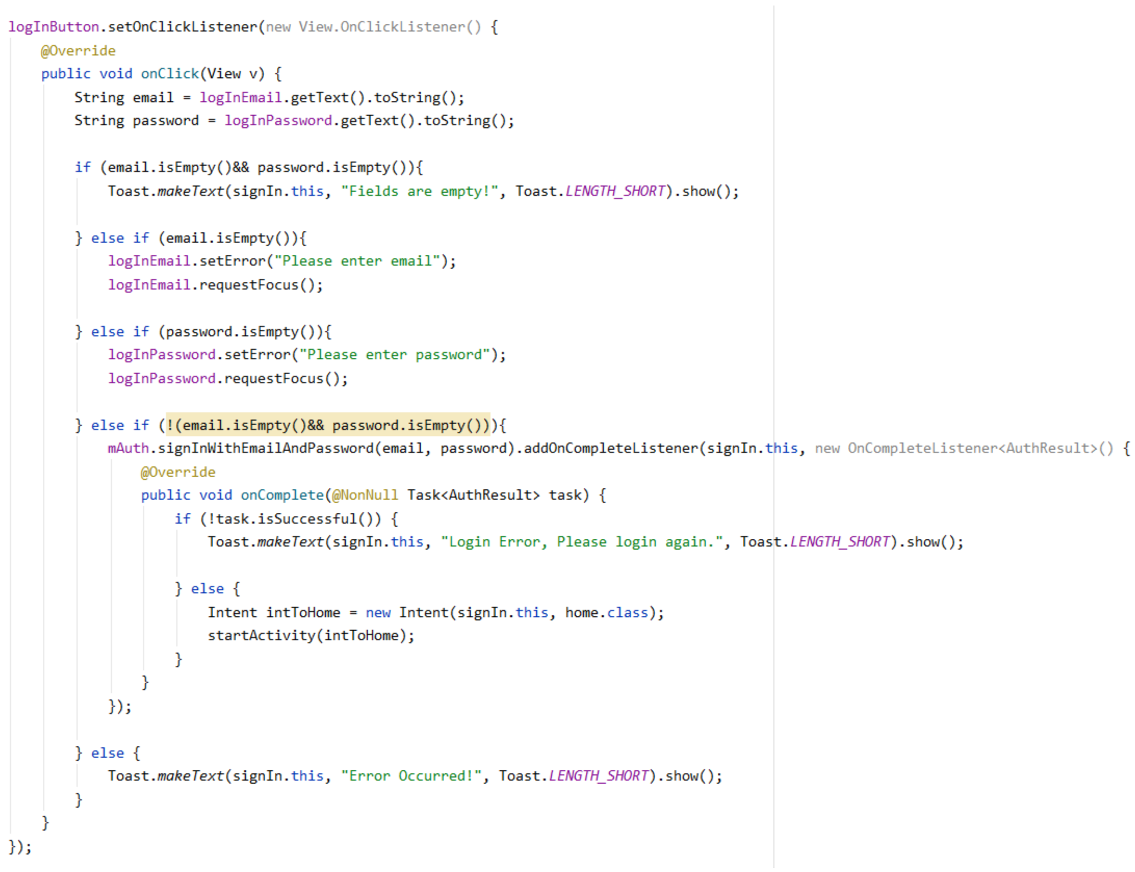
```
2xDxIptMgBWxZf58Vp8HILvfi6g2
4kJdSCP796PBHSgkmEwu6ni4OAD2
9Gna5GOA8tUVoEB8dtsIwwCP6VM2
LIVRZBPStaZJDZabwq6JUByvxMn2
ckJARJ2hUrfXFcuP98lSoc1CVtk2
uOrkQHdbznbCUEnJgnwm5ER7Doz2
vVsDK1RGaNN97IocW95vFfKNjG43
zlteZyIhBwTV0WqzAQ3dsUrMb7l1
```

```
email: "leeyingfoong@gmail.com"
name: "Ying Foong"
notiList: [{content: "Dear leeyingfo...]
password: "test1234"
phone: "0125380733"
role: "customer"
uid: "4kJdSCP796PBHSgkmEwu6ni4OAD2"
vehicleList: [{plateNo: "ABC1234", colo...]
```

Figure 3-6-2-2: User collection in Cloud Firestore

Figure above shows the “user” collection in Cloud Firestore. All details of user are stored here after they created the profile. Even if the user updates their profile, the details will be stored and updated here.

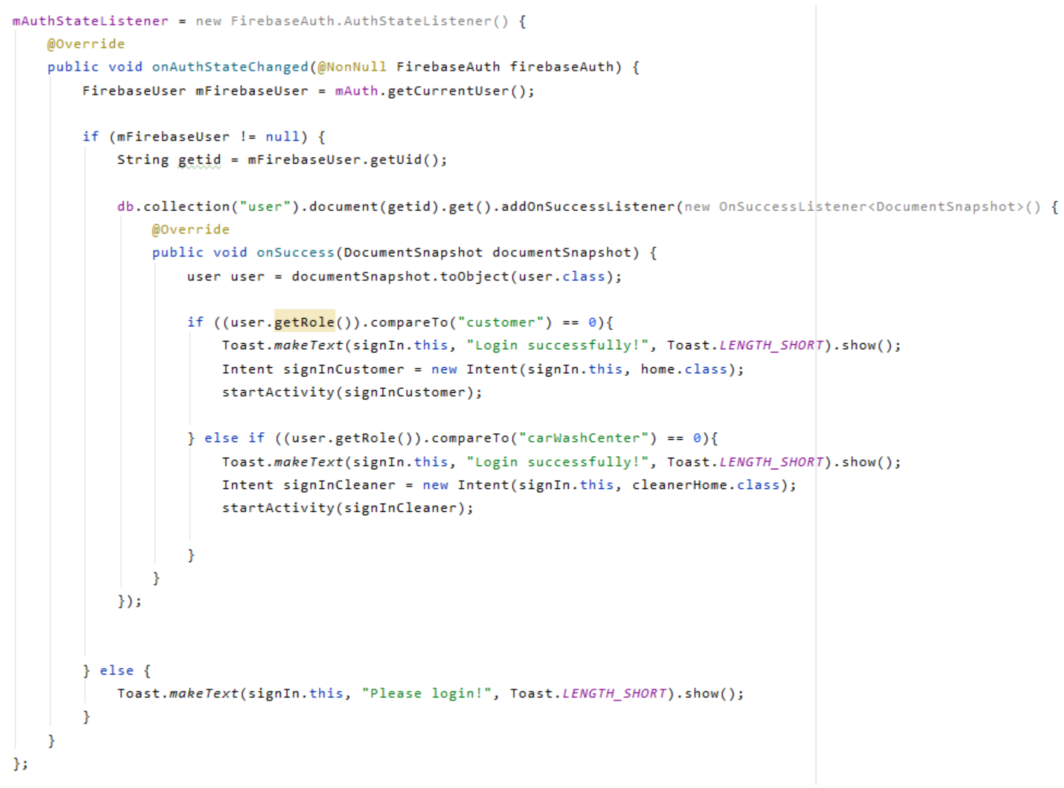
3.6.3 Sign In



```java
logInButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String email = logInEmail.getText().toString();
        String password = logInPassword.getText().toString();

        if (email.isEmpty()&& password.isEmpty()){
            Toast.makeText(signIn.this, "Fields are empty!", Toast.LENGTH_SHORT).show();

        } else if (email.isEmpty()){
            logInEmail.setError("Please enter email");
            logInEmail.requestFocus();

        } else if (password.isEmpty()){
            logInPassword.setError("Please enter password");
            logInPassword.requestFocus();

        } else if (!(email.isEmpty()&& password.isEmpty())){
            mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(signIn.this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (!task.isSuccessful()) {
                        Toast.makeText(signIn.this, "Login Error, Please login again.", Toast.LENGTH_SHORT).show();

                    } else {
                        Intent intToHome = new Intent(signIn.this, home.class);
                        startActivity(intToHome);
                    }
                }
            });

        } else {
            Toast.makeText(signIn.this, "Error Occurred!", Toast.LENGTH_SHORT).show();
        }
    }
});
```

Figure 3-6-3-1: Sign In

Figure 3-6-3-1 shows the coding of sign in function. When user clicks the sign in button, sign in function is triggered. System will retrieve the inputs from user and determine whether there are any missing fields. If there is missing fields, the system will display error message to inform user. Else system will call the function signInWithEmailAndPassword() function. signInWithEmailAndPassword() function is use to authenticate the user. If the user signs in successfully, user is directed to home page. Else, error message is displayed to user.



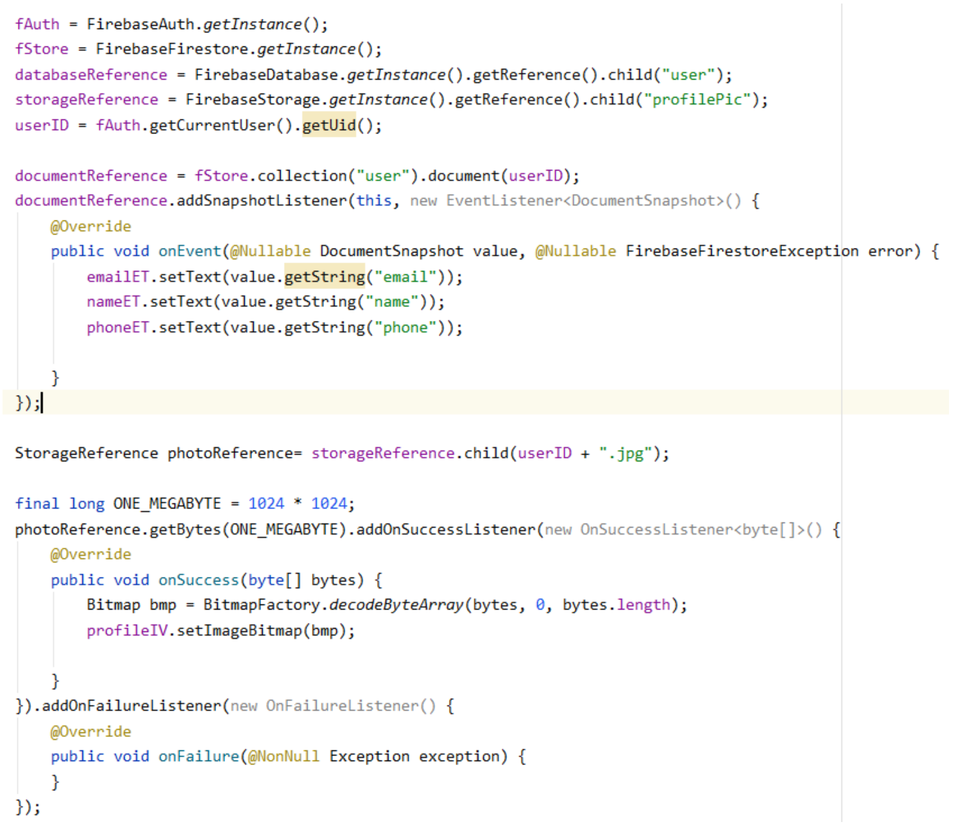
```
mAuthStateListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        FirebaseUser mFirebaseUser = mAuth.getCurrentUser();

        if (mFirebaseUser != null) {
            String getid = mFirebaseUser.getUid();

            db.collection("user").document(getid).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
                @Override
                public void onSuccess(DocumentSnapshot documentSnapshot) {
                    user user = documentSnapshot.toObject(user.class);

                    if ((user.getRole()).compareTo("customer") == 0){
                        Toast.makeText(signIn.this, "Login successfully!", Toast.LENGTH_SHORT).show();
                        Intent signInCustomer = new Intent(signIn.this, home.class);
                        startActivity(signInCustomer);

                    } else if ((user.getRole()).compareTo("carWashCenter") == 0){
                        Toast.makeText(signIn.this, "Login successfully!", Toast.LENGTH_SHORT).show();
                        Intent signInCleaner = new Intent(signIn.this, cleanerHome.class);
                        startActivity(signInCleaner);

                    }
                }
            });


        } else {
            Toast.makeText(signIn.this, "Please login!", Toast.LENGTH_SHORT).show();
        }
    }
};
```

Figure 3-6-3-2: Identity user role

Figure 3-6-3-2 shows the coding of identity user role function. When the user successfully authenticates identity, the system will get user’s details. If the user’s role is “customer”, user is directed to customer home page. If the user’s role is “carWashCenter”, user is directed to cleaner home page.

3.6.4 Manage Profile

3.6.4.1 Display Profile

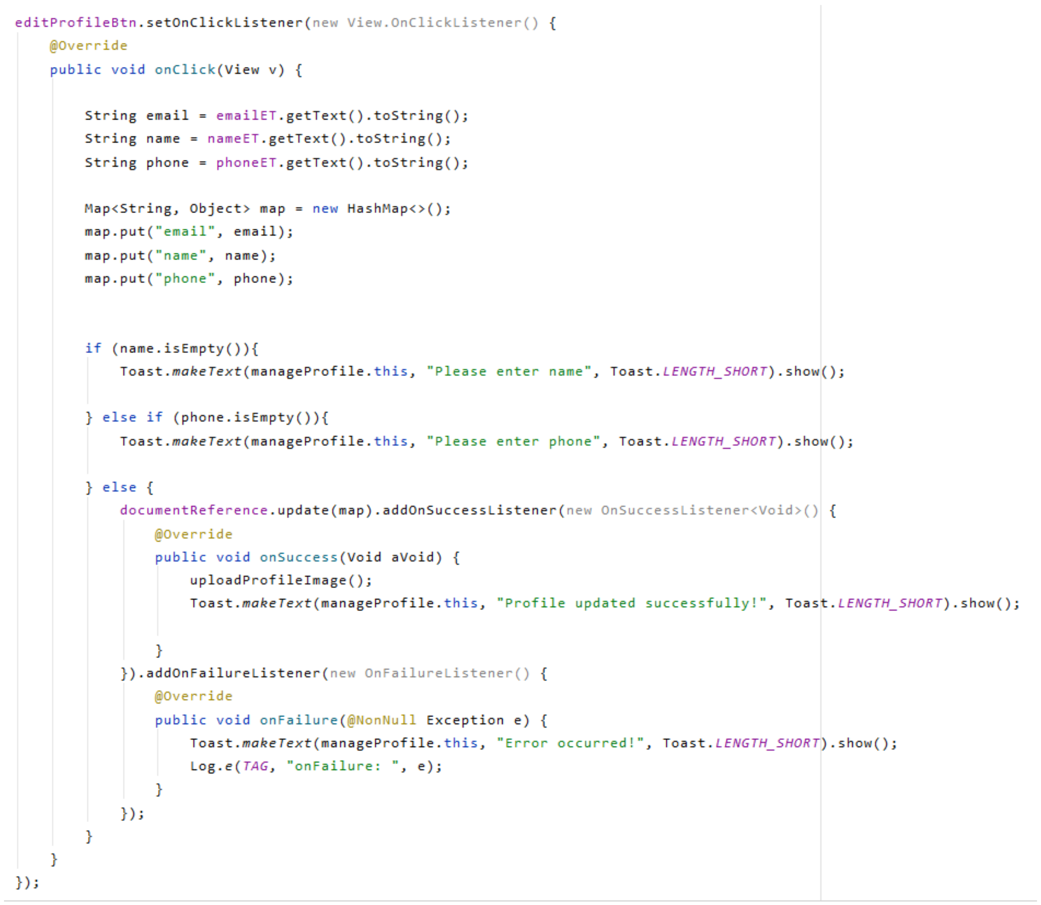


```
fAuth = FirebaseAuth.getInstance();
fStore = FirebaseFirestore.getInstance();
databaseReference = FirebaseDatabase.getInstance().getReference().child("user");
storageReference = FirebaseStorage.getInstance().getReference().child("profilePic");
userID = fAuth.getCurrentUser().getUid();

documentReference = fStore.collection("user").document(userID);
documentReference.addSnapshotListener(this, new EventListener<DocumentSnapshot>() {
    @Override
    public void onEvent(@Nullable DocumentSnapshot value, @Nullable FirebaseFirestoreException error) {
        emailET.setText(value.getString("email"));
        nameET.setText(value.getString("name"));
        phoneET.setText(value.getString("phone"));

    }
});

StorageReference photoReference= storageReference.child(userID + ".jpg");

final long ONE_MEGABYTE = 1024 * 1024;
photoReference.getBytes(ONE_MEGABYTE).addOnSuccessListener(new OnSuccessListener<byte[]>() {
    @Override
    public void onSuccess(byte[] bytes) {
        Bitmap bmp = BitmapFactory.decodeByteArray(bytes, 0, bytes.length);
        profileIV.setImageBitmap(bmp);

    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception exception) {
    }
});
```

Figure 3-6-4-1-1: Display user profile

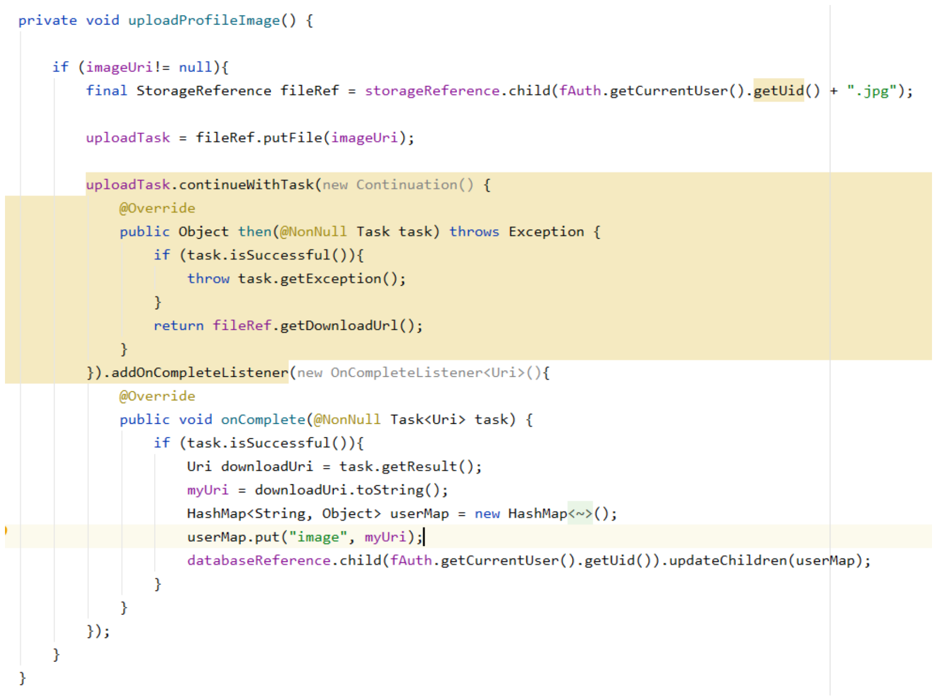
Figure 3-6-4-1-1 shows the coding of display user profile. Firstly, the user details are retrieved from the Firestore. Then, system display the user details in user profile page. Moreover, the profile picture is retrieved from the Firebase Storage using the user ID as the image path.

3.6.4.2 Update profile



```java
editProfileBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String email = emailET.getText().toString();
        String name = nameET.getText().toString();
        String phone = phoneET.getText().toString();

        Map<String, Object> map = new HashMap<>();
        map.put("email", email);
        map.put("name", name);
        map.put("phone", phone);


        if (name.isEmpty()){
            Toast.makeText(manageProfile.this, "Please enter name", Toast.LENGTH_SHORT).show();

        } else if (phone.isEmpty()){
            Toast.makeText(manageProfile.this, "Please enter phone", Toast.LENGTH_SHORT).show();

        } else {
            documentReference.update(map).addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    uploadProfileImage();
                    Toast.makeText(manageProfile.this, "Profile updated successfully!", Toast.LENGTH_SHORT).show();

                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(manageProfile.this, "Error occurred!", Toast.LENGTH_SHORT).show();
                    Log.e(TAG, "onFailure: ", e);
                }
            });
        }
    }
});
```

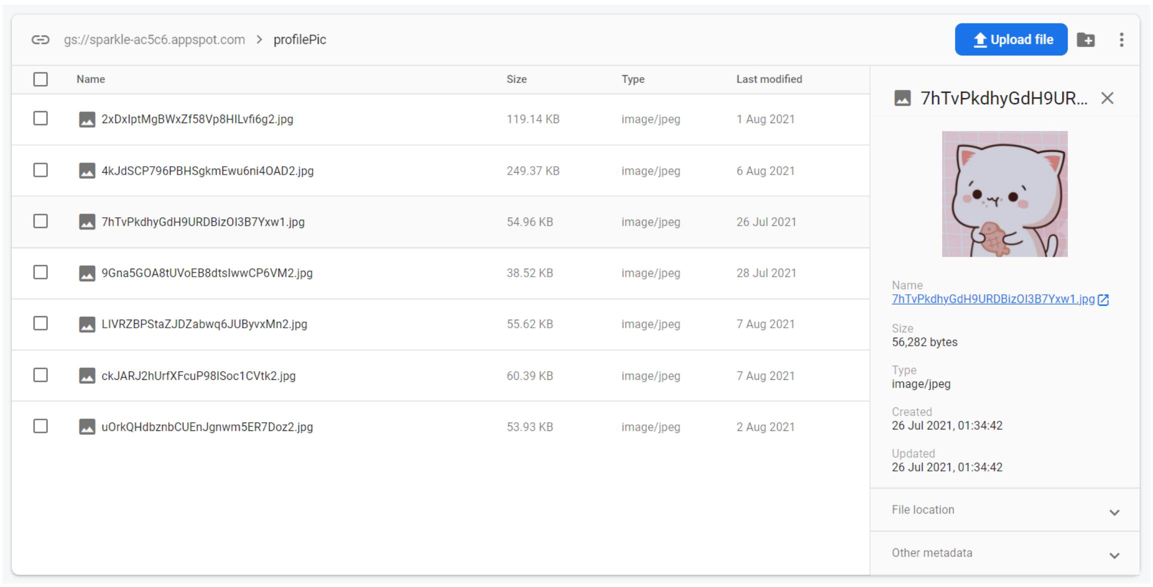
Figure 3-6-4-2-1: Update profile



```java
private void uploadProfileImage() {

    if (imageUri!= null){
        final StorageReference fileRef = storageReference.child(fAuth.getCurrentUser().getUid() + ".jpg");

        uploadTask = fileRef.putFile(imageUri);

        uploadTask.continueWithTask(new Continuation() {
            @Override
            public Object then(@NonNull Task task) throws Exception {
                if (task.isSuccessful()){
                    throw task.getException();
                }
                return fileRef.getDownloadUrl();
            }
        }).addOnCompleteListener(new OnCompleteListener<Uri>(){
            @Override
            public void onComplete(@NonNull Task<Uri> task) {
                if (task.isSuccessful()){
                    Uri downloadUri = task.getResult();
                    myUri = downloadUri.toString();
                    HashMap<String, Object> userMap = new HashMap<~>();
                    userMap.put("image", myUri);
                    databaseReference.child(fAuth.getCurrentUser().getUid()).updateChildren(userMap);
                }
            }
        });
    }
}
```

Figure 3-6-4-2-2: Upload Profile Picture

Figure 3-6-4-2-1 shows the coding of update profile function and Figure 3-6-4-2-2 shows the coding of upload profile picture. When user clicks the “update” button, the update profile function is triggered. The system will get all the user inputs and store into HashMap. If the user input is not complete, the system will display error message to notify the user. Else, if the input is completed, the user profile is updated successfully in Firestore. Besides, the profile picture is also updated in Firebase Storage. Figure 3-6-4-2-3 below shows the profile picture stored in Firebase Storage.



gs://sparkle-ac5c6.appspot.com > profilePic

Name	Size	Type	Last modified
2xDxIptMgBWxZf58Vp8HILvfi6g2.jpg	119.14 KB	image/jpeg	1 Aug 2021
4kJdSCP796PBHSgkmEwu6ni4OAD2.jpg	249.37 KB	image/jpeg	6 Aug 2021
7hTvPkdhyGdH9URDBizOI3B7Yxw1.jpg	54.96 KB	image/jpeg	26 Jul 2021
9Gna5GOA8tUVoEB8dtsIwwCP6VM2.jpg	38.52 KB	image/jpeg	28 Jul 2021
LIVRZBPStaZJDZabwq6JUByvxMn2.jpg	55.62 KB	image/jpeg	7 Aug 2021
ckJARJ2hUrfXFcuP98ISoc1CVtk2.jpg	60.39 KB	image/jpeg	7 Aug 2021
uOrkQHdbznbCUEnJgnwm5ER7Doz2.jpg	53.93 KB	image/jpeg	2 Aug 2021

Name
7hTvPkdhyGdH9URDBizOI3B7Yxw1.jpg

Size
56,282 bytes

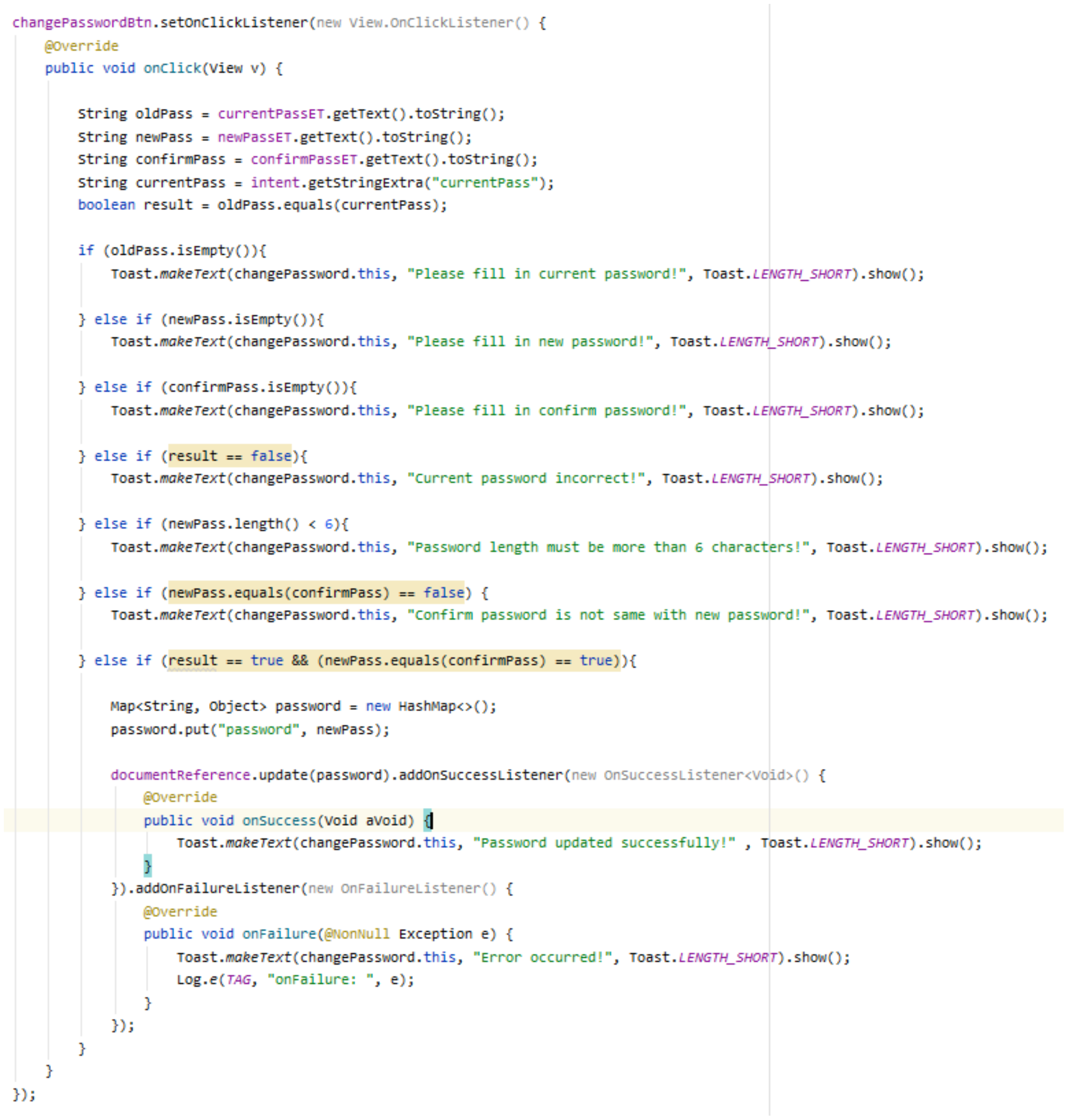
Type
image/jpeg

Created
26 Jul 2021, 01:34:42

Updated
26 Jul 2021, 01:34:42

Figure 3-6-4-2-3: Profile Picture in Firebase Storage

3.6.4.3 Change password

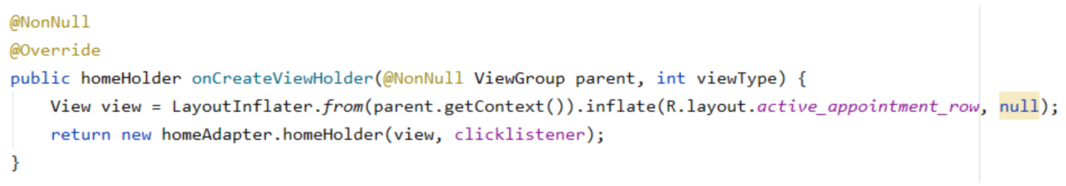


```java
changePasswordBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String oldPass = currentPassET.getText().toString();
        String newPass = newPassET.getText().toString();
        String confirmPass = confirmPassET.getText().toString();
        String currentPass = intent.getStringExtra("currentPass");
        boolean result = oldPass.equals(currentPass);

        if (oldPass.isEmpty()){
            Toast.makeText(changePassword.this, "Please fill in current password!", Toast.LENGTH_SHORT).show();

        } else if (newPass.isEmpty()){
            Toast.makeText(changePassword.this, "Please fill in new password!", Toast.LENGTH_SHORT).show();

        } else if (confirmPass.isEmpty()){
            Toast.makeText(changePassword.this, "Please fill in confirm password!", Toast.LENGTH_SHORT).show();

        } else if (result == false){
            Toast.makeText(changePassword.this, "Current password incorrect!", Toast.LENGTH_SHORT).show();

        } else if (newPass.length() < 6){
            Toast.makeText(changePassword.this, "Password length must be more than 6 characters!", Toast.LENGTH_SHORT).show();

        } else if (newPass.equals(confirmPass) == false) {
            Toast.makeText(changePassword.this, "Confirm password is not same with new password!", Toast.LENGTH_SHORT).show();

        } else if (result == true && (newPass.equals(confirmPass) == true)){

            Map<String, Object> password = new HashMap<>();
            password.put("password", newPass);

            documentReference.update(password).addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    Toast.makeText(changePassword.this, "Password updated successfully!" , Toast.LENGTH_SHORT).show();
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(changePassword.this, "Error occurred!", Toast.LENGTH_SHORT).show();
                    Log.e(TAG, "onFailure: ", e);
                }
            });
        }
    }
});
```

Figure 3-6-4-3-1: Change Password

Figure 3-6-4-3-1 shows the coding of change password function. When user clicks the “change password” button, the change password function is triggered. The system will get all the user inputs. If the user input is not complete, the system will display error message to notify the user. In addition, the old password need to be same as the password stored in the Cloud Firestore. New password and confirmation new password also need to match in order to change password successfully. Else, error message will be displayed to user. If the input is completed, and the password correct, the user password is changed successfully in Firestore.

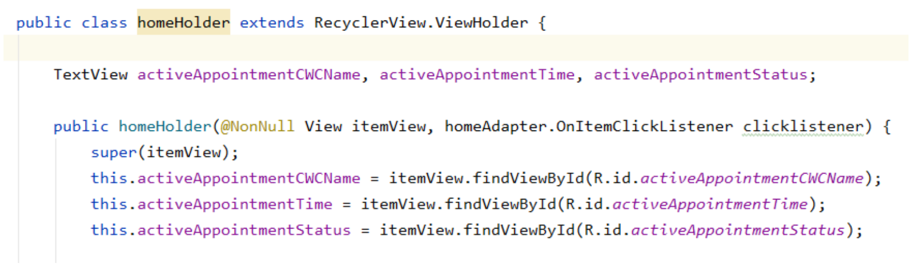
3.6.5 Display Active Appointment



```
@NonNull
@Override
public homeHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.active_appointment_row, null);
    return new homeAdapter.homeHolder(view, clicklistener);
}
```

Figure 3-6-5-1: Display Active Appointment Adapter (1)

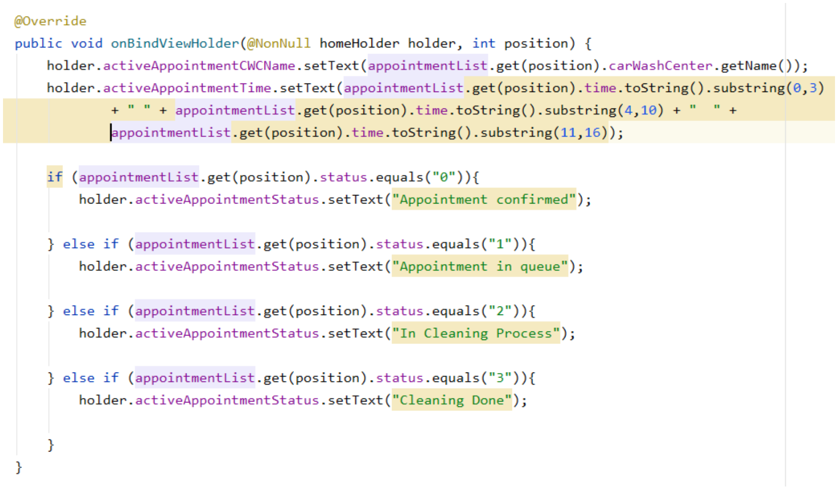
In home page, active appointment is displayed out for user to have a view. Firstly, system will retrieve all the active appointment from Firestore and put in an array list defined as "activeAppointmentList". The active appointment is displayed in home page in a recycler view. Recycler view is used to hold all the active appointment. The adapter is used to handle all active appointment component displayed in recycler view. The inflator is used to inflate the active appointment row design file into a view.



```
public class homeHolder extends RecyclerView.ViewHolder {

    TextView activeAppointmentCWCName, activeAppointmentTime, activeAppointmentStatus;

    public homeHolder(@NonNull View itemView, homeAdapter.OnItemClickListener clicklistener) {
        super(itemView);
        this.activeAppointmentCWCName = itemView.findViewById(R.id.activeAppointmentCWCName);
        this.activeAppointmentTime = itemView.findViewById(R.id.activeAppointmentTime);
        this.activeAppointmentStatus = itemView.findViewById(R.id.activeAppointmentStatus);
```

Figure 3-6-5-2: Display Active Appointment Adapter (2)

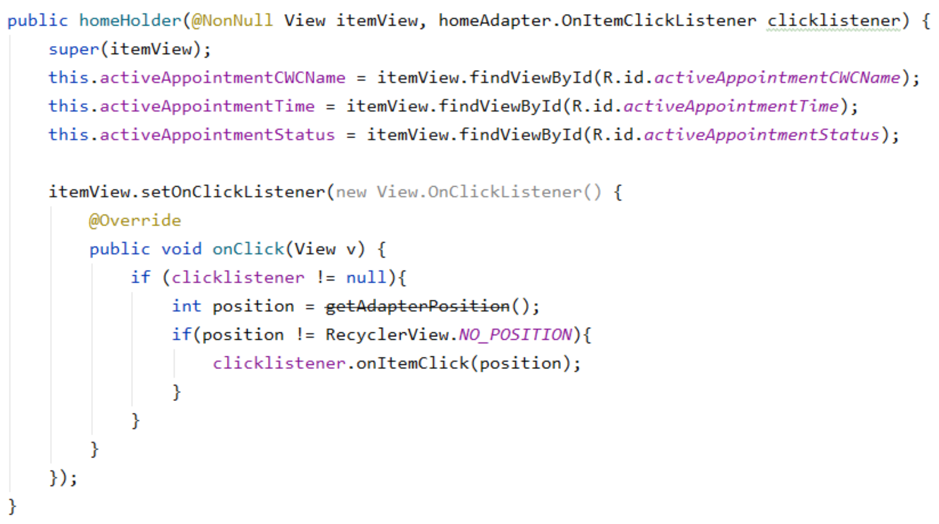
Firstly, all the components in the inflated view need to be defined and initiate. The component in the inflated view is defined using their view ID.



```
@Override
public void onBindViewHolder(@NonNull homeHolder holder, int position) {
    holder.activeAppointmentCWCName.setText(appointmentList.get(position).carWashCenter.getName());
    holder.activeAppointmentTime.setText(appointmentList.get(position).time.toString().substring(0,3)
            + " " + appointmentList.get(position).time.toString().substring(4,10) + "  " +
            appointmentList.get(position).time.toString().substring(11,16));

    if (appointmentList.get(position).status.equals("0")){
        holder.activeAppointmentStatus.setText("Appointment confirmed");

    } else if (appointmentList.get(position).status.equals("1")){
        holder.activeAppointmentStatus.setText("Appointment in queue");

    } else if (appointmentList.get(position).status.equals("2")){
        holder.activeAppointmentStatus.setText("In Cleaning Process");

    } else if (appointmentList.get(position).status.equals("3")){
        holder.activeAppointmentStatus.setText("Cleaning Done");

    }
}
```

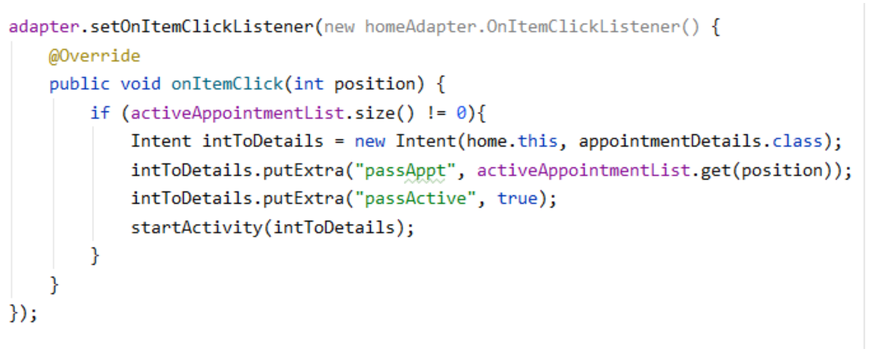
Figure 3-6-5-3: Display Active Appointment Adapter (3)

In onBindViewHolder(), each of the active appointment in the active appointment array list is retrieved and the details is displayed. The array list is storing appointment object, hence, the getter and setter function is used to retrieved the data from array list.



```java
public homeHolder(@NonNull View itemView, homeAdapter.OnItemClickListener clicklistener) {
    super(itemView);
    this.activeAppointmentCWCName = itemView.findViewById(R.id.activeAppointmentCWCName);
    this.activeAppointmentTime = itemView.findViewById(R.id.activeAppointmentTime);
    this.activeAppointmentStatus = itemView.findViewById(R.id.activeAppointmentStatus);

    itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (clicklistener != null){
                int position = ~~getAdapterPosition~~();
                if(position != RecyclerView.NO_POSITION){
                    clicklistener.onItemClick(position);
                }
            }
        }
    });
}
```

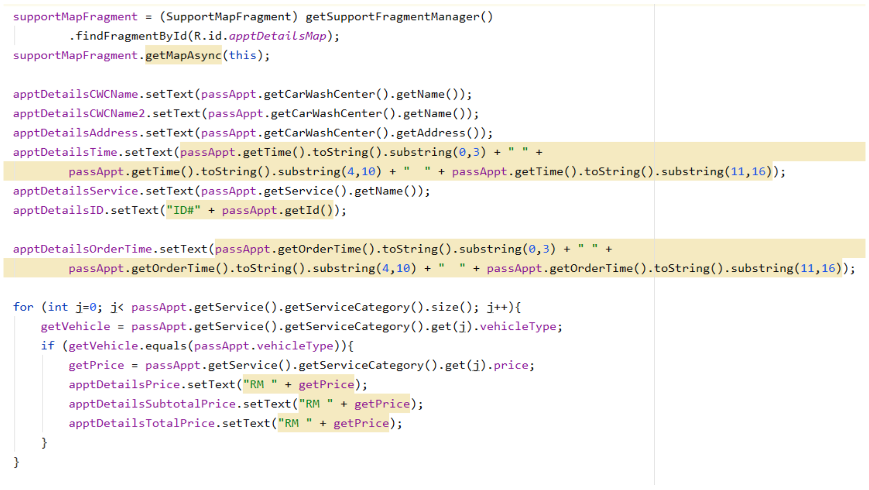
Figure 3-6-5-4: Display Active Appointment Adapter (4)



```java
adapter.setOnItemClickListener(new homeAdapter.OnItemClickListener() {
    @Override
    public void onItemClick(int position) {
        if (activeAppointmentList.size() != 0){
            Intent intToDetails = new Intent(home.this, appointmentDetails.class);
            intToDetails.putExtra("passAppt", activeAppointmentList.get(position));
            intToDetails.putExtra("passActive", true);
            startActivity(intToDetails);
        }
    }
});
```

Figure 3-6-5-5: Display Active Appointment Adapter (5)

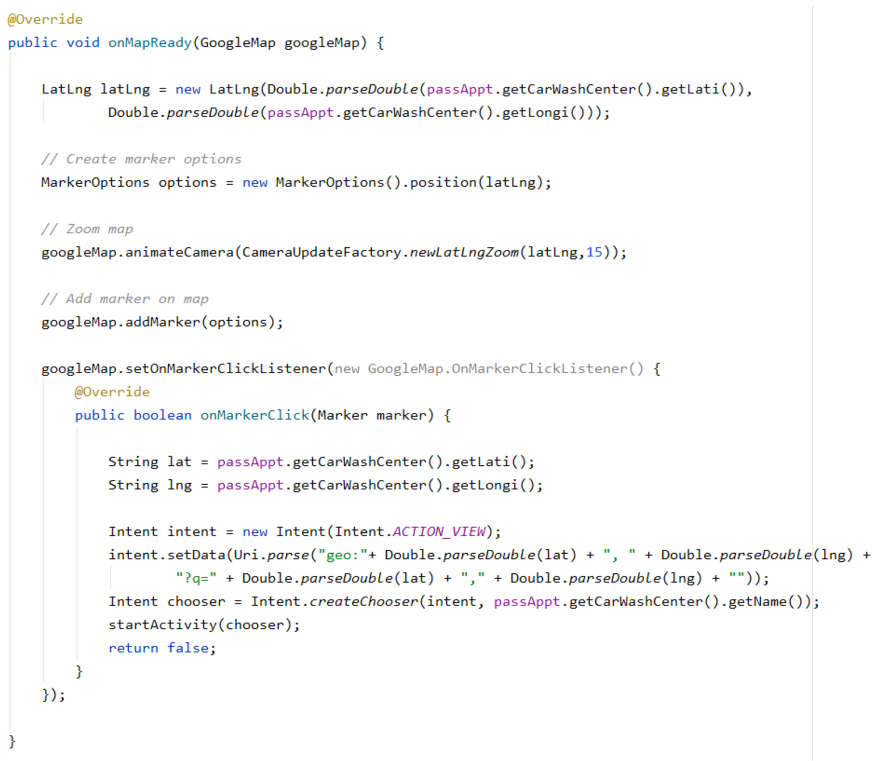
In Figure 3-6-5-4, onClick method is set in the holder view. The on click listener is used to perform action when user clicked on the item in the recycler view. In this function, when user clicks on the item, the function will return the position of the item in the active appointment array list. In Figure 3-6-5-5, the position is get in this function and the active appointment in the active appointment array list is retrieved and passed to the appointment details page that display all the details of the active appointment.



```java
supportMapFragment = (SupportMapFragment) getSupportFragmentManager()
        .findFragmentById(R.id.apptDetailsMap);
supportMapFragment.getMapAsync(this);

apptDetailsCWCName.setText(passAppt.getCarWashCenter().getName());
apptDetailsCWCName2.setText(passAppt.getCarWashCenter().getName());
apptDetailsAddress.setText(passAppt.getCarWashCenter().getAddress());
apptDetailsTime.setText(passAppt.getTime().toString().substring(0,3) + " " +
        passAppt.getTime().toString().substring(4,10) + "  " + passAppt.getTime().toString().substring(11,16));
apptDetailsService.setText(passAppt.getService().getName());
apptDetailsID.setText("ID#" + passAppt.getId());

apptDetailsOrderTime.setText(passAppt.getOrderTime().toString().substring(0,3) + " " +
        passAppt.getOrderTime().toString().substring(4,10) + "  " + passAppt.getOrderTime().toString().substring(11,16));

for (int j=0; j< passAppt.getService().getServiceCategory().size(); j++){
    getVehicle = passAppt.getService().getServiceCategory().get(j).vehicleType;
    if (getVehicle.equals(passAppt.vehicleType)){
        getPrice = passAppt.getService().getServiceCategory().get(j).price;
        apptDetailsPrice.setText("RM " + getPrice);
        apptDetailsSubtotalPrice.setText("RM " + getPrice);
        apptDetailsTotalPrice.setText("RM " + getPrice);
    }
}
```

Figure 3-6-5-6: Display Appointment Details

Figure 3-6-5-6 shows the coding of display appointment details. The appointment details are passed from the active appointment function. The details are retrieved using the getter and setter function defined in the class. Then, the data is set in the text view in appointment details page.



```java
@Override
public void onMapReady(GoogleMap googleMap) {

    LatLng latLng = new LatLng(Double.parseDouble(passAppt.getCarWashCenter().getLati()),
            Double.parseDouble(passAppt.getCarWashCenter().getLongi()));

    // Create marker options
    MarkerOptions options = new MarkerOptions().position(latLng);

    // Zoom map
    googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,15));

    // Add marker on map
    googleMap.addMarker(options);

    googleMap.setOnMarkerClickListener(new GoogleMap.OnMarkerClickListener() {
        @Override
        public boolean onMarkerClick(Marker marker) {

            String lat = passAppt.getCarWashCenter().getLati();
            String lng = passAppt.getCarWashCenter().getLongi();

            Intent intent = new Intent(Intent.ACTION_VIEW);
            intent.setData(Uri.parse("geo:"+ Double.parseDouble(lat) + ", " + Double.parseDouble(lng) +
                    "?q=" + Double.parseDouble(lat) + "," + Double.parseDouble(lng) + ""));
            Intent chooser = Intent.createChooser(intent, passAppt.getCarWashCenter().getName());
            startActivity(chooser);
            return false;
        }
    });

}
```

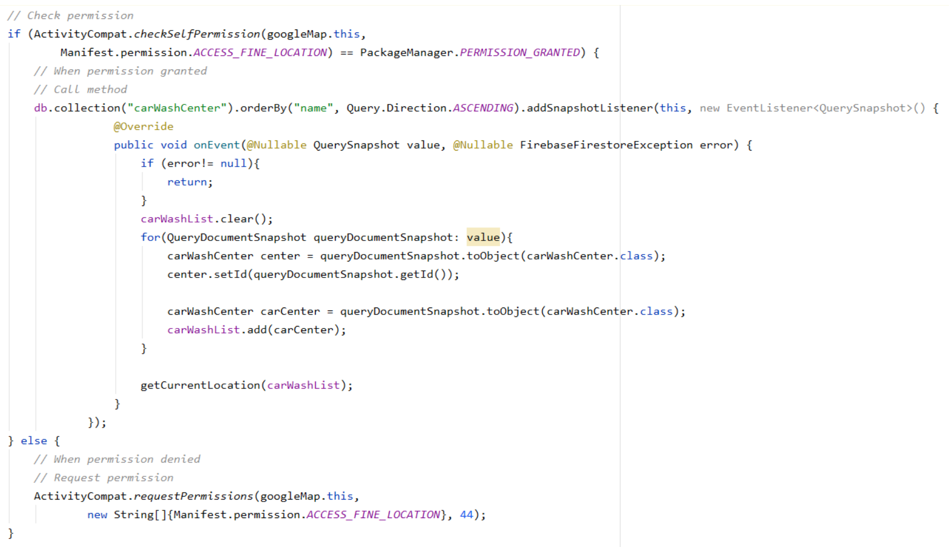
Figure 3-6-5-7: Pin marker in Google Map in Appointment Details

Figure 3-6-5-7 shows the coding of pinning a marker in Google Map function. Firstly, the latitude and longitude stored in the appointment is set as the location of the marker.

Then, the marker is added into the Google Map. When user click on the marker, the system will direct user to the Google Map application and provides direction to user.

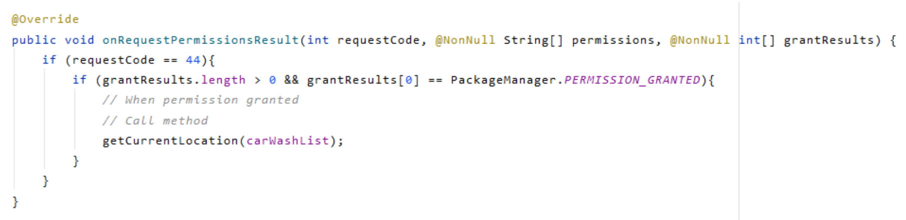
3.6.6 Make Appointment

3.6.6.1 View Nearby Car Wash Center in Map



```java
// Check permission
if (ActivityCompat.checkSelfPermission(googleMap.this,
        Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
    // When permission granted
    // Call method
    db.collection("carWashCenter").orderBy("name", Query.Direction.ASCENDING).addSnapshotListener(this, new EventListener<QuerySnapshot>() {
                @Override
                public void onEvent(@Nullable QuerySnapshot value, @Nullable FirebaseFirestoreException error) {
                    if (error!= null){
                        return;
                    }
                    carWashList.clear();
                    for(QueryDocumentSnapshot queryDocumentSnapshot: value){
                        carWashCenter center = queryDocumentSnapshot.toObject(carWashCenter.class);
                        center.setId(queryDocumentSnapshot.getId());

                        carWashCenter carCenter = queryDocumentSnapshot.toObject(carWashCenter.class);
                        carWashList.add(carCenter);
                    }

                    getCurrentLocation(carWashList);
                }
            });
} else {
    // When permission denied
    // Request permission
    ActivityCompat.requestPermissions(googleMap.this,
            new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 44);
}
```

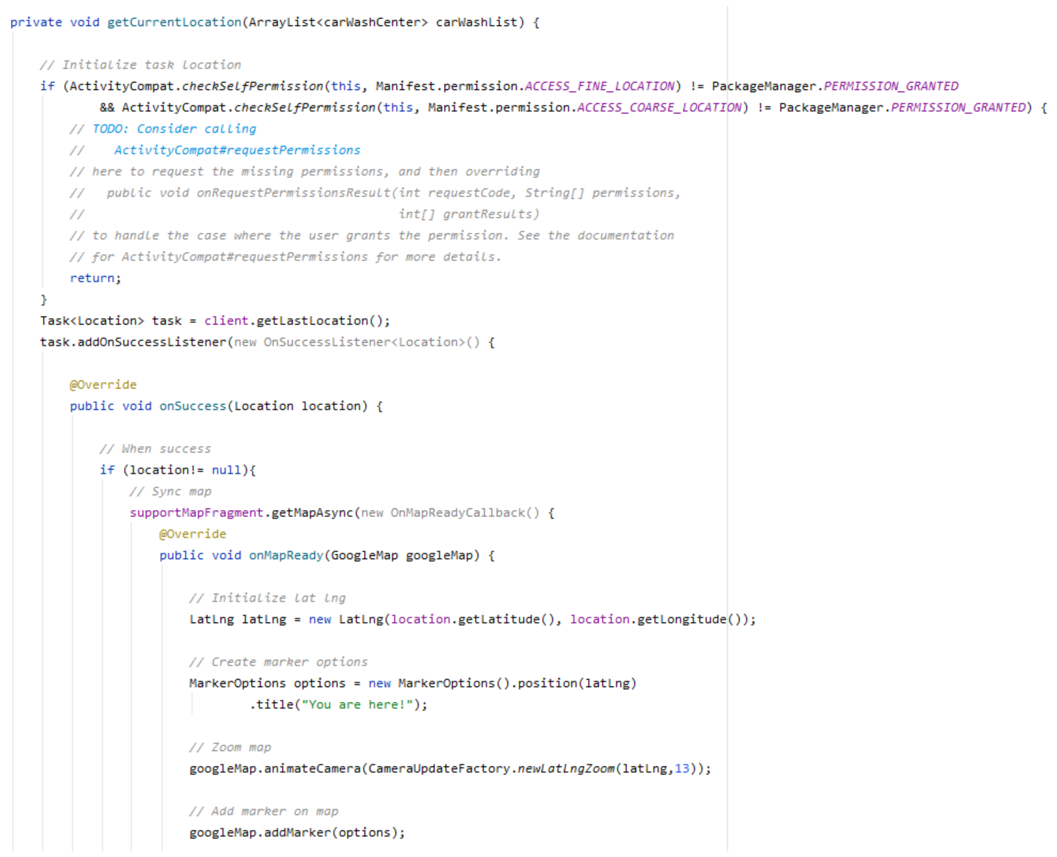
Figure 3-6-6-1-1: Grant for user permission to access location (1)



```java
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    if (requestCode == 44){
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
            // When permission granted
            // Call method
            getCurrentLocation(carWashList);
        }
    }
}
```

Figure 3-6-6-1-2: Grant for user permission to access location (1)

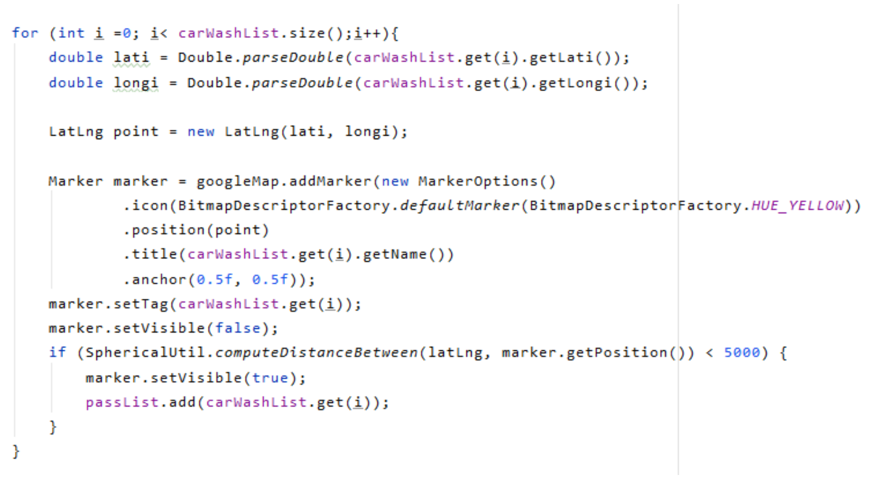
Figure 3-6-6-1-1 and 3-6-6-1-2 shows the coding of granting user permission to access the user’s location. If user gives the permission to let the application access to the location, the system then retrieves all the car wash center and put into an array list.



```
private void getCurrentLocation(ArrayList<carWashCenter> carWashList) {

    // Initialize task location
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
            && ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        // TODO: Consider calling
        //    ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        //   public void onRequestPermissionsResult(int requestCode, String[] permissions,
        //                                          int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for ActivityCompat#requestPermissions for more details.
        return;
    }
    Task<Location> task = client.getLastLocation();
    task.addOnSuccessListener(new OnSuccessListener<Location>() {

        @Override
        public void onSuccess(Location location) {

            // When success
            if (location!= null){
                // Sync map
                supportMapFragment.getMapAsync(new OnMapReadyCallback() {
                    @Override
                    public void onMapReady(GoogleMap googleMap) {

                        // Initialize lat lng
                        LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());

                        // Create marker options
                        MarkerOptions options = new MarkerOptions().position(latLng)
                                .title("You are here!");

                        // Zoom map
                        googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,13));

                        // Add marker on map
                        googleMap.addMarker(options);
```

Figure 3-6-6-1-3: Get Current Location

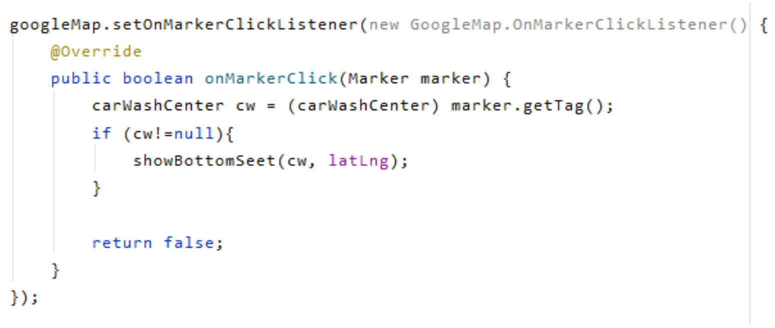
Figure 3-6-6-1-3 shows the coding of getting current location function. Firstly, the system will check whether the permission to get the location is granted. If grant the permission successfully, the system will sync the Google Map. Then, the latitude and longitude is initialized. A marker is created which indicates the current location of user. The marker of the current location of user is then pinned to the Google Map.



```
for (int i =0; i< carWashList.size();i++){
    double lati = Double.parseDouble(carWashList.get(i).getLati());
    double longi = Double.parseDouble(carWashList.get(i).getLongi());

    LatLng point = new LatLng(lati, longi);

    Marker marker = googleMap.addMarker(new MarkerOptions()
            .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_YELLOW))
            .position(point)
            .title(carWashList.get(i).getName())
            .anchor(0.5f, 0.5f));
    marker.setTag(carWashList.get(i));
    marker.setVisible(false);
    if (SphericalUtil.computeDistanceBetween(latLng, marker.getPosition()) < 5000) {
        marker.setVisible(true);
        passList.add(carWashList.get(i));
    }
}
```

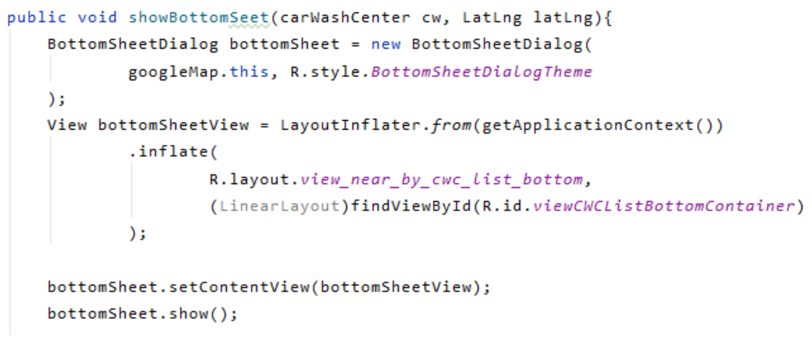
Figure 3-6-6-1-4: Add Marker for Car Wash Center Nearby

Figure 3-6-6-1-4 shows the coding of adding marker for car wash center nearby. To check whether the car wash center in 5 kilometers away from user's location, for loop is used. If the distance between the car wash center and user's current location is in 5 kilometers, the marker is added to the Google Map.



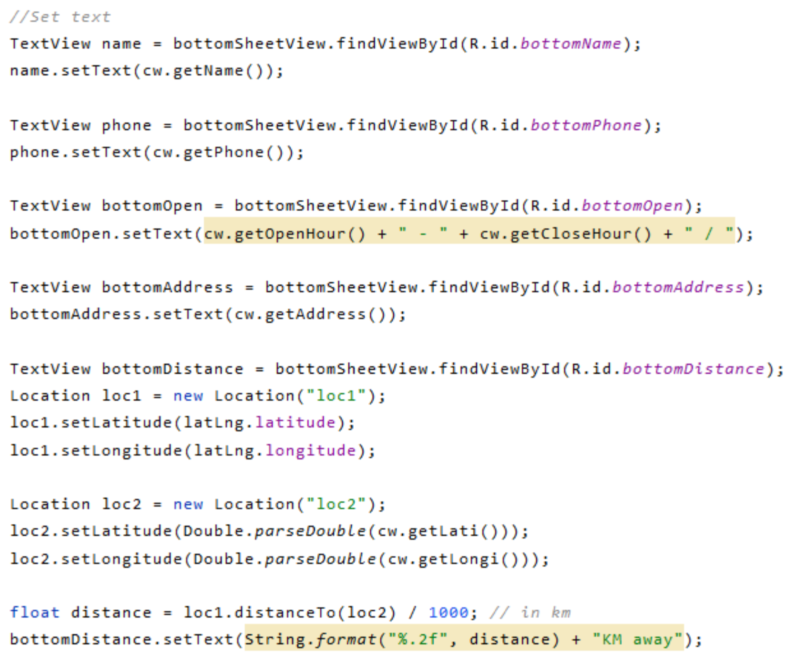
```
googleMap.setOnMarkerClickListener(new GoogleMap.OnMarkerClickListener() {
    @Override
    public boolean onMarkerClick(Marker marker) {
        carWashCenter cw = (carWashCenter) marker.getTag();
        if (cw!=null){
            showBottomSeet(cw, latLng);
        }

        return false;
    }
});
```

Figure 3-6-6-1-5: Display Details of Car Wash Center (1)



```
public void showBottomSeet(carWashCenter cw, LatLng latLng){
    BottomSheetDialog bottomSheet = new BottomSheetDialog(
            googleMap.this, R.style.BottomSheetDialogTheme
    );
    View bottomSheetView = LayoutInflater.from(getApplicationContext())
            .inflate(
                    R.layout.view_near_by_cwc_list_bottom,
                    (LinearLayout)findViewById(R.id.viewCWCListBottomContainer)
            );

    bottomSheet.setContentView(bottomSheetView);
    bottomSheet.show();
```

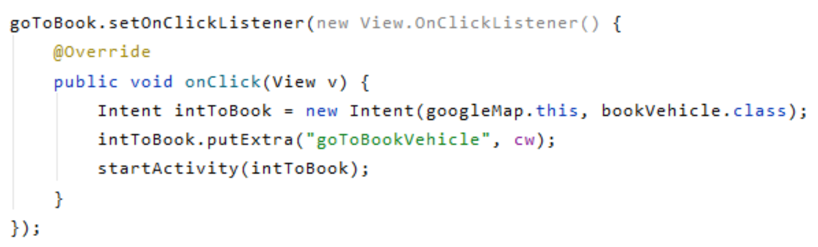
Figure 3-6-6-1-6: Display Details of Car Wash Center (2)



```
//Set text
TextView name = bottomSheetView.findViewById(R.id.bottomName);
name.setText(cw.getName());

TextView phone = bottomSheetView.findViewById(R.id.bottomPhone);
phone.setText(cw.getPhone());

TextView bottomOpen = bottomSheetView.findViewById(R.id.bottomOpen);
bottomOpen.setText(cw.getOpenHour() + " - " + cw.getCloseHour() + " / ");

TextView bottomAddress = bottomSheetView.findViewById(R.id.bottomAddress);
bottomAddress.setText(cw.getAddress());

TextView bottomDistance = bottomSheetView.findViewById(R.id.bottomDistance);
Location loc1 = new Location("loc1");
loc1.setLatitude(latLng.latitude);
loc1.setLongitude(latLng.longitude);

Location loc2 = new Location("loc2");
loc2.setLatitude(Double.parseDouble(cw.getLati()));
loc2.setLongitude(Double.parseDouble(cw.getLongi()));

float distance = loc1.distanceTo(loc2) / 1000; // in km
bottomDistance.setText(String.format("%.2f", distance) + "KM away");
```

Figure 3-6-6-1-7: Display Details of Car Wash Center (3)

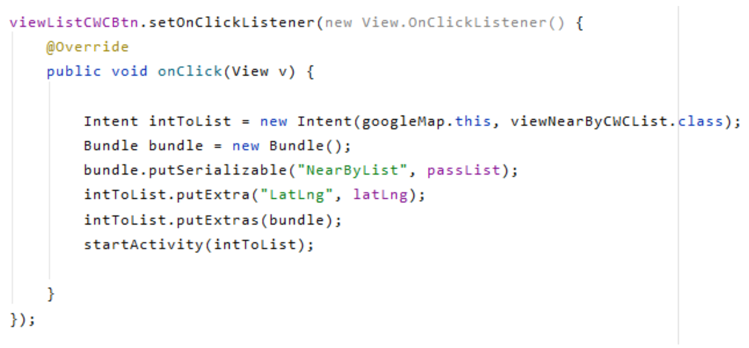
When user clicks on the marker, showBottomSheet() function is called. This function is to pop out a bottom view and display the details of car wash center. The inflator is used to inflate the bottom sheet design file into a view. The marker contains the car wash center and store the car wash center as object. Then, the car wash center is passed into showBottomSheet() function. The details of the car wash center are retrieved using the defined getter function in car wash center class.



```java
goToBook.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intToBook = new Intent(googleMap.this, bookVehicle.class);
        intToBook.putExtra("goToBookVehicle", cw);
        startActivity(intToBook);
    }
});
```

Figure 3-6-6-1-8: Select Car Wash Center

When user clicks on the “book” button, the car wash center selected is passed to the next page to continue the booking process. The car wash center is passed using intent.

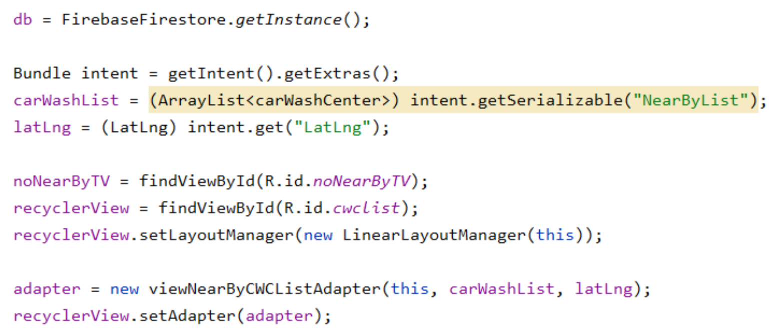


```java
viewListCWCBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Intent intToList = new Intent(googleMap.this, viewNearByCWCList.class);
        Bundle bundle = new Bundle();
        bundle.putSerializable("NearByList", passList);
        intToList.putExtra("LatLng", latLng);
        intToList.putExtras(bundle);
        startActivity(intToList);

    }
});
```

Figure 3-6-6-1-9: Go to View Car Wash Center Nearby in List Activity

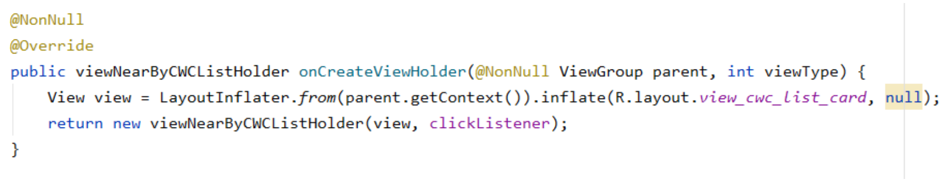
Figure 3-6-6-1-9 shows the coding of go to view car wash center nearby in list page when the “view list” button is clicked. When user clicked “view list” button, user is direct to view car wash center nearby in list page that display all the car wash center and its details in a list. The car wash center nearby is put in an array list and pass to next activity using intent.

3.6.6.2 View Nearby Car Wash Center in List



```
db = FirebaseFirestore.getInstance();

Bundle intent = getIntent().getExtras();
carWashList = (ArrayList<carWashCenter>) intent.getSerializable("NearByList");
latLng = (LatLng) intent.get("LatLng");

noNearByTV = findViewById(R.id.noNearByTV);
recyclerView = findViewById(R.id.cwclist);
recyclerView.setLayoutManager(new LinearLayoutManager(this));

adapter = new viewNearByCWCListAdapter(this, carWashList, latLng);
recyclerView.setAdapter(adapter);
```

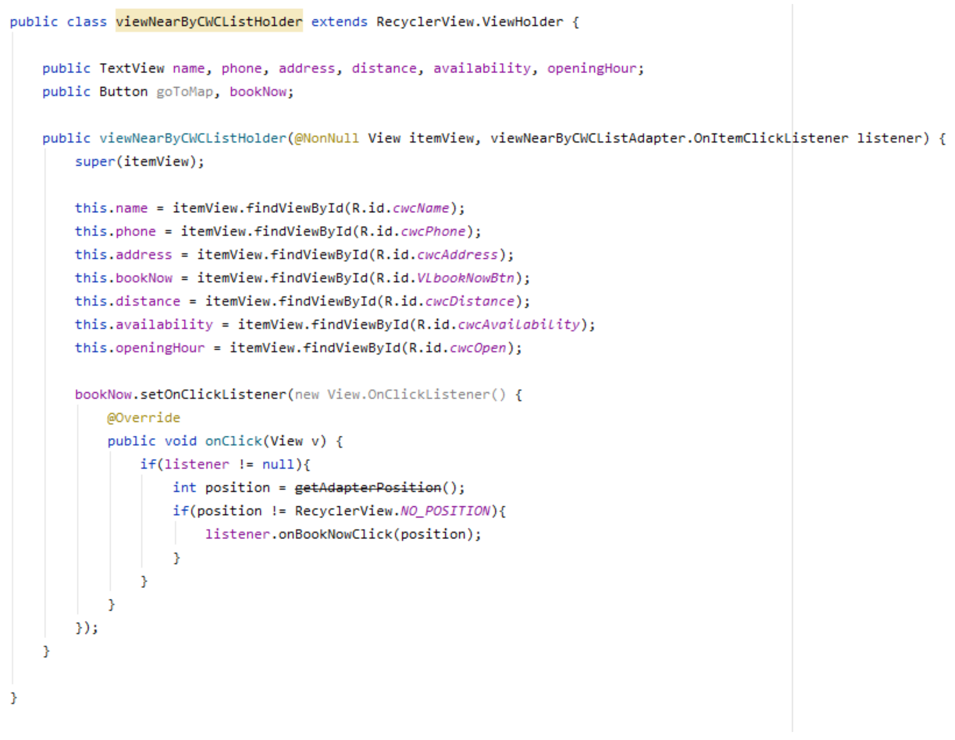
Figure 3-6-6-2-1: Display Car Wash Center Nearby in List (1)



```
@NonNull
@Override
public viewNearByCWCListHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.view_cwc_list_card, null);
    return new viewNearByCWCListHolder(view, clickListener);
}
```

Figure 3-6-6-2-2: Display Car Wash Center Nearby in List (2)

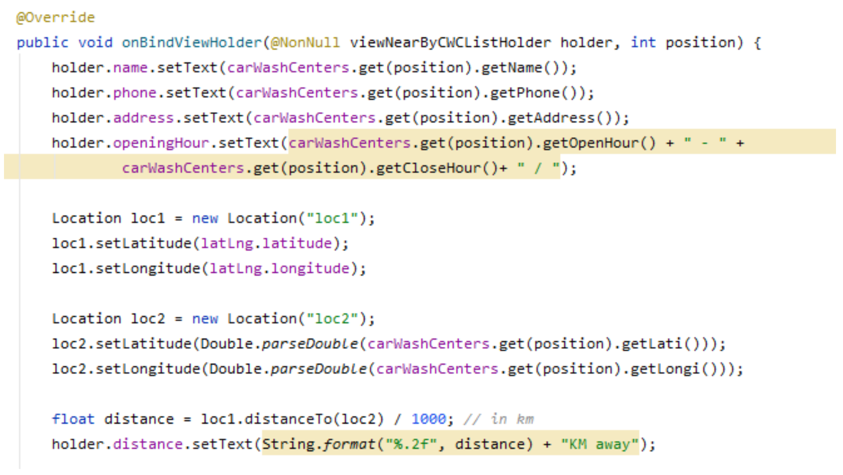
In this activity, system get the car wash center nearby in array list by using getIntent() function. The nearby car wash center is display to user using recycler view. Recycler view is used to hold all the car wash center. The adapter is used to handle each car wash center displayed in recycler view. In figure 3-6-6-2-2, the inflator is used to inflate the car wash center row design file into a view.



```
public class viewNearByCWCListHolder extends RecyclerView.ViewHolder {

    public TextView name, phone, address, distance, availability, openingHour;
    public Button goToMap, bookNow;

    public viewNearByCWCListHolder(@NonNull View itemView, viewNearByCWCListAdapter.OnItemClickListener listener) {
        super(itemView);

        this.name = itemView.findViewById(R.id.cwcName);
        this.phone = itemView.findViewById(R.id.cwcPhone);
        this.address = itemView.findViewById(R.id.cwcAddress);
        this.bookNow = itemView.findViewById(R.id.VLbookNowBtn);
        this.distance = itemView.findViewById(R.id.cwcDistance);
        this.availability = itemView.findViewById(R.id.cwcAvailability);
        this.openingHour = itemView.findViewById(R.id.cwcOpen);

        bookNow.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(listener != null){
                    int position = ~~getAdapterPosition~~();
                    if(position != RecyclerView.NO_POSITION){
                        listener.onBookNowClick(position);
                    }
                }
            }
        });
    }

}
```

Figure 3-6-6-2-3: Display Car Wash Center Nearby in List (3)

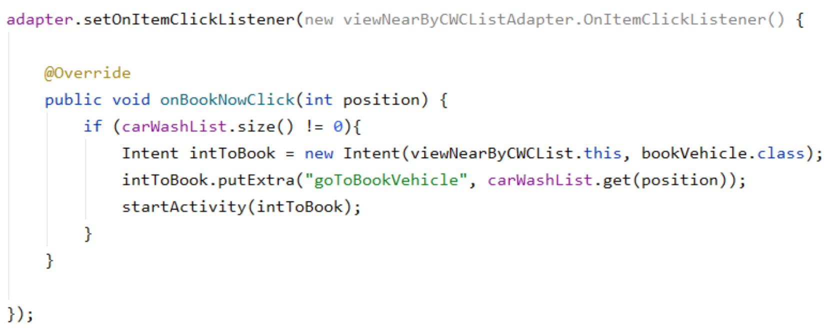
Firstly, all the components in the inflated view need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-6-2-3, component such as the name, phone, address, opening hour and distance is declared here.



```
@Override
public void onBindViewHolder(@NonNull viewNearByCWCListHolder holder, int position) {
    holder.name.setText(carWashCenters.get(position).getName());
    holder.phone.setText(carWashCenters.get(position).getPhone());
    holder.address.setText(carWashCenters.get(position).getAddress());
    holder.openingHour.setText(carWashCenters.get(position).getOpenHour() + " - " +
            carWashCenters.get(position).getCloseHour()+ " / ");

    Location loc1 = new Location("loc1");
    loc1.setLatitude(latLng.latitude);
    loc1.setLongitude(latLng.longitude);

    Location loc2 = new Location("loc2");
    loc2.setLatitude(Double.parseDouble(carWashCenters.get(position).getLati()));
    loc2.setLongitude(Double.parseDouble(carWashCenters.get(position).getLongi()));

    float distance = loc1.distanceTo(loc2) / 1000; // in km
    holder.distance.setText(String.format("%.2f", distance) + "KM away");
```

Figure 3-6-6-2-4: Display Car Wash Center Nearby in List (4)

Each of the car wash center in the car wash center array list is retrieved and the details is displayed. The holder is used to hold each of the component value. The array list is storing car wash center object, hence, the getter and setter function is used to retrieved the data from the array list.

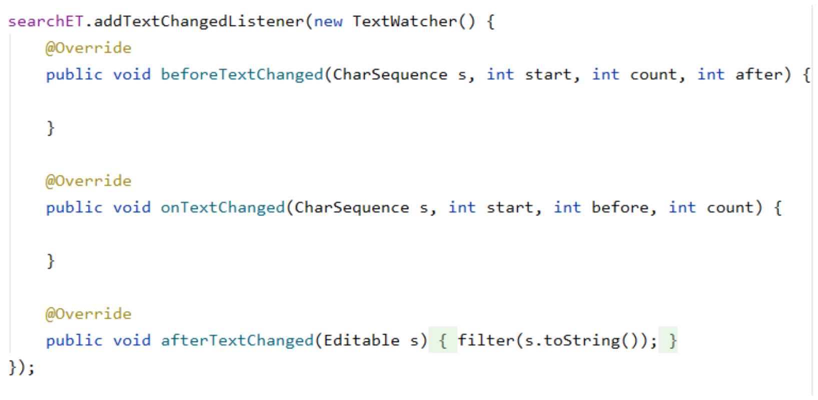


```
adapter.setOnItemClickListener(new viewNearByCWCListAdapter.OnItemClickListener() {

    @Override
    public void onBookNowClick(int position) {
        if (carWashList.size() != 0){
            Intent intToBook = new Intent(viewNearByCWCList.this, bookVehicle.class);
            intToBook.putExtra("goToBookVehicle", carWashList.get(position));
            startActivity(intToBook);
        }
    }

});
```

Figure 3-6-6-2-5: Display Car Wash Center Nearby in List (5)

When user clicks on the “book” button of the item in recycler view, the car wash center selected is passed to the next page to continue the booking process. The car wash center is get from the car wash center array list and passed using intent.

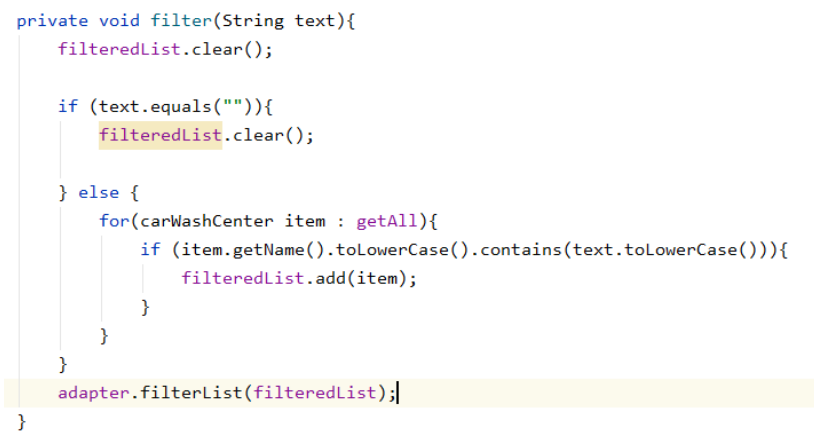
3.6.6.3 Search Car Wash Center



```
searchET.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {

    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {

    }

    @Override
    public void afterTextChanged(Editable s) { filter(s.toString()); }
});
```

Figure 3-6-6-3-1: Search Car Wash Center Text Listener

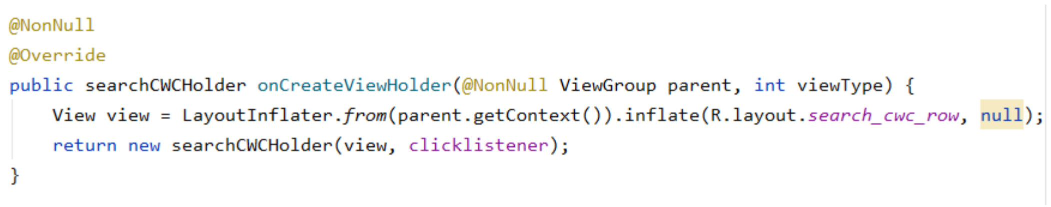
Figure 3-6-6-3-1 shows the coding of search car wash center text listener. When user enter car wash center name, the afterTextChanged() function will listen to the text changes Then, it call the filter() function and change the input to string format and pass to the function.



```
private void filter(String text){
    filteredList.clear();

    if (text.equals("")){
        filteredList.clear();

    } else {
        for(carWashCenter item : getAll){
            if (item.getName().toLowerCase().contains(text.toLowerCase())){
                filteredList.add(item);
            }
        }
    }
    adapter.filterList(filteredList);
}
```

Figure 3-6-6-3-2: Filter Car Wash Center By Name

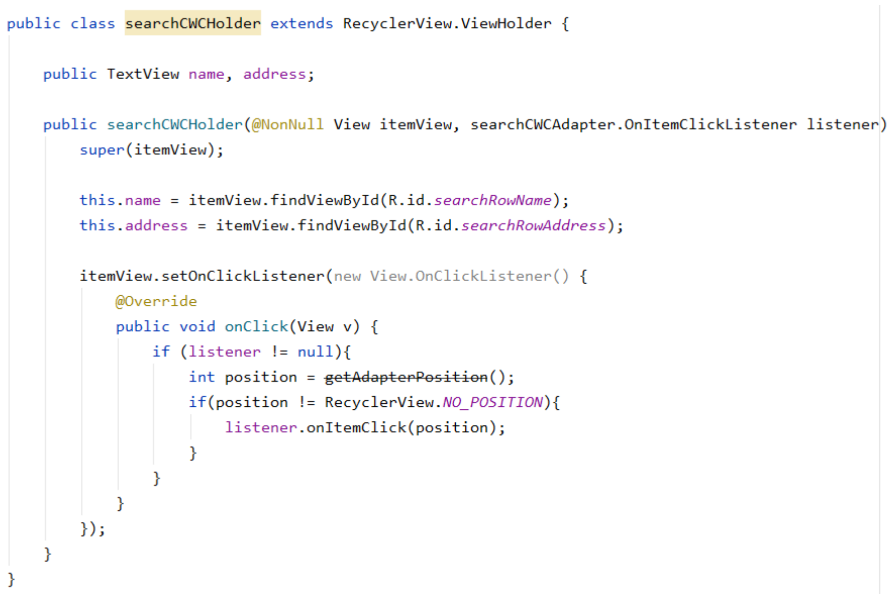
Figure 3-6-6-3-2 shows the coding of filter car wash center by name function. For loop is used to check the name of car wash center in the array list one by one with the user input. If the name of car wash center contains the user input, the car wash center is added to an array list called “filteredList”.



```
@NonNull
@Override
public searchCWCHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.search_cwc_row, null);
    return new searchCWCHolder(view, clicklistener);
}
```

Figure 3-6-6-3-3: Search Car Wash Center Adapter (1)

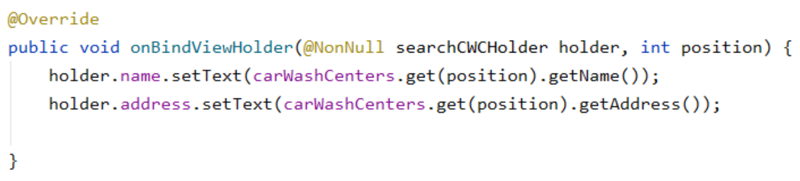
The filtered car wash center array list is display using recycler view. Recycler view is used to hold all the filtered car wash center. The adapter is used to handle each component of the filtered car wash center displayed in recycler view. In figure 3-6-6-3-3, the inflator is used to inflate the filtered car wash center row design file into a view.



```
public class searchCWCHolder extends RecyclerView.ViewHolder {

    public TextView name, address;

    public searchCWCHolder(@NonNull View itemView, searchCWCAdapter.OnItemClickListener listener) {
        super(itemView);

        this.name = itemView.findViewById(R.id.searchRowName);
        this.address = itemView.findViewById(R.id.searchRowAddress);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (listener != null){
                    int position = ~~getAdapterPosition~~();
                    if(position != RecyclerView.NO_POSITION){
                        listener.onItemClick(position);
                    }
                }
            }
        });
    }
}
```

Figure 3-6-6-3-4: Search Car Wash Center Adapter (2)

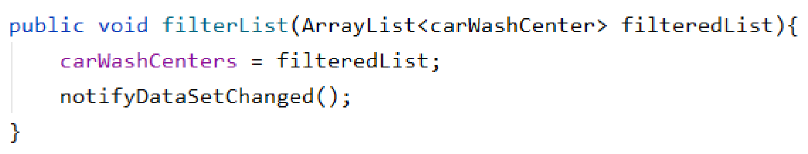
Firstly, all the components in the inflated view need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-6-3-4, component such as the name and address are declared.



```
@Override
public void onBindViewHolder(@NonNull searchCWCHolder holder, int position) {
    holder.name.setText(carWashCenters.get(position).getName());
    holder.address.setText(carWashCenters.get(position).getAddress());

}
```

Figure 3-6-6-3-5: Search Car Wash Center Adapter (3)

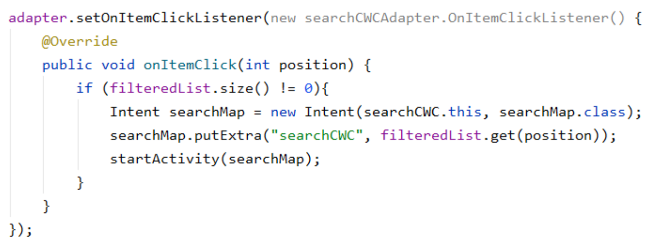
Each of the car wash center in the filtered car wash center array list is retrieved and the details is displayed. The holder is used to hold each of the component value in the inflated view. The array list is storing car wash center object, hence, the getter function is used to retrieved the data from the array list.



```
public void filterList(ArrayList<carWashCenter> filteredList){
    carWashCenters = filteredList;
    notifyDataSetChanged();
}
```

Figure 3-6-6-3-6: Search Car Wash Center Adapter (4)

The filterList() function is used to display the filtered car wash center and notify the changes made.

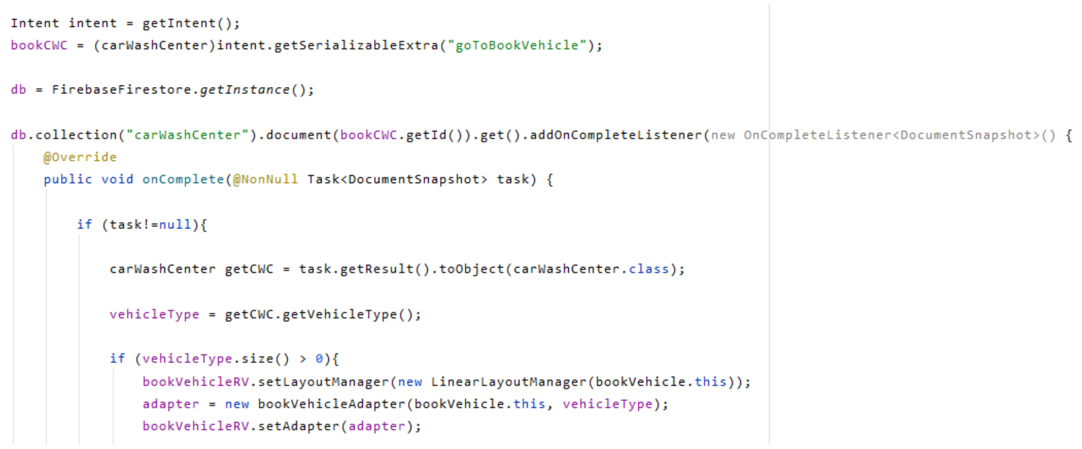


```
adapter.setOnItemClickListener(new searchCWCAdapter.OnItemClickListener() {
    @Override
    public void onItemClick(int position) {
        if (filteredList.size() != 0){
            Intent searchMap = new Intent(searchCWC.this, searchMap.class);
            searchMap.putExtra("searchCWC", filteredList.get(position));
            startActivity(searchMap);
        }
    }
});
```

Figure 3-6-6-3-7: Search Car Wash Center Adapter (5)

When user clicks on the car wash center in recycler view, the car wash center selected is passed to the Google Map page and display the location. The car wash center is get from the car wash center array list and passed using intent.

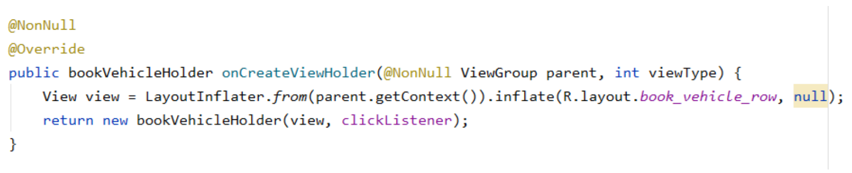
3.6.6.4 Select Vehicle Type



```
Intent intent = getIntent();
bookCWC = (carWashCenter)intent.getSerializableExtra("goToBookVehicle");

db = FirebaseFirestore.getInstance();

db.collection("carWashCenter").document(bookCWC.getId()).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {

        if (task!=null){

            carWashCenter getCWC = task.getResult().toObject(carWashCenter.class);

            vehicleType = getCWC.getVehicleType();

            if (vehicleType.size() > 0){
                bookVehicleRV.setLayoutManager(new LinearLayoutManager(bookVehicle.this));
                adapter = new bookVehicleAdapter(bookVehicle.this, vehicleType);
                bookVehicleRV.setAdapter(adapter);
```

Figure 3-6-6-4-1: Display Vehicle Type Adapter (1)

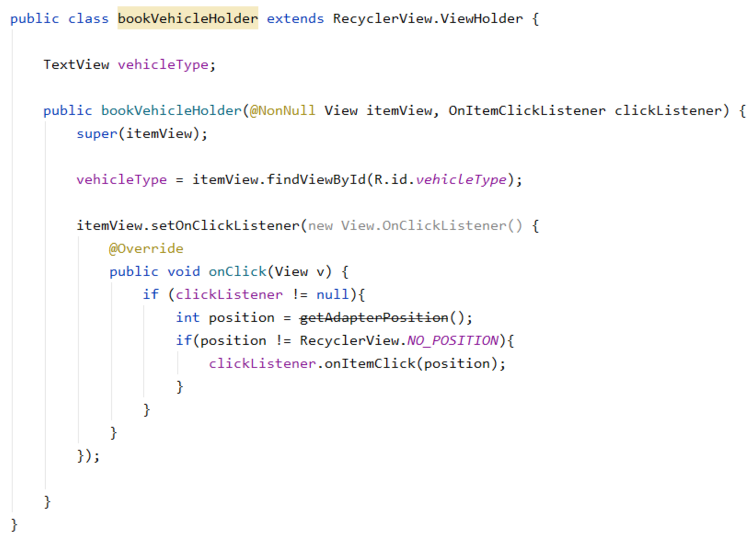
After user has select the car wash center, the car wash center is then passed to this activity using intent. Each of the car wash center provides service to different vehicle type. The vehicle type is display using recycler view. Hence, the vehicle type is retrieved from the car wash center. The vehicle type retrieved is in array list. The array list is then pass to the adapter of the recycler view.



```
@NonNull
@Override
public bookVehicleHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.book_vehicle_row, null);
    return new bookVehicleHolder(view, clickListener);
}
```

Figure 3-6-6-4-2: Display Vehicle Type Adapter (2)

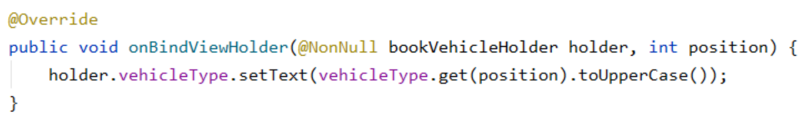
The vehicle type array list is display using recycler view. Recycler view is used to hold all the vehicle type. The adapter is used to handle each component of vehicle type displayed in recycler view. In figure 3-6-6-4-2, the inflator is used to inflate the vehicle type row design file into a view.



```
public class bookVehicleHolder extends RecyclerView.ViewHolder {

    TextView vehicleType;

    public bookVehicleHolder(@NonNull View itemView, OnItemClickListener clickListener) {
        super(itemView);

        vehicleType = itemView.findViewById(R.id.vehicleType);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (clickListener != null){
                    int position = ~~getAdapterPosition~~();
                    if(position != RecyclerView.NO_POSITION){
                        clickListener.onItemClick(position);
                    }
                }
            }
        });

    }
}
```

Figure 3-6-6-4-3: Display Vehicle Type Adapter (3)

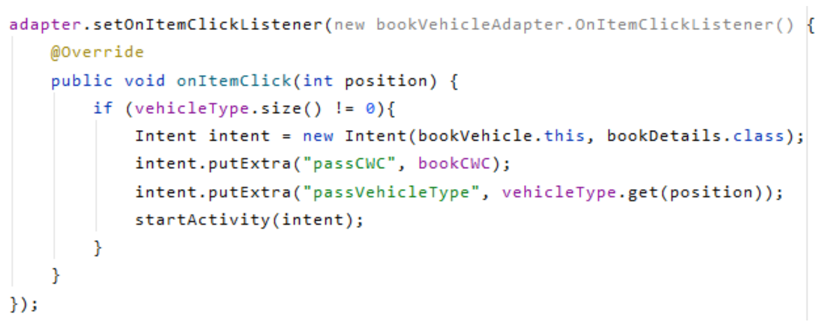
Firstly, all the components in the inflated view need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-6-4-3, component such as the vehicle type is declared.



```
@Override
public void onBindViewHolder(@NonNull bookVehicleHolder holder, int position) {
    holder.vehicleType.setText(vehicleType.get(position).toUpperCase());
}
```

Figure 3-6-6-4-4: Display Vehicle Type Adapter (4)

Each of the vehicle type in the vehicle type array list is retrieved and the details is displayed. The onBindViewHolder() is used to hold each of the component value of the inflated view.

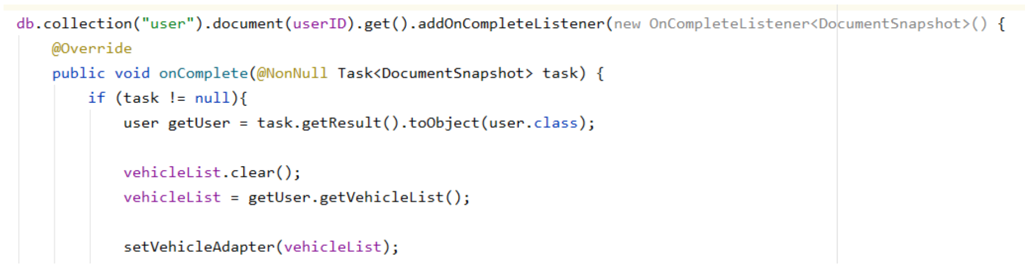


```
adapter.setOnItemClickListener(new bookVehicleAdapter.OnItemClickListener() {
    @Override
    public void onItemClick(int position) {
        if (vehicleType.size() != 0){
            Intent intent = new Intent(bookVehicle.this, bookDetails.class);
            intent.putExtra("passCWC", bookCWC);
            intent.putExtra("passVehicleType", vehicleType.get(position));
            startActivity(intent);
        }
    }
});
```

Figure 3-6-6-4-5: Display Vehicle Type Adapter (5)

When user clicks on the vehicle type in recycler view, the vehicle type selected is selected and passed to the next activity and continue the booking procedure. The vehicle type is get from the array list and passed using intent.

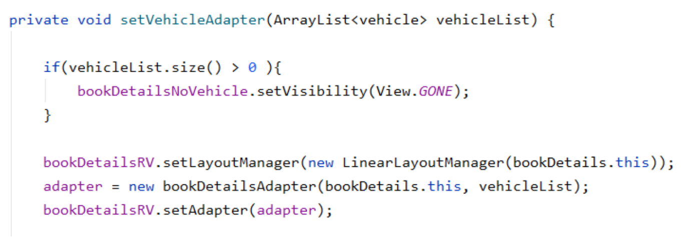
3.6.6.5 Select Vehicle



```
db.collection("user").document(userID).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        if (task != null){
            user getUser = task.getResult().toObject(user.class);

            vehicleList.clear();
            vehicleList = getUser.getVehicleList();

            setVehicleAdapter(vehicleList);
```

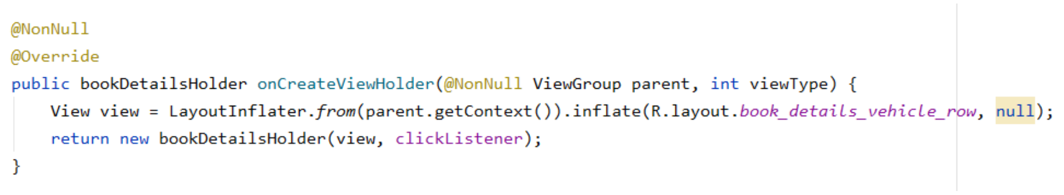
Figure 3-6-6-5-1: Retrieve user vehicle

Figure 3-6-6-5-1 shows the coding of retrieve user vehicle list from Firestore. User ID is used to retrieve the user from the "user" collection in Firestore. the vehicle list is in array list format. Then the vehicle list array list is passed into the setVehicleAdapter() function. The function is called after the system get the vehicle list from Firestore.



```
private void setVehicleAdapter(ArrayList<vehicle> vehicleList) {

    if(vehicleList.size() > 0 ){
        bookDetailsNoVehicle.setVisibility(View.GONE);
    }

    bookDetailsRV.setLayoutManager(new LinearLayoutManager(bookDetails.this));
    adapter = new bookDetailsAdapter(bookDetails.this, vehicleList);
    bookDetailsRV.setAdapter(adapter);
```

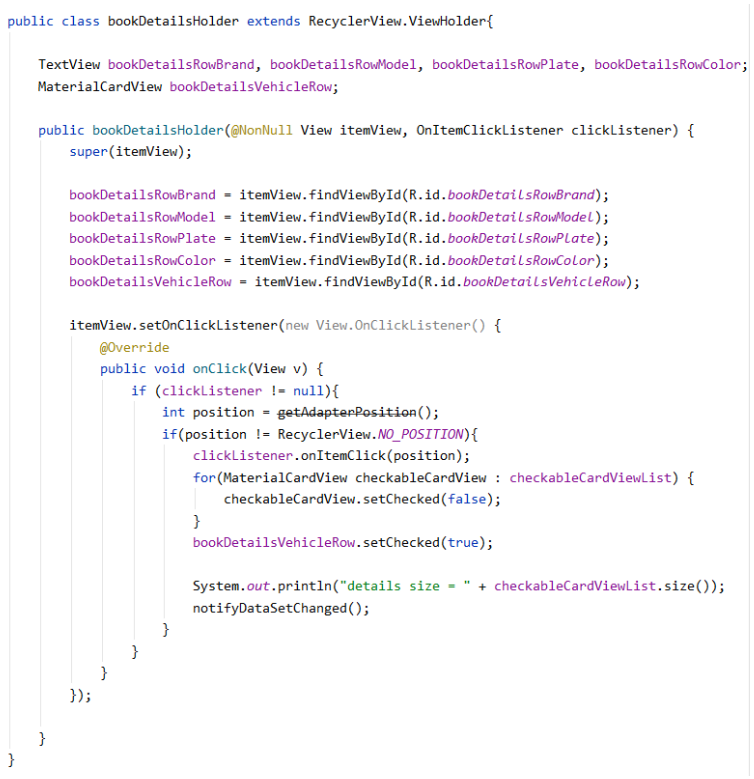
Figure 3-6-6-5-2: Display User Vehicle Adapter (1)



```
@NonNull
@Override
public bookDetailsHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.book_details_vehicle_row, null);
    return new bookDetailsHolder(view, clickListener);
}
```

Figure 3-6-6-5-3: Display User Vehicle Adapter (2)

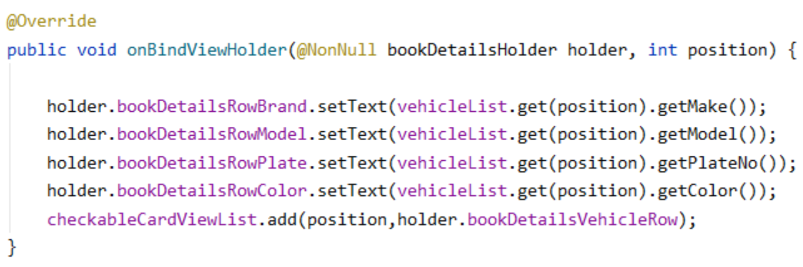
In figure 3-6-6-5-2, the vehicle array list is passed to the recycler view adapter. The vehicle list is display using recycler view. Recycler view is used to hold all the user’s vehicle list. The adapter is used to handle each vehicle displayed in recycler view. In figure 3-6-6-5-3, the inflator is used to inflate the user’s vehicle row design file into a view.



```java
public class bookDetailsHolder extends RecyclerView.ViewHolder{

    TextView bookDetailsRowBrand, bookDetailsRowModel, bookDetailsRowPlate, bookDetailsRowColor;
    MaterialCardView bookDetailsVehicleRow;

    public bookDetailsHolder(@NonNull View itemView, OnItemClickListener clickListener) {
        super(itemView);

        bookDetailsRowBrand = itemView.findViewById(R.id.bookDetailsRowBrand);
        bookDetailsRowModel = itemView.findViewById(R.id.bookDetailsRowModel);
        bookDetailsRowPlate = itemView.findViewById(R.id.bookDetailsRowPlate);
        bookDetailsRowColor = itemView.findViewById(R.id.bookDetailsRowColor);
        bookDetailsVehicleRow = itemView.findViewById(R.id.bookDetailsVehicleRow);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (clickListener != null){
                    int position = ~~getAdapterPosition~~();
                    if(position != RecyclerView.NO_POSITION){
                        clickListener.onItemClick(position);
                        for(MaterialCardView checkableCardView : checkableCardViewList) {
                            checkableCardView.setChecked(false);
                        }
                        bookDetailsVehicleRow.setChecked(true);

                        System.out.println("details size = " + checkableCardViewList.size());
                        notifyDataSetChanged();
                    }
                }
            }
        });

    }
}
```

Figure 3-6-6-5-4: Display User Vehicle Adapter (3)

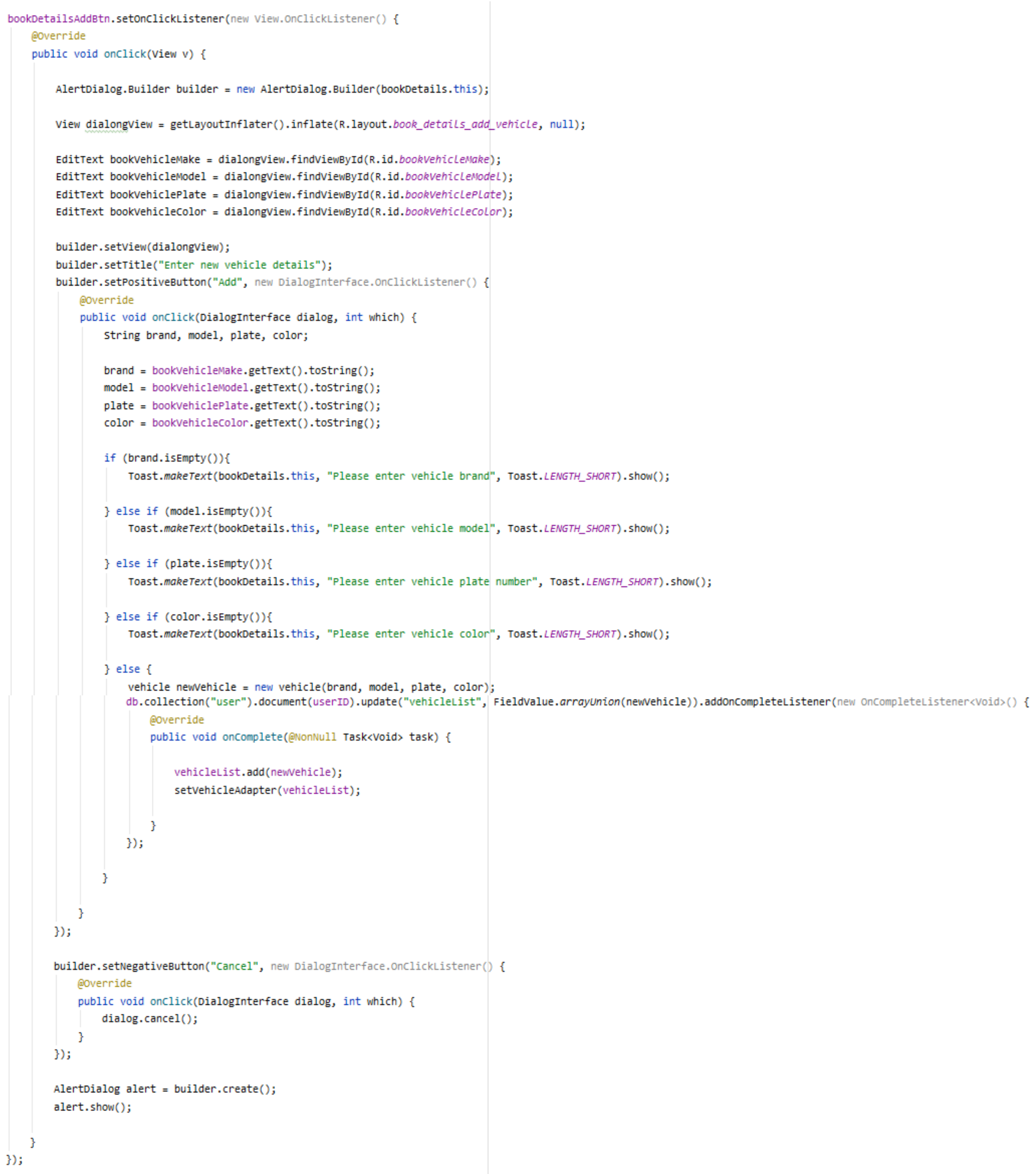
Firstly, all the components in the inflated view need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-6-5-4, component such as the vehicle model, vehicle brand, vehicle number plate and vehicle color is declared. When the user clicked on the inflated view, the system will uncheck all the item in the recycler view first, then only the system will check the selected item.



```java
@Override
public void onBindViewHolder(@NonNull bookDetailsHolder holder, int position) {

    holder.bookDetailsRowBrand.setText(vehicleList.get(position).getMake());
    holder.bookDetailsRowModel.setText(vehicleList.get(position).getModel());
    holder.bookDetailsRowPlate.setText(vehicleList.get(position).getPlateNo());
    holder.bookDetailsRowColor.setText(vehicleList.get(position).getColor());
    checkableCardViewList.add(position,holder.bookDetailsVehicleRow);
}
```

Figure 3-6-6-5-5: Display User Vehicle Adapter (4)

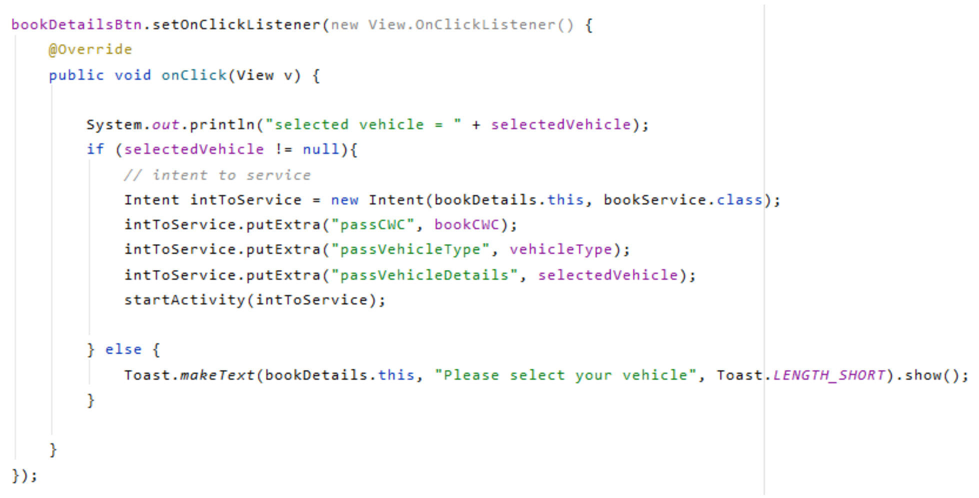
Each of the vehicle in the vehicle array list is retrieved and the details is displayed. The onBindViewHolder() is used to hold each of the component value of the inflated view. The getter function defined in the vehicle class is used to get the value from the vehicle in vehicle array list.



```
bookDetailsAddBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        AlertDialog.Builder builder = new AlertDialog.Builder(bookDetails.this);

        View dialongView = getLayoutInflater().inflate(R.layout.book_details_add_vehicle, null);

        EditText bookVehicleMake = dialongView.findViewById(R.id.bookVehicleMake);
        EditText bookVehicleModel = dialongView.findViewById(R.id.bookVehicleModel);
        EditText bookVehiclePlate = dialongView.findViewById(R.id.bookVehiclePlate);
        EditText bookVehicleColor = dialongView.findViewById(R.id.bookVehicleColor);

        builder.setView(dialongView);
        builder.setTitle("Enter new vehicle details");
        builder.setPositiveButton("Add", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String brand, model, plate, color;

                brand = bookVehicleMake.getText().toString();
                model = bookVehicleModel.getText().toString();
                plate = bookVehiclePlate.getText().toString();
                color = bookVehicleColor.getText().toString();

                if (brand.isEmpty()){
                    Toast.makeText(bookDetails.this, "Please enter vehicle brand", Toast.LENGTH_SHORT).show();

                } else if (model.isEmpty()){
                    Toast.makeText(bookDetails.this, "Please enter vehicle model", Toast.LENGTH_SHORT).show();

                } else if (plate.isEmpty()){
                    Toast.makeText(bookDetails.this, "Please enter vehicle plate number", Toast.LENGTH_SHORT).show();

                } else if (color.isEmpty()){
                    Toast.makeText(bookDetails.this, "Please enter vehicle color", Toast.LENGTH_SHORT).show();

                } else {
                    vehicle newVehicle = new vehicle(brand, model, plate, color);
                    db.collection("user").document(userID).update("vehicleList", FieldValue.arrayUnion(newVehicle)).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {

                            vehicleList.add(newVehicle);
                            setVehicleAdapter(vehicleList);

                        }
                    });

                }

            }
        });

        builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });

        AlertDialog alert = builder.create();
        alert.show();

    }
});
```

Figure 3-6-6-5-6: Add User Vehicle

Figure 3-6-6-5-6 shows the coding of add user vehicle function. When the add vehicle button is clicked by user, the onClick() function is triggered. An alert dialog is pop out and user is required to input the vehicle details such as the vehicle model, brand, plate number and color. When the positive button which is the “Add” button is clicked, the system will check whether the input field is completed. If the inputs are not completed, the system will display error message to inform user. Else, the system will update the Firestore. The system will insert the newly added vehicle to the user's database. Besides, the system will also insert the new vehicle to the vehicle list and call setVehicleAdapter() function again to display the latest vehicle list to user.

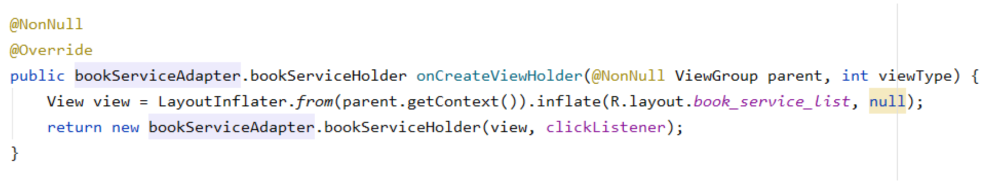


```java
bookDetailsBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        System.out.println("selected vehicle = " + selectedVehicle);
        if (selectedVehicle != null){
            // intent to service
            Intent intToService = new Intent(bookDetails.this, bookService.class);
            intToService.putExtra("passCWC", bookCWC);
            intToService.putExtra("passVehicleType", vehicleType);
            intToService.putExtra("passVehicleDetails", selectedVehicle);
            startActivity(intToService);

        } else {
            Toast.makeText(bookDetails.this, "Please select your vehicle", Toast.LENGTH_SHORT).show();
        }

    }
});
```

Figure 3-6-6-5-7: Select user vehicle

When user clicks "continue" button, the onClick() function is triggered. The system will check whether the selected vehicle is null or not. If the user did not select any vehicle, the selected vehicle will become null. Thus, the system will display error message. If user had already selected a vehicle, the system will direct user to the next page. The car wash center select, vehicle type and vehicle is pass to the next activity using intent.

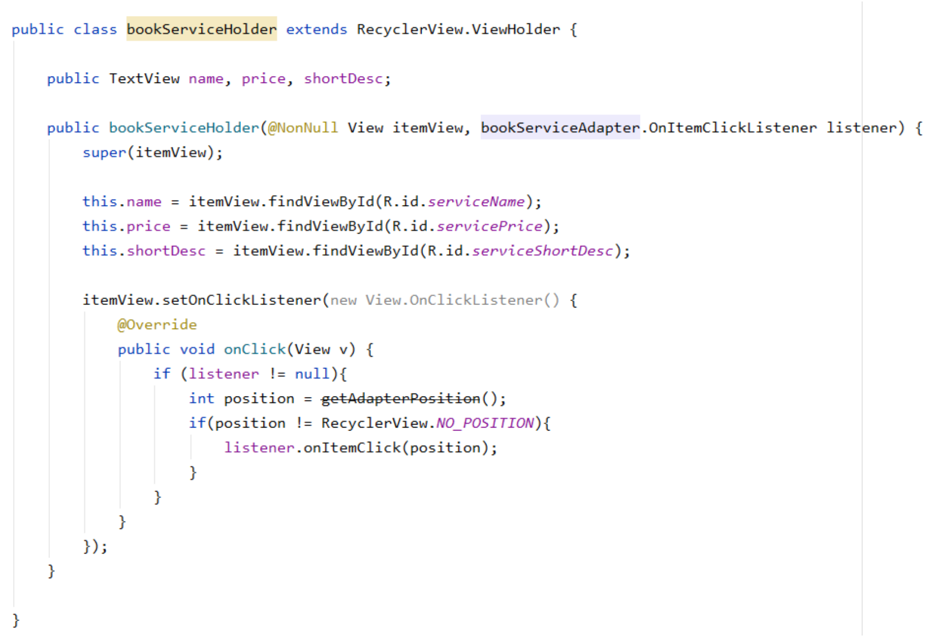
3.6.6.6 Select Service



```java
@NonNull
@Override
public bookServiceAdapter.bookServiceHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.book_service_list, null);
    return new bookServiceAdapter.bookServiceHolder(view, clickListener);
}
```

Figure 3-6-6-6-1: Display Service Adapter (1)

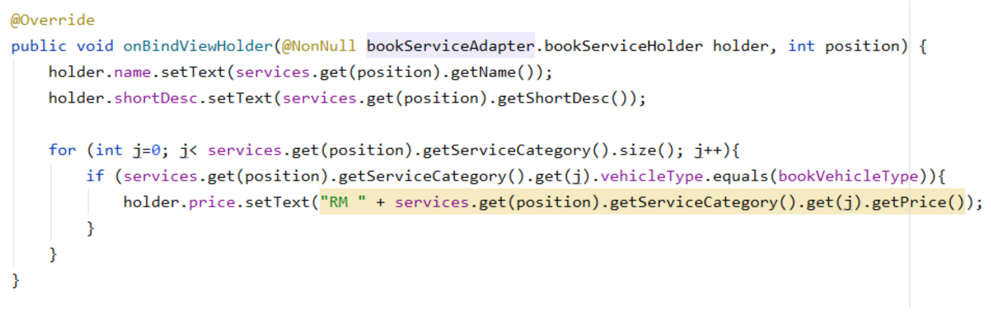
Figure 3-6-6-6-1 shows the coding of display service of car wash center adapter. The service list is get from the car wash center passed from the previous activity using intent. The service is display using recycler view. Recycler view is used to hold all the service in car wash center's service list. The adapter is used to handle each service displayed in recycler view. In figure 3-6-6-6-1, the inflator is used to inflate the car wash center’s service row design file into a view.



```
public class bookServiceHolder extends RecyclerView.ViewHolder {

    public TextView name, price, shortDesc;

    public bookServiceHolder(@NonNull View itemView, bookServiceAdapter.OnItemClickListener listener) {
        super(itemView);

        this.name = itemView.findViewById(R.id.serviceName);
        this.price = itemView.findViewById(R.id.servicePrice);
        this.shortDesc = itemView.findViewById(R.id.serviceShortDesc);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (listener != null){
                    int position = getAdapterPosition();
                    if(position != RecyclerView.NO_POSITION){
                        listener.onItemClick(position);
                    }
                }
            }
        });
    }

}
```

Figure 3-6-6-6-2: Display Service Adapter (2)

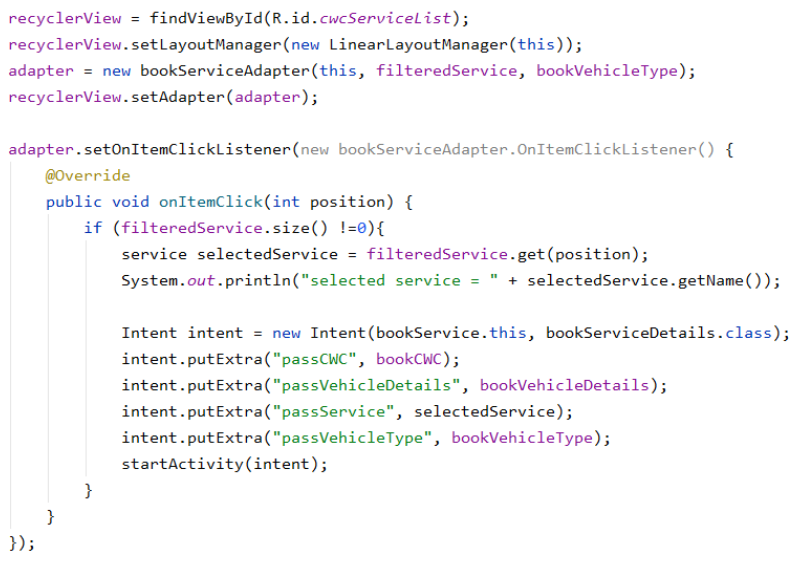
Firstly, all the components in the inflated view need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-6-6-2, component such as the service name, price and description is declared. Onclick() function is triggered when the user clicked on the item in the recycler view. It will return the position of the service in the service list.



```
@Override
public void onBindViewHolder(@NonNull bookServiceAdapter.bookServiceHolder holder, int position) {
    holder.name.setText(services.get(position).getName());
    holder.shortDesc.setText(services.get(position).getShortDesc());

    for (int j=0; j< services.get(position).getServiceCategory().size(); j++){
        if (services.get(position).getServiceCategory().get(j).vehicleType.equals(bookVehicleType)){
            holder.price.setText("RM " + services.get(position).getServiceCategory().get(j).getPrice());
        }
    }
}
```

Figure 3-6-6-6-3: Display Service Adapter (3)

Each of the service in the service array list is retrieved and the details is displayed. The onBindViewHolder() is used to hold each of the component value of the inflated view. The getter and setter function defined in the service class is used to get the value from the service in service array list.

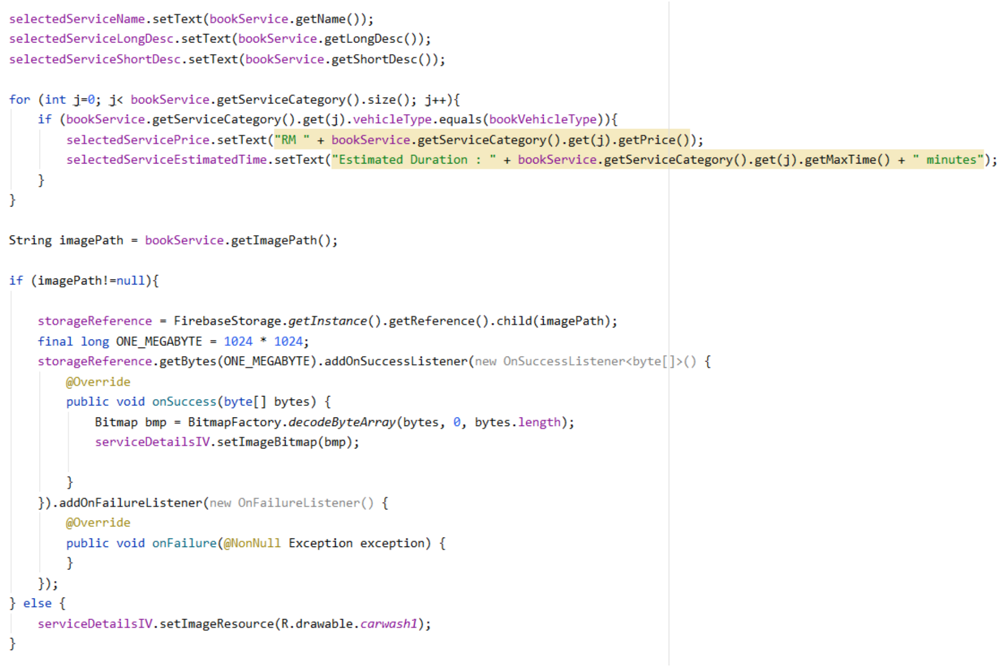


```java
recyclerView = findViewById(R.id.cwcServiceList);
recyclerView.setLayoutManager(new LinearLayoutManager(this));
adapter = new bookServiceAdapter(this, filteredService, bookVehicleType);
recyclerView.setAdapter(adapter);

adapter.setOnItemClickListener(new bookServiceAdapter.OnItemClickListener() {
    @Override
    public void onItemClick(int position) {
        if (filteredService.size() !=0){
            service selectedService = filteredService.get(position);
            System.out.println("selected service = " + selectedService.getName());

            Intent intent = new Intent(bookService.this, bookServiceDetails.class);
            intent.putExtra("passCWC", bookCWC);
            intent.putExtra("passVehicleDetails", bookVehicleDetails);
            intent.putExtra("passService", selectedService);
            intent.putExtra("passVehicleType", bookVehicleType);
            startActivity(intent);
        }
    }
});
```

Figure 3-6-6-6-4: Display Service Adapter (4)

OnItemClick() function is triggered when the user clicked on the item in the recycler view. The position of the service in service list is passed from the view holder in Figure 3-6-6-6-4. When user clicked on the service, it indicates the user had chosen their preferable service. The system will then direct user to the next activity that display the details of the service. The system will pass the selected car wash center, vehicle type, vehicle and service to the next activity using intent.

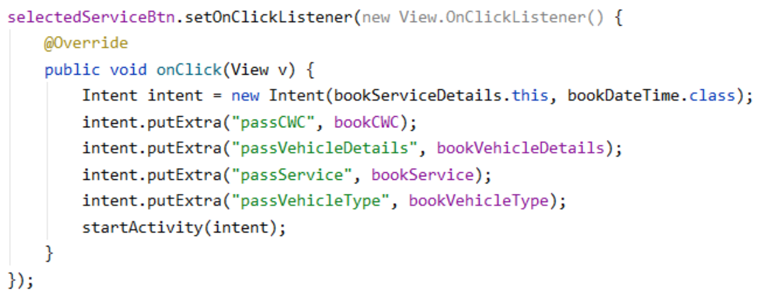
3.6.6.7 Display Service Details



```java
selectedServiceName.setText(bookService.getName());
selectedServiceLongDesc.setText(bookService.getLongDesc());
selectedServiceShortDesc.setText(bookService.getShortDesc());

for (int j=0; j< bookService.getServiceCategory().size(); j++){
    if (bookService.getServiceCategory().get(j).vehicleType.equals(bookVehicleType)){
        selectedServicePrice.setText("RM " + bookService.getServiceCategory().get(j).getPrice());
        selectedServiceEstimatedTime.setText("Estimated Duration : " + bookService.getServiceCategory().get(j).getMaxTime() + " minutes");
    }
}

String imagePath = bookService.getImagePath();

if (imagePath!=null){

    storageReference = FirebaseStorage.getInstance().getReference().child(imagePath);
    final long ONE_MEGABYTE = 1024 * 1024;
    storageReference.getBytes(ONE_MEGABYTE).addOnSuccessListener(new OnSuccessListener<byte[]>() {
        @Override
        public void onSuccess(byte[] bytes) {
            Bitmap bmp = BitmapFactory.decodeByteArray(bytes, 0, bytes.length);
            serviceDetailsIV.setImageBitmap(bmp);

        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception exception) {
        }
    });
} else {
    serviceDetailsIV.setImageResource(R.drawable.carwash1);
}
```

Figure 3-6-6-7-1: Display service details

Figure 3-6-6-7-1 shows the coding of displaying service details function. The selected service details are obtained from the service passed from the previous activity using intent. Then the data obtained is set to the respective text view in the service details page.

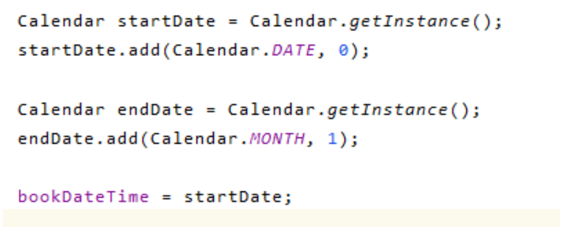


```
selectedServiceBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(bookServiceDetails.this, bookDateTime.class);
        intent.putExtra("passCWC", bookCWC);
        intent.putExtra("passVehicleDetails", bookVehicleDetails);
        intent.putExtra("passService", bookService);
        intent.putExtra("passVehicleType", bookVehicleType);
        startActivity(intent);
    }
});
```

Figure 3-6-6-7-2: Intent to Select Time Slot Activity

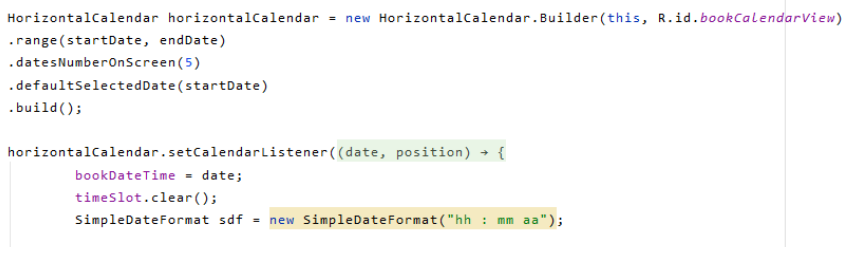
When user clicked on the “continue” button, the system will direct user to the next activity to continue the booking appointment process. The system will pass the selected car wash center, vehicle type, vehicle and service to the next activity using intent.

3.6.6.8 Select Time Slot



```
Calendar startDate = Calendar.getInstance();
startDate.add(Calendar.DATE, 0);

Calendar endDate = Calendar.getInstance();
endDate.add(Calendar.MONTH, 1);

bookDateTime = startDate;
```

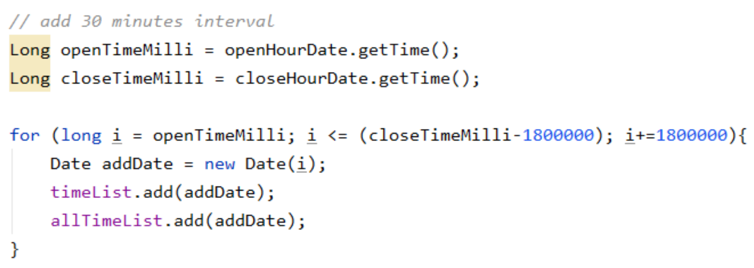
Figure 3-6-6-8-1: Select Date for Appointment (1)



```
HorizontalCalendar horizontalCalendar = new HorizontalCalendar.Builder(this, R.id.bookCalendarView)
.range(startDate, endDate)
.datesNumberOnScreen(5)
.defaultSelectedDate(startDate)
.build();

horizontalCalendar.setCalendarListener((date, position) → {
        bookDateTime = date;
        timeSlot.clear();
        SimpleDateFormat sdf = new SimpleDateFormat("hh : mm aa");
```

Figure 3-6-6-8-2: Select Date for Appointment (2)

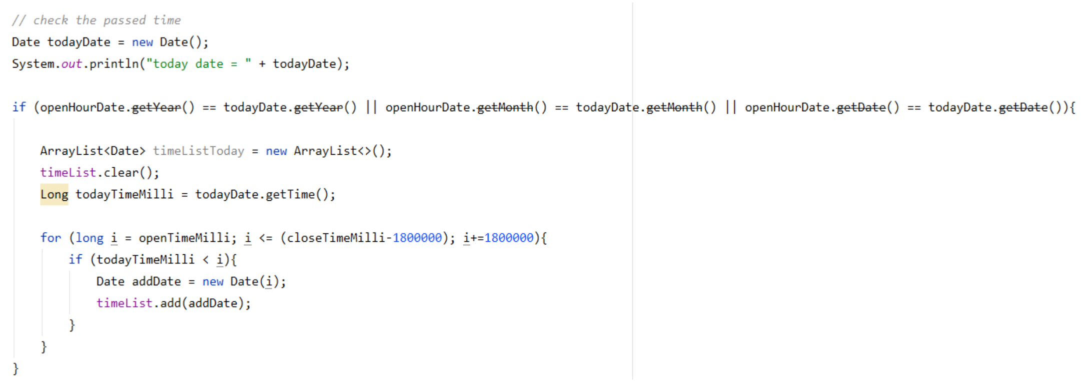
Figure 3-6-6-8-1 and figure 3-6-6-8-2 shows the coding of selecting date for appointment. Horizontal Calendar is used to let the user select the date they want to book the appointment. A range of date is set. The start date defined in the coding is the current date and the end date defined is 1 month from the start date. Calendar listener is used to get the selected date when user switch the date.



```java
// add 30 minutes interval
Long openTimeMilli = openHourDate.getTime();
Long closeTimeMilli = closeHourDate.getTime();

for (long i = openTimeMilli; i <= (closeTimeMilli-1800000); i+=1800000){
    Date addDate = new Date(i);
    timeList.add(addDate);
    allTimeList.add(addDate);
}
```

Figure 3-6-6-8-3: Display Available Time Slot

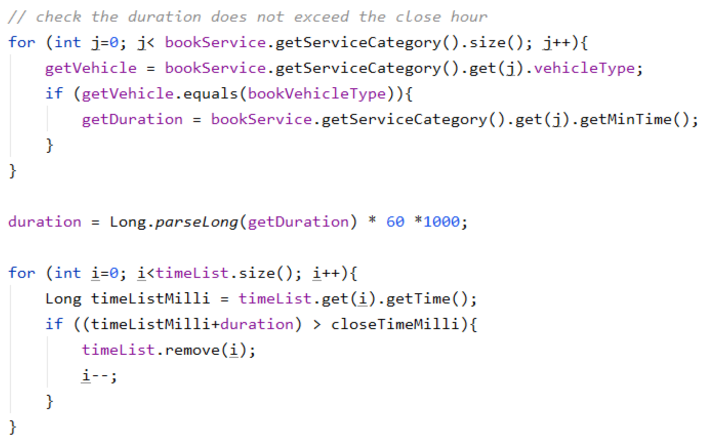
The opening hour and the closing hour of the selected car wash center by user is retrieved from the data passed from the previous activity using intent. An array used to store the available time slot is declared as “timeList” at first. 1-time slot is set as 30 minutes. The first date time added to the time slot array list must be the car wash center’s opening time and then the increase 30 minutes as the interval of a time slot is 30 minutes using for loop. If the date time had exceeded the car wash center’s closing time, the for loop will be exit.



```java
// check the passed time
Date todayDate = new Date();
System.out.println("today date = " + todayDate);

if (openHourDate.getYear() == todayDate.getYear() || openHourDate.getMonth() == todayDate.getMonth() || openHourDate.getDate() == todayDate.getDate()){

    ArrayList<Date> timeListToday = new ArrayList<>();
    timeList.clear();
    Long todayTimeMilli = todayDate.getTime();

    for (long i = openTimeMilli; i <= (closeTimeMilli-1800000); i+=1800000){
        if (todayTimeMilli < i){
            Date addDate = new Date(i);
            timeList.add(addDate);
        }
    }
}
```

Figure 3-6-6-8-4: Check for Available Time Slot (1)

Next, user is not allowed to select the time slot that has already passed. If the user had selected the current date in the Horizontal Calendar, the system will check the whether the current time has exceeded the time in the time slot. If the current time had exceeded the time slot, the time slot will remove from the available time slot list. For example, if user selected the current date in Horizontal Calendar, the current time is at 2.50 p.m. Hence, the user is not able to select the time slot at 11 a.m. of the current date.

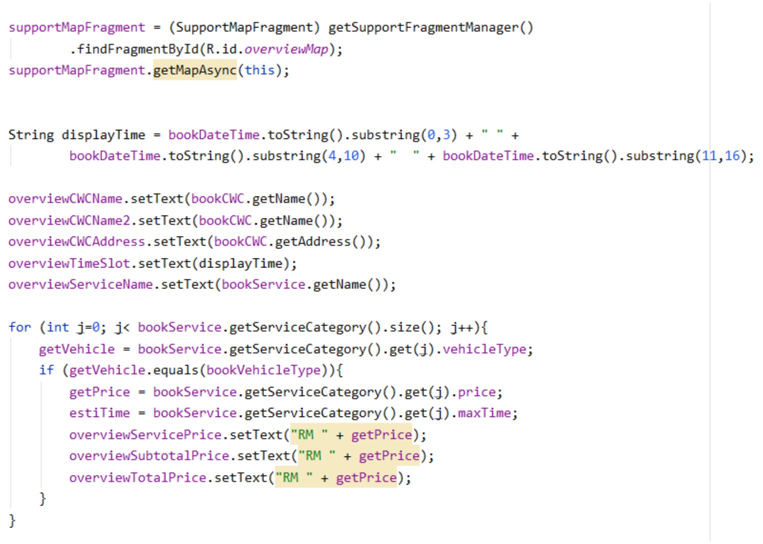


```
// check the duration does not exceed the close hour
for (int j=0; j< bookService.getServiceCategory().size(); j++){
    getVehicle = bookService.getServiceCategory().get(j).vehicleType;
    if (getVehicle.equals(bookVehicleType)){
        getDuration = bookService.getServiceCategory().get(j).getMinTime();
    }
}

duration = Long.parseLong(getDuration) * 60 *1000;

for (int i=0; i<timeList.size(); i++){
    Long timeListMilli = timeList.get(i).getTime();
    if ((timeListMilli+duration) > closeTimeMilli){
        timeList.remove(i);
        i--;
    }
}
```

Figure 3-6-6-8-5: Check for Available Time Slot (2)

In addition, the system will also check whether the service booked at a time able to finish before the car wash center’s closing hour. Hence, the time needed to finish the service is retrieved from the user’s selected car wash center. Then, the system will check all the time slot in the time slot array list one by one using for loop. If the time slot not able to finish the service before closing hour, the then time slot will be remove from the available time slot list.

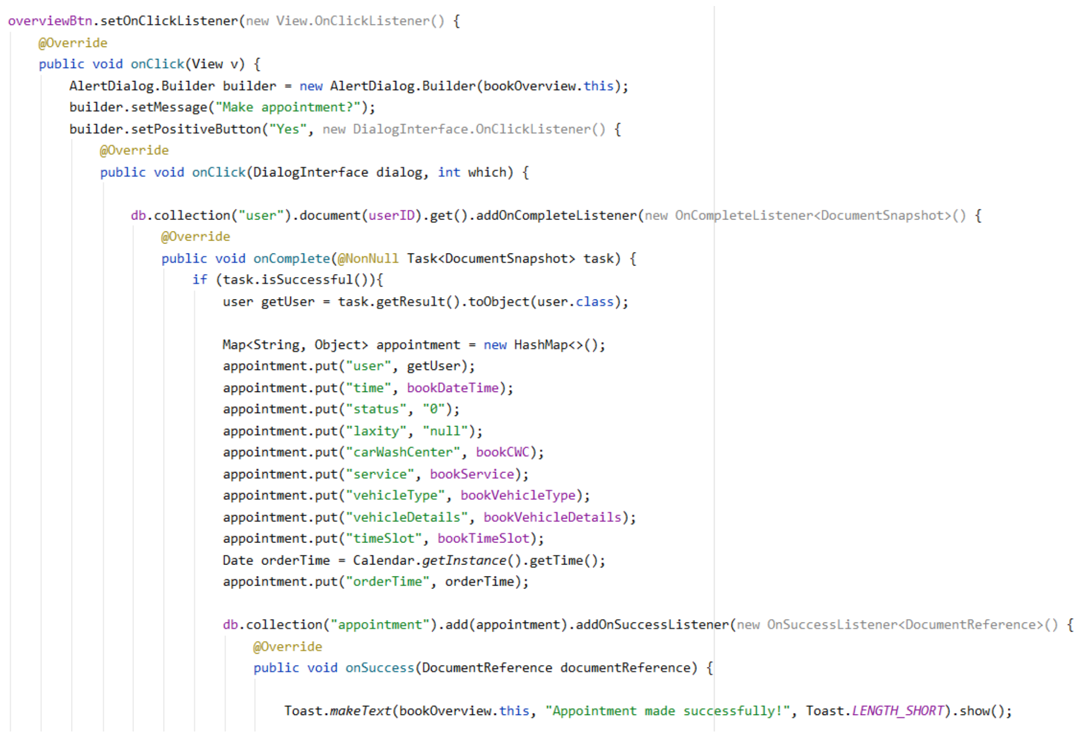
3.6.6.9 Appointment Overview



```
supportMapFragment = (SupportMapFragment) getSupportFragmentManager()
        .findFragmentById(R.id.overviewMap);
supportMapFragment.getMapAsync(this);


String displayTime = bookDateTime.toString().substring(0,3) + " " +
        bookDateTime.toString().substring(4,10) + "  " + bookDateTime.toString().substring(11,16);

overviewCWCName.setText(bookCWC.getName());
overviewCWCName2.setText(bookCWC.getName());
overviewCWCAddress.setText(bookCWC.getAddress());
overviewTimeSlot.setText(displayTime);
overviewServiceName.setText(bookService.getName());

for (int j=0; j< bookService.getServiceCategory().size(); j++){
    getVehicle = bookService.getServiceCategory().get(j).vehicleType;
    if (getVehicle.equals(bookVehicleType)){
        getPrice = bookService.getServiceCategory().get(j).price;
        estiTime = bookService.getServiceCategory().get(j).maxTime;
        overviewServicePrice.setText("RM " + getPrice);
        overviewSubtotalPrice.setText("RM " + getPrice);
        overviewTotalPrice.setText("RM " + getPrice);
    }
}
```

Figure 3-6-6-9-1: Display Appointment Overview

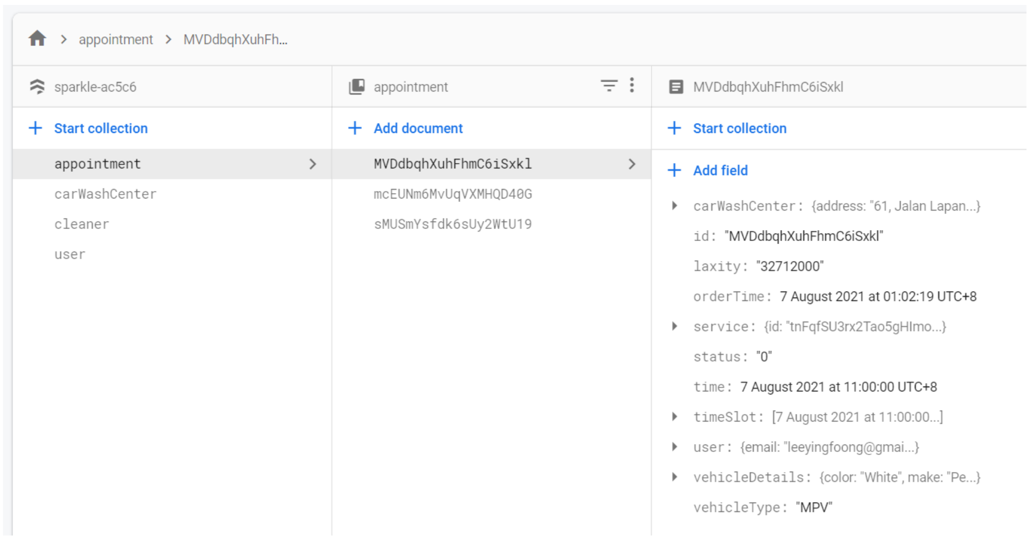
The details of the appointment are passed from the select time slot activity using intent. The data is set to the respective text field using setText() function.



```
overviewBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        AlertDialog.Builder builder = new AlertDialog.Builder(bookOverview.this);
        builder.setMessage("Make appointment?");
        builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

                db.collection("user").document(userID).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                        if (task.isSuccessful()){
                            user getUser = task.getResult().toObject(user.class);

                            Map<String, Object> appointment = new HashMap<>();
                            appointment.put("user", getUser);
                            appointment.put("time", bookDateTime);
                            appointment.put("status", "0");
                            appointment.put("laxity", "null");
                            appointment.put("carWashCenter", bookCWC);
                            appointment.put("service", bookService);
                            appointment.put("vehicleType", bookVehicleType);
                            appointment.put("vehicleDetails", bookVehicleDetails);
                            appointment.put("timeSlot", bookTimeSlot);
                            Date orderTime = Calendar.getInstance().getTime();
                            appointment.put("orderTime", orderTime);

                            db.collection("appointment").add(appointment).addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
                                @Override
                                public void onSuccess(DocumentReference documentReference) {

                                    Toast.makeText(bookOverview.this, "Appointment made successfully!", Toast.LENGTH_SHORT).show();
```

Figure 3-6-6-9-2: Add Appointment to Firestore

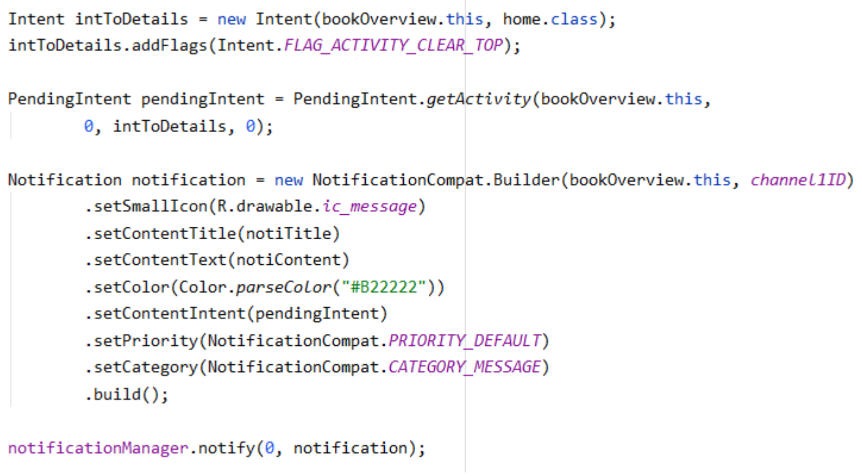
Figure 3-6-6-9-2 shows the coding of add appointment to Firestore function. When user clicks the "confirm" button the onClick() function is triggered. All the data such as user, car wash center, booking time, vehicle type, service is store in a HashMap. Then, the appointment is added to the Firestore "appointment" collection. The appointment in Firestore is as shown in Figure 3-6-6-9-3.



```
> appointment > MVDdbqhXuhFh...

sparkle-ac5c6          appointment                 MVDdbqhXuhFhmC6iSxkl
+ Start collection     + Add document              + Start collection
appointment        >   MVDdbqhXuhFhmC6iSxkl   >    + Add field
carWashCenter          mcEUNm6MvUqVXMHQD40G        ▸ carWashCenter: {address: "61, Jalan Lapan...}
cleaner                sMUSmYsfdk6sUy2WtU19          id: "MVDdbqhXuhFhmC6iSxkl"
user                                                 laxity: "32712000"
                                                     orderTime: 7 August 2021 at 01:02:19 UTC+8
                                                   ▸ service: {id: "tnFqfSU3rx2Tao5gHlmo...}
                                                     status: "0"
                                                     time: 7 August 2021 at 11:00:00 UTC+8
                                                   ▸ timeSlot: [7 August 2021 at 11:00:00...]
                                                   ▸ user: {email: "leeyingfoong@gmai...}
                                                   ▸ vehicleDetails: {color: "White", make: "Pe...}
                                                     vehicleType: "MPV"
```

Figure 3-6-6-9-3: Appointment in Firestore

3.6.7 Received Notification

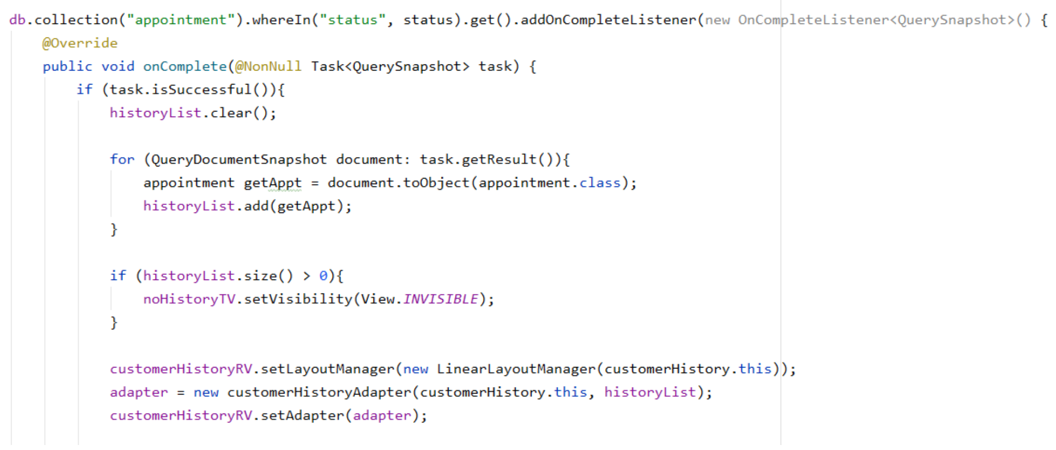


```
Intent intToDetails = new Intent(bookOverview.this, home.class);
intToDetails.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

PendingIntent pendingIntent = PendingIntent.getActivity(bookOverview.this,
        0, intToDetails, 0);

Notification notification = new NotificationCompat.Builder(bookOverview.this, channel1ID)
        .setSmallIcon(R.drawable.ic_message)
        .setContentTitle(notiTitle)
        .setContentText(notiContent)
        .setColor(Color.parseColor("#B22222"))
        .setContentIntent(pendingIntent)
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setCategory(NotificationCompat.CATEGORY_MESSAGE)
        .build();

notificationManager.notify(0, notification);
```

Figure 3-6-7-1: Sends Notification

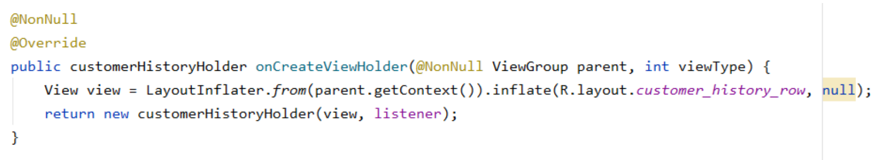
The design and content of the notification is customized. The small icon, content title, content text, colour and many more also able to customize in this coding. User will receive the customized notification.

3.6.8 View Customer’s Appointment History



```
db.collection("appointment").whereIn("status", status).get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
        if (task.isSuccessful()){
            historyList.clear();

            for (QueryDocumentSnapshot document: task.getResult()){
                appointment getAppt = document.toObject(appointment.class);
                historyList.add(getAppt);
            }

            if (historyList.size() > 0){
                noHistoryTV.setVisibility(View.INVISIBLE);
            }

            customerHistoryRV.setLayoutManager(new LinearLayoutManager(customerHistory.this));
            adapter = new customerHistoryAdapter(customerHistory.this, historyList);
            customerHistoryRV.setAdapter(adapter);
```

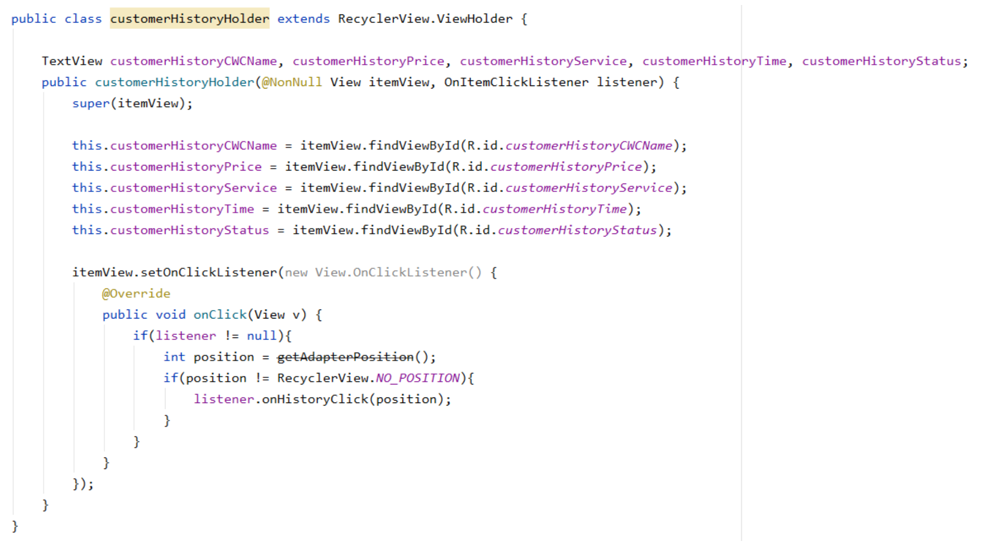
Figure 3-6-8-1: Retrieve Appointment From Firestore



```
@NonNull
@Override
public customerHistoryHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.customer_history_row, null);
    return new customerHistoryHolder(view, listener);
}
```

Figure 3-6-8-2: Display Appointment History Adapter (1)

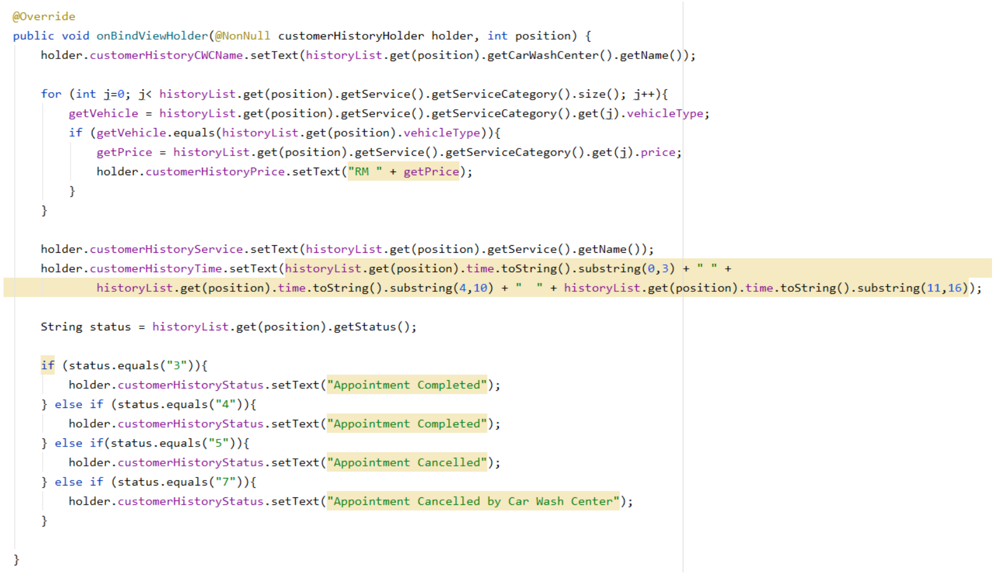
Figure 3-6-8-1 shows the coding of getting appointment from Firestore. The appointment is retrieved from the “appointment” collection from Firestore. System had used the user’s ID to filter all the appointment. All the appointment is then added into an array list defined as “historyList” in the coding above. The appointment history is display using recycler view. Recycler view is used to hold all the appointment in appointment history array list. The array list is then passed into the adapter. The adapter is used to handle each appointment displayed in recycler view. In figure 3-6-8-2, the inflator is used to inflate the appointment history row design file into a view.



```java
public class customerHistoryHolder extends RecyclerView.ViewHolder {

    TextView customerHistoryCWCName, customerHistoryPrice, customerHistoryService, customerHistoryTime, customerHistoryStatus;
    public customerHistoryHolder(@NonNull View itemView, OnItemClickListener listener) {
        super(itemView);

        this.customerHistoryCWCName = itemView.findViewById(R.id.customerHistoryCWCName);
        this.customerHistoryPrice = itemView.findViewById(R.id.customerHistoryPrice);
        this.customerHistoryService = itemView.findViewById(R.id.customerHistoryService);
        this.customerHistoryTime = itemView.findViewById(R.id.customerHistoryTime);
        this.customerHistoryStatus = itemView.findViewById(R.id.customerHistoryStatus);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(listener != null){
                    int position = ~~getAdapterPosition~~();
                    if(position != RecyclerView.NO_POSITION){
                        listener.onHistoryClick(position);
                    }
                }
            }
        });
    }
}
```

Figure 3-6-8-3: Display Appointment History Adapter (2)

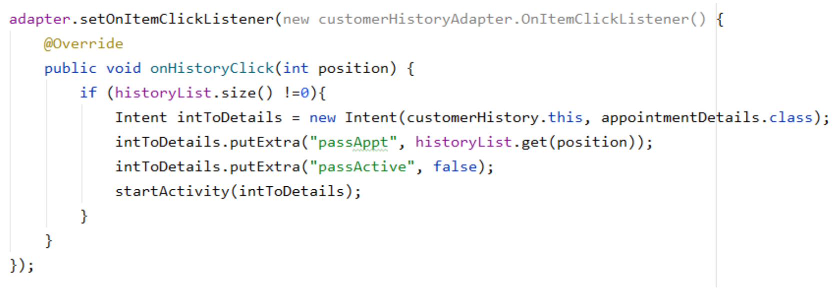
Firstly, all the components in the inflated view need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-8-3, component such as the car wash center name, service, price and many more is declared. Onclick() function is triggered when the user clicked on the item in the recycler view. It will return the position of the appointment in the appointment history list.



```java
@Override
public void onBindViewHolder(@NonNull customerHistoryHolder holder, int position) {
    holder.customerHistoryCWCName.setText(historyList.get(position).getCarWashCenter().getName());

    for (int j=0; j< historyList.get(position).getService().getServiceCategory().size(); j++){
        getVehicle = historyList.get(position).getService().getServiceCategory().get(j).vehicleType;
        if (getVehicle.equals(historyList.get(position).vehicleType)){
            getPrice = historyList.get(position).getService().getServiceCategory().get(j).price;
            holder.customerHistoryPrice.setText("RM " + getPrice);
        }
    }

    holder.customerHistoryService.setText(historyList.get(position).getService().getName());
    holder.customerHistoryTime.setText(historyList.get(position).time.toString().substring(0,3) + " " +
            historyList.get(position).time.toString().substring(4,10) + "  " + historyList.get(position).time.toString().substring(11,16));

    String status = historyList.get(position).getStatus();

    if (status.equals("3")){
        holder.customerHistoryStatus.setText("Appointment Completed");
    } else if (status.equals("4")){
        holder.customerHistoryStatus.setText("Appointment Completed");
    } else if(status.equals("5")){
        holder.customerHistoryStatus.setText("Appointment Cancelled");
    } else if (status.equals("7")){
        holder.customerHistoryStatus.setText("Appointment Cancelled by Car Wash Center");
    }

}
```

Figure 3-6-8-4: Display Appointment History Adapter (3)

Each of the appointment in the appointment history array list is retrieved and the details is displayed. The onBindViewHolder() is used to hold each of the component value of the inflated view. The getter function defined in the appointment class is used to get the value from the appointment in appointment history list.



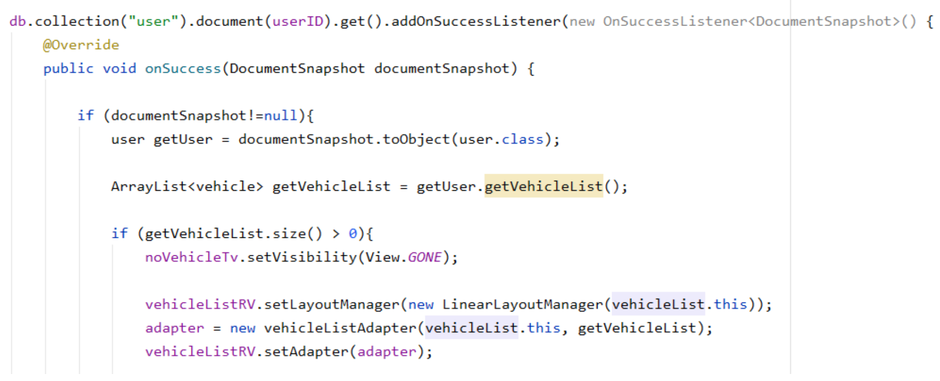
```
adapter.setOnItemClickListener(new customerHistoryAdapter.OnItemClickListener() {
    @Override
    public void onHistoryClick(int position) {
        if (historyList.size() !=0){
            Intent intToDetails = new Intent(customerHistory.this, appointmentDetails.class);
            intToDetails.putExtra("passAppt", historyList.get(position));
            intToDetails.putExtra("passActive", false);
            startActivity(intToDetails);
        }
    }
});
```

Figure 3-6-8-5: Display Appointment History Adapter (4)

OnHistoryClick() function is triggered when the user clicked on the item in the recycler view. The position of the appointment in appointment history list is passed from the view holder in Figure 3-6-8-5. When user clicked on the appointment, the system will then direct user to the next activity that display the details of the appointment. The system will pass the appointment to the next activity using intent.

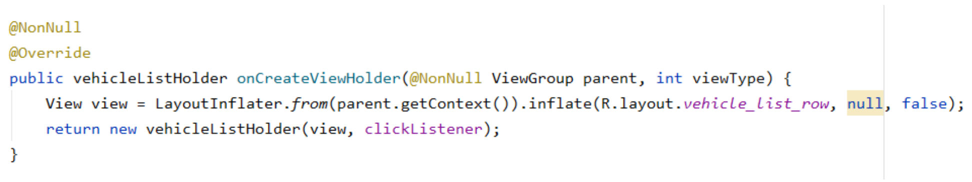
3.6.9 Manage Vehicle

3.6.9.1 View Vehicle



```
db.collection("user").document(userID).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
    @Override
    public void onSuccess(DocumentSnapshot documentSnapshot) {

        if (documentSnapshot!=null){
            user getUser = documentSnapshot.toObject(user.class);

            ArrayList<vehicle> getVehicleList = getUser.getVehicleList();

            if (getVehicleList.size() > 0){
                noVehicleTv.setVisibility(View.GONE);

                vehicleListRV.setLayoutManager(new LinearLayoutManager(vehicleList.this));
                adapter = new vehicleListAdapter(vehicleList.this, getVehicleList);
                vehicleListRV.setAdapter(adapter);
```

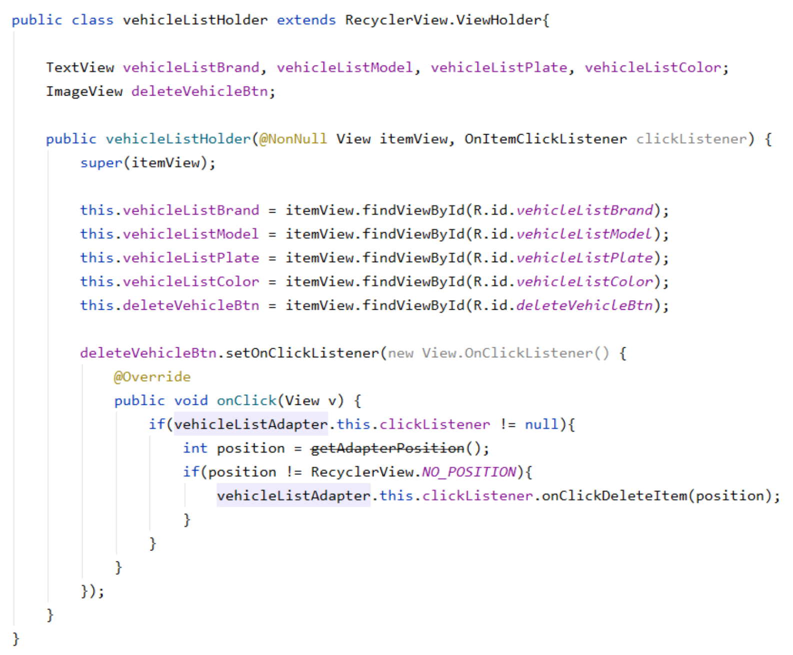
Figure 3-6-9-1-1: Retrieve User’s Vehicle from Firestore



```
@NonNull
@Override
public vehicleListHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.vehicle_list_row, null, false);
    return new vehicleListHolder(view, clickListener);
}
```

Figure 3-6-9-1-2: Display User’s Vehicle Adapter (1)

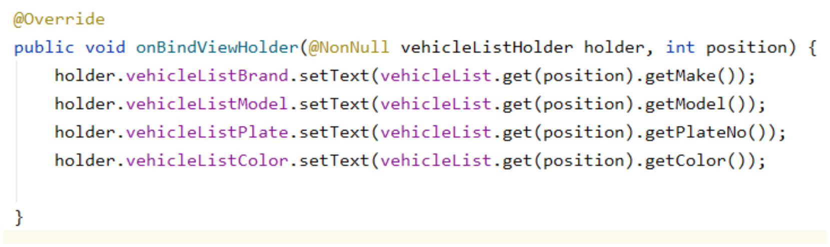
Figure 3-6-9-1-1 shows the coding of retrieve user's vehicle from Firestore function. The vehicle is retrieved from the "user" collection from Firestore. System get the vehicle list from user's document. The vehicle list is display using recycler view. Recycler view is used to hold all the vehicle in vehicle list. The list is then passed into the adapter. The adapter is used to handle each vehicle displayed in recycler view. In figure 3-6-9-1-2, the inflator is used to inflate the vehicle row design file into a view.



```
public class vehicleListHolder extends RecyclerView.ViewHolder{

    TextView vehicleListBrand, vehicleListModel, vehicleListPlate, vehicleListColor;
    ImageView deleteVehicleBtn;

    public vehicleListHolder(@NonNull View itemView, OnItemClickListener clickListener) {
        super(itemView);

        this.vehicleListBrand = itemView.findViewById(R.id.vehicleListBrand);
        this.vehicleListModel = itemView.findViewById(R.id.vehicleListModel);
        this.vehicleListPlate = itemView.findViewById(R.id.vehicleListPlate);
        this.vehicleListColor = itemView.findViewById(R.id.vehicleListColor);
        this.deleteVehicleBtn = itemView.findViewById(R.id.deleteVehicleBtn);

        deleteVehicleBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(vehicleListAdapter.this.clickListener != null){
                    int position = ~~getAdapterPosition~~();
                    if(position != RecyclerView.NO_POSITION){
                        vehicleListAdapter.this.clickListener.onClickDeleteItem(position);
                    }
                }
            }
        });
    }
}
```

Figure 3-6-9-1-3: Display User's Vehicle Adapter (2)

Firstly, all the components in the inflated view need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-9-1-3, component such as the vehicle model, brand, plate number and color is declared. Onclick() function is triggered when the user clicked on the delete button in the recycler view. It will return the position of the vehicle in the vehicle list.



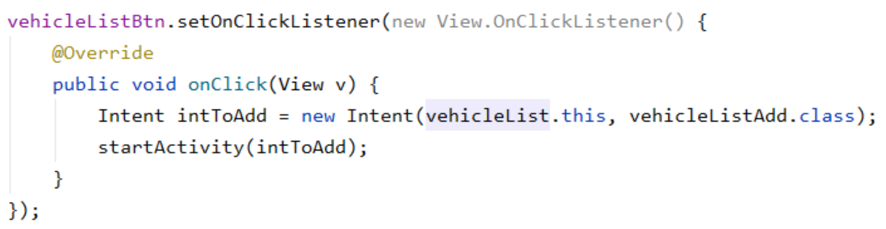
```
@Override
public void onBindViewHolder(@NonNull vehicleListHolder holder, int position) {
    holder.vehicleListBrand.setText(vehicleList.get(position).getMake());
    holder.vehicleListModel.setText(vehicleList.get(position).getModel());
    holder.vehicleListPlate.setText(vehicleList.get(position).getPlateNo());
    holder.vehicleListColor.setText(vehicleList.get(position).getColor());

}
```

Figure 3-6-9-1-4: Display User's Vehicle Adapter (3)

Each of the vehicle in the vehicle array list is retrieved and the details is displayed. The onBindViewHolder() is used to hold each of the component value of the inflated view.

The getter and setter function defined in the vehicle class is used to get the value from the vehicle in vehicle list.

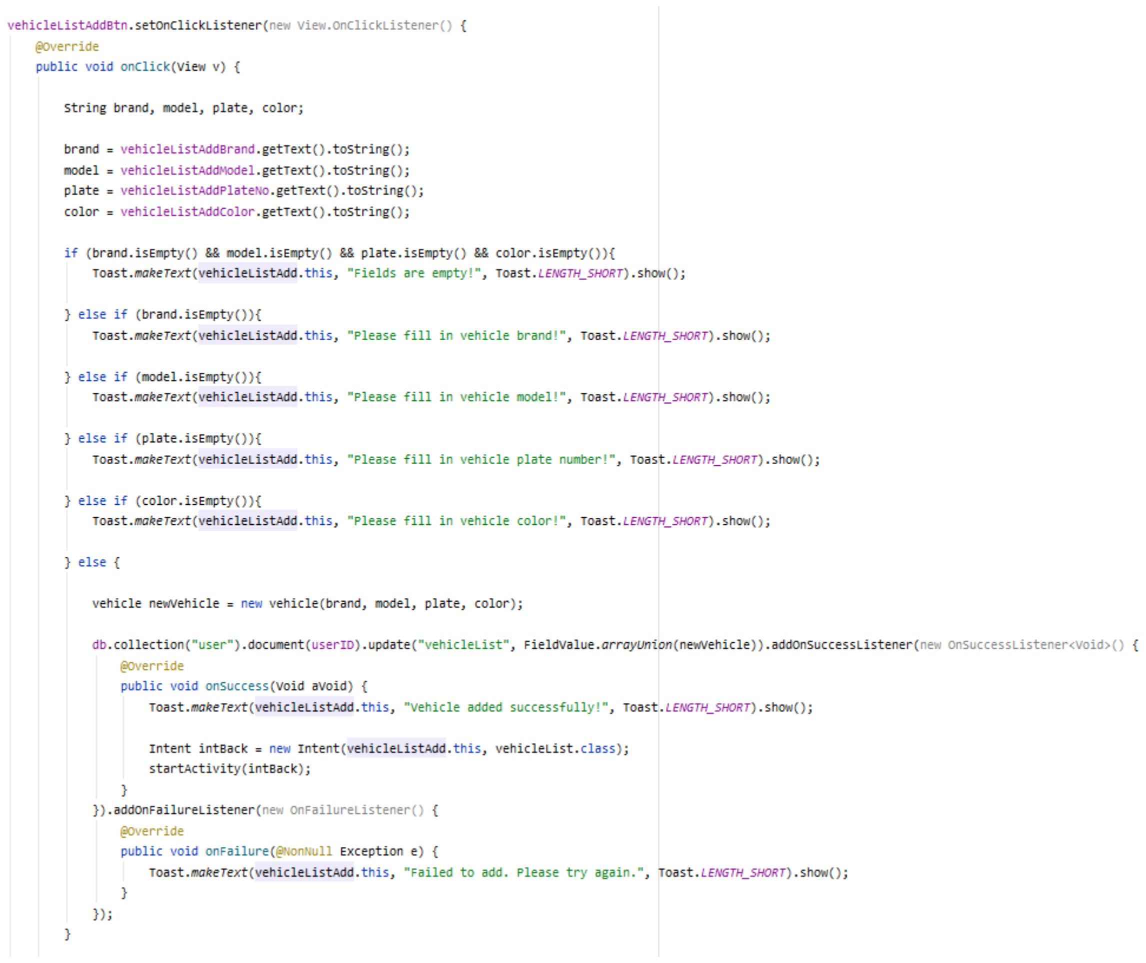
3.6.9.2 Add Vehicle



```java
vehicleListBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intToAdd = new Intent(vehicleList.this, vehicleListAdd.class);
        startActivity(intToAdd);
    }
});
```

Figure 3-6-9-2-1: Intent to Add Vehicle Activity

When user clicks on the “add vehicle” button, the system will direct user to the add vehicle page.

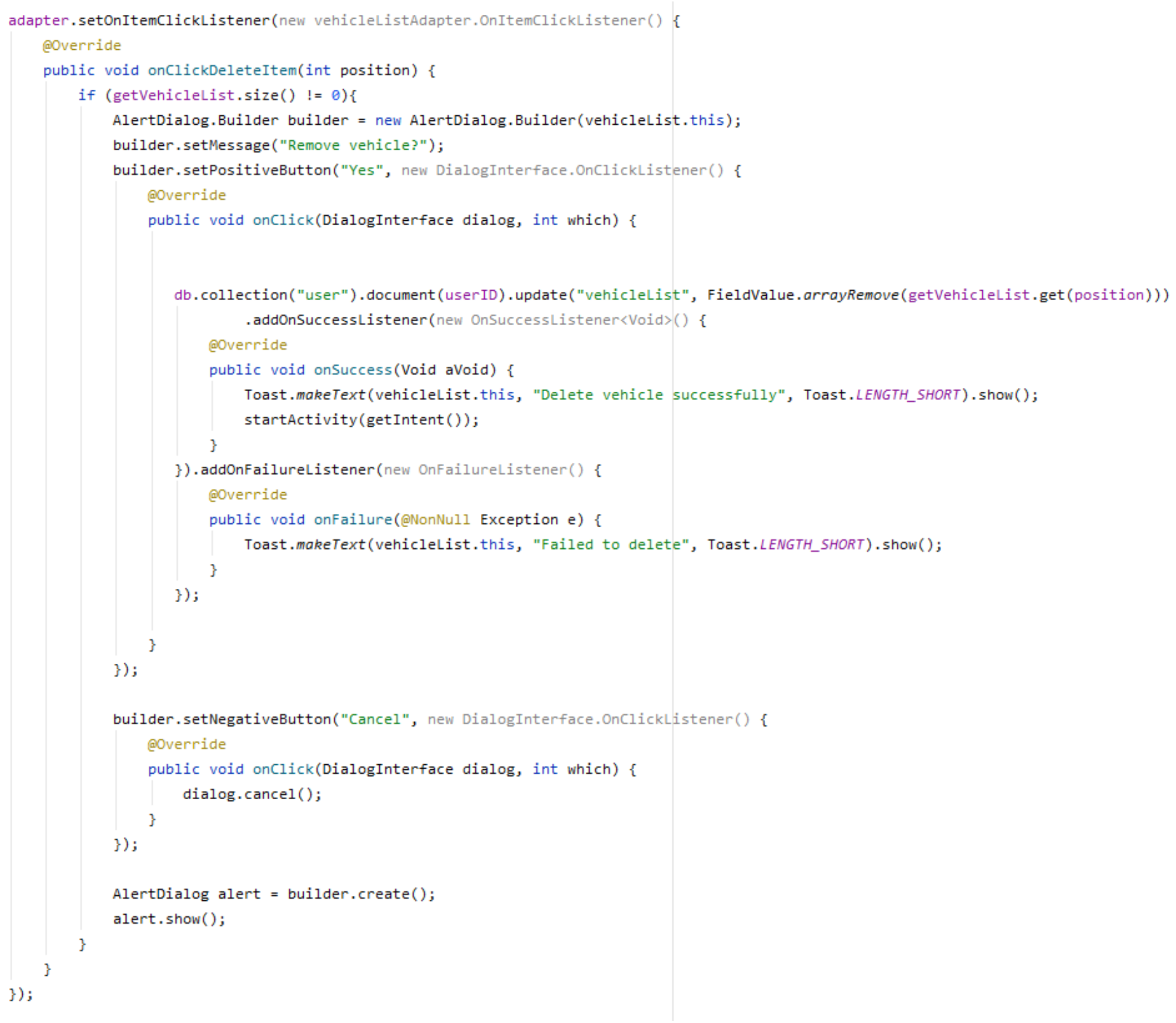


```java
vehicleListAddBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String brand, model, plate, color;

        brand = vehicleListAddBrand.getText().toString();
        model = vehicleListAddModel.getText().toString();
        plate = vehicleListAddPlateNo.getText().toString();
        color = vehicleListAddColor.getText().toString();

        if (brand.isEmpty() && model.isEmpty() && plate.isEmpty() && color.isEmpty()){
            Toast.makeText(vehicleListAdd.this, "Fields are empty!", Toast.LENGTH_SHORT).show();

        } else if (brand.isEmpty()){
            Toast.makeText(vehicleListAdd.this, "Please fill in vehicle brand!", Toast.LENGTH_SHORT).show();

        } else if (model.isEmpty()){
            Toast.makeText(vehicleListAdd.this, "Please fill in vehicle model!", Toast.LENGTH_SHORT).show();

        } else if (plate.isEmpty()){
            Toast.makeText(vehicleListAdd.this, "Please fill in vehicle plate number!", Toast.LENGTH_SHORT).show();

        } else if (color.isEmpty()){
            Toast.makeText(vehicleListAdd.this, "Please fill in vehicle color!", Toast.LENGTH_SHORT).show();

        } else {

            vehicle newVehicle = new vehicle(brand, model, plate, color);

            db.collection("user").document(userID).update("vehicleList", FieldValue.arrayUnion(newVehicle)).addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    Toast.makeText(vehicleListAdd.this, "Vehicle added successfully!", Toast.LENGTH_SHORT).show();

                    Intent intBack = new Intent(vehicleListAdd.this, vehicleList.class);
                    startActivity(intBack);
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(vehicleListAdd.this, "Failed to add. Please try again.", Toast.LENGTH_SHORT).show();
                }
            });
        }
```

Figure 3-6-9-2-2: Add User Vehicle

Figure 3-6-9-2-2 shows the coding of add user vehicle function. When user clicks on the “add” button in add user vehicle page, the onClick() function is triggered. The user input is retrieved by the system. The system will then check if there are any missing inputs. If there is missing inputs, error message is displayed to user. If the input is complete, the system will update the database by adding the vehicle to the user's document in Firestore.

3.6.9.3 Delete Vehicle

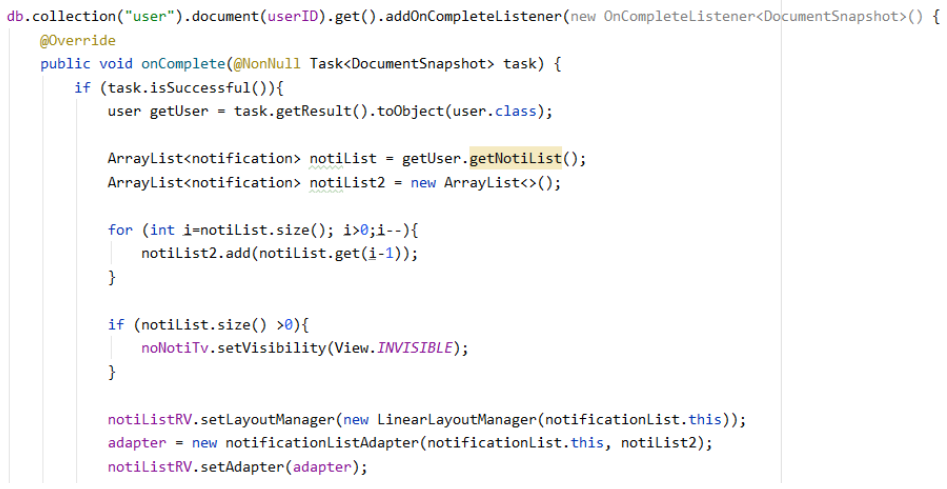


```
adapter.setOnItemClickListener(new vehicleListAdapter.OnItemClickListener() {
    @Override
    public void onClickDeleteItem(int position) {
        if (getVehicleList.size() != 0){
            AlertDialog.Builder builder = new AlertDialog.Builder(vehicleList.this);
            builder.setMessage("Remove vehicle?");
            builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {


                    db.collection("user").document(userID).update("vehicleList", FieldValue.arrayRemove(getVehicleList.get(position)))
                            .addOnSuccessListener(new OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            Toast.makeText(vehicleList.this, "Delete vehicle successfully", Toast.LENGTH_SHORT).show();
                            startActivity(getIntent());
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            Toast.makeText(vehicleList.this, "Failed to delete", Toast.LENGTH_SHORT).show();
                        }
                    });

                }
            });

            builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            });

            AlertDialog alert = builder.create();
            alert.show();
        }
    }
});
```

Figure 3-6-9-3-1: Delete User's Vehicle

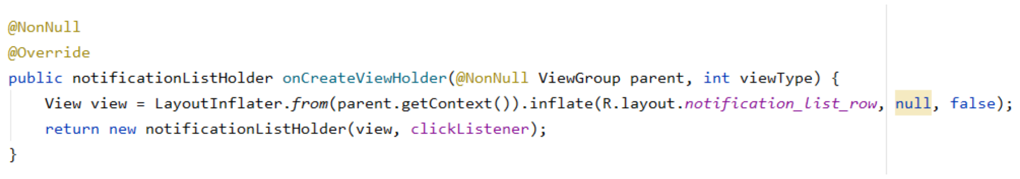
Figure 3-6-9-3-1 shows the delete user's vehicle function. When the delete button of the vehicle is clicked, onClickDeleteItem() function is triggered. The system will pop out an alert dialog to confirm the deletion. If the user clicked "Yes", the vehicle will be removed from the user's document in Firestore. if the user clicked "Cancel", the alert dialog will be closed and no changes happened.

3.6.10 View Notification



```
db.collection("user").document(userID).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        if (task.isSuccessful()){
            user getUser = task.getResult().toObject(user.class);

            ArrayList<notification> notiList = getUser.getNotiList();
            ArrayList<notification> notiList2 = new ArrayList<>();

            for (int i=notiList.size(); i>0;i--){
                notiList2.add(notiList.get(i-1));
            }

            if (notiList.size() >0){
                noNotiTv.setVisibility(View.INVISIBLE);
            }

            notiListRV.setLayoutManager(new LinearLayoutManager(notificationList.this));
            adapter = new notificationListAdapter(notificationList.this, notiList2);
            notiListRV.setAdapter(adapter);
```

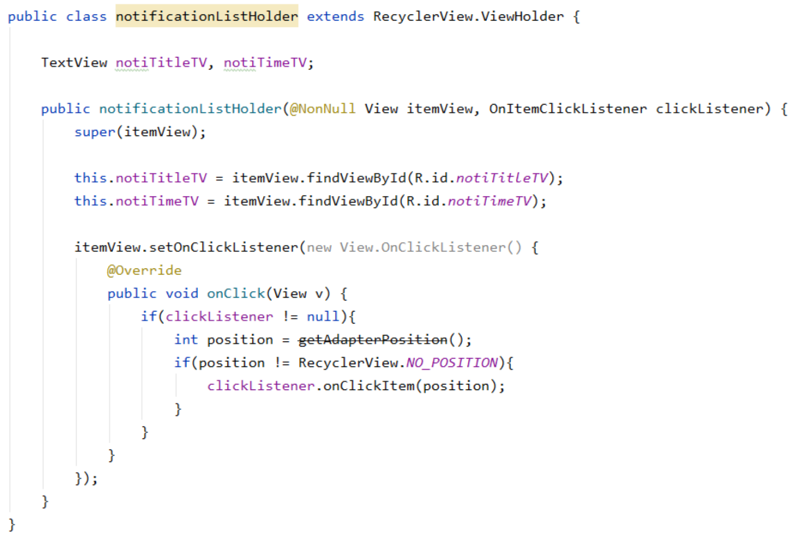
Figure 3-6-10-1: Get Notification from Firestore



```
@NonNull
@Override
public notificationListHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.notification_list_row, null, false);
    return new notificationListHolder(view, clickListener);
}
```

Figure 3-6-10-2: Display Notification Adapter (1)

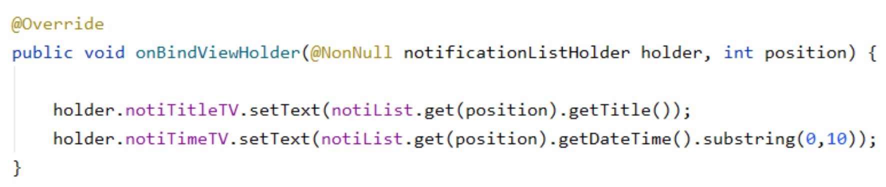
Figure 3-6-10-1shows the coding of retrieve user’s notification from Firestore function. The notification is retrieved from the “user” collection from Firestore. System get the notification list from user’s document. The notification list is display using recycler view. Recycler view is used to hold all the notification in notification list. The list is then passed into the adapter. The adapter is used to handle each notification displayed in recycler view. In figure 3-6-10-2, the inflator is used to inflate the notification row design file into a view.



```
public class notificationListHolder extends RecyclerView.ViewHolder {

    TextView notiTitleTV, notiTimeTV;

    public notificationListHolder(@NonNull View itemView, OnItemClickListener clickListener) {
        super(itemView);

        this.notiTitleTV = itemView.findViewById(R.id.notiTitleTV);
        this.notiTimeTV = itemView.findViewById(R.id.notiTimeTV);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(clickListener != null){
                    int position = ~~getAdapterPosition~~();
                    if(position != RecyclerView.NO_POSITION){
                        clickListener.onClickItem(position);
                    }
                }
            }
        });
    }
}
```

Figure 3-6-10-3: Display Notification Adapter (2)

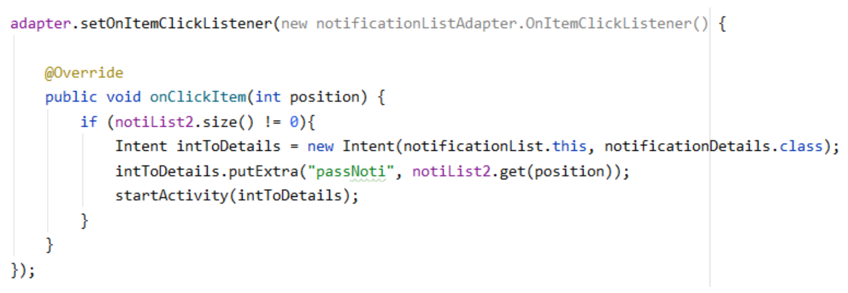
View holder need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-10-3, component such as the notification title, content and date are declared. Onclick() function is triggered when the user clicked on the notification in the recycler view. It will return the position of the notification in the notification list.



```
@Override
public void onBindViewHolder(@NonNull notificationListHolder holder, int position) {

    holder.notiTitleTV.setText(notiList.get(position).getTitle());
    holder.notiTimeTV.setText(notiList.get(position).getDateTime().substring(0,10));
}
```

Figure 3-6-10-4: Display Notification Adapter (3)

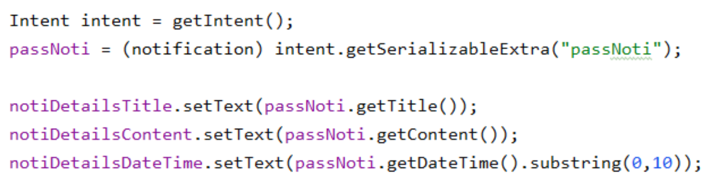
Each of the notification in the notification array list is retrieved and the details is displayed. The holder is used to hold each of the component value. The getter and setter function defined in the notification class is used to get the value from the notification in notification list.



```
adapter.setOnItemClickListener(new notificationListAdapter.OnItemClickListener() {

    @Override
    public void onClickItem(int position) {
        if (notiList2.size() != 0){
            Intent intToDetails = new Intent(notificationList.this, notificationDetails.class);
            intToDetails.putExtra("passNoti", notiList2.get(position));
            startActivity(intToDetails);
        }
    }
});
```

Figure 3-6-10-5: Intent to Notification Details Activity

When user clicked on the notification, the system will direct user to the next activity which is the notification details activity. The system gets the position of the notification in the notification list and get the notification passed to the next activity using intent.



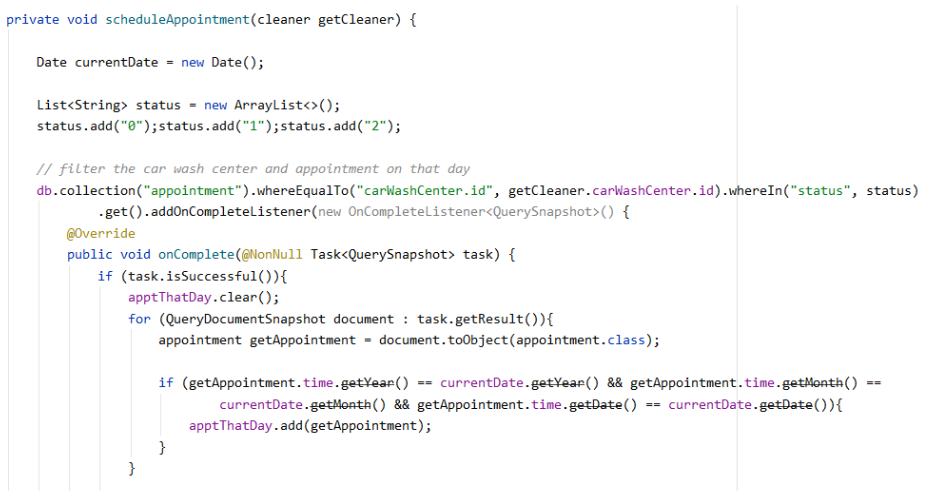
```
Intent intent = getIntent();
passNoti = (notification) intent.getSerializableExtra("passNoti");

notiDetailsTitle.setText(passNoti.getTitle());
notiDetailsContent.setText(passNoti.getContent());
notiDetailsDateTime.setText(passNoti.getDateTime().substring(0,10));
```

Figure 3-6-10-6: Display Notification Details

The notification passed from the previous activity is received here. The system gets the notification using intent. Then, the system retrieve the notification data using getter()

function defined in the notification class. The value get is then set to the respective text view using setText() function.

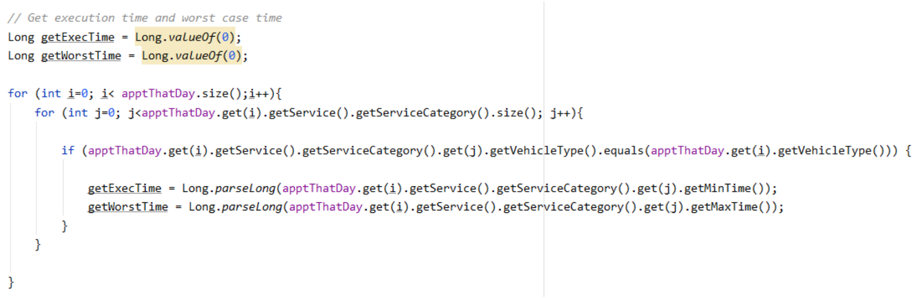
3.6.11 Scheduled Appointment



```java
private void scheduleAppointment(cleaner getCleaner) {

    Date currentDate = new Date();

    List<String> status = new ArrayList<>();
    status.add("0");status.add("1");status.add("2");

    // filter the car wash center and appointment on that day
    db.collection("appointment").whereEqualTo("carWashCenter.id", getCleaner.carWashCenter.id).whereIn("status", status)
            .get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()){
                apptThatDay.clear();
                for (QueryDocumentSnapshot document : task.getResult()){
                    appointment getAppointment = document.toObject(appointment.class);

                    if (getAppointment.time.~~getYear~~() == currentDate.~~getYear~~() && getAppointment.time.~~getMonth~~() ==
                            currentDate.~~getMonth~~() && getAppointment.time.~~getDate~~() == currentDate.~~getDate~~()){
                        apptThatDay.add(getAppointment);
                    }
                }
```

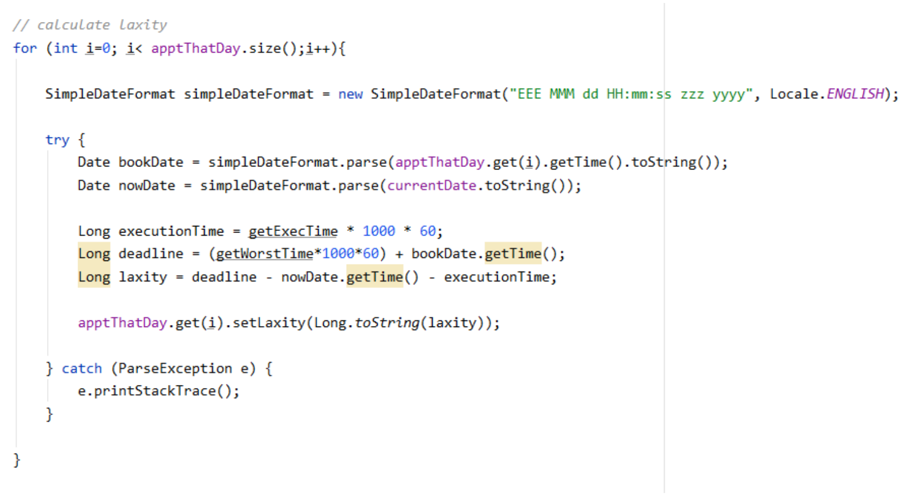
Figure 3-6-11-1: Get Appointment From Firestore

Figure 3-6-11-1 shows the coding of get appointment from "appointment" the Firestore. The system will filter the appointment using the car wash center ID. Then, the system will filter the appointment today by checking the year, month and date. The appointment made for today is then added into an array list.



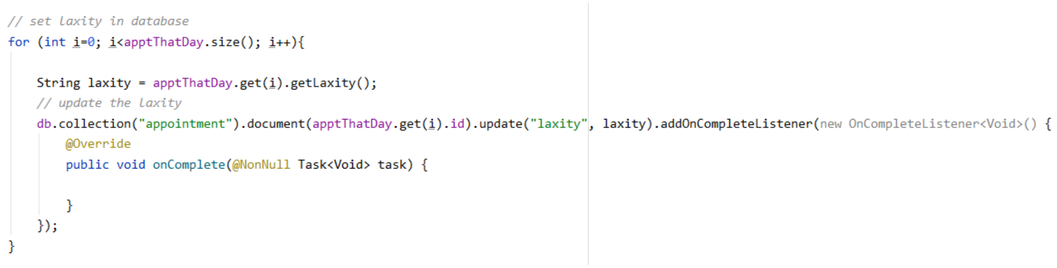
```java
// Get execution time and worst case time
Long getExecTime = Long.valueOf(0);
Long getWorstTime = Long.valueOf(0);

for (int i=0; i< apptThatDay.size();i++){
    for (int j=0; j<apptThatDay.get(i).getService().getServiceCategory().size(); j++){

        if (apptThatDay.get(i).getService().getServiceCategory().get(j).getVehicleType().equals(apptThatDay.get(i).getVehicleType())) {

            getExecTime = Long.parseLong(apptThatDay.get(i).getService().getServiceCategory().get(j).getMinTime());
            getWorstTime = Long.parseLong(apptThatDay.get(i).getService().getServiceCategory().get(j).getMaxTime());
        }
    }

}
```

Figure 3-6-11-2: Schedule Appointment (1)



```
// calculate laxity
for (int i=0; i< apptThatDay.size();i++){

    SimpleDateFormat simpleDateFormat = new SimpleDateFormat("EEE MMM dd HH:mm:ss zzz yyyy", Locale.ENGLISH);

    try {
        Date bookDate = simpleDateFormat.parse(apptThatDay.get(i).getTime().toString());
        Date nowDate = simpleDateFormat.parse(currentDate.toString());

        Long executionTime = getExecTime * 1000 * 60;
        Long deadline = (getWorstTime*1000*60) + bookDate.getTime();
        Long laxity = deadline - nowDate.getTime() - executionTime;

        apptThatDay.get(i).setLaxity(Long.toString(laxity));

    } catch (ParseException e) {
        e.printStackTrace();
    }

}
```

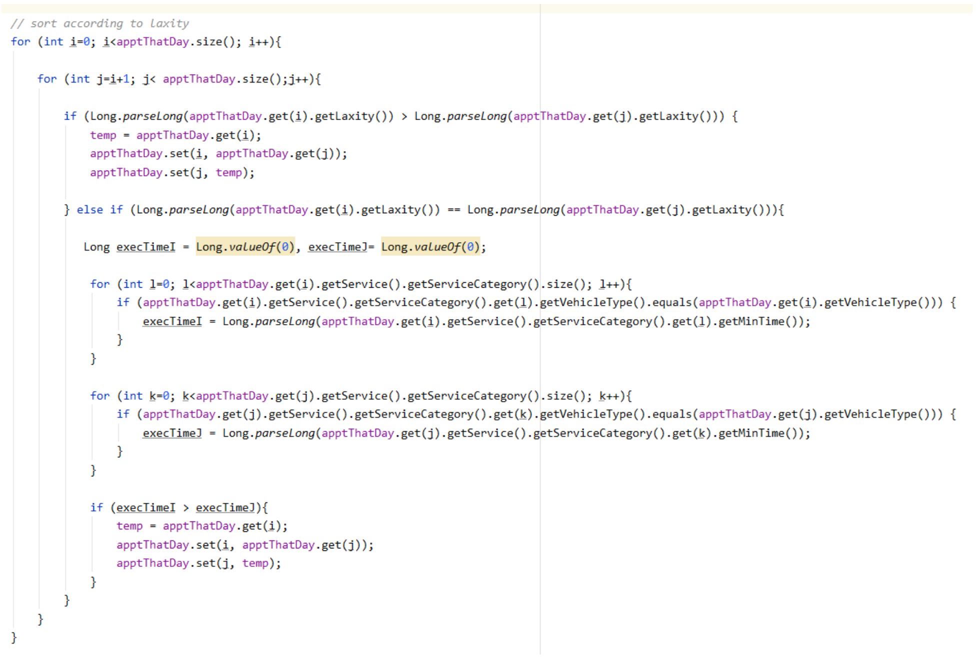
Figure 3-6-11-3: Schedule Appointment (2)



```
// set laxity in database
for (int i=0; i<apptThatDay.size(); i++){

    String laxity = apptThatDay.get(i).getLaxity();
    // update the laxity
    db.collection("appointment").document(apptThatDay.get(i).id).update("laxity", laxity).addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {

        }
    });
}
```

Figure 3-6-11-4: Schedule Appointment (3)

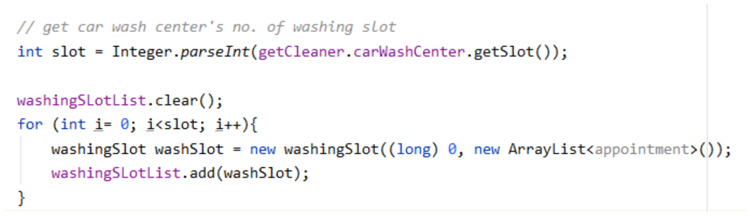
Firstly, in Figure 3-6-11-2, the execution time and the worst case execution time is obtained from the appointment retrieved from the Firestore. Then, the execution time and worst case execution time is used to calculate the laxity of each of the appointment as shown in Figure 3-6-11-3. All the variable such as execution time, deadline, and laxity is calculated in milliseconds and stored as Long. The laxity is calculated and then it is updated in the Firestore in the particular appointment's document.



```java
// sort according to laxity
for (int i=0; i<apptThatDay.size(); i++){

    for (int j=i+1; j< apptThatDay.size();j++){

        if (Long.parseLong(apptThatDay.get(i).getLaxity()) > Long.parseLong(apptThatDay.get(j).getLaxity())) {
            temp = apptThatDay.get(i);
            apptThatDay.set(i, apptThatDay.get(j));
            apptThatDay.set(j, temp);

        } else if (Long.parseLong(apptThatDay.get(i).getLaxity()) == Long.parseLong(apptThatDay.get(j).getLaxity())){

           Long execTimeI = Long.valueOf(0), execTimeJ= Long.valueOf(0);

            for (int l=0; l<apptThatDay.get(i).getService().getServiceCategory().size(); l++){
                if (apptThatDay.get(i).getService().getServiceCategory().get(l).getVehicleType().equals(apptThatDay.get(i).getVehicleType())) {
                    execTimeI = Long.parseLong(apptThatDay.get(i).getService().getServiceCategory().get(l).getMinTime());
                }
            }

            for (int k=0; k<apptThatDay.get(j).getService().getServiceCategory().size(); k++){
                if (apptThatDay.get(j).getService().getServiceCategory().get(k).getVehicleType().equals(apptThatDay.get(j).getVehicleType())) {
                    execTimeJ = Long.parseLong(apptThatDay.get(j).getService().getServiceCategory().get(k).getMinTime());
                }
            }

            if (execTimeI > execTimeJ){
                temp = apptThatDay.get(i);
                apptThatDay.set(i, apptThatDay.get(j));
                apptThatDay.set(j, temp);
            }
        }
    }
}
```

Figure 3-6-11-5: Schedule Appointment (4)

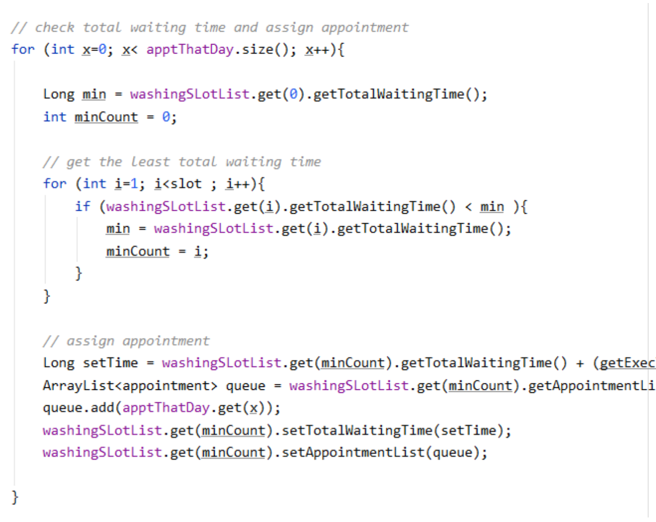
In figure 3-6-11-5 shows the appointment in the appointment array list is sorted according to the laxity value. The laxity is sorted in ascending order. Hence, the appointment that has the smallest laxity value will at the first place in array list. If both appointment has the same laxity value, the system will choose the appointment that has the shorter execution time means that the appointment has shorter execution time has the higher priority.



```java
// get car wash center's no. of washing slot
int slot = Integer.parseInt(getCleaner.carWashCenter.getSlot());

washingSLotList.clear();
for (int i= 0; i<slot; i++){
    washingSlot washSlot = new washingSlot((long) 0, new ArrayList<appointment>());
    washingSLotList.add(washSlot);
}
```

Figure 3-6-11-6: Schedule Appointment (5)

Next, the number of washing slot in the car wash center is obtain from the document of “car wash center” collection in the Firestore. The system will then create a number of washing slot according to the number of washing slot the car wash center has.

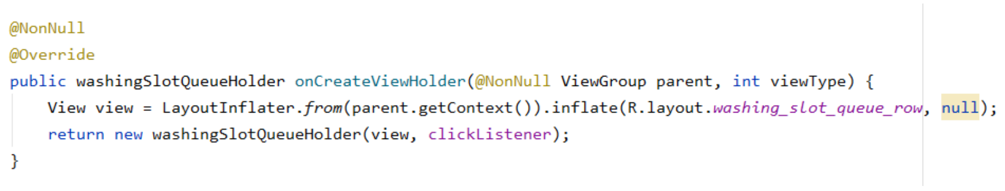


```
// check total waiting time and assign appointment
for (int x=0; x< apptThatDay.size(); x++){

    Long min = washingSLotList.get(0).getTotalWaitingTime();
    int minCount = 0;

    // get the least total waiting time
    for (int i=1; i<slot ; i++){
        if (washingSLotList.get(i).getTotalWaitingTime() < min ){
            min = washingSLotList.get(i).getTotalWaitingTime();
            minCount = i;
        }
    }

    // assign appointment
    Long setTime = washingSLotList.get(minCount).getTotalWaitingTime() + (getExecTime*1000*60);
    ArrayList<appointment> queue = washingSLotList.get(minCount).getAppointmentList();
    queue.add(apptThatDay.get(x));
    washingSLotList.get(minCount).setTotalWaitingTime(setTime);
    washingSLotList.get(minCount).setAppointmentList(queue);

}
```

Figure 3-6-11-7: Schedule Appointment (6)

The following steps of scheduling the appointment are check the total waiting time of each of the washing slot. By default, each of the car wash center’s washing slot’s waiting time is 0. The system will check all the washing slot and find the washing slot that has the shortest waiting time. And lastly, the appointment is added into the washing slot that has the shortest waiting time. The execution time of appointment is added to the total waiting time of the washing slot.

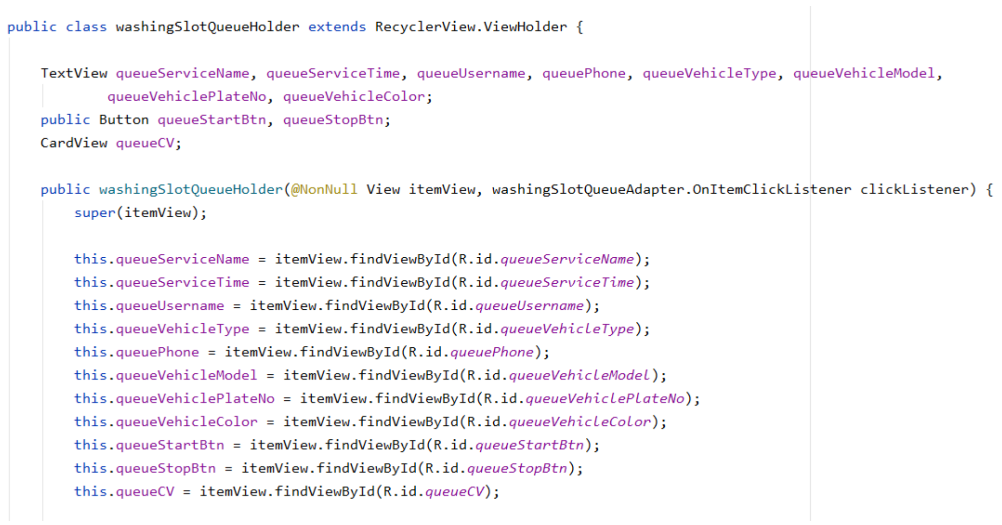
3.6.12 Display Scheduled Appointment



```
@NonNull
@Override
public washingSlotQueueHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.washing_slot_queue_row, null);
    return new washingSlotQueueHolder(view, clickListener);
}
```

Figure 3-6-12-1: Display Scheduled Appointment Adapter (1)

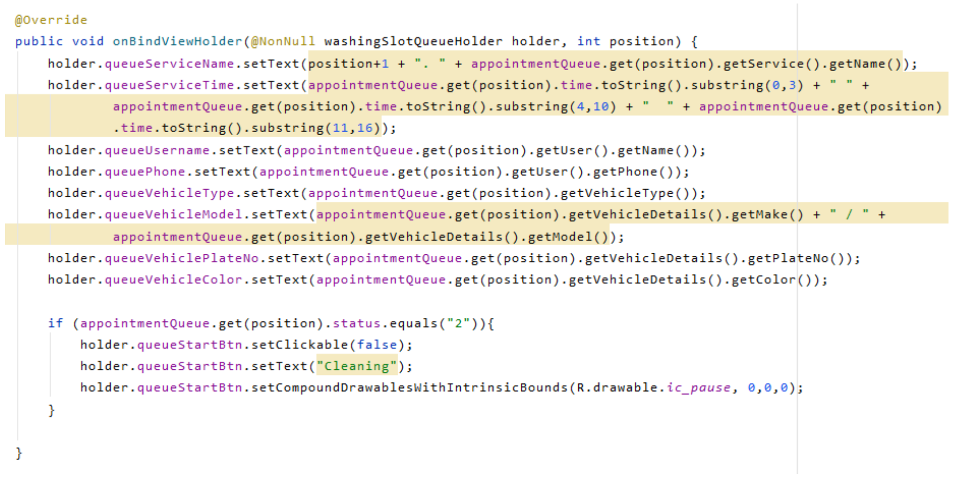
The scheduled appointment list is display using recycler view. Recycler view is used to hold all the appointment in scheduled appointment list. The adapter is used to handle each appointment displayed in recycler view. In figure 3-6-12-1, the inflator is used to inflate the scheduled appointment row design file into a view.



```java
public class washingSlotQueueHolder extends RecyclerView.ViewHolder {

    TextView queueServiceName, queueServiceTime, queueUsername, queuePhone, queueVehicleType, queueVehicleModel,
            queueVehiclePlateNo, queueVehicleColor;
    public Button queueStartBtn, queueStopBtn;
    CardView queueCV;

    public washingSlotQueueHolder(@NonNull View itemView, washingSlotQueueAdapter.OnItemClickListener clickListener) {
        super(itemView);

        this.queueServiceName = itemView.findViewById(R.id.queueServiceName);
        this.queueServiceTime = itemView.findViewById(R.id.queueServiceTime);
        this.queueUsername = itemView.findViewById(R.id.queueUsername);
        this.queueVehicleType = itemView.findViewById(R.id.queueVehicleType);
        this.queuePhone = itemView.findViewById(R.id.queuePhone);
        this.queueVehicleModel = itemView.findViewById(R.id.queueVehicleModel);
        this.queueVehiclePlateNo = itemView.findViewById(R.id.queueVehiclePlateNo);
        this.queueVehicleColor = itemView.findViewById(R.id.queueVehicleColor);
        this.queueStartBtn = itemView.findViewById(R.id.queueStartBtn);
        this.queueStopBtn = itemView.findViewById(R.id.queueStopBtn);
        this.queueCV = itemView.findViewById(R.id.queueCV);
```

Figure 3-6-12-2: Display Scheduled Appointment Adapter (2)

View holder need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-12-2, component such as the user details, service details and vehicle details are declared.

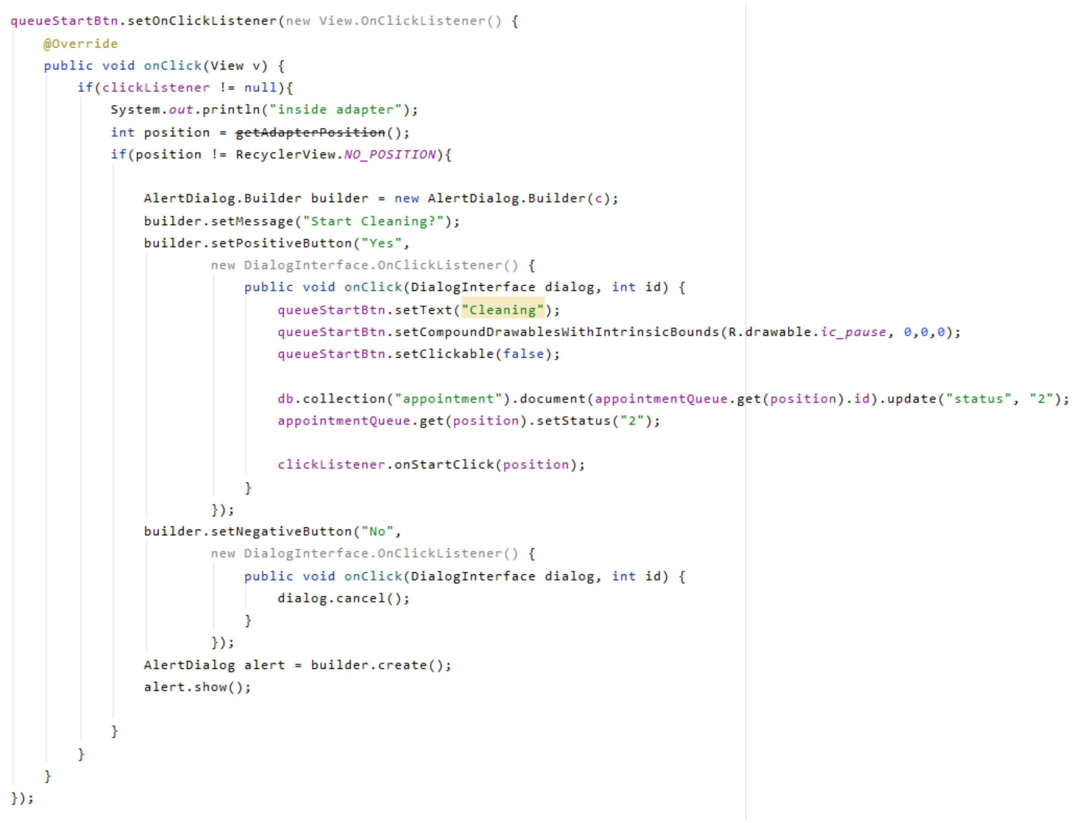


```java
@Override
public void onBindViewHolder(@NonNull washingSlotQueueHolder holder, int position) {
    holder.queueServiceName.setText(position+1 + ". " + appointmentQueue.get(position).getService().getName());
    holder.queueServiceTime.setText(appointmentQueue.get(position).time.toString().substring(0,3) + " " +
            appointmentQueue.get(position).time.toString().substring(4,10) + "  " + appointmentQueue.get(position)
            .time.toString().substring(11,16));
    holder.queueUsername.setText(appointmentQueue.get(position).getUser().getName());
    holder.queuePhone.setText(appointmentQueue.get(position).getUser().getPhone());
    holder.queueVehicleType.setText(appointmentQueue.get(position).getVehicleType());
    holder.queueVehicleModel.setText(appointmentQueue.get(position).getVehicleDetails().getMake() + " / " +
            appointmentQueue.get(position).getVehicleDetails().getModel());
    holder.queueVehiclePlateNo.setText(appointmentQueue.get(position).getVehicleDetails().getPlateNo());
    holder.queueVehicleColor.setText(appointmentQueue.get(position).getVehicleDetails().getColor());

    if (appointmentQueue.get(position).status.equals("2")){
        holder.queueStartBtn.setClickable(false);
        holder.queueStartBtn.setText("Cleaning");
        holder.queueStartBtn.setCompoundDrawablesWithIntrinsicBounds(R.drawable.ic_pause, 0,0,0);
    }

}
```

Figure 3-6-12-3: Display Scheduled Appointment Adapter (3)

Each of the appointment in the scheduled appointment list is retrieved by position and the details is displayed. The holder is used to hold each of the component value. The getter function defined in the appointment class is used to get the value from the appointment. When the status of the appointment is equal to 2 which indicates it is in cleaning process, the status of the appointment will be displayed.

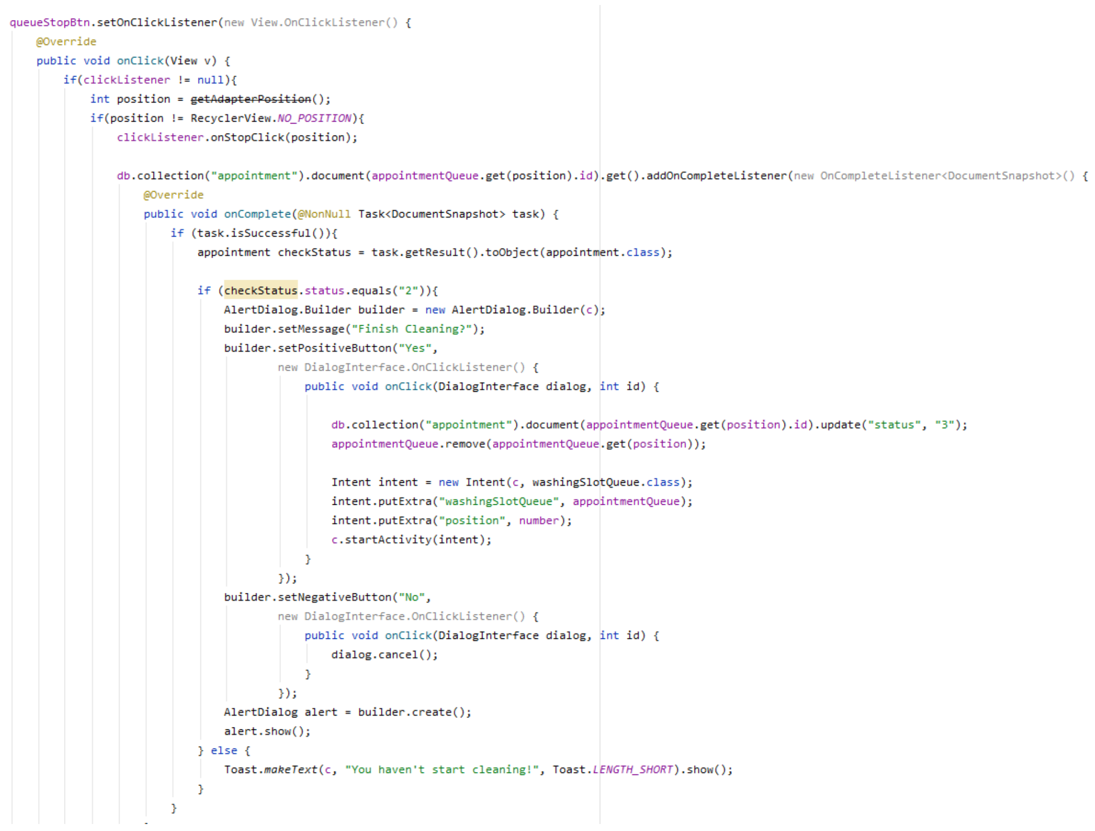
3.6.13 Manage Appointment



```
queueStartBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(clickListener != null){
            System.out.println("inside adapter");
            int position = ~~getAdapterPosition~~();
            if(position != RecyclerView.NO_POSITION){

                AlertDialog.Builder builder = new AlertDialog.Builder(c);
                builder.setMessage("Start Cleaning?");
                builder.setPositiveButton("Yes",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                queueStartBtn.setText("Cleaning");
                                queueStartBtn.setCompoundDrawablesWithIntrinsicBounds(R.drawable.ic_pause, 0,0,0);
                                queueStartBtn.setClickable(false);

                                db.collection("appointment").document(appointmentQueue.get(position).id).update("status", "2");
                                appointmentQueue.get(position).setStatus("2");

                                clickListener.onStartClick(position);
                            }
                        });
                builder.setNegativeButton("No",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                dialog.cancel();
                            }
                        });
                AlertDialog alert = builder.create();
                alert.show();

            }
        }
    }
});
```

Figure 3-6-13-1: Manage Appointment (1)

When user clicked on the “Start” button in the item of the appointment view, it indicates the appointment is going to start cleaning process. The onClick() function is then triggered. An alert dialog is pop out for user to confirm the starting of cleaning process. If the user clicked “Yes”, then icon display will change to a cleaning icon. System will update the status of appointment in the “appointment” collection in Firestore.



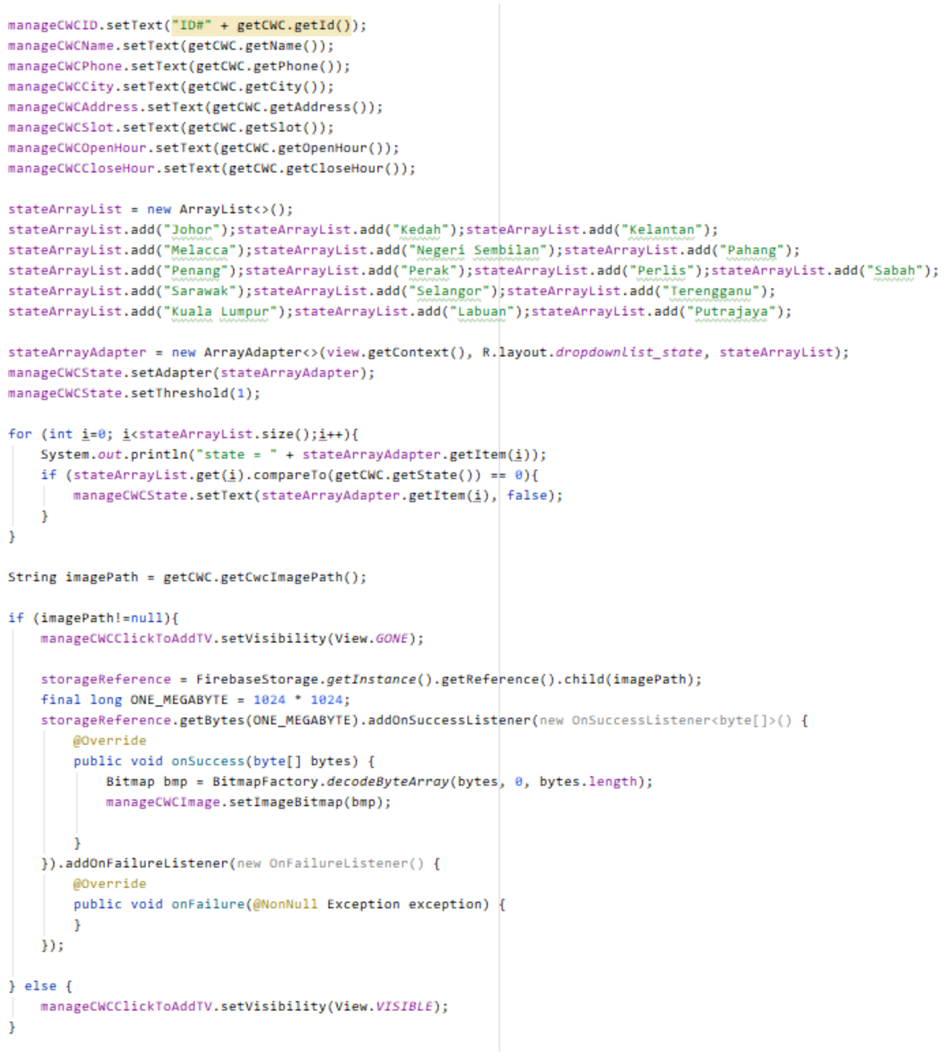
```
queueStopBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(clickListener != null){
            int position = getAdapterPosition();
            if(position != RecyclerView.NO_POSITION){
                clickListener.onStopClick(position);

                db.collection("appointment").document(appointmentQueue.get(position).id).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                        if (task.isSuccessful()){
                            appointment checkStatus = task.getResult().toObject(appointment.class);

                            if (checkStatus.status.equals("2")){
                                AlertDialog.Builder builder = new AlertDialog.Builder(c);
                                builder.setMessage("Finish Cleaning?");
                                builder.setPositiveButton("Yes",
                                        new DialogInterface.OnClickListener() {
                                            public void onClick(DialogInterface dialog, int id) {

                                                db.collection("appointment").document(appointmentQueue.get(position).id).update("status", "3");
                                                appointmentQueue.remove(appointmentQueue.get(position));

                                                Intent intent = new Intent(c, washingSlotQueue.class);
                                                intent.putExtra("washingSlotQueue", appointmentQueue);
                                                intent.putExtra("position", number);
                                                c.startActivity(intent);
                                            }
                                        });
                                builder.setNegativeButton("No",
                                        new DialogInterface.OnClickListener() {
                                            public void onClick(DialogInterface dialog, int id) {
                                                dialog.cancel();
                                            }
                                        });
                                AlertDialog alert = builder.create();
                                alert.show();
                            } else {
                                Toast.makeText(c, "You haven't start cleaning!", Toast.LENGTH_SHORT).show();
                            }
                        }
```

Figure 3-6-13-2: Manage Appointment (2)

When user clicked on the "Finish" button in the item of the appointment view, it indicates the appointment is going to end cleaning process. The onClick() function is then triggered. An alert dialog is pop out for user to confirm the starting of cleaning process. If the user clicked "Yes", then icon display will change to a cleaning icon. The system will firstly check whether the appointment had already started cleaning process as an appointment must be in cleaning process first before the appointment can be ended. If the appointment has not started yet, the system will display error message to user. If the appointment has already in cleaning process, the system will update the status of appointment in the "appointment" collection in Firestore.

3.6.14 Manage Car Wash Center Details

3.6.14.1 View Car Wash Center Details

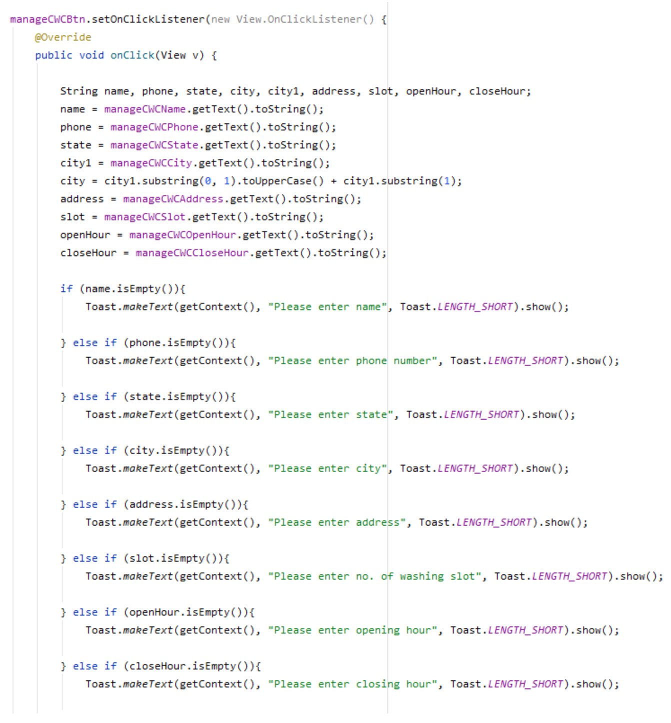


```
manageCWCID.setText("ID#" + getCWC.getId());
manageCWCName.setText(getCWC.getName());
manageCWCPhone.setText(getCWC.getPhone());
manageCWCCity.setText(getCWC.getCity());
manageCWCAddress.setText(getCWC.getAddress());
manageCWCSlot.setText(getCWC.getSlot());
manageCWCOpenHour.setText(getCWC.getOpenHour());
manageCWCCloseHour.setText(getCWC.getCloseHour());

stateArrayList = new ArrayList<>();
stateArrayList.add("Johor");stateArrayList.add("Kedah");stateArrayList.add("Kelantan");
stateArrayList.add("Melacca");stateArrayList.add("Negeri Sembilan");stateArrayList.add("Pahang");
stateArrayList.add("Penang");stateArrayList.add("Perak");stateArrayList.add("Perlis");stateArrayList.add("Sabah");
stateArrayList.add("Sarawak");stateArrayList.add("Selangor");stateArrayList.add("Terengganu");
stateArrayList.add("Kuala Lumpur");stateArrayList.add("Labuan");stateArrayList.add("Putrajaya");

stateArrayAdapter = new ArrayAdapter<>(view.getContext(), R.layout.dropdownlist_state, stateArrayList);
manageCWCState.setAdapter(stateArrayAdapter);
manageCWCState.setThreshold(1);

for (int i=0; i<stateArrayList.size();i++){
    System.out.println("state = " + stateArrayAdapter.getItem(i));
    if (stateArrayList.get(i).compareTo(getCWC.getState()) == 0){
        manageCWCState.setText(stateArrayAdapter.getItem(i), false);
    }
}

String imagePath = getCWC.getCwcImagePath();

if (imagePath!=null){
    manageCWCClickToAddTV.setVisibility(View.GONE);

    storageReference = FirebaseStorage.getInstance().getReference().child(imagePath);
    final long ONE_MEGABYTE = 1024 * 1024;
    storageReference.getBytes(ONE_MEGABYTE).addOnSuccessListener(new OnSuccessListener<byte[]>() {
        @Override
        public void onSuccess(byte[] bytes) {
            Bitmap bmp = BitmapFactory.decodeByteArray(bytes, 0, bytes.length);
            manageCWCImage.setImageBitmap(bmp);

        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception exception) {
        }
    });

} else {
    manageCWCClickToAddTV.setVisibility(View.VISIBLE);
}
```

Figure 3-6-14-1-1: View Car Wash Center Details

Figure 3-6-14-1-1 shows the coding of view car wash center details function. The car wash center details are obtained from Firestore. Car wash center's data is stored at the "car wash center" collection in the Firestore. The getter function defined in the car wash center class is used to get the value of the car wash center details.

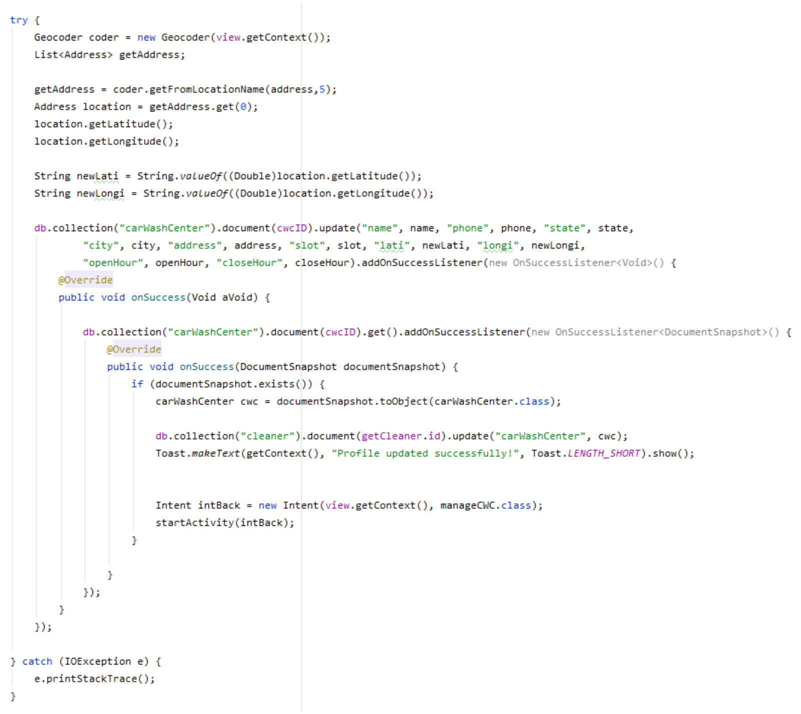
3.6.14.2 Update Car Wash Center Details



```java
manageCWCBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String name, phone, state, city, city1, address, slot, openHour, closeHour;
        name = manageCWCName.getText().toString();
        phone = manageCWCPhone.getText().toString();
        state = manageCWCState.getText().toString();
        city1 = manageCWCCity.getText().toString();
        city = city1.substring(0, 1).toUpperCase() + city1.substring(1);
        address = manageCWCAddress.getText().toString();
        slot = manageCWCSlot.getText().toString();
        openHour = manageCWCOpenHour.getText().toString();
        closeHour = manageCWCCloseHour.getText().toString();

        if (name.isEmpty()){
            Toast.makeText(getContext(), "Please enter name", Toast.LENGTH_SHORT).show();

        } else if (phone.isEmpty()){
            Toast.makeText(getContext(), "Please enter phone number", Toast.LENGTH_SHORT).show();

        } else if (state.isEmpty()){
            Toast.makeText(getContext(), "Please enter state", Toast.LENGTH_SHORT).show();

        } else if (city.isEmpty()){
            Toast.makeText(getContext(), "Please enter city", Toast.LENGTH_SHORT).show();

        } else if (address.isEmpty()){
            Toast.makeText(getContext(), "Please enter address", Toast.LENGTH_SHORT).show();

        } else if (slot.isEmpty()){
            Toast.makeText(getContext(), "Please enter no. of washing slot", Toast.LENGTH_SHORT).show();

        } else if (openHour.isEmpty()){
            Toast.makeText(getContext(), "Please enter opening hour", Toast.LENGTH_SHORT).show();

        } else if (closeHour.isEmpty()){
            Toast.makeText(getContext(), "Please enter closing hour", Toast.LENGTH_SHORT).show();
```

Figure 3-6-14-2-1: Update Car Wash Center Details (1)

When user clicked the “update” button in the car wash center details page, the onClick() function of the button is triggered. Firstly, the system will get all the input of user from the edit text in the layout file. Then, the system will check whether there is any missing input. If there is missing input, the system will display error message to inform user.



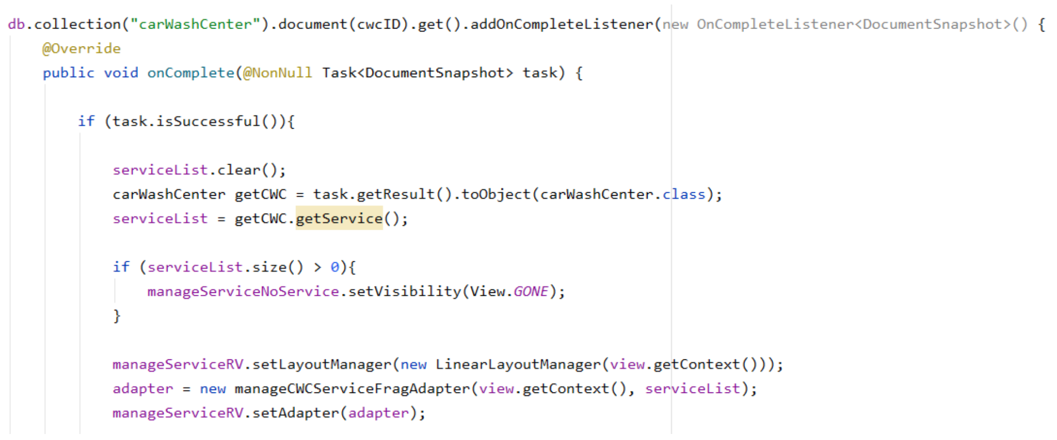
```java
try {
    Geocoder coder = new Geocoder(view.getContext());
    List<Address> getAddress;

    getAddress = coder.getFromLocationName(address,5);
    Address location = getAddress.get(0);
    location.getLatitude();
    location.getLongitude();

    String newLati = String.valueOf((Double)location.getLatitude());
    String newLongi = String.valueOf((Double)location.getLongitude());

    db.collection("carWashCenter").document(cwcID).update("name", name, "phone", phone, "state", state,
            "city", city, "address", address, "slot", slot, "lati", newLati, "longi", newLongi,
            "openHour", openHour, "closeHour", closeHour).addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {

            db.collection("carWashCenter").document(cwcID).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
                @Override
                public void onSuccess(DocumentSnapshot documentSnapshot) {
                    if (documentSnapshot.exists()) {
                        carWashCenter cwc = documentSnapshot.toObject(carWashCenter.class);

                        db.collection("cleaner").document(getCleaner.id).update("carWashCenter", cwc);
                        Toast.makeText(getContext(), "Profile updated successfully!", Toast.LENGTH_SHORT).show();


                        Intent intBack = new Intent(view.getContext(), manageCWC.class);
                        startActivity(intBack);
                    }

                }
            });
        }
    });

} catch (IOException e) {
    e.printStackTrace();
}
```

Figure 3-6-14-2-2: Update Car Wash Center Details (2)

If the user input is complete, the system will get the value of the address of the car wash center and transform it into longitude and latitude. Next, all the car wash center details is updated in the Firestore using update() function.

3.6.15 Manage Car Wash Center Service

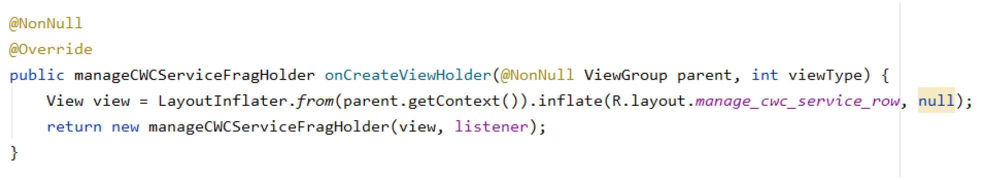
3.6.15.1 View Car Wash Center Service



```java
db.collection("carWashCenter").document(cwcID).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {

        if (task.isSuccessful()){

            serviceList.clear();
            carWashCenter getCWC = task.getResult().toObject(carWashCenter.class);
            serviceList = getCWC.getService();

            if (serviceList.size() > 0){
                manageServiceNoService.setVisibility(View.GONE);
            }

            manageServiceRV.setLayoutManager(new LinearLayoutManager(view.getContext()));
            adapter = new manageCWCServiceFragAdapter(view.getContext(), serviceList);
            manageServiceRV.setAdapter(adapter);
```

Figure 3-6-15-1-1: View Car Wash Center Service Adapter (1)

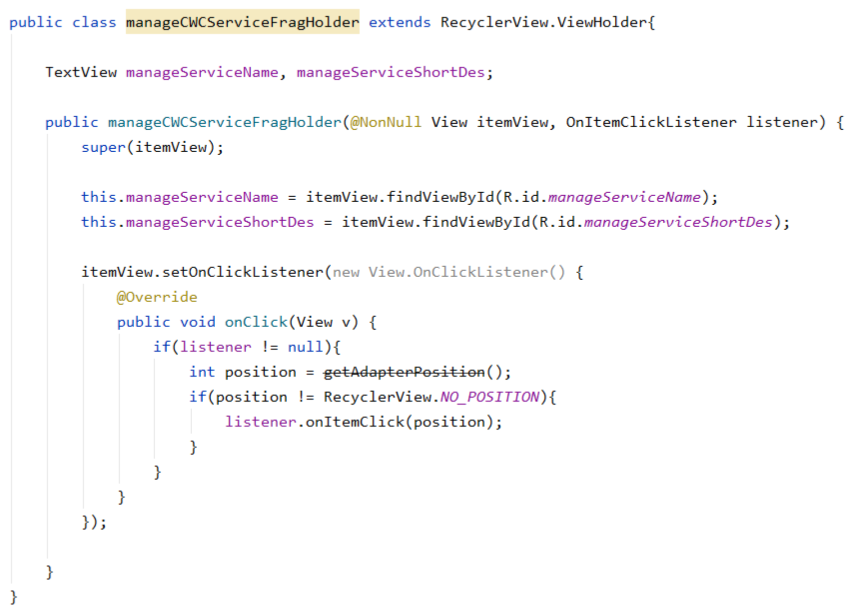
In Figure 3-6-15-1-1, the service of car wash center is retrieved from the "car wash center" collection in Firestore. All the services are stored in an array list. The services are displayed using recycler view. Recycler view is used to hold all the services in service list. The adapter is used to handle each appointment displayed in recycler view. Hence, the service list is pass into the adapter.



```java
@NonNull
@Override
public manageCWCServiceFragHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.manage_cwc_service_row, null);
    return new manageCWCServiceFragHolder(view, listener);
}
```

Figure 3-6-15-1-2: View Car Wash Center Service Adapter (2)

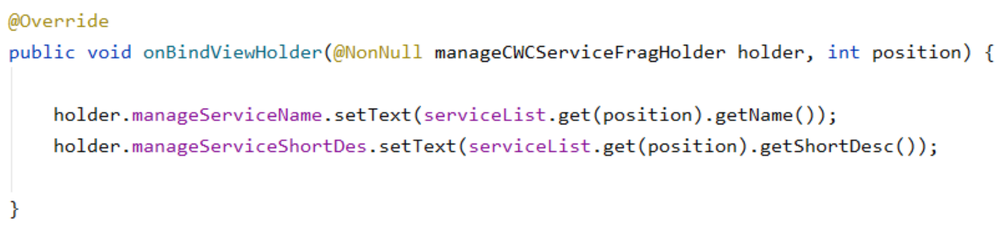
In figure 3-6-15-1-2, the inflator is used to inflate the service row design file into a view.



```
public class manageCWCServiceFragHolder extends RecyclerView.ViewHolder{

    TextView manageServiceName, manageServiceShortDes;

    public manageCWCServiceFragHolder(@NonNull View itemView, OnItemClickListener listener) {
        super(itemView);

        this.manageServiceName = itemView.findViewById(R.id.manageServiceName);
        this.manageServiceShortDes = itemView.findViewById(R.id.manageServiceShortDes);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(listener != null){
                    int position = ~~getAdapterPosition~~();
                    if(position != RecyclerView.NO_POSITION){
                        listener.onItemClick(position);
                    }
                }
            }
        });

    }
}
```

Figure 3-6-15-1-3: View Car Wash Center Service Adapter (3)

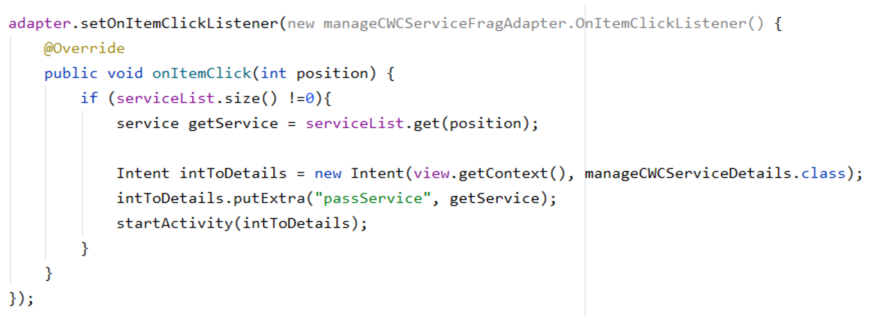
Firstly, all the components in the inflated view need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-15-1-3, component such as the service name and description are declared. When user clicked on the inflated view, the system will return the position of the service clicked in the service array list.



```
@Override
public void onBindViewHolder(@NonNull manageCWCServiceFragHolder holder, int position) {

    holder.manageServiceName.setText(serviceList.get(position).getName());
    holder.manageServiceShortDes.setText(serviceList.get(position).getShortDesc());

}
```

Figure 3-6-15-1-4: View Car Wash Center Service Adapter (4)

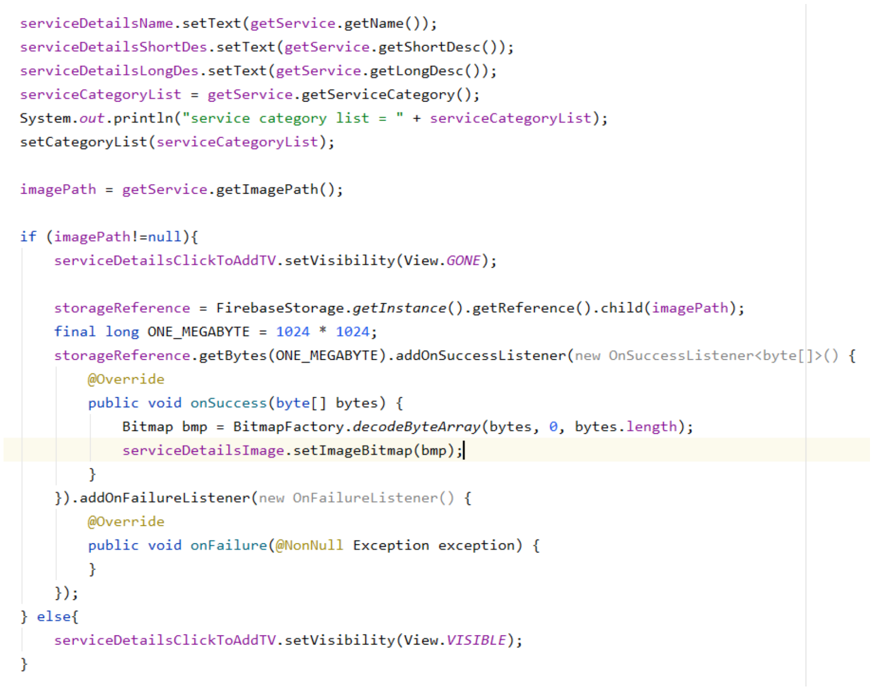
Each of the service in the service list is retrieved by position and the details is displayed. The onBindViewHolder() is used to hold each of the component value of the inflated view. The getter function defined in the service class is used to get the value from the service.



```
adapter.setOnItemClickListener(new manageCWCServiceFragAdapter.OnItemClickListener() {
    @Override
    public void onItemClick(int position) {
        if (serviceList.size() !=0){
            service getService = serviceList.get(position);

            Intent intToDetails = new Intent(view.getContext(), manageCWCServiceDetails.class);
            intToDetails.putExtra("passService", getService);
            startActivity(intToDetails);
        }
    }
});
```

Figure 3-6-15-1-5: View Car Wash Center Service Adapter (5)

When user clicked on the service of the recycler view, the position of the service in the recycler view is obtained from the view holder. The service is obtained and pass to the service details page using intent. Then, the system will direct user to a new activity that display the details of the service.

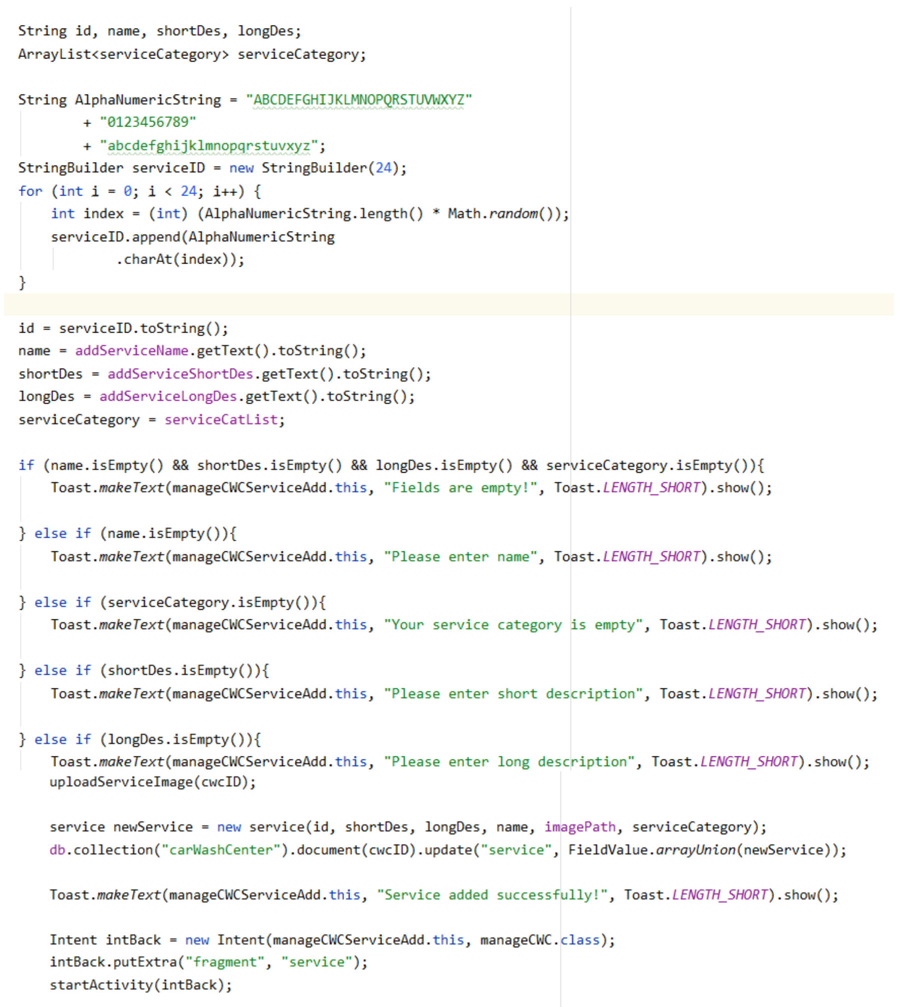


```
serviceDetailsName.setText(getService.getName());
serviceDetailsShortDes.setText(getService.getShortDesc());
serviceDetailsLongDes.setText(getService.getLongDesc());
serviceCategoryList = getService.getServiceCategory();
System.out.println("service category list = " + serviceCategoryList);
setCategoryList(serviceCategoryList);

imagePath = getService.getImagePath();

if (imagePath!=null){
    serviceDetailsClickToAddTV.setVisibility(View.GONE);

    storageReference = FirebaseStorage.getInstance().getReference().child(imagePath);
    final long ONE_MEGABYTE = 1024 * 1024;
    storageReference.getBytes(ONE_MEGABYTE).addOnSuccessListener(new OnSuccessListener<byte[]>() {
        @Override
        public void onSuccess(byte[] bytes) {
            Bitmap bmp = BitmapFactory.decodeByteArray(bytes, 0, bytes.length);
            serviceDetailsImage.setImageBitmap(bmp);
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception exception) {
        }
    });
} else{
    serviceDetailsClickToAddTV.setVisibility(View.VISIBLE);
}
```

Figure 3-6-15-1-6: Display Car Wash Center Service Details

Figure 3-6-15-1-6 shows the coding of display car wash center's service details function. The service details are passed from the previous activity using intent. Then, the system use the getter function defined in the service class to get the value of the service. The value is then displayed by using the setText() function. Image of the service is also retrieved from the Cloud Storage and displayed on the image view.

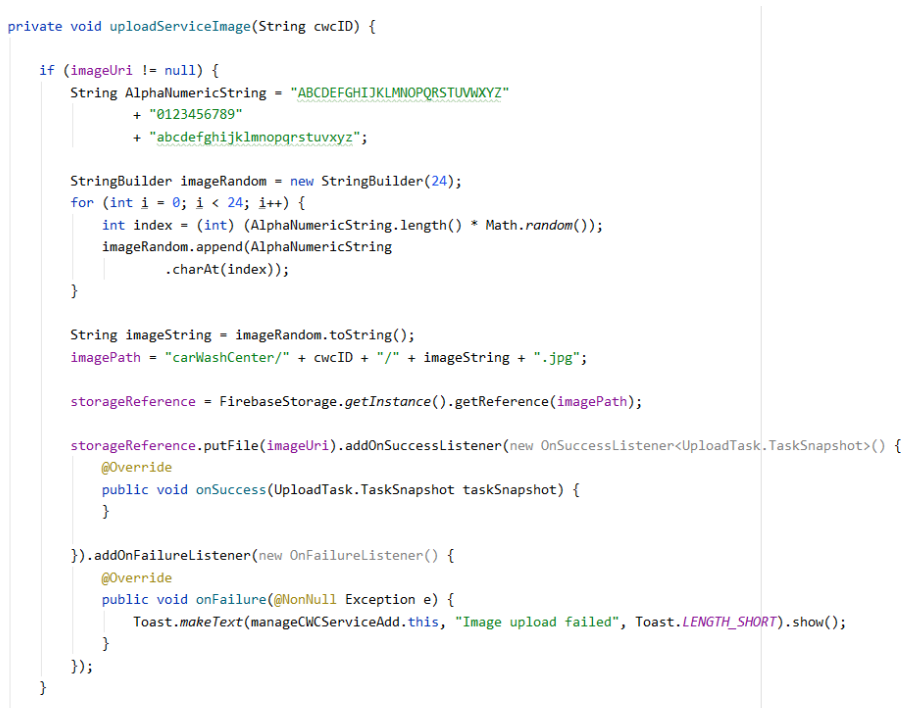
3.6.15.2 Add Car Wash Center Service



```
String id, name, shortDes, longDes;
ArrayList<serviceCategory> serviceCategory;

String AlphaNumericString = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        + "0123456789"
        + "abcdefghijklmnopqrstuvxyz";
StringBuilder serviceID = new StringBuilder(24);
for (int i = 0; i < 24; i++) {
    int index = (int) (AlphaNumericString.length() * Math.random());
    serviceID.append(AlphaNumericString
            .charAt(index));
}

id = serviceID.toString();
name = addServiceName.getText().toString();
shortDes = addServiceShortDes.getText().toString();
longDes = addServiceLongDes.getText().toString();
serviceCategory = serviceCatList;

if (name.isEmpty() && shortDes.isEmpty() && longDes.isEmpty() && serviceCategory.isEmpty()){
    Toast.makeText(manageCWCServiceAdd.this, "Fields are empty!", Toast.LENGTH_SHORT).show();

} else if (name.isEmpty()){
    Toast.makeText(manageCWCServiceAdd.this, "Please enter name", Toast.LENGTH_SHORT).show();

} else if (serviceCategory.isEmpty()){
    Toast.makeText(manageCWCServiceAdd.this, "Your service category is empty", Toast.LENGTH_SHORT).show();

} else if (shortDes.isEmpty()){
    Toast.makeText(manageCWCServiceAdd.this, "Please enter short description", Toast.LENGTH_SHORT).show();

} else if (longDes.isEmpty()){
    Toast.makeText(manageCWCServiceAdd.this, "Please enter long description", Toast.LENGTH_SHORT).show();
    uploadServiceImage(cwcID);

    service newService = new service(id, shortDes, longDes, name, imagePath, serviceCategory);
    db.collection("carWashCenter").document(cwcID).update("service", FieldValue.arrayUnion(newService));

    Toast.makeText(manageCWCServiceAdd.this, "Service added successfully!", Toast.LENGTH_SHORT).show();

    Intent intBack = new Intent(manageCWCServiceAdd.this, manageCWC.class);
    intBack.putExtra("fragment", "service");
    startActivity(intBack);
```

Figure 3-6-15-2-1: Add Car Wash Center Service

Figure 3-6-15-2-1 shows the coding of add car wash center service function. User’s need to complete a form in order to add a new service. When user clicked on “add” button, the onClick() function of the button is triggered. Firstly, the system will get all the user’s input from the edit text. Then, the system will check whether there are any missing inputs. If there is missing input, the system will display error message to user. Else, if the input is complete, the system will update the car wash center service in Firestore. The system will add new service into the car wash center document. Besides, the image of the service will also be upload to the Cloud Storage.

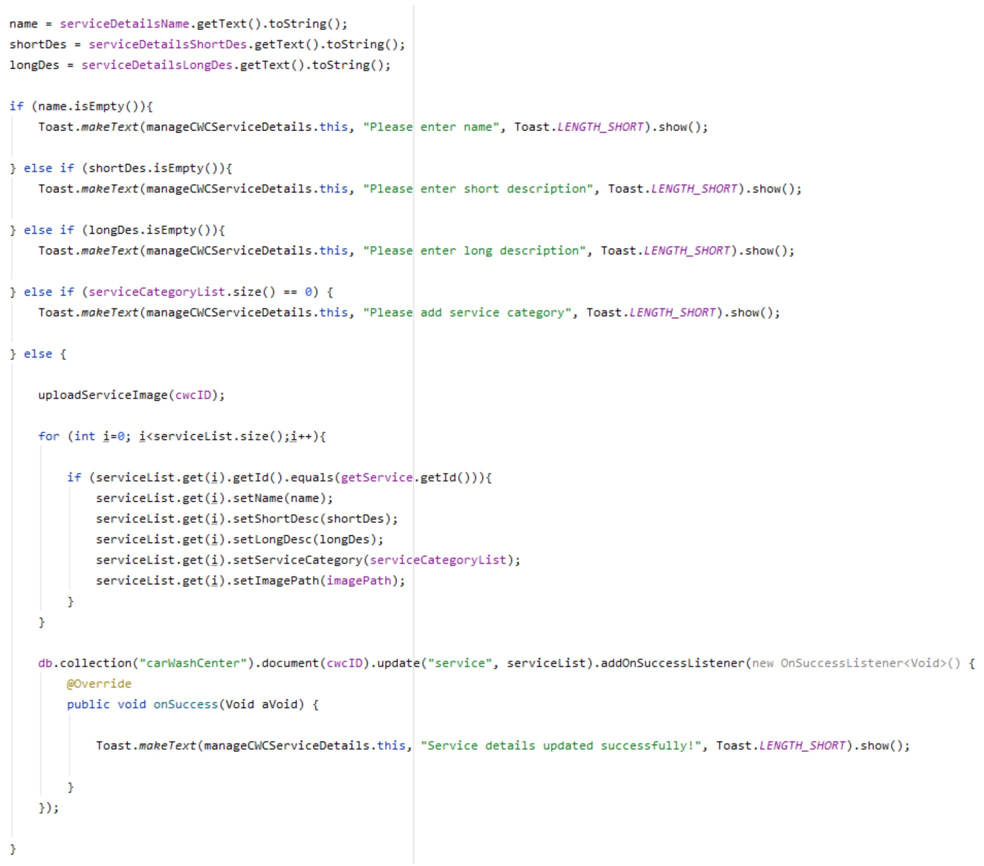


```
private void uploadServiceImage(String cwcID) {

    if (imageUri != null) {
        String AlphaNumericString = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                + "0123456789"
                + "abcdefghijklmnopqrstuvxyz";

        StringBuilder imageRandom = new StringBuilder(24);
        for (int i = 0; i < 24; i++) {
            int index = (int) (AlphaNumericString.length() * Math.random());
            imageRandom.append(AlphaNumericString
                    .charAt(index));
        }

        String imageString = imageRandom.toString();
        imagePath = "carWashCenter/" + cwcID + "/" + imageString + ".jpg";

        storageReference = FirebaseStorage.getInstance().getReference(imagePath);

        storageReference.putFile(imageUri).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            }

        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(manageCWCServiceAdd.this, "Image upload failed", Toast.LENGTH_SHORT).show();
            }
        });
    }
```

Figure 3-6-15-2-2: Upload Car Wash Center Service Image

Figure 3-6-15-2-2 shows the coding of upload car wash center service image. The name of the service image is defined using random string generator. A string of 24 characters is generated and the string is used as the name of image. The image is then stored into the Cloud Storage.

3.6.15.3 Update Service

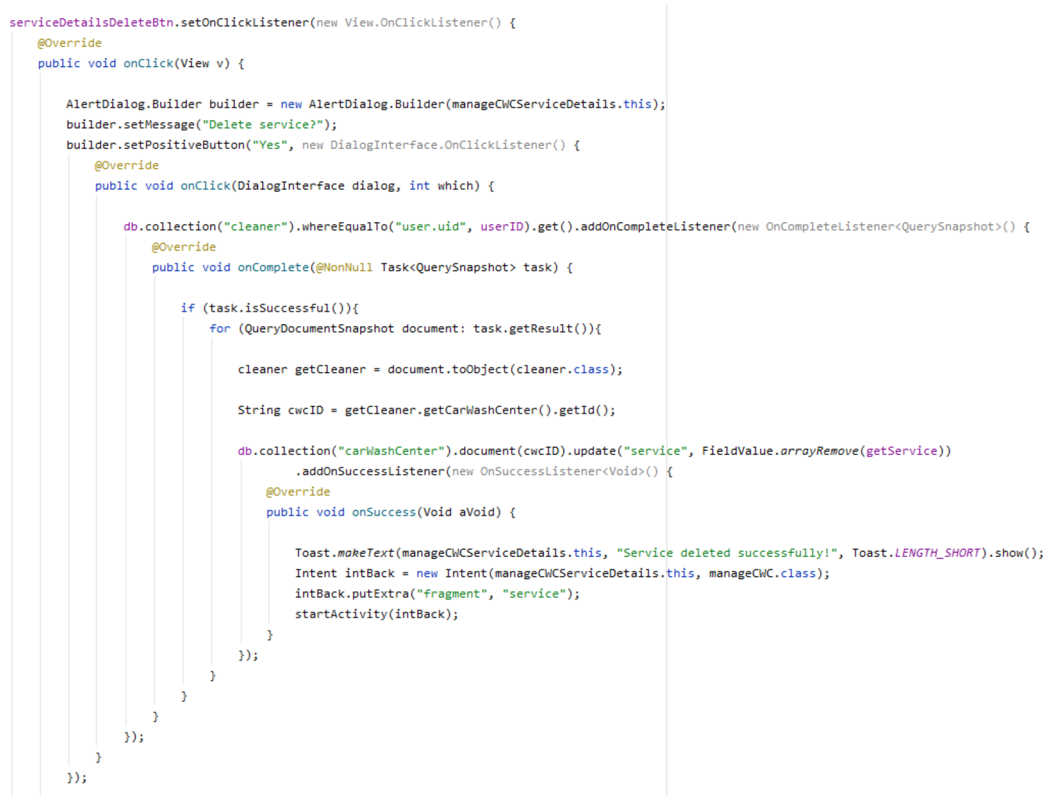


```java
name = serviceDetailsName.getText().toString();
shortDes = serviceDetailsShortDes.getText().toString();
longDes = serviceDetailsLongDes.getText().toString();

if (name.isEmpty()){
    Toast.makeText(manageCWCServiceDetails.this, "Please enter name", Toast.LENGTH_SHORT).show();

} else if (shortDes.isEmpty()){
    Toast.makeText(manageCWCServiceDetails.this, "Please enter short description", Toast.LENGTH_SHORT).show();

} else if (longDes.isEmpty()){
    Toast.makeText(manageCWCServiceDetails.this, "Please enter long description", Toast.LENGTH_SHORT).show();

} else if (serviceCategoryList.size() == 0) {
    Toast.makeText(manageCWCServiceDetails.this, "Please add service category", Toast.LENGTH_SHORT).show();

} else {

    uploadServiceImage(cwcID);

    for (int i=0; i<serviceList.size();i++){

        if (serviceList.get(i).getId().equals(getService.getId())){
            serviceList.get(i).setName(name);
            serviceList.get(i).setShortDesc(shortDes);
            serviceList.get(i).setLongDesc(longDes);
            serviceList.get(i).setServiceCategory(serviceCategoryList);
            serviceList.get(i).setImagePath(imagePath);
        }
    }

    db.collection("carWashCenter").document(cwcID).update("service", serviceList).addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {

            Toast.makeText(manageCWCServiceDetails.this, "Service details updated successfully!", Toast.LENGTH_SHORT).show();

        }
    });

}
```

Figure 3-6-15-3-1: Update Car Wash Center Service

Figure 3-6-15-3-1 shows the coding of update car wash center service function. User can edit the details of the service in the edit text. When user clicked on “update” button, the onClick() function of the button is triggered. Firstly, the system will get all the user’s input from the edit text. Then, the system will check whether there are any missing inputs. If there is missing input, the system will display error message to user. Else, if the input is complete, the system will update the car wash center service in Firestore. Besides, the image of the service will also be upload to the Cloud Storage.

3.6.15.4 Delete Service



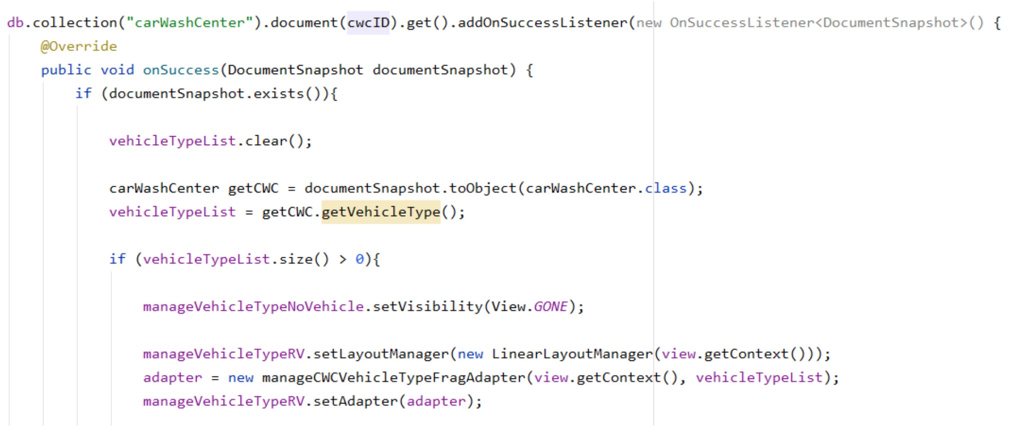
```java
serviceDetailsDeleteBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        AlertDialog.Builder builder = new AlertDialog.Builder(manageCWCServiceDetails.this);
        builder.setMessage("Delete service?");
        builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

                db.collection("cleaner").whereEqualTo("user.uid", userID).get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot> task) {

                        if (task.isSuccessful()){
                            for (QueryDocumentSnapshot document: task.getResult()){

                                cleaner getCleaner = document.toObject(cleaner.class);

                                String cwcID = getCleaner.getCarWashCenter().getId();

                                db.collection("carWashCenter").document(cwcID).update("service", FieldValue.arrayRemove(getService))
                                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                                    @Override
                                    public void onSuccess(Void aVoid) {

                                        Toast.makeText(manageCWCServiceDetails.this, "Service deleted successfully!", Toast.LENGTH_SHORT).show();
                                        Intent intBack = new Intent(manageCWCServiceDetails.this, manageCWC.class);
                                        intBack.putExtra("fragment", "service");
                                        startActivity(intBack);
                                    }
                                });
                            }
                        }
                    }
                });
            }
        });
```

Figure 3-6-15-4-1: Delete Car Wash Center Service

Figure 3-6-15-4-1 shows the coding of delete car wash center service function. When user clicked the “delete” button, the onClick() function of the button is triggered. The system will pop out an alert dialog and ask for the confirmation to delete the service. If the user clicked “yes”, then the service will be deleted. The system will remove the service from the car wash center stored in the Firestore.

3.6.16 Manage Car Wash Center Vehicle Type

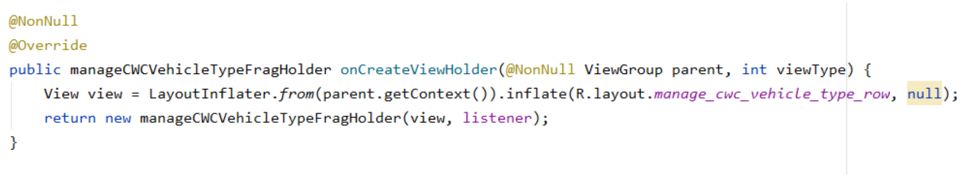
3.6.16.1 View Car Wash Center Vehicle Type



```
db.collection("carWashCenter").document(cwcID).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
    @Override
    public void onSuccess(DocumentSnapshot documentSnapshot) {
        if (documentSnapshot.exists()){

            vehicleTypeList.clear();

            carWashCenter getCWC = documentSnapshot.toObject(carWashCenter.class);
            vehicleTypeList = getCWC.getVehicleType();

            if (vehicleTypeList.size() > 0){

                manageVehicleTypeNoVehicle.setVisibility(View.GONE);

                manageVehicleTypeRV.setLayoutManager(new LinearLayoutManager(view.getContext()));
                adapter = new manageCWCVehicleTypeFragAdapter(view.getContext(), vehicleTypeList);
                manageVehicleTypeRV.setAdapter(adapter);
```

Figure 3-6-16-1-1: View Car Wash Center Vehicle Type Adapter (1)

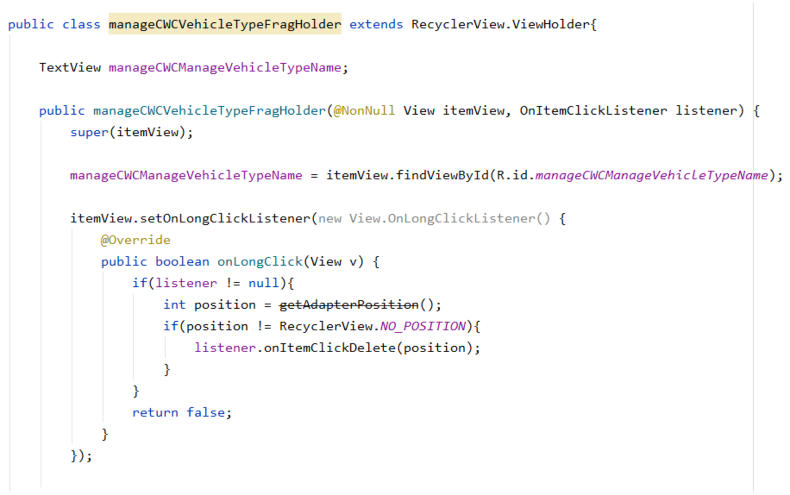
In Figure 3-6-16-1-1, the vehicle type of car wash center is retrieved from the "car wash center" collection in Firestore. All the vehicle types are stored in an array list. The services are displayed using recycler view. Recycler view is used to hold all the vehicle type in vehicle type list. The adapter is used to handle each vehicle type displayed in recycler view. Hence, the vehicle type list is pass into the adapter.



```
@NonNull
@Override
public manageCWCVehicleTypeFragHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.manage_cwc_vehicle_type_row, null);
    return new manageCWCVehicleTypeFragHolder(view, listener);
}
```

Figure 3-6-16-1-2: View Car Wash Center Vehicle Type Adapter (2)

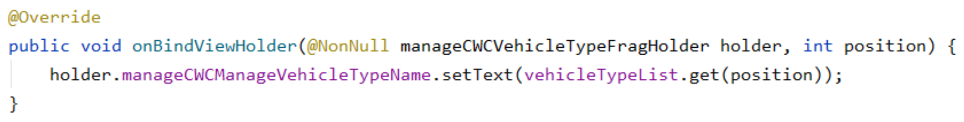
In figure 3-6-16-1-2, the inflator is used to inflate the vehicle type row design file into a view.



```
public class manageCWCVehicleTypeFragHolder extends RecyclerView.ViewHolder{

    TextView manageCWCManageVehicleTypeName;

    public manageCWCVehicleTypeFragHolder(@NonNull View itemView, OnItemClickListener listener) {
        super(itemView);

        manageCWCManageVehicleTypeName = itemView.findViewById(R.id.manageCWCManageVehicleTypeName);

        itemView.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View v) {
                if(listener != null){
                    int position = getAdapterPosition();
                    if(position != RecyclerView.NO_POSITION){
                        listener.onItemClickDelete(position);
                    }
                }
                return false;
            }
        });
```

Figure 3-6-16-1-3: View Car Wash Center Vehicle Type Adapter (3)

View holder need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-16-1-3, component such as the vehicle type name is declared.

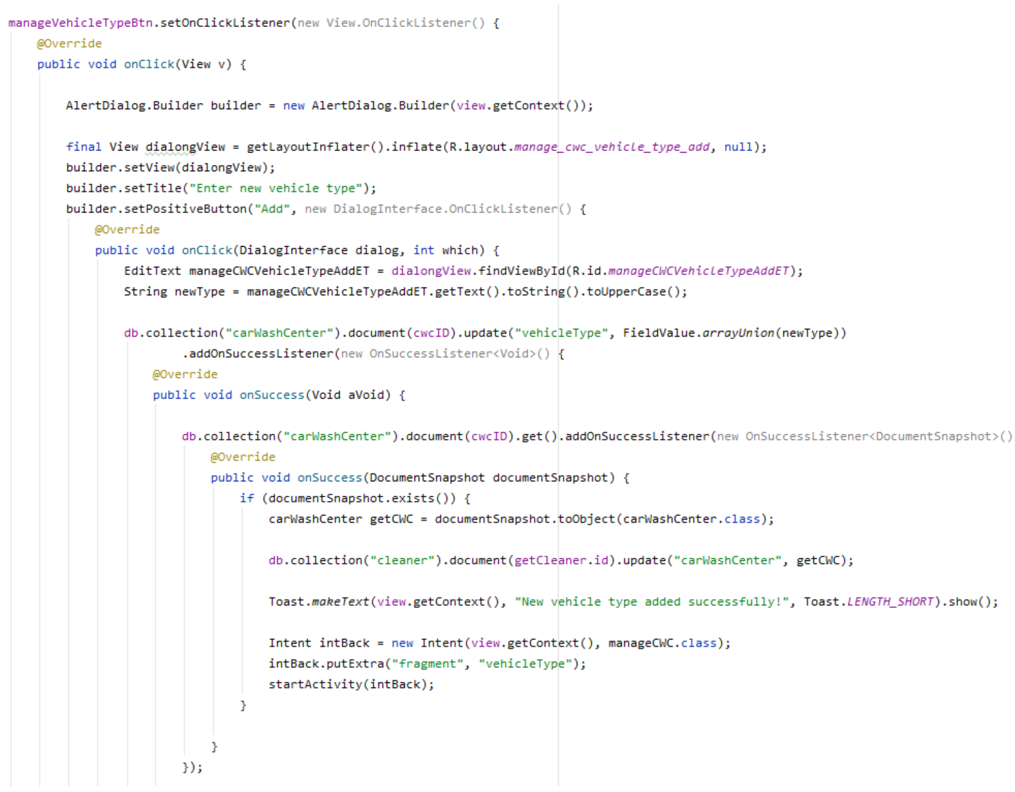


```java
@Override
public void onBindViewHolder(@NonNull manageCWCVehicleTypeFragHolder holder, int position) {
    holder.manageCWCManageVehicleTypeName.setText(vehicleTypeList.get(position));
}
```

Figure 3-6-16-1-4: View Car Wash Center Vehicle Type Adapter (4)

Each of the vehicle type in the vehicle type list is retrieved by position and the details is displayed. The holder is used to hold each of the component value.

3.6.16.2 Add Car Wash Center Vehicle Type

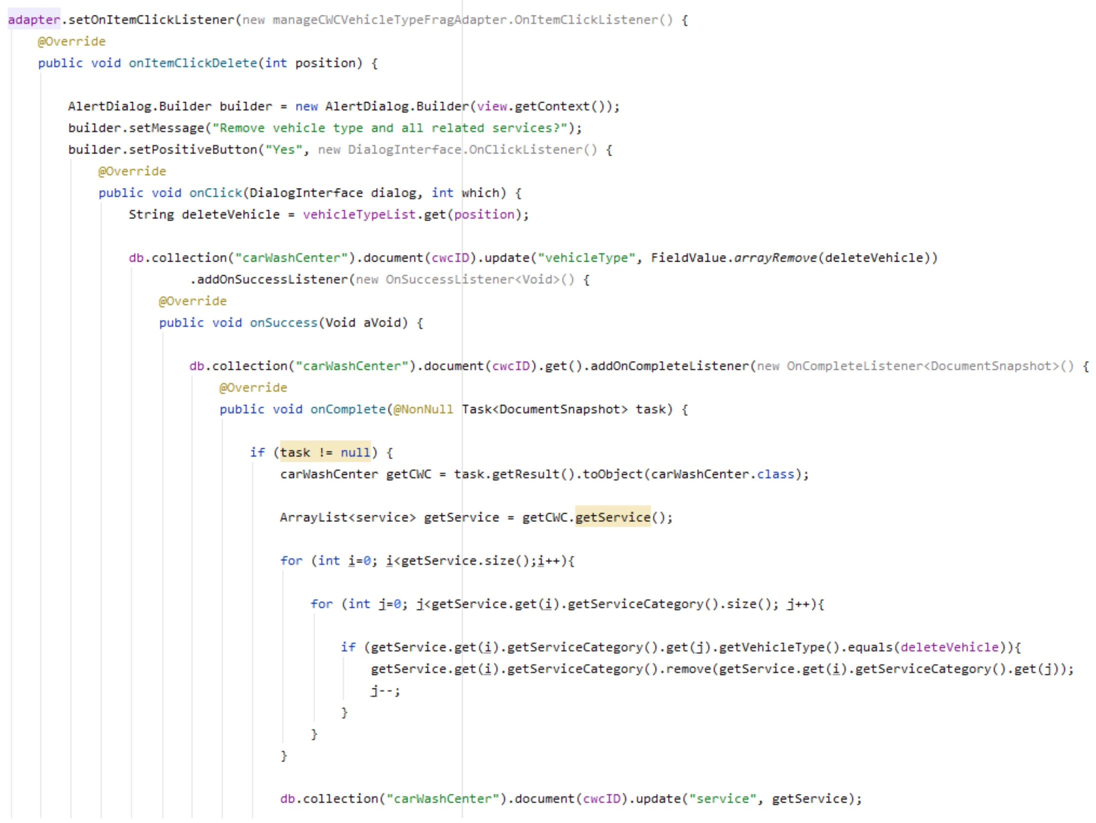


```java
manageVehicleTypeBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        AlertDialog.Builder builder = new AlertDialog.Builder(view.getContext());

        final View dialongView = getLayoutInflater().inflate(R.layout.manage_cwc_vehicle_type_add, null);
        builder.setView(dialongView);
        builder.setTitle("Enter new vehicle type");
        builder.setPositiveButton("Add", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                EditText manageCWCVehicleTypeAddET = dialongView.findViewById(R.id.manageCWCVehicleTypeAddET);
                String newType = manageCWCVehicleTypeAddET.getText().toString().toUpperCase();

                db.collection("carWashCenter").document(cwcID).update("vehicleType", FieldValue.arrayUnion(newType))
                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {

                        db.collection("carWashCenter").document(cwcID).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
                            @Override
                            public void onSuccess(DocumentSnapshot documentSnapshot) {
                                if (documentSnapshot.exists()) {
                                    carWashCenter getCWC = documentSnapshot.toObject(carWashCenter.class);

                                    db.collection("cleaner").document(getCleaner.id).update("carWashCenter", getCWC);

                                    Toast.makeText(view.getContext(), "New vehicle type added successfully!", Toast.LENGTH_SHORT).show();

                                    Intent intBack = new Intent(view.getContext(), manageCWC.class);
                                    intBack.putExtra("fragment", "vehicleType");
                                    startActivity(intBack);
                                }

                            }
                        });
```

Figure 3-6-16-2-1: Add Car Wash Center Vehicle Type

Figure 3-6-16-2-1 shows the coding of add car wash center vehicle type function. When user clicked the “add vehicle” button, the system will pop out an alert dialog and prompt for the vehicle type name. User needs to fill in the vehicle type name and click “ok”. Next, the system will add the new vehicle type into the car wash center in Firestore.

3.6.16.3 Delete Car Wash Center Vehicle Type



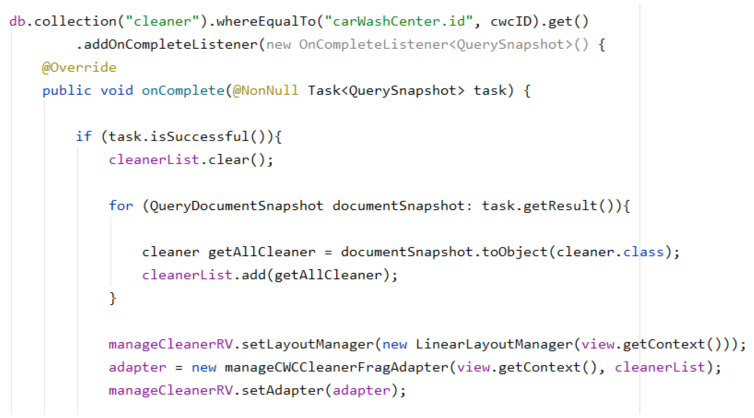
```java
adapter.setOnItemClickListener(new manageCWCVehicleTypeFragAdapter.OnItemClickListener() {
    @Override
    public void onItemClickDelete(int position) {

        AlertDialog.Builder builder = new AlertDialog.Builder(view.getContext());
        builder.setMessage("Remove vehicle type and all related services?");
        builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String deleteVehicle = vehicleTypeList.get(position);

                db.collection("carWashCenter").document(cwcID).update("vehicleType", FieldValue.arrayRemove(deleteVehicle))
                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {

                        db.collection("carWashCenter").document(cwcID).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                            @Override
                            public void onComplete(@NonNull Task<DocumentSnapshot> task) {

                                if (task != null) {
                                    carWashCenter getCWC = task.getResult().toObject(carWashCenter.class);

                                    ArrayList<service> getService = getCWC.getService();

                                    for (int i=0; i<getService.size();i++){

                                        for (int j=0; j<getService.get(i).getServiceCategory().size(); j++){

                                            if (getService.get(i).getServiceCategory().get(j).getVehicleType().equals(deleteVehicle)){
                                                getService.get(i).getServiceCategory().remove(getService.get(i).getServiceCategory().get(j));
                                                j--;
                                            }
                                        }
                                    }

                                    db.collection("carWashCenter").document(cwcID).update("service", getService);
```

Figure 3-6-16-3-1: Delete Car Wash Center Vehicle Type

Figure 3-6-16-3-1 shows the coding of delete car wash center vehicle type function. When user long clicked the “delete” button, the onClick() function of the button is triggered. The system will pop out an alert dialog and ask for the confirmation to delete the vehicle type. If the user clicked “yes”, then the vehicle type will be deleted. The system will remove the vehicle type from the car wash center stored in the Firestore. Besides, the system will also delete the service of the vehicle type in Firestore.

3.6.17 Manage Car Wash Center Cleaner

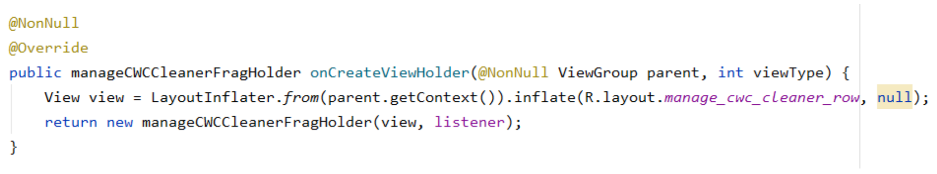
3.6.17.1 View Car Wash Center Cleaner



```
db.collection("cleaner").whereEqualTo("carWashCenter.id", cwcID).get()
        .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {

        if (task.isSuccessful()){
            cleanerList.clear();

            for (QueryDocumentSnapshot documentSnapshot: task.getResult()){

                cleaner getAllCleaner = documentSnapshot.toObject(cleaner.class);
                cleanerList.add(getAllCleaner);
            }

            manageCleanerRV.setLayoutManager(new LinearLayoutManager(view.getContext()));
            adapter = new manageCWCCleanerFragAdapter(view.getContext(), cleanerList);
            manageCleanerRV.setAdapter(adapter);
```

Figure 3-6-17-1-1: View Car Wash Center Cleaner Adapter (1)

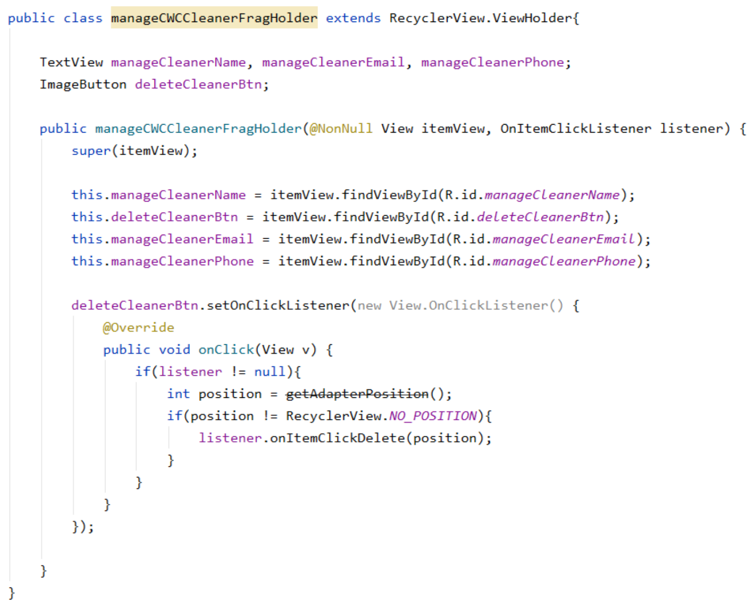
In Figure 3-6-17-1-1, the cleaner of car wash center is retrieved from the “cleaner” collection in Firestore. The system will filter using the car wash center ID to filter out the cleaner that belongs to the car wash center. All the cleaner are stored in an array list. The cleaner are displayed using recycler view. Recycler view is used to hold all the cleaner in cleaner list. The adapter is used to handle each cleaner displayed in recycler view. Hence, the cleaner list is pass into the adapter.



```
@NonNull
@Override
public manageCWCCleanerFragHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.manage_cwc_cleaner_row, null);
    return new manageCWCCleanerFragHolder(view, listener);
}
```

Figure 3-6-17-1-2: View Car Wash Center Cleaner Adapter (2)

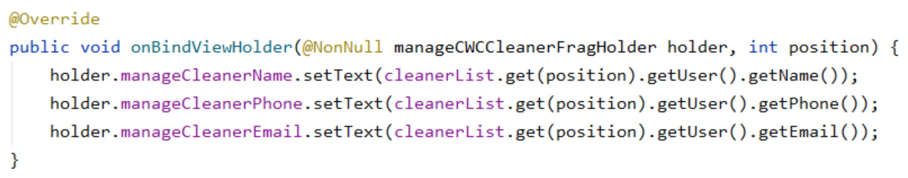
In figure 3-6-17-1-2, the inflator is used to inflate the cleaner row design file into a view.



```
public class manageCWCCleanerFragHolder extends RecyclerView.ViewHolder{

    TextView manageCleanerName, manageCleanerEmail, manageCleanerPhone;
    ImageButton deleteCleanerBtn;

    public manageCWCCleanerFragHolder(@NonNull View itemView, OnItemClickListener listener) {
        super(itemView);

        this.manageCleanerName = itemView.findViewById(R.id.manageCleanerName);
        this.deleteCleanerBtn = itemView.findViewById(R.id.deleteCleanerBtn);
        this.manageCleanerEmail = itemView.findViewById(R.id.manageCleanerEmail);
        this.manageCleanerPhone = itemView.findViewById(R.id.manageCleanerPhone);

        deleteCleanerBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(listener != null){
                    int position = getAdapterPosition();
                    if(position != RecyclerView.NO_POSITION){
                        listener.onItemClickDelete(position);
                    }
                }
            }
        });

    }
}
```

Figure 3-6-17-1-3: View Car Wash Center Cleaner Adapter (3)

View holder need to be defined and initiate. The component in the inflated view is defined using their view ID. In figure 3-6-17-1-3, component such as the cleaner name, phone number and email are declared. When user clicked on the “delete” button in itemView, the system will return the position of the cleaner clicked on the cleaner array list.

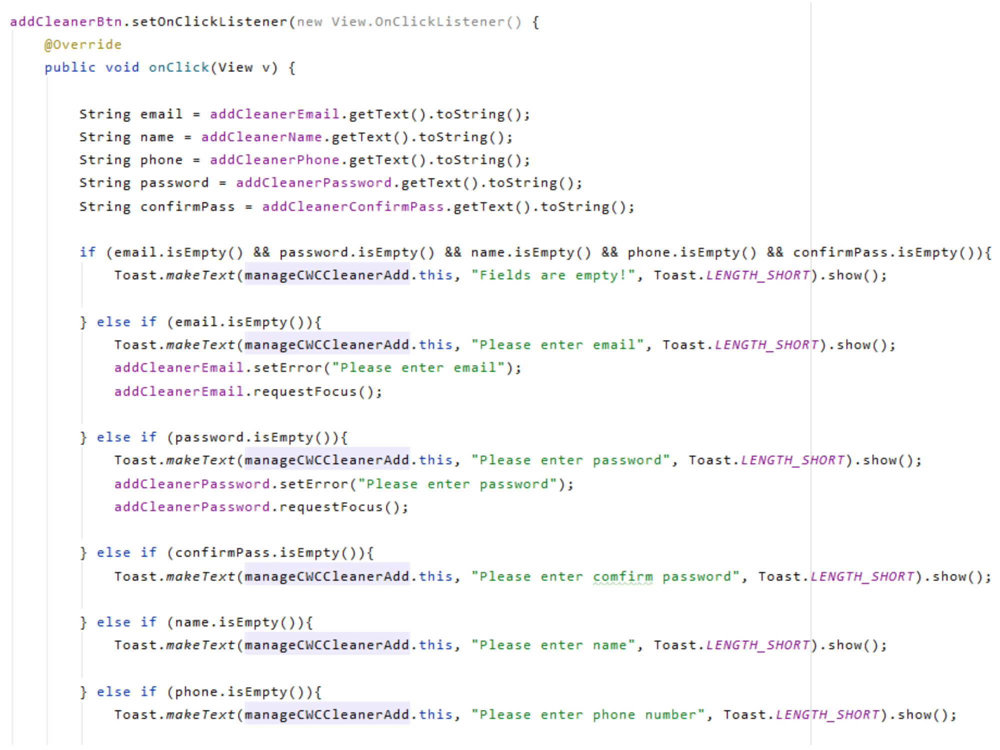


```
@Override
public void onBindViewHolder(@NonNull manageCWCCleanerFragHolder holder, int position) {
    holder.manageCleanerName.setText(cleanerList.get(position).getUser().getName());
    holder.manageCleanerPhone.setText(cleanerList.get(position).getUser().getPhone());
    holder.manageCleanerEmail.setText(cleanerList.get(position).getUser().getEmail());
}
```

Figure 3-6-17-1-4: View Car Wash Center Cleaner Adapter (4)

Each of the cleaner in the cleaner list is retrieved by position and the details is displayed. The holder is used to hold each of the component value. The getter function defined in the cleaner class is used to get the value from the cleaner.

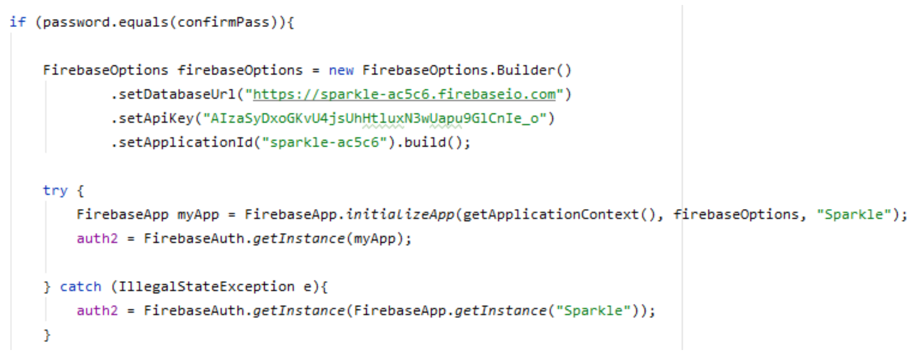
3.6.17.2 Add Car Wash Center Cleaner



```
addCleanerBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String email = addCleanerEmail.getText().toString();
        String name = addCleanerName.getText().toString();
        String phone = addCleanerPhone.getText().toString();
        String password = addCleanerPassword.getText().toString();
        String confirmPass = addCleanerConfirmPass.getText().toString();

        if (email.isEmpty() && password.isEmpty() && name.isEmpty() && phone.isEmpty() && confirmPass.isEmpty()){
            Toast.makeText(manageCWCCleanerAdd.this, "Fields are empty!", Toast.LENGTH_SHORT).show();

        } else if (email.isEmpty()){
            Toast.makeText(manageCWCCleanerAdd.this, "Please enter email", Toast.LENGTH_SHORT).show();
            addCleanerEmail.setError("Please enter email");
            addCleanerEmail.requestFocus();

        } else if (password.isEmpty()){
            Toast.makeText(manageCWCCleanerAdd.this, "Please enter password", Toast.LENGTH_SHORT).show();
            addCleanerPassword.setError("Please enter password");
            addCleanerPassword.requestFocus();

        } else if (confirmPass.isEmpty()){
            Toast.makeText(manageCWCCleanerAdd.this, "Please enter comfirm password", Toast.LENGTH_SHORT).show();

        } else if (name.isEmpty()){
            Toast.makeText(manageCWCCleanerAdd.this, "Please enter name", Toast.LENGTH_SHORT).show();

        } else if (phone.isEmpty()){
            Toast.makeText(manageCWCCleanerAdd.this, "Please enter phone number", Toast.LENGTH_SHORT).show();
```

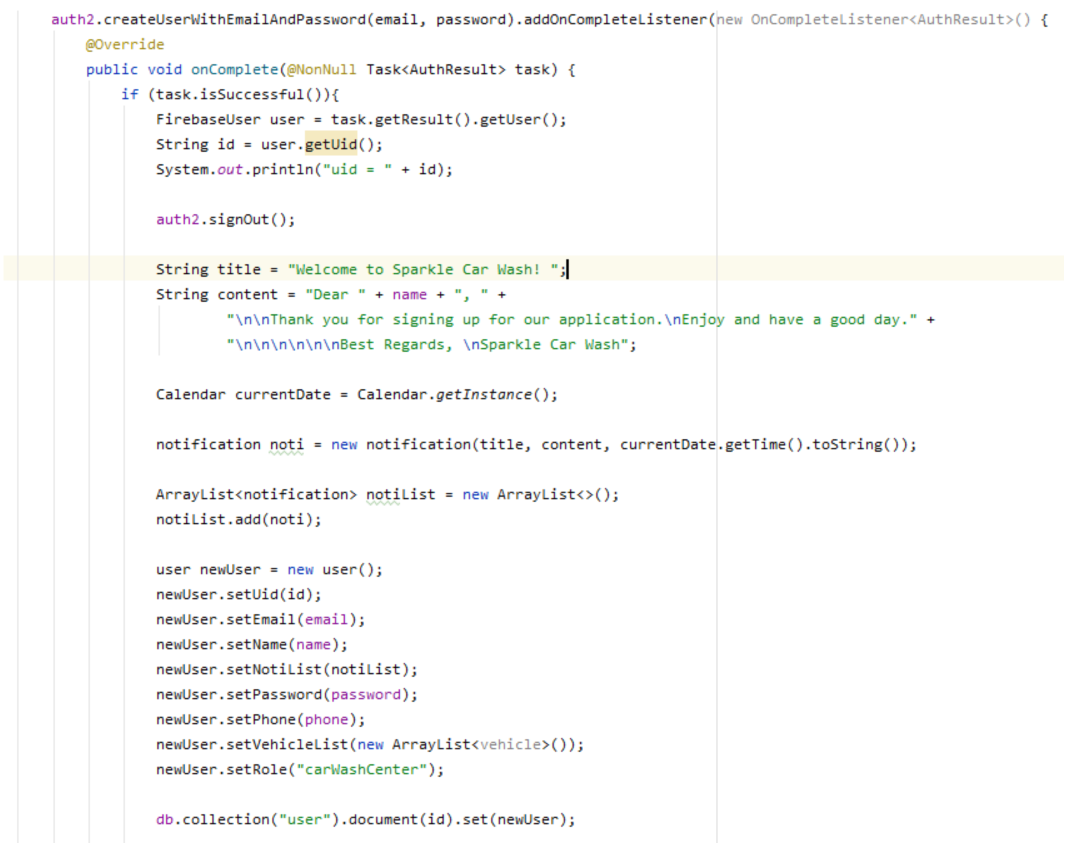
Figure 3-6-17-2-1: Add Car Wash Center Cleaner (1)

Figure 3-6-17-2-1 shows the coding of add car wash center cleaner function. When user clicked the "add cleaner" button, the system will pop out an alert dialog and prompt for the cleaner details. User needs to fill in the cleaner details and click "ok". Next, the system will check the inputs. If there are missing inputs, the system will display error message to user. System will also check the similarity of password and confirmation password. If the password and confirmation does not match, error message will be displayed to user. Next, if there is no missing inputs and password and confirmation password is the same, the system will continue the next steps.



```
if (password.equals(confirmPass)){

    FirebaseOptions firebaseOptions = new FirebaseOptions.Builder()
            .setDatabaseUrl("https://sparkle-ac5c6.firebaseio.com")
            .setApiKey("AIzaSyDxoGKvU4jsUhHtluxN3wUapu9GlCnIe_o")
            .setApplicationId("sparkle-ac5c6").build();

    try {
        FirebaseApp myApp = FirebaseApp.initializeApp(getApplicationContext(), firebaseOptions, "Sparkle");
        auth2 = FirebaseAuth.getInstance(myApp);

    } catch (IllegalStateException e){
        auth2 = FirebaseAuth.getInstance(FirebaseApp.getInstance("Sparkle"));
    }
```

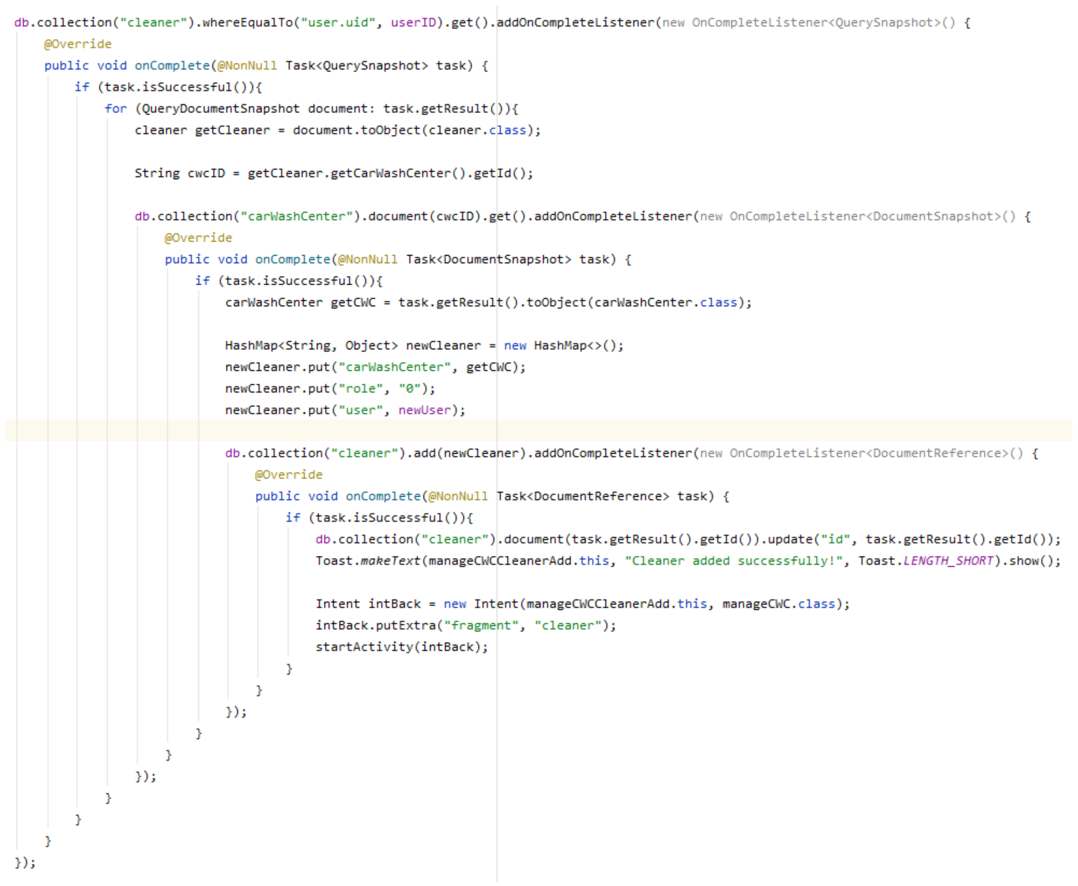
Figure 3-6-17-2-2: Add Car Wash Center Cleaner (2)



```
auth2.createUserWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()){
            FirebaseUser user = task.getResult().getUser();
            String id = user.getUid();
            System.out.println("uid = " + id);

            auth2.signOut();

            String title = "Welcome to Sparkle Car Wash! ";
            String content = "Dear " + name + ", " +
                    "\n\nThank you for signing up for our application.\nEnjoy and have a good day." +
                    "\n\n\n\n\n\nBest Regards, \nSparkle Car Wash";

            Calendar currentDate = Calendar.getInstance();

            notification noti = new notification(title, content, currentDate.getTime().toString());

            ArrayList<notification> notiList = new ArrayList<>();
            notiList.add(noti);

            user newUser = new user();
            newUser.setUid(id);
            newUser.setEmail(email);
            newUser.setName(name);
            newUser.setNotiList(notiList);
            newUser.setPassword(password);
            newUser.setPhone(phone);
            newUser.setVehicleList(new ArrayList<vehicle>());
            newUser.setRole("carWashCenter");

            db.collection("user").document(id).set(newUser);
```

Figure 3-6-17-2-3: Add Car Wash Center Cleaner (3)

After completing the cleaner details, the system will create a new Firebase Options, set database url, set API key and set application. In figure 3-6-17-2-2 and figure 3-6-17-2-3, the system will create a new user by using the createUserWithEmailAndPassword() function. The system will set the role of user to “carWashCenter” which indicates the user is a cleaner and the cleaner will log into the account with cleaner role.

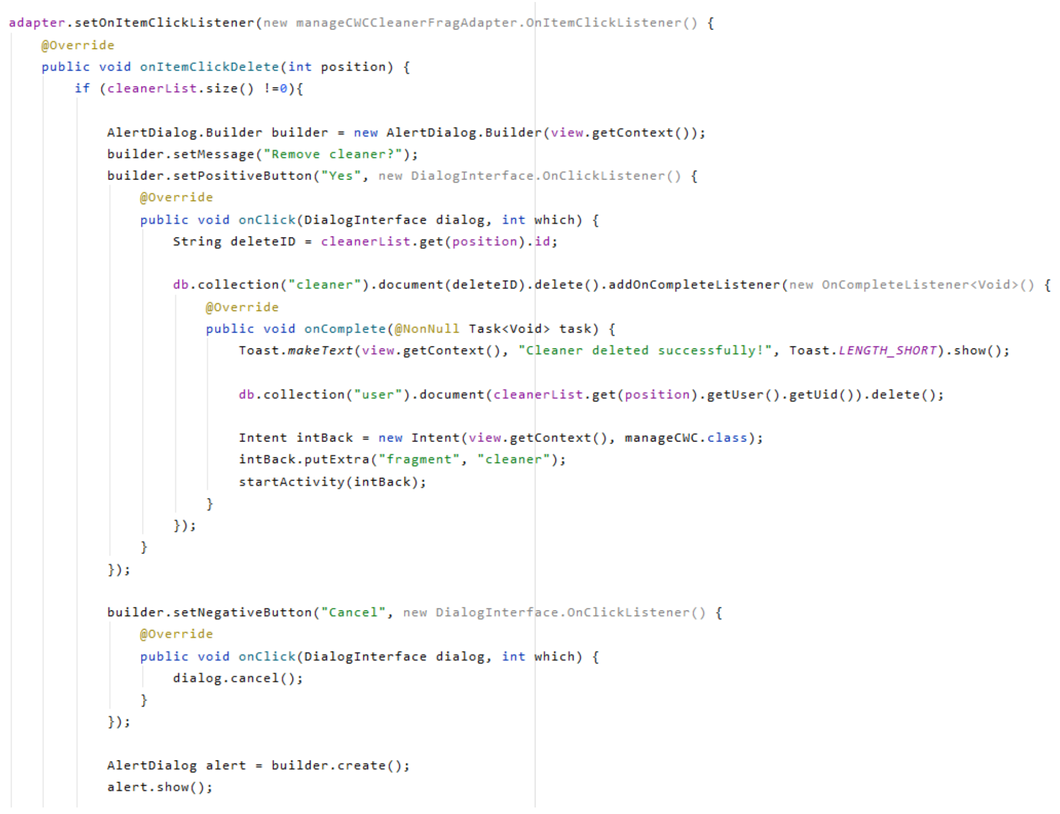


```
db.collection("cleaner").whereEqualTo("user.uid", userID).get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
        if (task.isSuccessful()){
            for (QueryDocumentSnapshot document: task.getResult()){
                cleaner getCleaner = document.toObject(cleaner.class);

                String cwcID = getCleaner.getCarWashCenter().getId();

                db.collection("carWashCenter").document(cwcID).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                        if (task.isSuccessful()){
                            carWashCenter getCWC = task.getResult().toObject(carWashCenter.class);

                            HashMap<String, Object> newCleaner = new HashMap<>();
                            newCleaner.put("carWashCenter", getCWC);
                            newCleaner.put("role", "0");
                            newCleaner.put("user", newUser);

                            db.collection("cleaner").add(newCleaner).addOnCompleteListener(new OnCompleteListener<DocumentReference>() {
                                @Override
                                public void onComplete(@NonNull Task<DocumentReference> task) {
                                    if (task.isSuccessful()){
                                        db.collection("cleaner").document(task.getResult().getId()).update("id", task.getResult().getId());
                                        Toast.makeText(manageCWCCleanerAdd.this, "Cleaner added successfully!", Toast.LENGTH_SHORT).show();

                                        Intent intBack = new Intent(manageCWCCleanerAdd.this, manageCWC.class);
                                        intBack.putExtra("fragment", "cleaner");
                                        startActivity(intBack);
                                    }
                                }
                            });
                        }
                    }
                });
            }
        }
    }
});
```

Figure 3-6-17-2-4: Add Car Wash Center Cleaner (4)

Lastly, the cleaner details inputted by the user is obtained and stored in HashMap. Then, the system will add the cleaner to the “cleaner” collection in Firestore.

3.6.17.3 Delete Car Wash Center Cleaner



```
adapter.setOnItemClickListener(new manageCWCCleanerFragAdapter.OnItemClickListener() {
    @Override
    public void onItemClickDelete(int position) {
        if (cleanerList.size() !=0){

            AlertDialog.Builder builder = new AlertDialog.Builder(view.getContext());
            builder.setMessage("Remove cleaner?");
            builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    String deleteID = cleanerList.get(position).id;

                    db.collection("cleaner").document(deleteID).delete().addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            Toast.makeText(view.getContext(), "Cleaner deleted successfully!", Toast.LENGTH_SHORT).show();

                            db.collection("user").document(cleanerList.get(position).getUser().getUid()).delete();

                            Intent intBack = new Intent(view.getContext(), manageCWC.class);
                            intBack.putExtra("fragment", "cleaner");
                            startActivity(intBack);
                        }
                    });
                }
            });

            builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            });

            AlertDialog alert = builder.create();
            alert.show();
```

Figure 3-6-17-3-1: Delete Car Wash Center Cleaner

Figure 3-6-17-3-1 shows the coding of delete car wash center cleaner function. When user clicked the “delete” button in the view of the cleaner, the onItemClickDelete() function of the button is triggered. The system will pop out an alert dialog and ask for the confirmation to delete the cleaner. If the user clicked “yes”, then the cleaner will be deleted. The system will remove the cleaner stored in the cleaner collection in the Firestore.

CHAPTER 4 METHODOLOGY AND TOOLS

4.1 Overview

In this chapter, the methodology implemented in this project is explained. Besides, the tools like hardware and software is listed and described in this chapter. And lastly, the timeline planned for this project is showed in order to ensure that the project is delivered on time.

4.2 Methodology

Methodology is a systematic approach to implementing the System Development Life Cycle (SDLC). Throw away prototype method is used to develop the mobile application in this project. This methodology also known as close-ended prototyping. The benefit of using this methodology is it can get feedback from users and able refine the user requirements in the early development of system. Throw away prototype can deal with the errors or issues until it is resolved. After the issues are resolved, the progress will move to the next phase which is implementation phase. Overall, throw away prototype can minimize the risk of user requirements that are poorly defined.

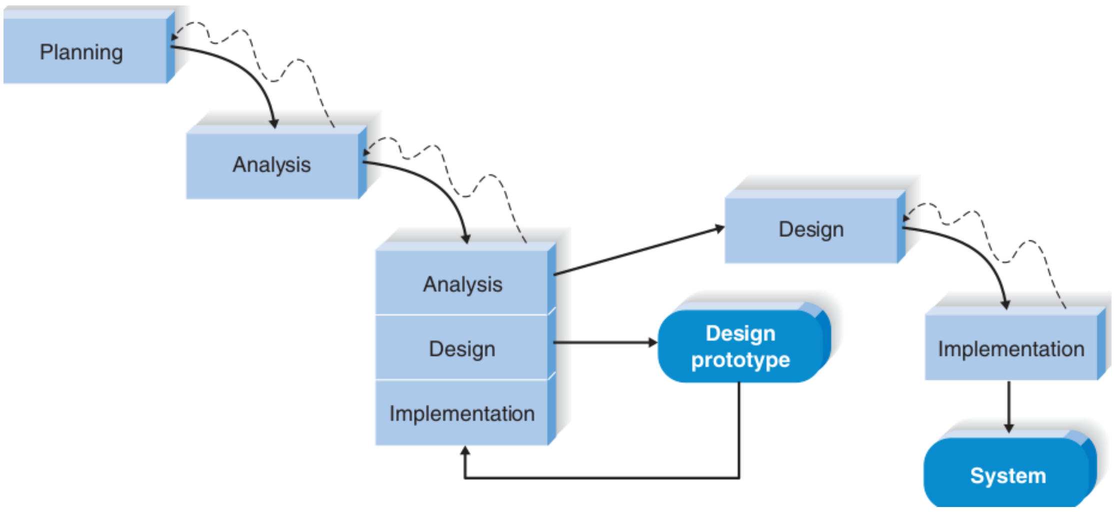


Figure 4-2-1: Throw-away Prototype Methodology

4.2.1 Planning Phase

The first phase is planning the project. Title of the project is determined through discussion with supervisor. Some research is also done in order to more understand the background of this project. Last but not least, the duration of this project is set at 21 weeks.

4.2.2 Analysis Phase

Analysis phase comes after planning phase in this project development. Literature review is done at this phase. The brief, strengths and weaknesses of previous work is identified and discussed. The purpose of literature review is to understand the knowledge or information presented in previous work on topic (Literature Review: Literature Review 2020). Then, recommendations are also proposed in order to solve the limitations of the previous work. The proposed solution is also compared and discussed in order to implement in this project and produce a better application. Problem statement and motivation is determined to initialize the project. The project scope and project objective is also decided. Then, project scope, objectives, problem statement and motivation of the project are collected.

4.2.3 Design Phase

In design phase, the technologies and tools involved in this project is identified. Besides, the methodology that going to be used in this project is also identified. Then, system framework is also designed in this phase. System framework provides an overview stated that what functions is included in the system.

4.2.4 Design Prototype

Analysis, design and implementation phase is repeated in this phases. A version of prototype is developed for every cycle of this phase. Then, the version of prototype will be given to users for evaluation. In order to understand more about the users' requirements, feedback and review is taken and recorded from users. The system will be improved according to users' feedback. Additional features are added in order to enhance the system. Design prototype phase will be repeated continuously until the system has meet the requirements of users and users are satisfied with it.

4.2.5 Implementation Phase

The designed system will start to be implemented into the system. The identified features are coded into the real system. After the development of prototype is finished, the prototype will be enhanced and any issue or problem will be solved. The cycle continues until all problems is solved.

4.3 System Requirements

Hardware and software are used to develop a mobile application.

4.3.1 Hardware Specification

Components	Requirements
Windows	Window 10
Processor	Intel® Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz
RAM	8.00 GB
Input	Keyboard and Mouse
Type of System	64-bit operating system, x64-based processor

Table 4-3-1-1: Components and Requirements of Hardware Specification

4.3.2 Software Specifications

i. Android Studio

Android Studio is the official integrated development environment (IDE) for Google’s Android operating system (Android Studio features: Android Developers). It is built on JetBrains’ IntelliJ IDEA software and specifically designed for Android. Besides, it supports programming languages such as Java, C++ and more with extensions. Android Studio has excellent emulator which allows users to create a new Graphical User Interface (GUI) easily as it provides drag and drop feature. Users can have real-time experience to the Android applications. It becomes more efficient and development time will decrease.

ii. Google Firebase

Firebase is a web and mobile application development platform with tools to help developer build and improve their applications (Stevenson, 2018). It provides various functions and features such as authentication, real-time databases, configuration, file storage, hosting and cloud function. Firebase will interact directly with services; hence users just have to query the database by writing code.

iii. Google Map API

Google Map is embedded to mobile application or website by using Google Map API. Google Map API can be used to retrieve information from Google Map. The map retrieved can be used for extensive customization.

4.4 Project Timeline

Table 4-4-1 shows the overview of the project preparation to develop a system in FYP I and FYP II in duration of 21 weeks. The application is developed according to this schedule as it has shown what should be done and the progress of the project. This timeline ensure the project is developed and completed on time.

Project Task	Project Week																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Planning Phase																					
Decide project title	▨																				
Research project background		▨																			
Analysis Phase																					
Literature review on existing application/ previous work			▨	▨																	
Determine problem statement, motivation			▨	▨																	
Determine project scope and project objective					▨	▨															
Determine impact and contribution of project					▨	▨															

Table 4-4-1: Project Timeline (1)

Design Phase																					
Identify methodology used in this project							▨														
Identify technologies and tools needed in this project							▨														
Design system framework								▨	▨	▨											
Design system using diagrams								▨	▨	▨											
Implementation Phase																					
Create prototype											▨	▨	▨	▨	▨	▨					
Enhance prototype incrementally											▨	▨	▨	▨	▨	▨					
Test the system																	▨	▨			
Deploy the system																	▨	▨			
Documentation																			▨	▨	▨

Table 4-4-2: Project Timeline (2)

CHAPTER 5 IMPLEMENTATION AND TESTING

5.1 Overview

In this chapter, the system is implemented and the performance is tested. It is also to prove that all the features and function in this system are able to achieve the objective of this project. The screenshots of the system are displayed and explained accordingly.

5.2 Project Screenshot and Explanation

Project screenshots and its explanation is listed in this section. Firstly, the name of this application is introduced. "Sparkle" is the name of this application. Sparkle means shine brightly with flashes of light in dictionary. It also stands for the stars in the sky, express the excitement, convey the feeling of love and symbolized cleanliness. Sparkle in this application implicates every customer deserved to have a quality car wash service or car detailing. The colour of the theme used in this application is red colour. Red colour is a strong colour and it symbolized the warm feelings and passion. Figure 5-2-1 shows the icon of the application.

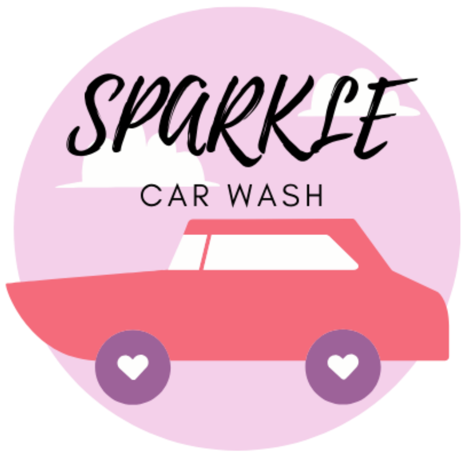


Figure 5-2-1: Icon of the Application

5.2.1 Sign Up Page and Create Profile Page

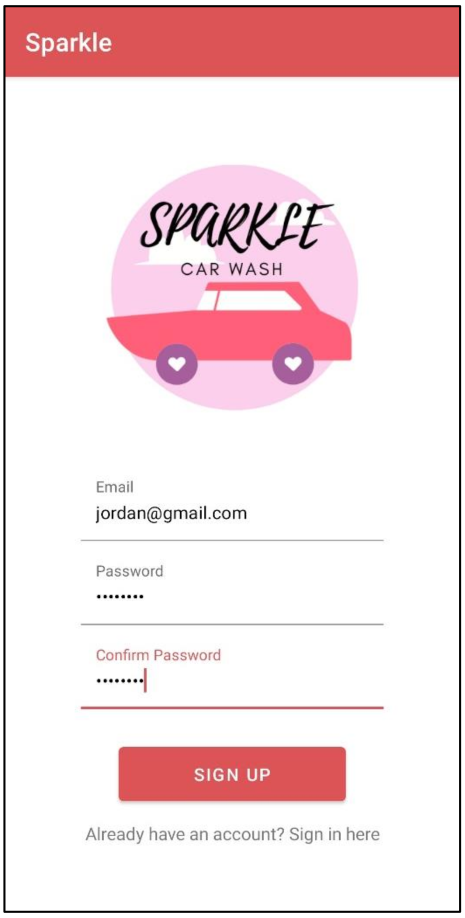


Figure 5-2-1-1: Sign Up Page

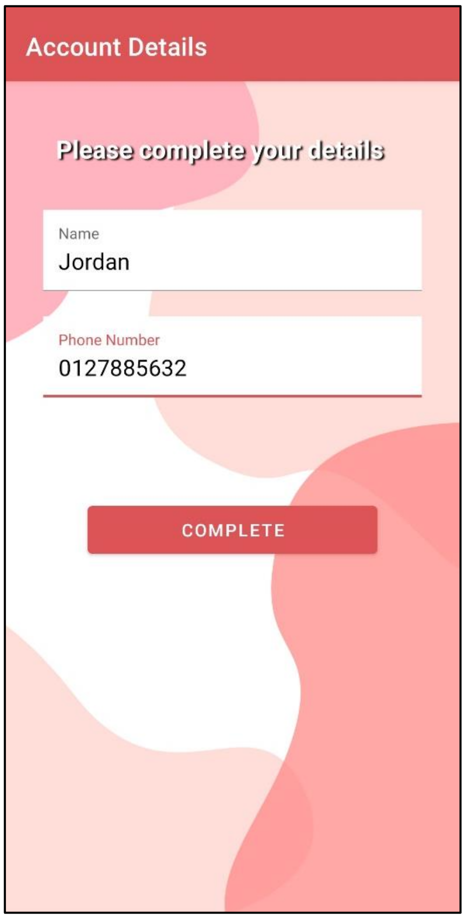


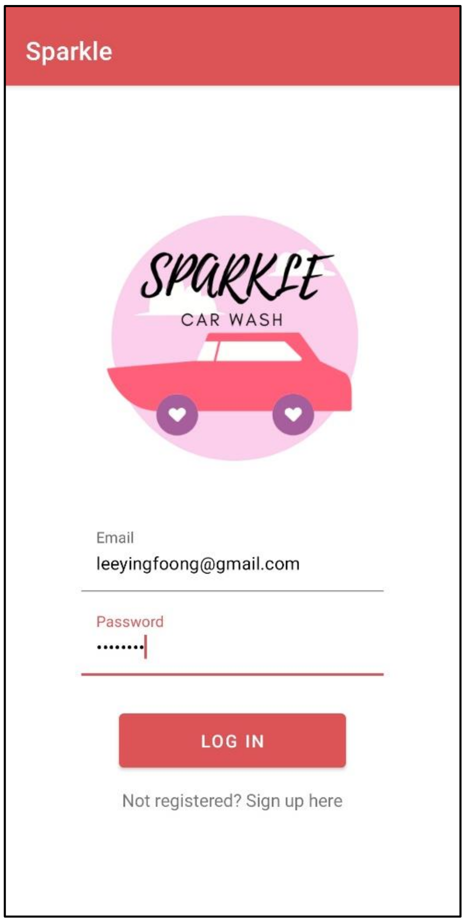
Figure 5-2-1-2: Create Profile Page

When user downloaded the application and they opened the application, the user is launched to the sign up page. In the sign up page, the icon of the application is shown. For first time user or user that does not have an account, they have to register an account in the sign up page. Firstly, user have to fill in 3 fields which is email, password and confirmation password. Email is needed in order to create an account and for log in purpose. The password and confirmation password is needed in order to verify the user during the authenticate process next time when user log in. The password and confirmation password must be the same. After user had completed all the input fields, the user can click the sign up button in order to continue the sign up process. If the user already has an account for this application, they user can click the text at the bottom of the page and the system will direct user to the sign in page.

The user is registered successfully now. However, the user profile is not stored in Cloud Firestore yet because the user is first time login user. Hence, when the system detected the first time user, the system will direct user to the create profile page as shown in

Figure 5-2-1-2. The user needs to fill in the name and phone number. User can click the complete button when they completed the form. The user input will store in the Cloud Firestore under the user collection.

5.2.2 Log In Page



Email
leeyingfoong@gmail.com
Password
••••••••

Figure 5-2-2-1: Log In Page

Figure 5-2-2-1 shows the log in page of the application. The icon of the application is displayed. 2 input is needed which is the email and password. User have to enter the email and password in order to log in the account. The system will authenticate the user when after the user clicked the log in button. If the user does not have an account for this application, the user can click the text at the bottom of the page and the system will direct user to register at sign up page.

5.2.3 Customer Home Page

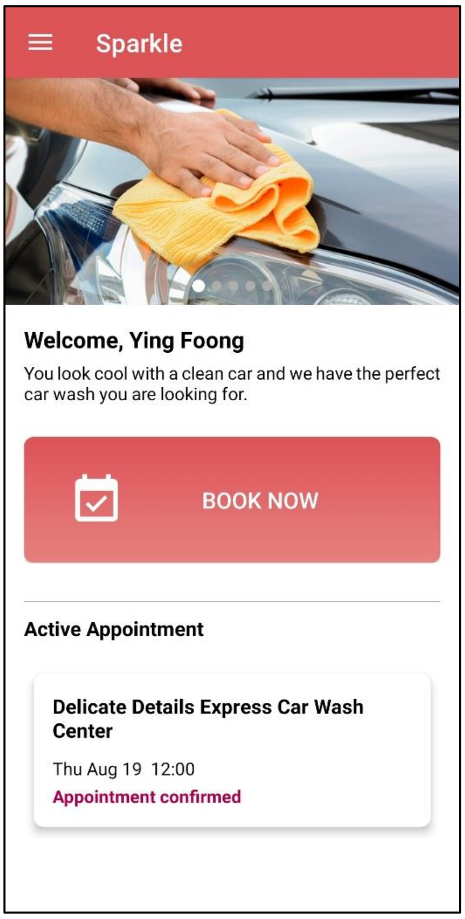

Sparkle
Welcome, Ying Foong
You look cool with a clean car and we have the perfect car wash you are looking for.
BOOK NOW
Active Appointment
Delicate Details Express Car Wash Center
Thu Aug 19 12:00
Appointment confirmed

Figure 5-2-3-1: Customer Home Page

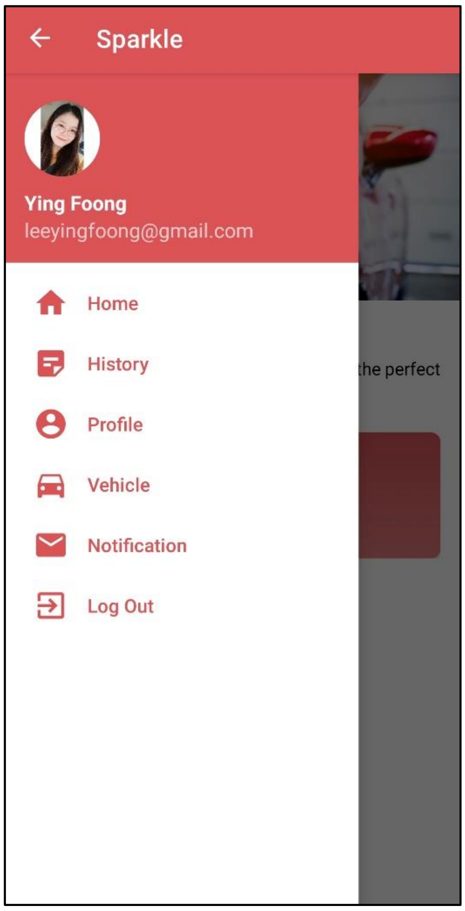


Figure 5-2-3-2: Customer Home Page's Navigation Bar

After user had logged in to the account, the system will check the role of the user. If the role of user is customer, the user will be directed to the customer's home page. In the customer home page, there is welcome greetings, book now button and a list of active appointment. Customer can click the "book now" button if they want to make a car wash appointment. Then, the active appointment is display at the bottom of the customer's home page. The active appointment is placed at home page because it is easier for customer to check their coming appointment or on going appointment. Besides, there is a navigation bar at the customer's home page. Profile picture and email of the customer is displayed at the top of the navigation bar. The menu that lead to different page is also listed in the navigation bar such as history, profile, vehicle and notification. It is convenient for customer to choose which features they want or which page they go.

5.2.4 View Nearby Car Wash Center in Map Page

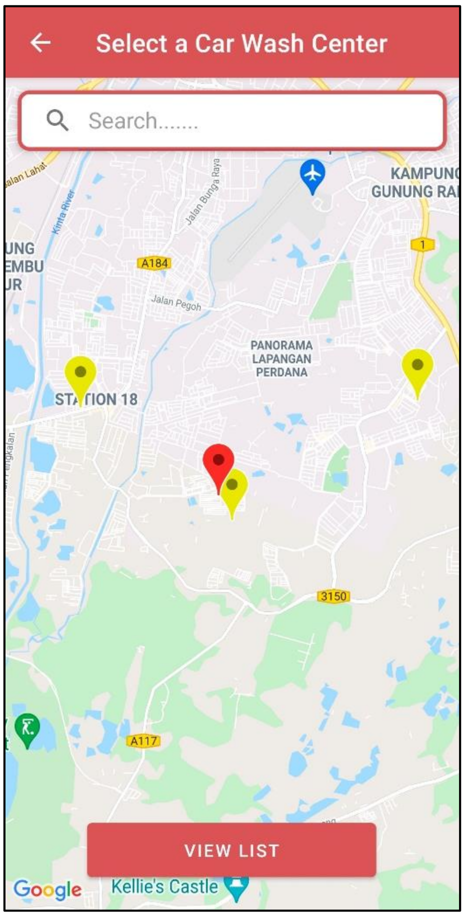
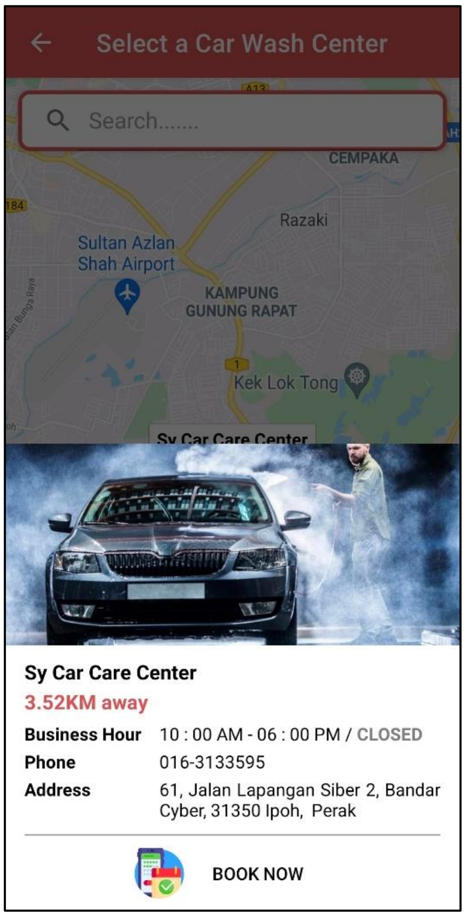


Figure 5-2-4-1: View Nearby Car Wash Center in Map Page



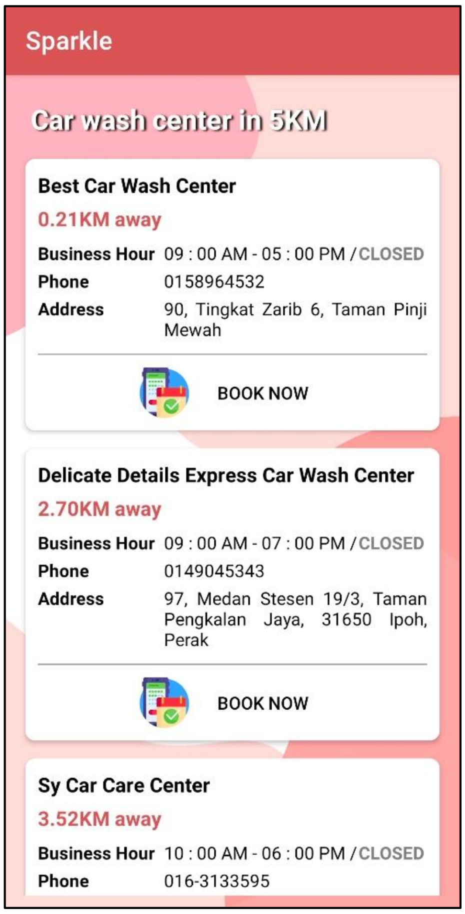
Sy Car Care Center
3.52KM away

Business Hour	10 : 00 AM - 06 : 00 PM / CLOSED
Phone	016-3133595
Address	61, Jalan Lapangan Siber 2, Bandar Cyber, 31350 Ipoh, Perak

Figure 5-2-4-2: View Nearby Car Wash Center Details in Map Page

Figure 5-2-4-1 shows the screenshot of view nearby car wash center in map page. In this page, the marker pinned on the map shows the location of the car wash center nearby. The red marker indicates the current location of the user while the yellow marker indicates the car wash center nearby. The user will see the car wash center nearby in 5KM. When the user clicked on the yellow marker, a bottom view is pop out. The bottom view contains the details of the car wash center such as the distance away, business hour, phone number and the address. If the user wants to make appointment at this car wash center, they can click the "book now" button to select the car wash center. On the top of the page, there is a search bar that allows user to search for the car wash center by name. There is a "view list" button on the bottom of the page.

5.2.5 View Nearby Car Wash Center in List Page


Sparkle

Car wash center in 5KM

Best Car Wash Center
0.21KM away

Business Hour	09 : 00 AM - 05 : 00 PM / CLOSED
Phone	0158964532
Address	90, Tingkat Zarib 6, Taman Pinji Mewah

Delicate Details Express Car Wash Center
2.70KM away

Business Hour	09 : 00 AM - 07 : 00 PM / CLOSED
Phone	0149045343
Address	97, Medan Stesen 19/3, Taman Pengkalan Jaya, 31650 Ipoh, Perak

Sy Car Care Center
3.52KM away

Business Hour	10 : 00 AM - 06 : 00 PM / CLOSED
Phone	016-3133595

Figure 5-2-5-1: View Nearby Car Wash Center in List Page

Figure 5-2-5-1 shows the list of nearby car wash center. All the car wash center in 5KM is displayed here. Each of the car wash center details is display inside a card view. The card view contains the details of the car wash center such as the distance away, business hour, phone number and the address. If the user wants to make appointment at this car wash center, they can click the “book now” button to select the car wash center.

5.2.6 Search Car Wash Center Page

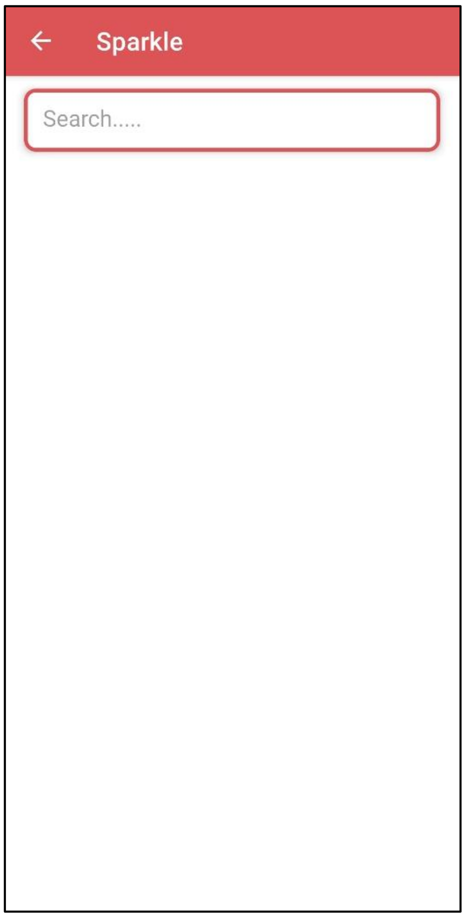
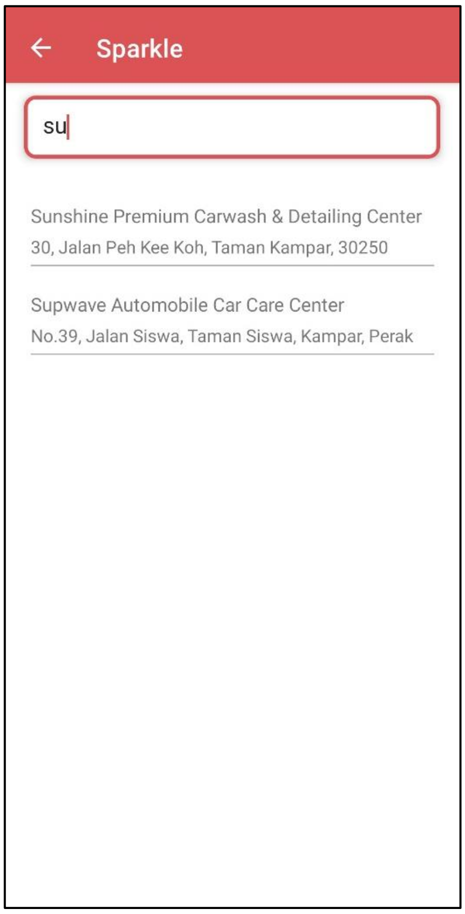


Figure 5-2-6-1: Search Car Wash Center Page (1)



Sunshine Premium Carwash & Detailing Center
30, Jalan Peh Kee Koh, Taman Kampar, 30250

Supwave Automobile Car Care Center
No.39, Jalan Siswa, Taman Siswa, Kampar, Perak

Figure 5-2-6-2: Search Car Wash Center Page (2)

Figure 5-2-6-1 shows the screenshot of the search car wash center page. It is used to search the car wash center when user wants to make an appointment at a specific car wash center. User can search the car wash center by name. When user enter the name of the car wash center, the system will filter the car wash center that contains the user's input. The car wash center filtered is then displayed in a list. The name and address of the car wash center is displayed. When user clicked on the car wash center displayed, user is directed to a map.

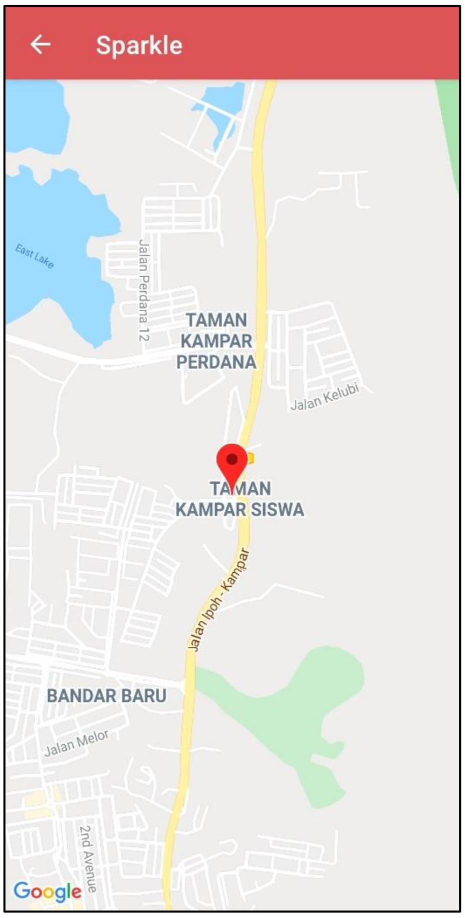
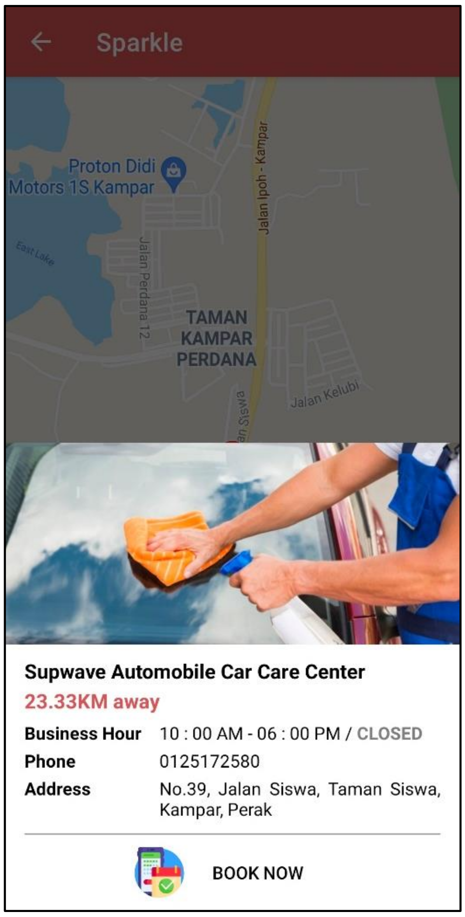


Figure 5-2-6-3: Search Car Wash Center Page (3)



Supwave Automobile Car Care Center
23.33KM away

Business Hour	10 : 00 AM - 06 : 00 PM / CLOSED
Phone	0125172580
Address	No.39, Jalan Siswa, Taman Siswa, Kampar, Perak

Figure 5-2-6-3: Search Car Wash Center Page (4)

When user clicked on the car wash center displayed on the filtered car wash center list, user is directed to a new map. The red marker indicates the location of the car wash center in map. When the user clicked on the red marker, a bottom view is pop out. The bottom view contains the details of the car wash center such as the distance away, business hour, phone number and the address. If the user wants to make appointment at this car wash center, they can click the “book now” button to select the car wash center.

5.2.7 Select Vehicle Type Page

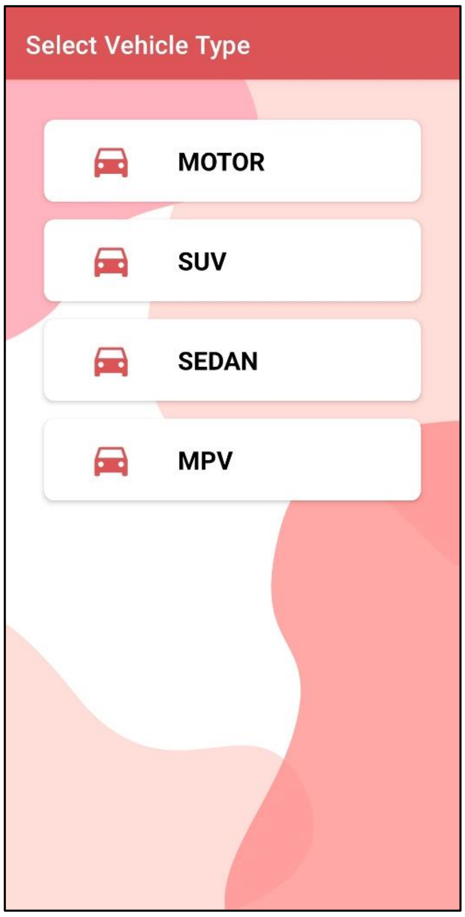
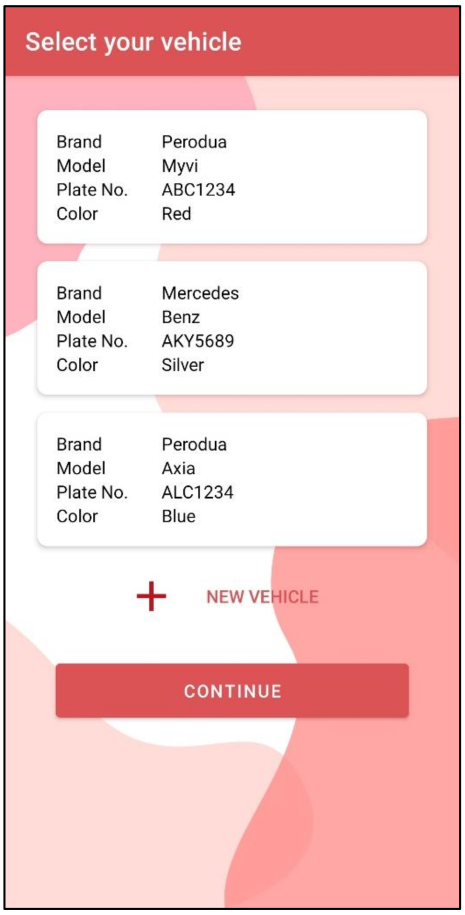


Figure 5-2-7-1: Select Vehicle Type Page

After customer has selected the car wash center they prefer, the customer is direct to select vehicle type page. In this page, the vehicle type that the car wash center selected serve is displayed. As shown in figure 5-2-7-1, the car wash center provide service to vehicle type such as motor, SUV, Sedan and MPV.

5.2.8 Select Vehicle Page



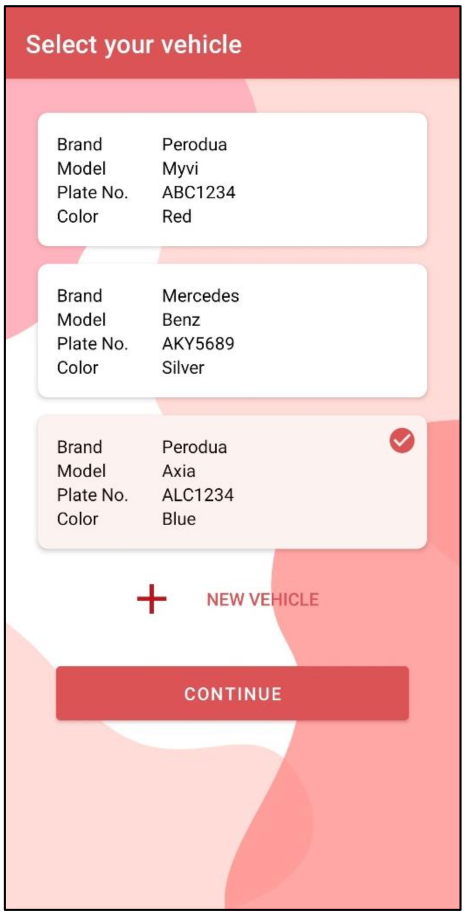
Select your vehicle

Brand	Perodua
Model	Myvi
Plate No.	ABC1234
Color	Red

Brand	Mercedes
Model	Benz
Plate No.	AKY5689
Color	Silver

Brand	Perodua
Model	Axia
Plate No.	ALC1234
Color	Blue

Figure 5-2-8-1: Select Vehicle Page (1)



Select your vehicle

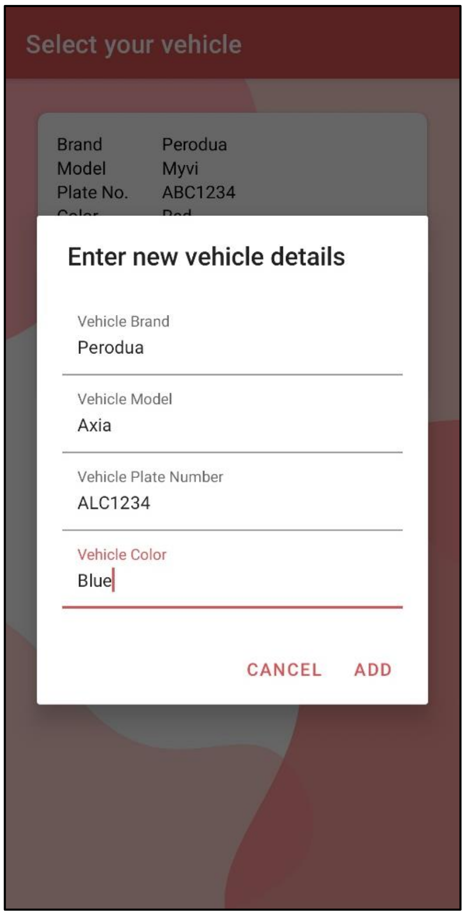
Brand	Perodua
Model	Myvi
Plate No.	ABC1234
Color	Red

Brand	Mercedes
Model	Benz
Plate No.	AKY5689
Color	Silver

Brand	Perodua ✓
Model	Axia
Plate No.	ALC1234
Color	Blue

Figure 5-2-8-2: Select Vehicle Page (2)

Next, figure 5-2-8-1 shows the select vehicle page. User needs to select the vehicle that they want to make appointment. In this page, all the vehicle of the user is displayed out with their details. It brings convenient to user as they no need to fill in the vehicle details again if they want to repeat booking for the same vehicle. Below the vehicle list is a “add vehicle” button and “continue” button. User can click on the “new vehicle” button if they wanted to add a new vehicle. User selects the vehicle that they wanted to book the appointment and a tick icon will appear on the selected vehicle.


Select your vehicle

Brand Perodua
Model Myvi
Plate No. ABC1234

Enter new vehicle details

Vehicle Brand
Perodua

Vehicle Model
Axia

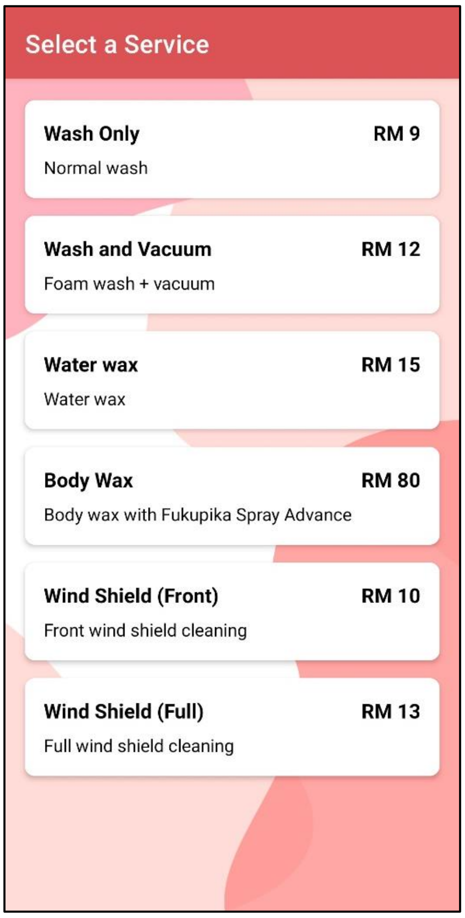
Vehicle Plate Number
ALC1234

Vehicle Color
Blue

Figure 5-2-8-3: Add New Vehicle Alert Dialog

When user clicked on the “add vehicle” button, an alert dialog is pop out is shown in figure 5-2-8-3. User need to fill in the new vehicle details such as the vehicle brand, vehicle model, vehicle plate number, and vehicle colour. After fill in the fields, user clicks on the “add” button. The new vehicle is added successfully. The new vehicle is also updated in the Cloud Firestore.

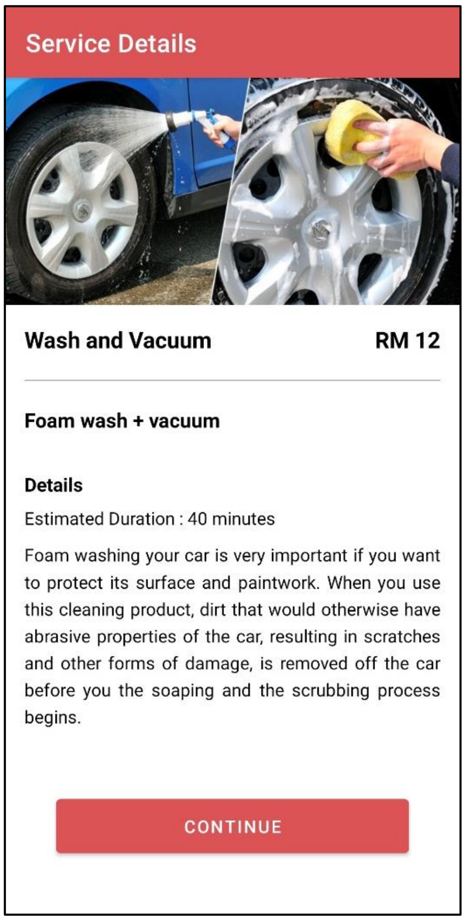
5.2.9 Select Service Page and Service Details Page



Select a Service

Wash Only Normal wash	**RM 9**
Wash and Vacuum Foam wash + vacuum	**RM 12**
Water wax Water wax	**RM 15**
Body Wax Body wax with Fukupika Spray Advance	**RM 80**
Wind Shield (Front) Front wind shield cleaning	**RM 10**
Wind Shield (Full) Full wind shield cleaning	**RM 13**

Figure 5-2-9-1: Select Service Page



Service Details

Wash and Vacuum **RM 12**

Foam wash + vacuum

Details

Estimated Duration : 40 minutes

Foam washing your car is very important if you want to protect its surface and paintwork. When you use this cleaning product, dirt that would otherwise have abrasive properties of the car, resulting in scratches and other forms of damage, is removed off the car before you the soaping and the scrubbing process begins.

Figure 5-2-9-2: Service Details Page

Figure 5-2-9-1 shows the screenshot of select service page. All the service provided from the selected car wash center is displayed in this page in a list. A card view contains the details of the service provided such as the service name, price and a short description is included. It is for customer to have simple introduction of the service. When customer clicked on the service, customer is directed to the service details page. In this page, the service details such as the service name, price, short description, estimated service duration and also the long description of the service is listed here. Customer can understand more about the service provided. To continue the booking process, customer can click on the “continue” button.

5.2.10 Select Time Slot Page

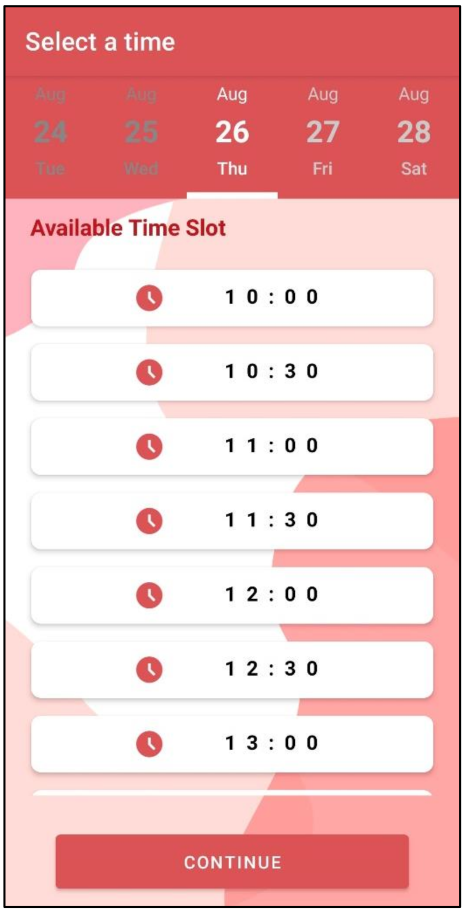


Figure 5-2-10-1: Select Time Slot Page (1)

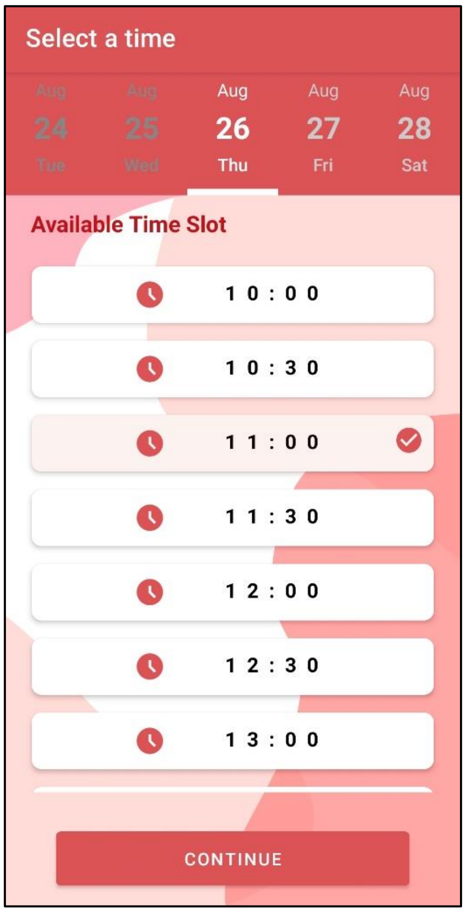
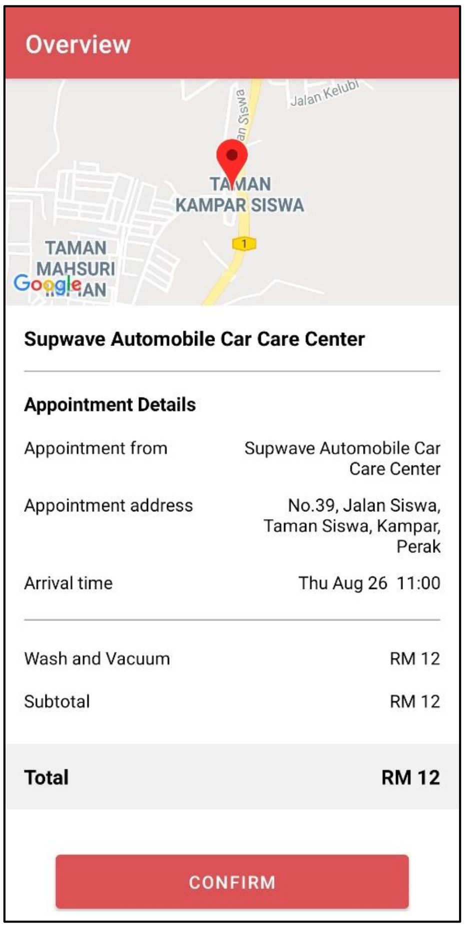


Figure 5-2-10-2: Select Time Slot Page (2)

Figure 5-2-10-1 shows the screenshot of select time slot page. On the top is the Horizontal Calendar and available time slot is displayed in this page. The horizontal calendar allows customer to select the date for the appointment. Customer can swipe to select the date. However, the customer cannot select the past date. Next, the available time slot is listed. Customer can select the time slot as they preferred. When customer selected the time slot, a tick will appear beside the selected time slot. To continue the booking process, customer clicks "continue" button.

5.2.11 Appointment Overview Page


Overview

Supwave Automobile Car Care Center

Appointment Details

Appointment from	Supwave Automobile Car Care Center
Appointment address	No.39, Jalan Siswa, Taman Siswa, Kampar, Perak
Arrival time	Thu Aug 26 11:00
Wash and Vacuum	RM 12
Subtotal	RM 12
Total	RM 12

Figure 5-2-11-1: Appointment Overview Page

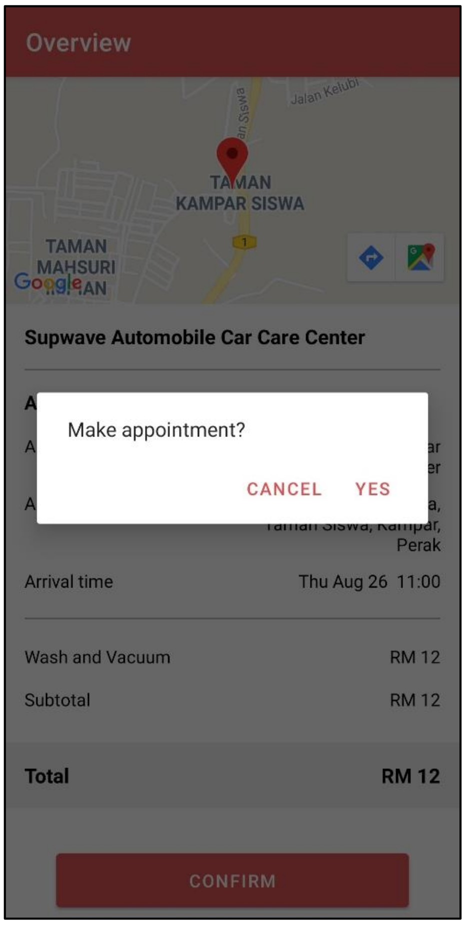
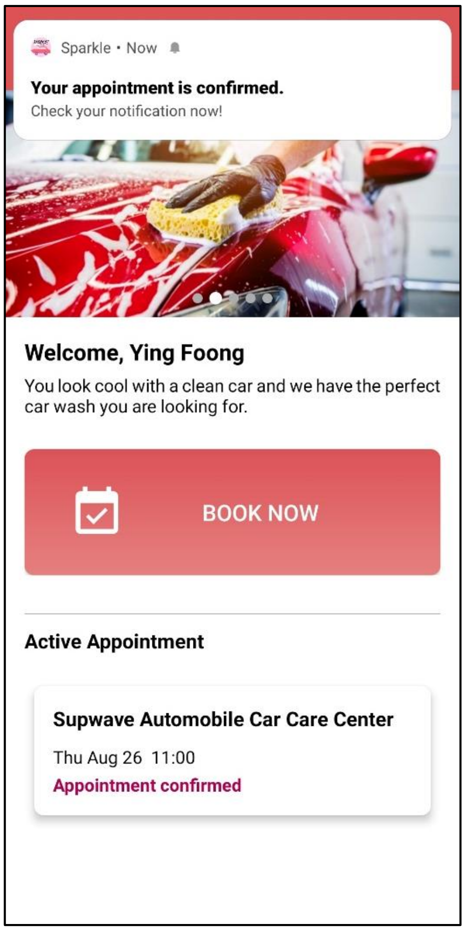

Make appointment?

Figure 5-2-11-2: Confirm Appointment

The overview of the appointment is displayed in appointment overview page as shown in figure 5-2-11-1. All the details are displayed in the page. Customer can have an overview and check the appointment for confirmation. The appointment details such as the car wash center details, appointment time, service details and total price is displayed. On the upper part of the page is a map with marker which indicates the location of the car wash center. To confirm the appointment, customer need to click on the “confirm” button. An alert dialog is pop out to ask for customer’s confirmation. Customer click “yes” if they confirm the appointment.



Sparkle • Now

Your appointment is confirmed.
Check your notification now!

Welcome, Ying Foong

You look cool with a clean car and we have the perfect car wash you are looking for.

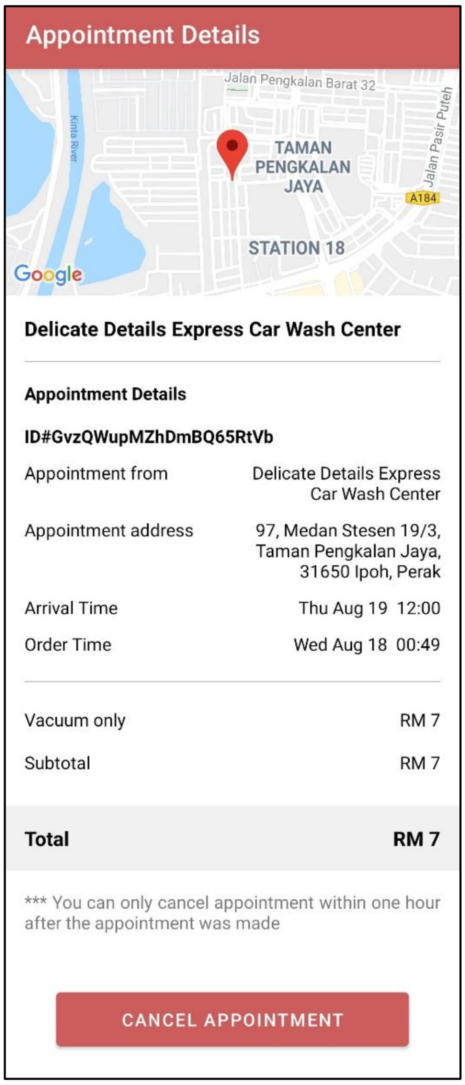
Active Appointment

Supwave Automobile Car Care Center
Thu Aug 26 11:00
Appointment confirmed

Figure 5-2-11-3: Notification Received

After customer confirm the appointment, customer is directed back to the customer home page. Then, a notification is pop out at the upper part of the customer home page. Moreover, the active appointment list in the home page is updated. The newly booked appointment is update at the Cloud Firestore and displayed out in the active appointment list.

5.2.12 Active Appointment Details Page



Delicate Details Express Car Wash Center

Appointment Details

ID#GvzQWupMZhDmBQ65RtVb

Appointment from	Delicate Details Express Car Wash Center
Appointment address	97, Medan Stesen 19/3, Taman Pengkalan Jaya, 31650 Ipoh, Perak
Arrival Time	Thu Aug 19 12:00
Order Time	Wed Aug 18 00:49
Vacuum only	RM 7
Subtotal	RM 7
Total	**RM 7**

*** You can only cancel appointment within one hour after the appointment was made

Figure 5-2-12-1: Active Appointment Details Page

Figure 5-2-12-1 shows the screenshot of active appointment details page. All the details are displayed in the page. Customer can have an overview of the appointment. The appointment details such as the car wash center details, appointment time, service details and total price is displayed. On the upper part of the page is a map with marker which indicates the location of the car wash center. If customer wants to cancel the appointment, they can click on the "cancel" button on the bottom of the page. However, the cancellation only can be made within an hour after the appointment was made.

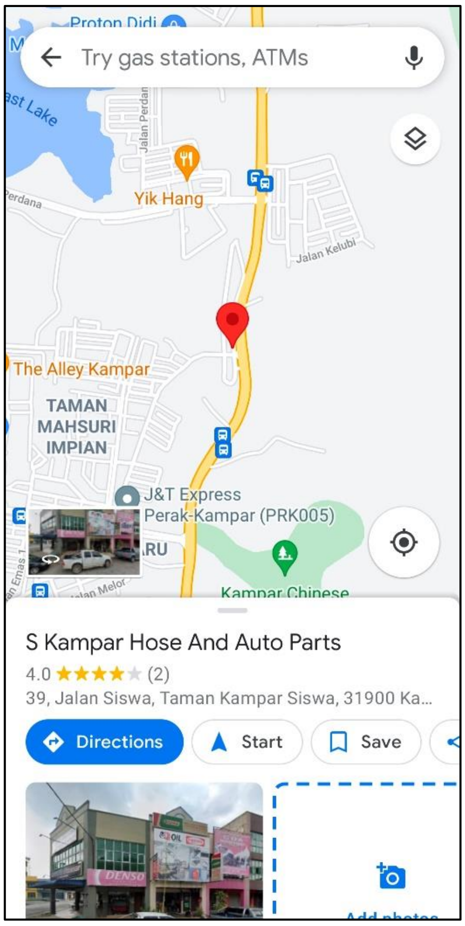
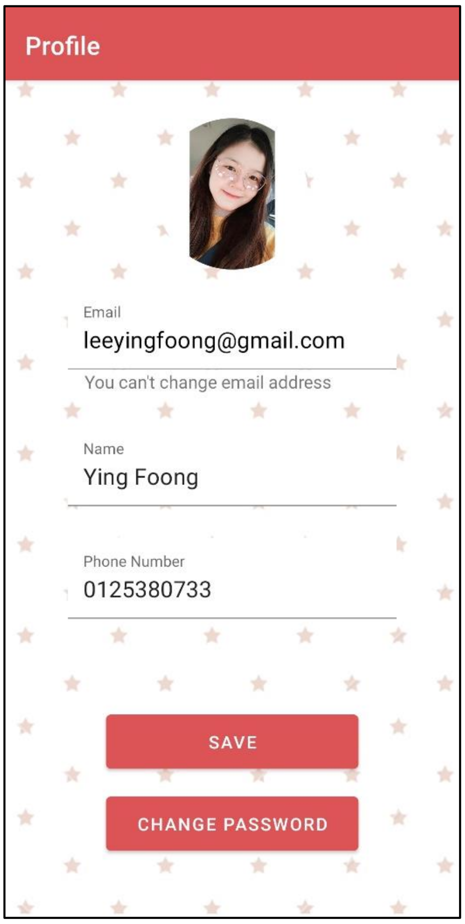

Try gas stations, ATMs
Yik Hang
The Alley Kampar
TAMAN MAHSURI IMPIAN
J&T Express Perak-Kampar (PRK005)
S Kampar Hose And Auto Parts
4.0 ★★★★★ (2)
39, Jalan Siswa, Taman Kampar Siswa, 31900 Ka...
Directions
Start
Save

Figure 5-2-12-2: Google Map Direction

When user clicked on the marker pinned on the map in active appointment details page, customer is directed to Google Map application. Then, user can click the “direction” button to get the direction to the car wash center. This function is to guide the customer to the destination when they are not familiar with the places. Customer no need to find the car wash center themselves as they system will lead them to the Google Map with destination set.

5.2.13 Manage Profile and Change Password

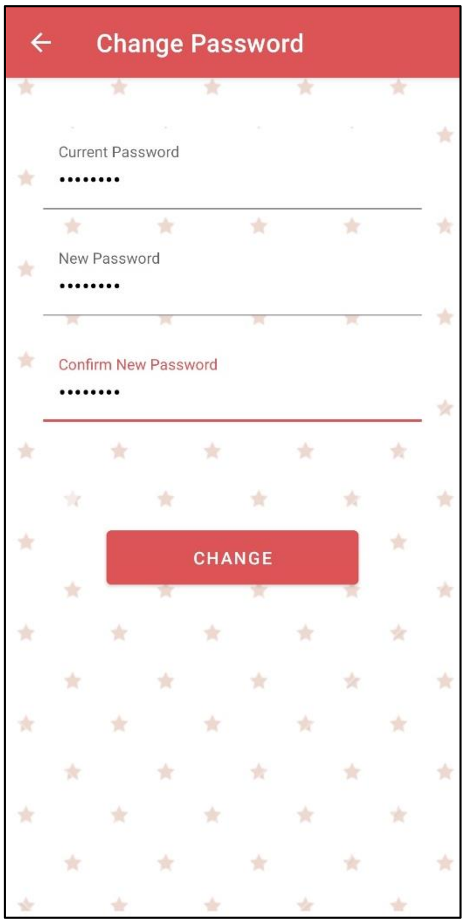

Profile

Email
leeyingfoong@gmail.com
You can't change email address

Name
Ying Foong

Phone Number
0125380733

Figure 5-2-13-1: Profile Page


Change Password

Current Password
••••••••

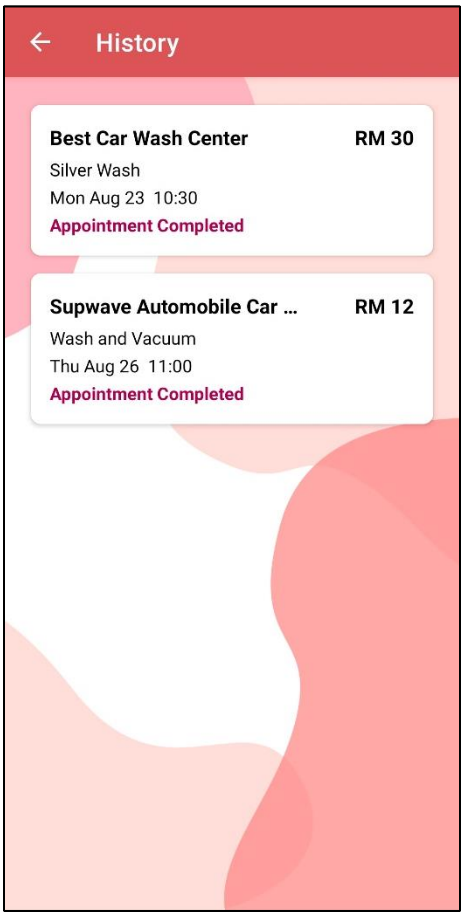
New Password
••••••••

Confirm New Password
••••••••

Figure 5-2-13-2: Change Password Page

In user profile page, the details of the user such as the email, name and phone number is displayed. The details of user are retrieved from the Cloud Firestore. User can change their name and phone number. However, they are not allowed to change the email because the system needs the email to authenticate the identity of user. If user wants to change their password, they need to click the “change password” button in the profile page. Then, they will be directed to the change password page. User need to key in the current password, new password and confirmation password. Current password is needed in order to authorized the user. next, the new password and confirmation password need to be the same as it is used to confirm the new password. When user clicks the “change” button, the password of the user is changed successfully.

5.2.14 Appointment History List and Appointment Details

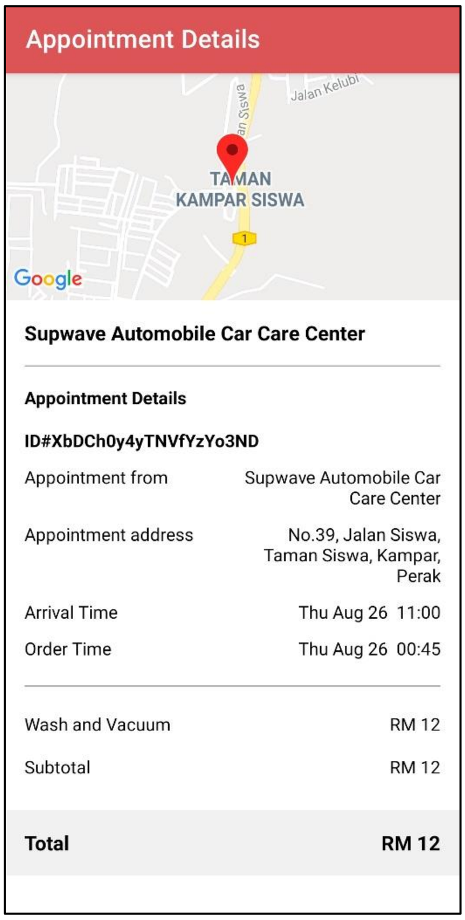


History

Best Car Wash Center RM 30
Silver Wash
Mon Aug 23 10:30
Appointment Completed

Supwave Automobile Car ... RM 12
Wash and Vacuum
Thu Aug 26 11:00
Appointment Completed

Figure 5-2-14-1: Appointment History List Page



Appointment Details

Supwave Automobile Car Care Center

Appointment Details

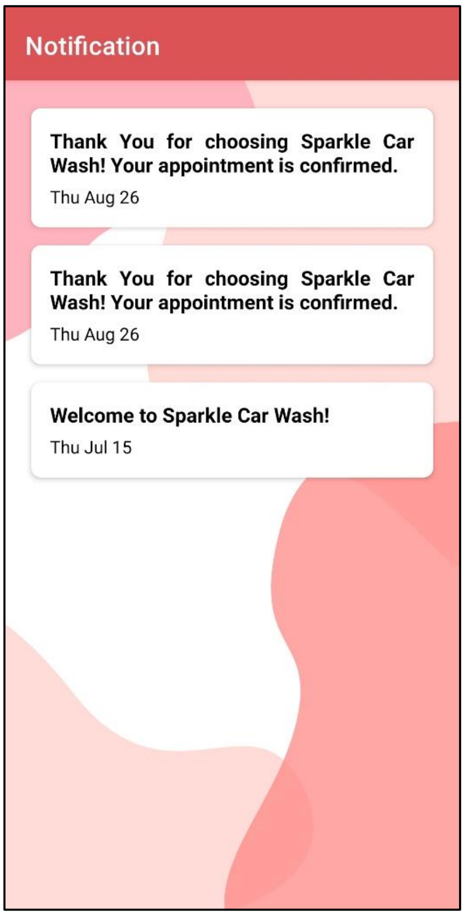
ID#XbDCh0y4yTNVfYzYo3ND

Appointment from	Supwave Automobile Car Care Center
Appointment address	No.39, Jalan Siswa, Taman Siswa, Kampar, Perak
Arrival Time	Thu Aug 26 11:00
Order Time	Thu Aug 26 00:45
Wash and Vacuum	RM 12
Subtotal	RM 12
Total	**RM 12**

Figure 5-2-14-2: Appointment Details Page

Figure 5-2-14-1 shows the screenshot of appointment history list. All the history of the customer is retrieved form the Cloud Firestore and displayed. In appointment history list page, the card view is used to hold the overall details of the appointment such as the car wash center name, service name, price, appointment time and appointment status. When customer clicked into the appointment, customer is directed to appointment details page as shown in figure 5-2-14-2. The appointment details such as the car wash center details, appointment time, service details and total price is displayed. On the upper part of the page is a map with marker which indicates the location of the car wash center.

5.2.15 Notification List and Notification Details

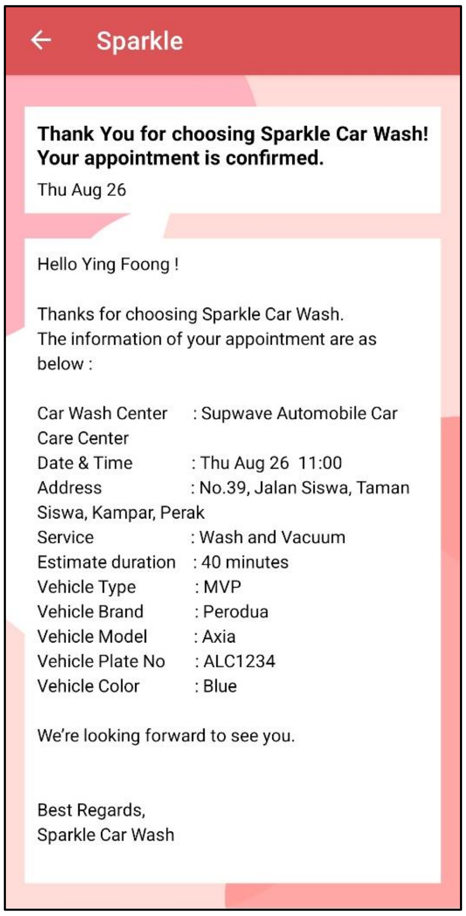

Notification

Thank You for choosing Sparkle Car Wash! Your appointment is confirmed.
Thu Aug 26

Thank You for choosing Sparkle Car Wash! Your appointment is confirmed.
Thu Aug 26

Welcome to Sparkle Car Wash!
Thu Jul 15

Figure 5-2-15-1: Notification List Page


Sparkle

Thank You for choosing Sparkle Car Wash! Your appointment is confirmed.
Thu Aug 26

Hello Ying Foong !

Thanks for choosing Sparkle Car Wash.
The information of your appointment are as below :

Car Wash Center : Supwave Automobile Car Care Center
Date & Time : Thu Aug 26 11:00
Address : No.39, Jalan Siswa, Taman Siswa, Kampar, Perak
Service : Wash and Vacuum
Estimate duration : 40 minutes
Vehicle Type : MVP
Vehicle Brand : Perodua
Vehicle Model : Axia
Vehicle Plate No : ALC1234
Vehicle Color : Blue

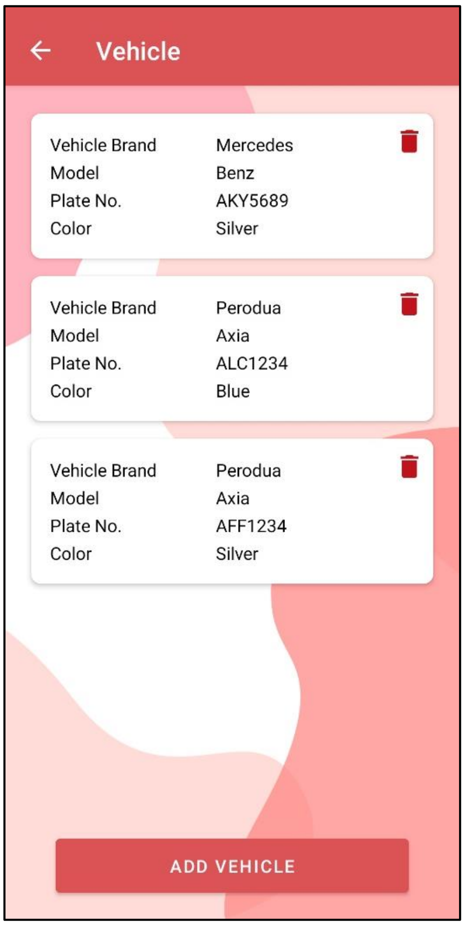
We're looking forward to see you.

Best Regards,
Sparkle Car Wash

Figure 5-2-15-2: Notification Details Page

Figure 5-2-15-1 shows the screenshot of notification list. All the notification received by the customer is retrieved form the Cloud Firestore and displayed. In notification list page, the card view is used to hold the overview of notification such as the notification title and time received. When customer clicked into the notification, customer is directed to notification details page as shown in figure 5-2-15-2. The notification details such as the title, time and content are displayed.

5.2.16 Manage Customer Vehicle List

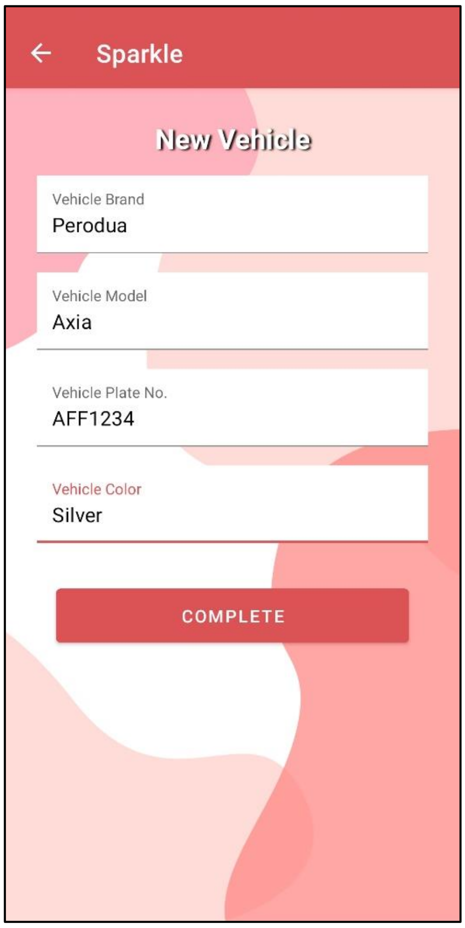


Vehicle Brand	Mercedes
Model	Benz
Plate No.	AKY5689
Color	Silver

Vehicle Brand	Perodua
Model	Axia
Plate No.	ALC1234
Color	Blue

Vehicle Brand	Perodua
Model	Axia
Plate No.	AFF1234
Color	Silver

Figure 5-2-16-1: Customer Vehicle List Page



Vehicle Brand
Perodua

Vehicle Model
Axia

Vehicle Plate No.
AFF1234

Vehicle Color
Silver

Figure 5-2-16-2: Add Customer Vehicle Page

Figure 5-2-16-1 shows the screenshot of customer's vehicle list. All the customer's vehicle is retrieved form the Cloud Firestore and displayed. In customer's vehicle list page, the card view is used to hold the details of the vehicle such as the vehicle brand, model, plate number and colour. On the top right side of the card view has a delete icon and it will perform delete function if the user clicked on it. Below of the customer vehicle list page has a "add vehicle" button. When customer clicked on the button, user is directed to add vehicle page. There are 4 inputs required which is the vehicle brand, model, plate number and colour. After customer had completed the form, they can click the "complete" button. The new vehicle is added to the Cloud Firestore and the customer vehicle list is updated.

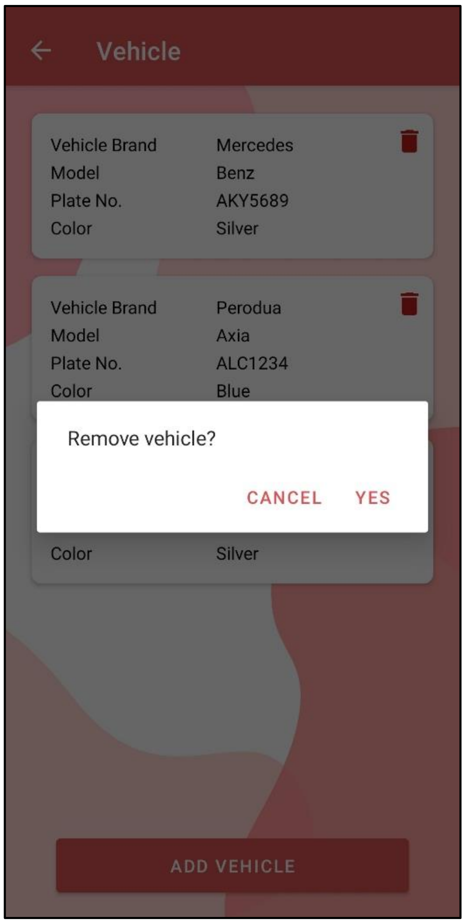


Figure 5-2-16-3: Remove Customer Vehicle Alert Dialog

If customer wishes to remove the vehicle from the customer's vehicle list, they can click the delete icon on the card view of each of the vehicle. An alert dialog is pop out to ask for the confirmation of deletion of the vehicle. Then, click "yes" to remove the vehicle. It is to ensure that the customer will not accidentally clicked on the delete icon and deleted the vehicle.

5.2.17 Cleaner's Home Page

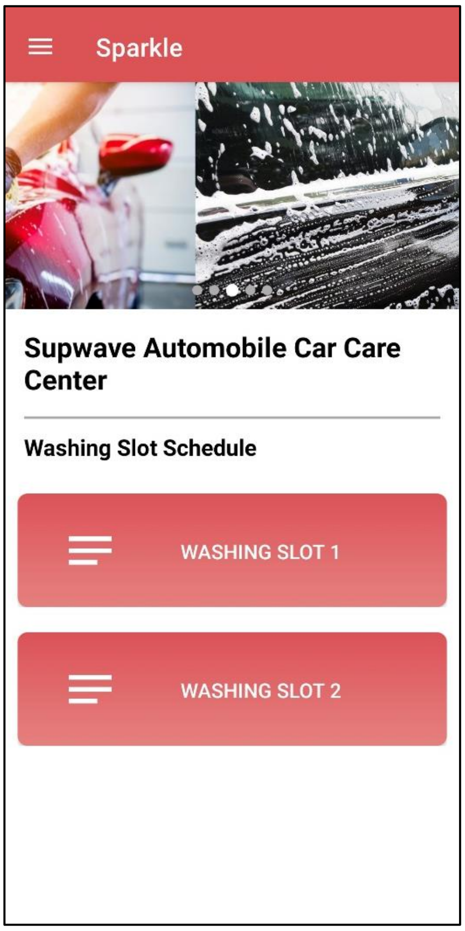


Figure 5-2-17-1: Cleaner's Home Page

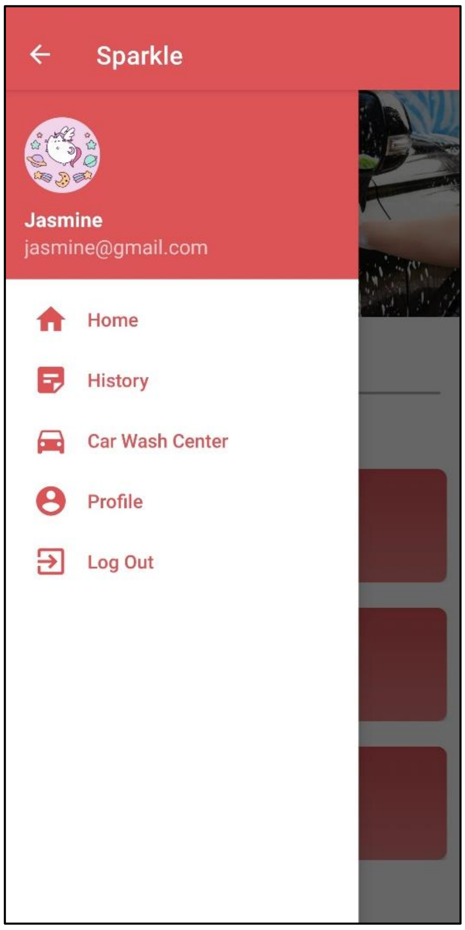
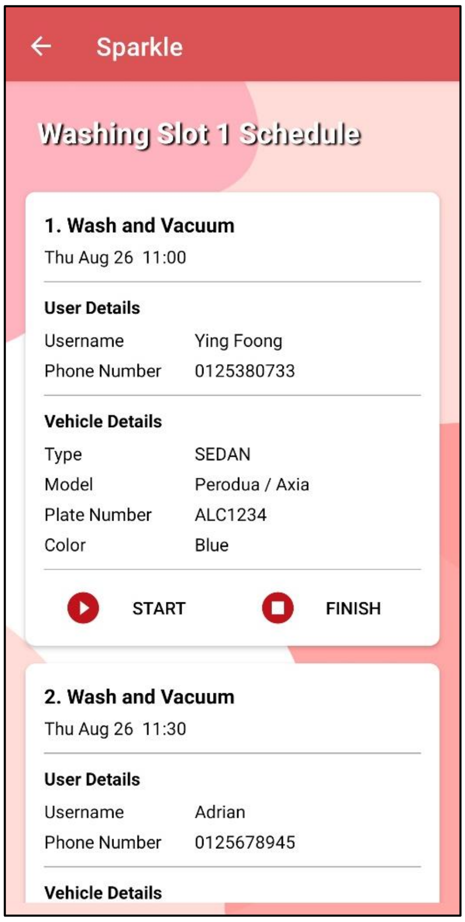


Figure 5-2-17-2: Cleaner's Home Page Navigation Bar

After user had logged in to the account, the system will check the role of the user. If the role of user is cleaner, the user will be directed to the cleaner's home page. In the cleaner home page, there is the name of the car wash center and the washing slot of the car wash center. Cleaner can click on the washing slot to see the scheduled appointment of the car wash center. Besides, there is a navigation bar at the cleaner's home page. Profile picture and email of the cleaner is displayed at the top of the navigation bar. The menu that lead to different page is also listed in the navigation bar such as history, car wash center and profile. It is convenient for cleaner to choose which features they want or which page they go.

5.2.18 Scheduled Appointment List Page


Sparkle

Washing Slot 1 Schedule

1. Wash and Vacuum
Thu Aug 26 11:00

User Details

Username	Ying Foong
Phone Number	0125380733

Vehicle Details

Type	SEDAN
Model	Perodua / Axia
Plate Number	ALC1234
Color	Blue

2. Wash and Vacuum
Thu Aug 26 11:30

User Details

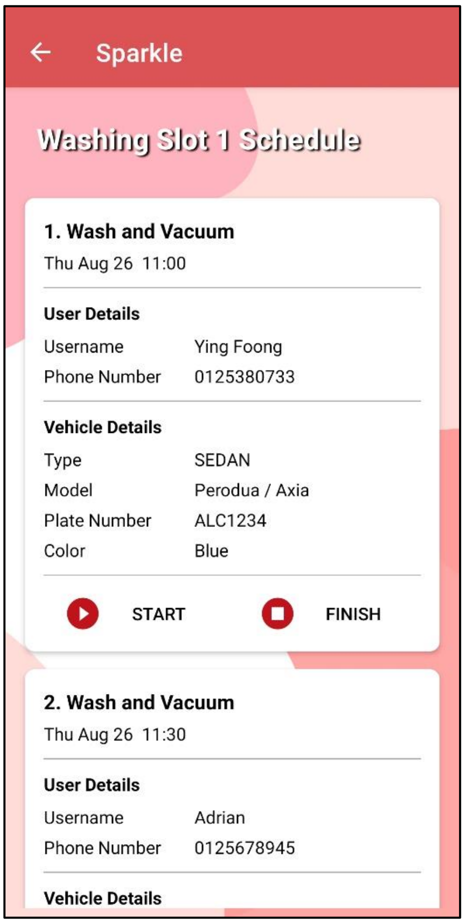
Username	Adrian
Phone Number	0125678945

Vehicle Details

Figure 5-2-18-1: Scheduled Appointment List Page

Figure 5-2-18-1 shows the screenshot of the scheduled appointment list. The appointment is scheduled by the system and display out according to the washing slot. Each of the appointment is displayed inside a card view. The card view contains the user details, service, booking time and vehicle details. Moreover, there are two button at the bottom of the card view which is the “start” button and “finish” button. “Start” button indicates the cleaner starts to clean the vehicle while the “finish” button indicates the cleaner had finished cleaning the vehicle. On the top left corner of the car view has a number which indicates the appointment should be executed first.

5.2.19 Manage Scheduled Appointment


Sparkle

Washing Slot 1 Schedule

1. Wash and Vacuum
Thu Aug 26 11:00

User Details

Username	Ying Foong
Phone Number	0125380733

Vehicle Details

Type	SEDAN
Model	Perodua / Axia
Plate Number	ALC1234
Color	Blue

2. Wash and Vacuum
Thu Aug 26 11:30

User Details

Username	Adrian
Phone Number	0125678945

Vehicle Details

Figure 5-2-19-1: Manage Scheduled Appointment Page (1)

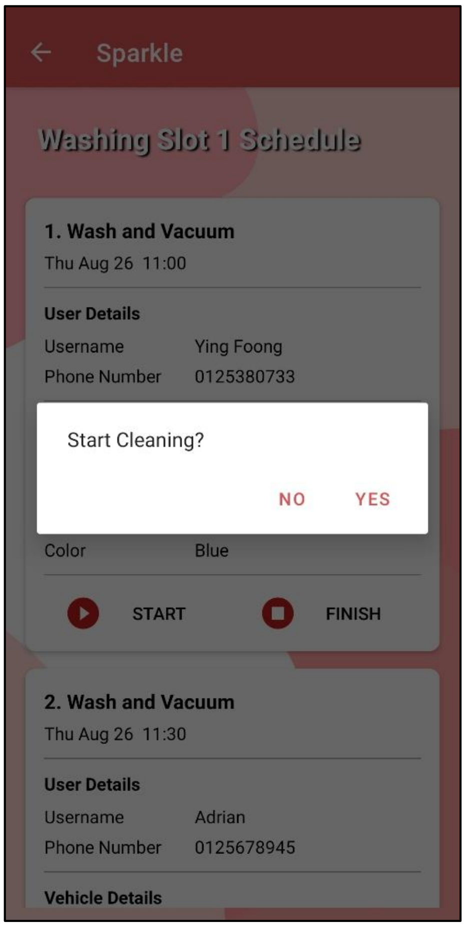
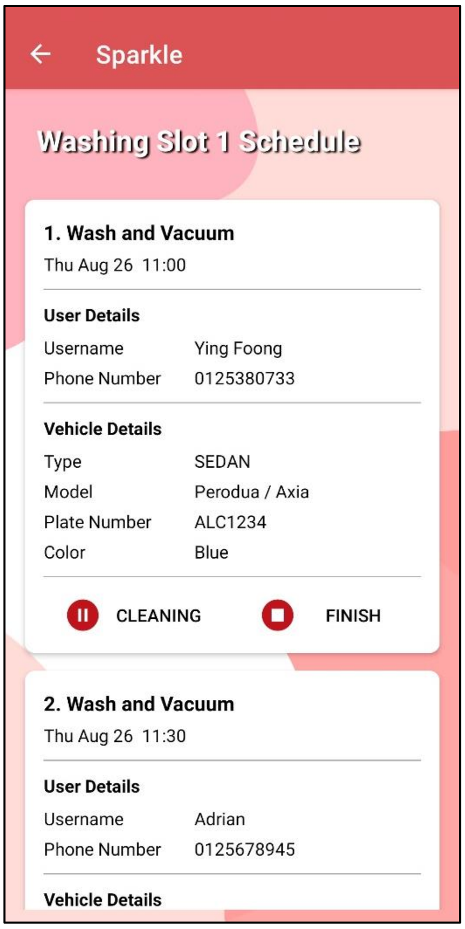

Start Cleaning?

Figure 5-2-19-2: Manage Scheduled Appointment Page (2)

Now, the cleaner needs to start the car wash service. Hence, cleaner clicked on the “start” button. An alert dialog is pop out as shown in figure 5-2-19-2. The alert dialog is to prompt for the confirmation to start the appointment. To confirm, cleaner needs to click on the “yes” button. The status of the appointment will be changed and the customer will also be informed.



Sparkle

Washing Slot 1 Schedule

1. Wash and Vacuum
Thu Aug 26 11:00

User Details

Username	Ying Foong
Phone Number	0125380733

Vehicle Details

Type	SEDAN
Model	Perodua / Axia
Plate Number	ALC1234
Color	Blue

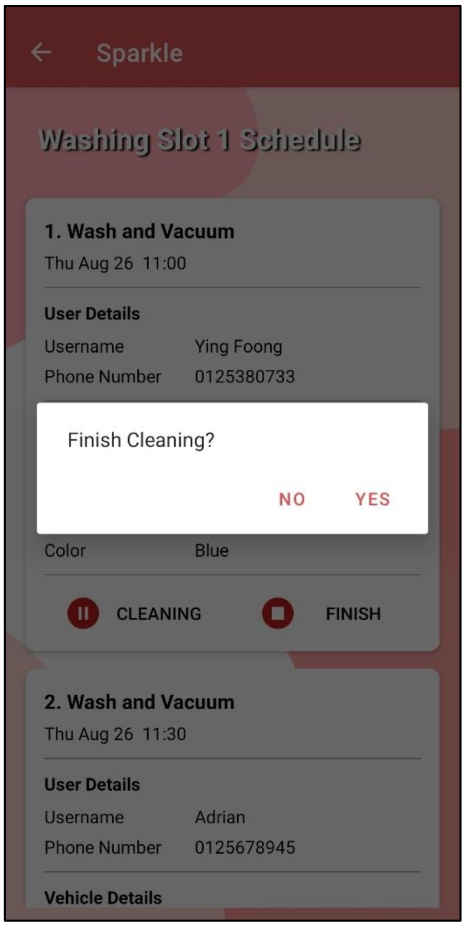
2. Wash and Vacuum
Thu Aug 26 11:30

User Details

Username	Adrian
Phone Number	0125678945

Vehicle Details

Figure 5-2-19-3: Manage Scheduled Appointment Page (3)



Sparkle

Washing Slot 1 Schedule

1. Wash and Vacuum
Thu Aug 26 11:00

User Details

Username	Ying Foong
Phone Number	0125380733

Finish Cleaning?

Color	Blue

2. Wash and Vacuum
Thu Aug 26 11:30

User Details

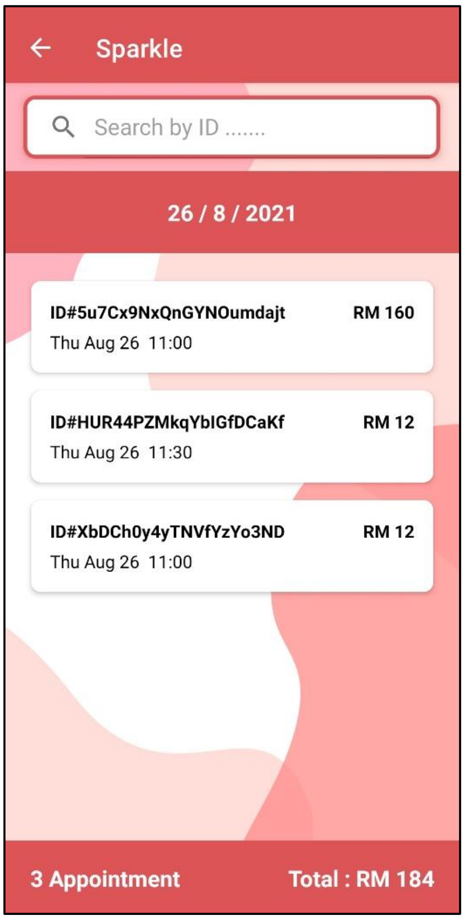
Username	Adrian
Phone Number	0125678945

Vehicle Details

Figure 5-2-19-4: Manage Scheduled Appointment Page (4)

Now, the cleaner needs to stop the car wash service. Hence, cleaner clicked on the “finish” button. An alert dialog is pop out as shown in figure 5-2-19-3. The alert dialog is to prompt for the confirmation to stop the appointment. To confirm, cleaner needs to click on the “yes” button. The status of the appointment will be changed and the customer will also be informed.

5.2.20 Car Wash Center Appointment History and Appointment Details



ID#5u7Cx9NxQnGYNOumdajt RM 160
Thu Aug 26 11:00

ID#HUR44PZMkqYblGfDCaKf RM 12
Thu Aug 26 11:30

ID#XbDCh0y4yTNVfYzYo3ND RM 12
Thu Aug 26 11:00

3 Appointment Total : RM 184

Figure 5-2-20-1: Car Wash Center Appointment History Page (1)

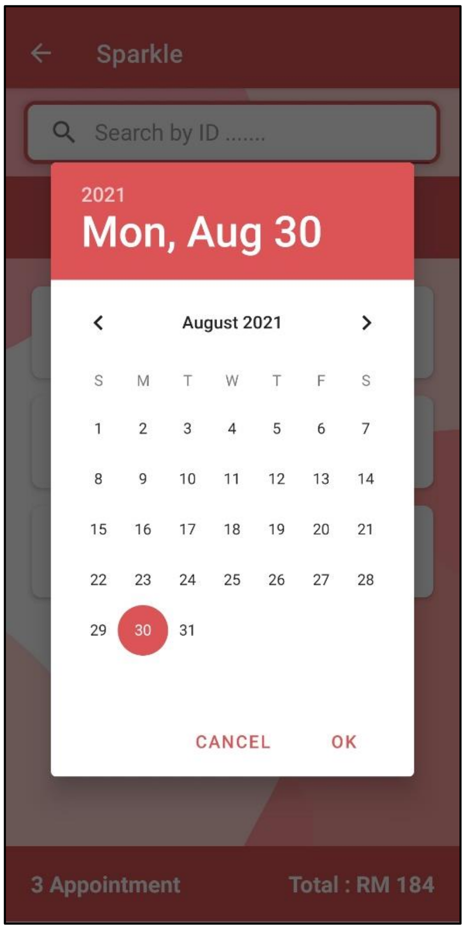


Figure 5-2-20-2: Car Wash Center Appointment History Page (2)

Figure 5-2-20-1 shows the screenshot of the car wash center appointment history page. This page displays all the appointment according to the date. Besides, cleaner also can view the total number of appointment and total profit earn on a certain date. On the top of the page has a search bar which allows cleaner to search for the appointment by appointment ID. Then, below the search bar is a banner that display the date. Cleaner can clicked on the date to change the date they want to view. When they clicked on the date, date picker dialog is pop out as shown in figure 5-2-20-2. Cleaner can select the date as they preferred. After the date is selected, the appointment of the selected day is displayed out in list. On the bottom of the page, the total number of appointment and total profit earned on the date selected is displayed.

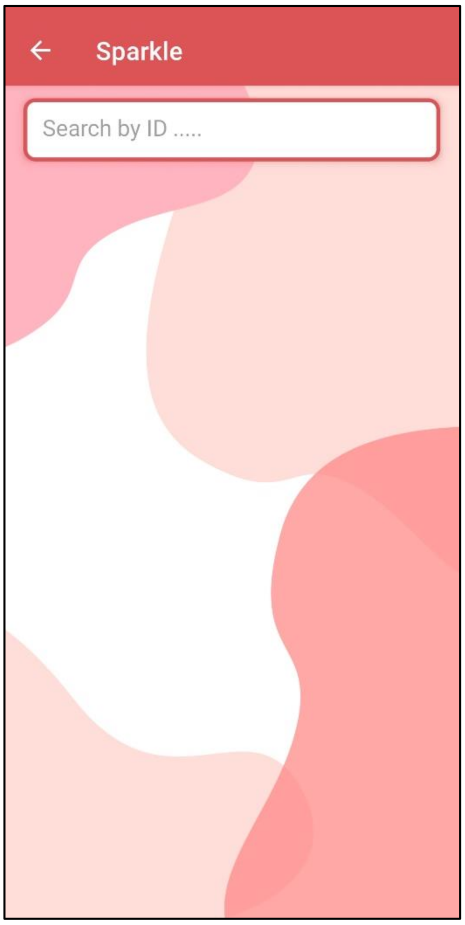
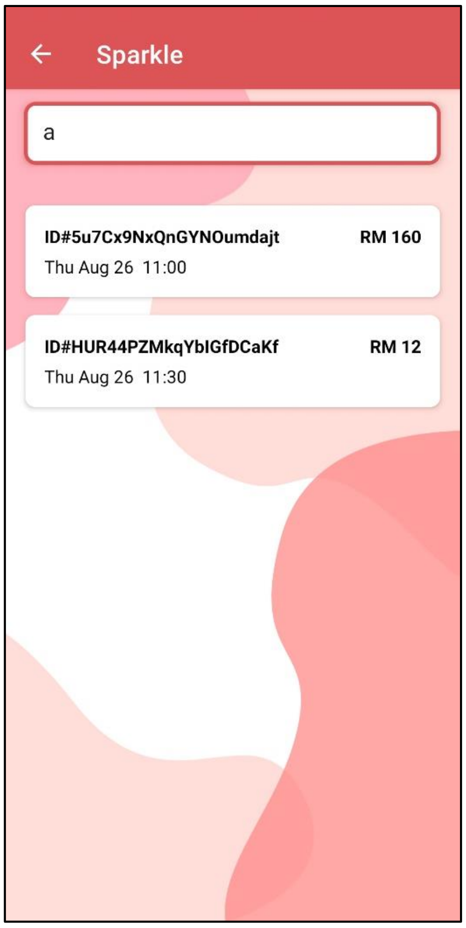


Figure 5-2-20-3: Search Car Wash Center's Appointment History Page (1)

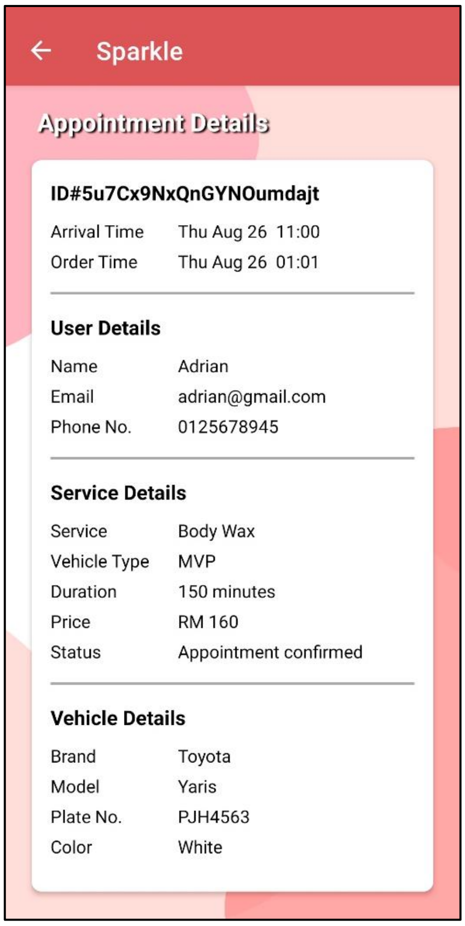


ID#5u7Cx9NxQnGYNOumdajt RM 160
Thu Aug 26 11:00

ID#HUR44PZMkqYblGfDCaKf RM 12
Thu Aug 26 11:30

Figure 5-2-20-4: Search Car Wash Center's Appointment History Page (2)

Figure 5-2-20-3 shows the screenshot of search car wash center's appointment history page. Cleaners are able to search the appointment by ID in this page. When cleaner enter the ID of the appointment, the system will filter the appointment that contains the cleaner's input. The appointment filtered is then displayed in a list. The ID, appointment time and price is displayed. When cleaner clicked on the appointment displayed, cleaner is directed to appointment details page.



ID#5u7Cx9NxQnGYNOumdajt

Arrival Time	Thu Aug 26 11:00
Order Time	Thu Aug 26 01:01

User Details

Name	Adrian
Email	adrian@gmail.com
Phone No.	0125678945

Service Details

Service	Body Wax
Vehicle Type	MVP
Duration	150 minutes
Price	RM 160
Status	Appointment confirmed

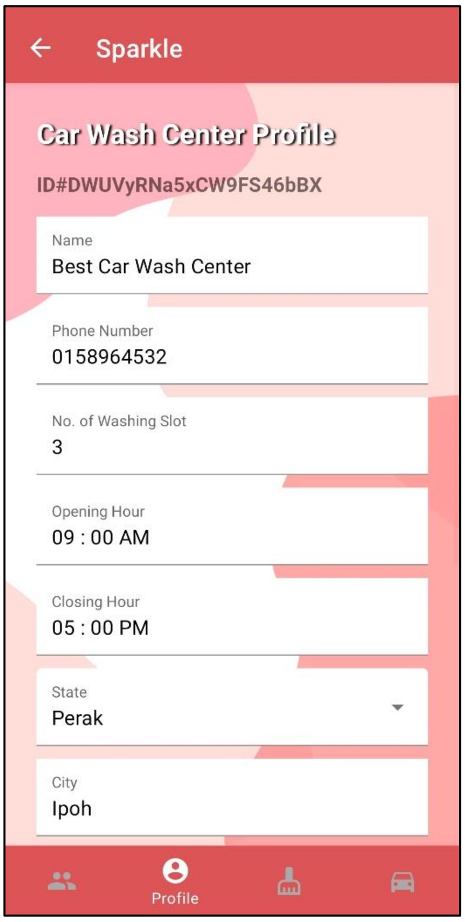
Vehicle Details

Brand	Toyota
Model	Yaris
Plate No.	PJH4563
Color	White

Figure 5-2-20-5: Car Wash Center’s Appointment Details Page

Figure 5-2-20-5 shows the screenshot of car wash center’s appointment details. When the cleaner clicked on the appointment in the previous page, the system will direct cleaner to this page. In this page, the details of the appointment are displayed such as the appointment arrival time, order time, user details, service details and the vehicle details. It is convenient to cleaner when they want to check the appointment details and they just have to find by the unique appointment ID.

5.2.21 Manage Car Wash Center Details


Sparkle

Car Wash Center Profile

ID#DWUVyRNa5xCW9FS46bBX

Name: Best Car Wash Center

Phone Number: 0158964532

No. of Washing Slot: 3

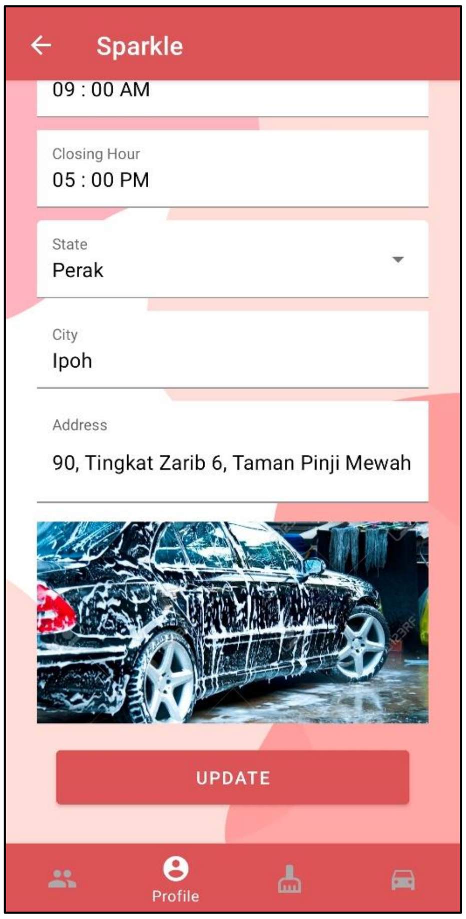
Opening Hour: 09 : 00 AM

Closing Hour: 05 : 00 PM

State: Perak

City: Ipoh

Figure 5-2-21-1: Car Wash Center Profile Page (1)


Sparkle

09 : 00 AM

Closing Hour: 05 : 00 PM

State: Perak

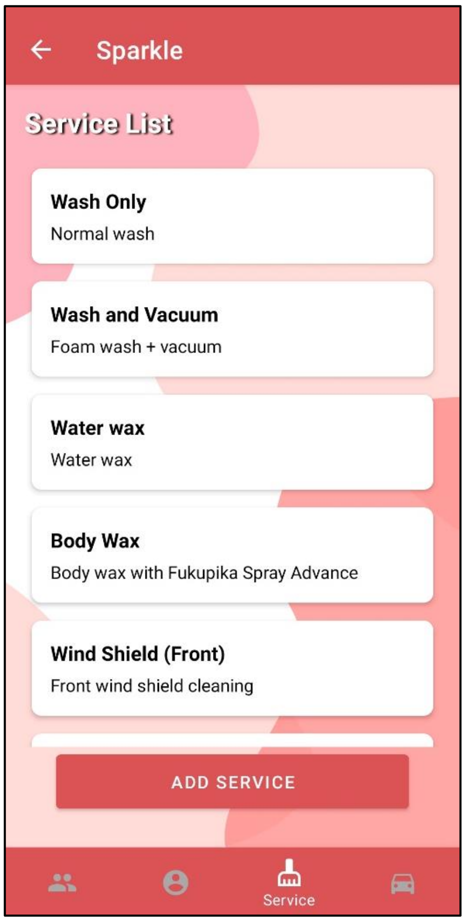
City: Ipoh

Address: 90, Tingkat Zarib 6, Taman Pinji Mewah

Figure 5-2-21-2: Car Wash Center Profile Page (2)

Figure 5-2-21-1 and figure 5-2-21-2 shows the screenshot of car wash center profile page. All the details of the car wash center profile are displayed in this page. The details such as the name, business hour, state, city, address, and profile picture is included in this page. If cleaner wanted to edit the details of the car wash center, they can just edit on the edit box and clicked on the "update" button. When the "update" button is clicked, the details of the profile will be updated in Cloud Firebase. In addition, the profile picture of the car wash center will be store in the Cloud Storage.

5.2.22 Manage Car Wash Center Service



Sparkle

Service List

Wash Only
Normal wash

Wash and Vacuum
Foam wash + vacuum

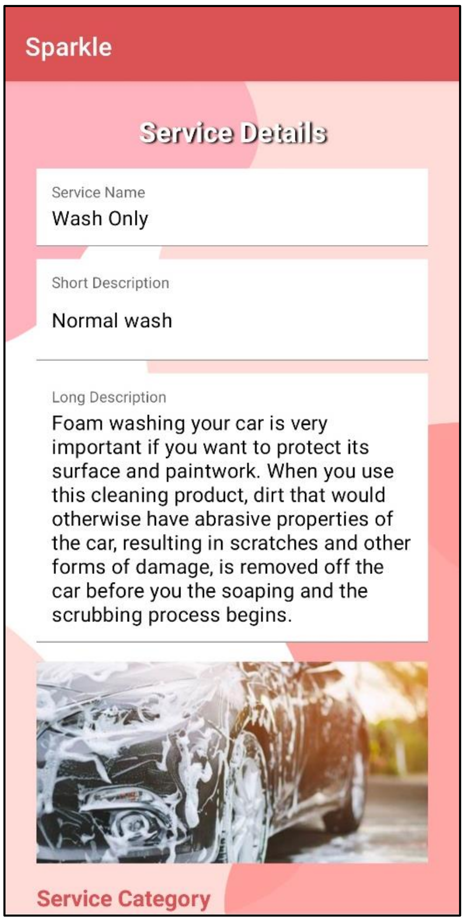
Water wax
Water wax

Body Wax
Body wax with Fukupika Spray Advance

Wind Shield (Front)
Front wind shield cleaning

Figure 5-2-22-1: Car Wash Center Service Page

Figure 5-2-22-1 shows the screenshot of the car wash center service page. All the service provided by the car wash center is displayed in a list. Each card contains the service name and its short description. A “add service” button is placed at the bottom of the page. Cleaner can click the button if they want to add new service to the car wash center. If cleaner clicked on the service, the system will direct cleaner to the car wash center’s service details page.


Sparkle

Service Details

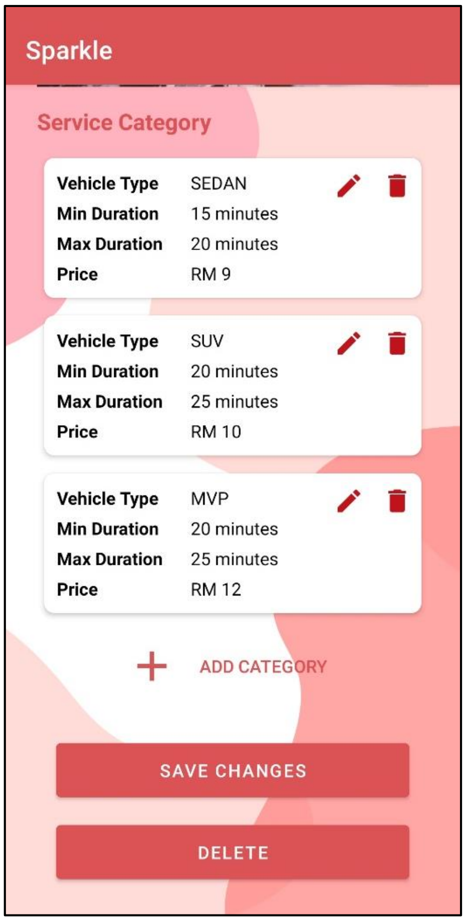
Service Name
Wash Only

Short Description
Normal wash

Long Description
Foam washing your car is very important if you want to protect its surface and paintwork. When you use this cleaning product, dirt that would otherwise have abrasive properties of the car, resulting in scratches and other forms of damage, is removed off the car before you the soaping and the scrubbing process begins.

Service Category

Figure 5-2-22-2: Car Wash Center Service Details Page (1)


Sparkle

Service Category

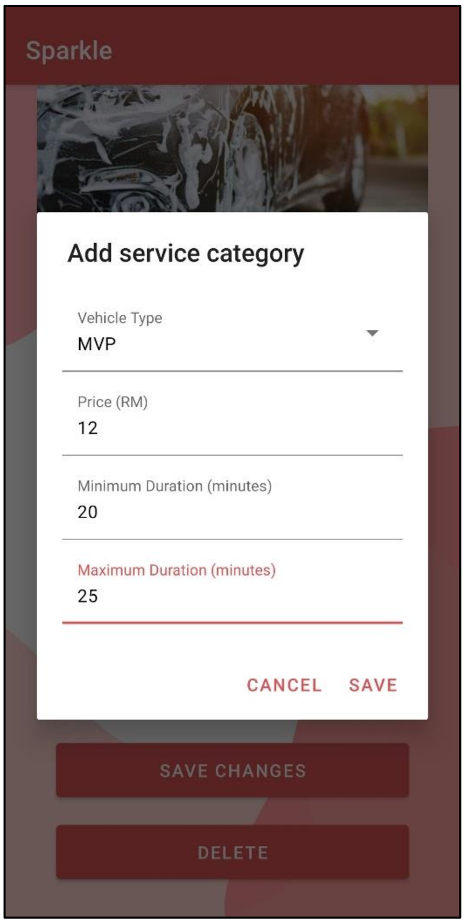
Vehicle Type	SEDAN
Min Duration	15 minutes
Max Duration	20 minutes
Price	RM 9

Vehicle Type	SUV
Min Duration	20 minutes
Max Duration	25 minutes
Price	RM 10

Vehicle Type	MVP
Min Duration	20 minutes
Max Duration	25 minutes
Price	RM 12

Figure 5-2-22-3: Car Wash Center Service Details Page (2)

Figure 5-2-22-2 and figure 5-2-22-3 shows the screenshot of car wash center service details page. All the details of the service are displayed here. In this page, the details of service such as service name, short description, long description, service photo, and service category is displayed. Service category is the service that provides to different vehicle type. The service provided to different type of vehicle has different price and duration. Each of the service category contains a few details which is the vehicle type, minimum duration, maximum duration and the price. There are also an edit icon and a delete icon at the top right corner of the card view. Below the service category list has a “add category” button. Cleaner can add service category by clicking the button. If the cleaner edited the details of the service, they need to click on the “save changes” button in order to save the changes and update the database.



Add service category

Vehicle Type: MVP

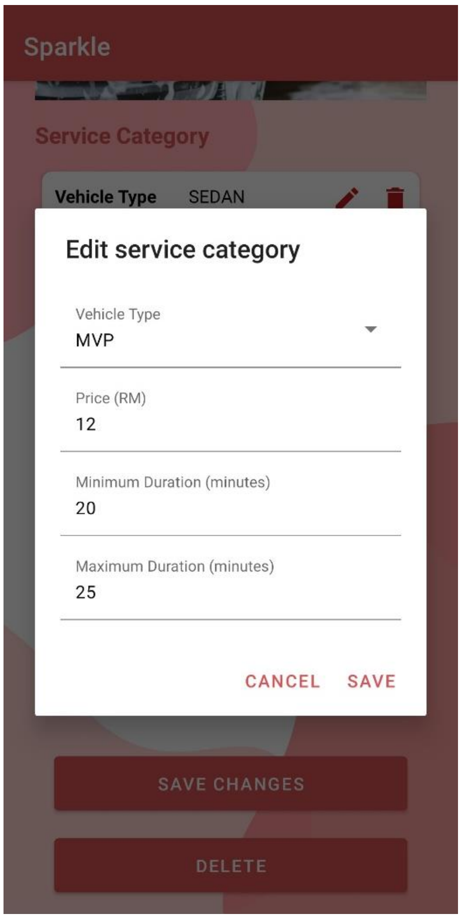
Price (RM): 12

Minimum Duration (minutes): 20

Maximum Duration (minutes): 25

CANCEL SAVE

Figure 5-2-22-4: Car Wash Center Service Details Page (3)



Edit service category

Vehicle Type: MVP

Price (RM): 12

Minimum Duration (minutes): 20

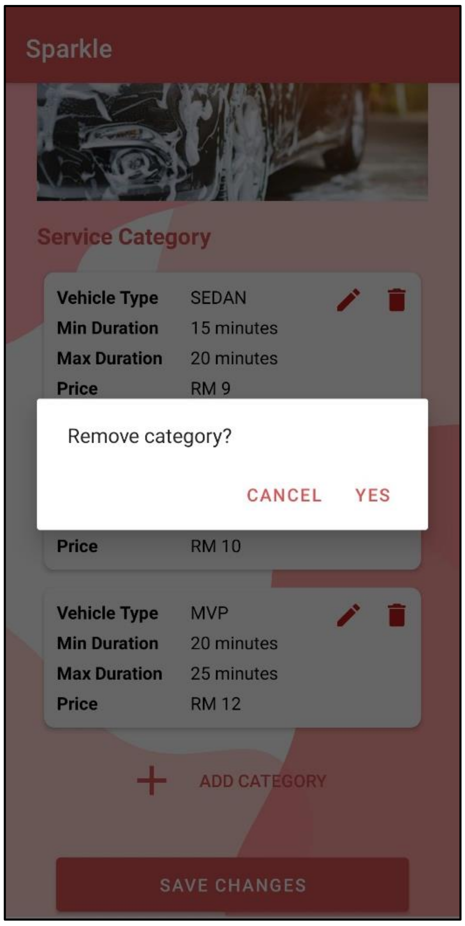
Maximum Duration (minutes): 25

CANCEL SAVE

Figure 5-2-22-5: Car Wash Center Service Details Page (4)

When cleaner clicked on edit button at the top right corner of the service category card view, a dialog is pop out and display all the details of the service category as shown in figure 5-2-22-5. Cleaner can edit the price, minimum duration and maximum duration of the service category. After finish editing, cleaner need to click on the “save” button to save the changes.

When cleaner clicked on the “add category” button, a dialog is pop out and prompt for the details of the service category as shown in figure 5-2-22-4.Cleaner need to slect the vehicle type, fill in the price, minimum duration and maximum duration of the service category. Then, click “save” button to add the service category.



Remove category?

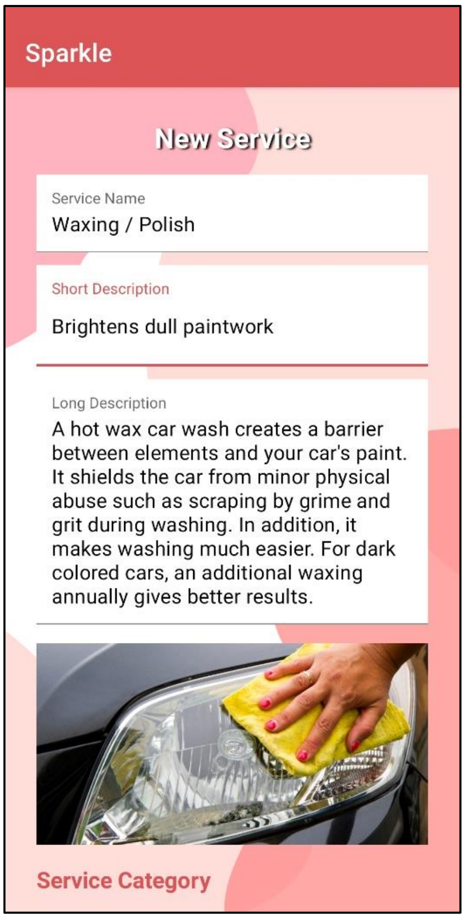
Vehicle Type	SEDAN
Min Duration	15 minutes
Max Duration	20 minutes
Price	RM 9

Price	RM 10

Vehicle Type	MVP
Min Duration	20 minutes
Max Duration	25 minutes
Price	RM 12

Figure 5-2-22-6: Car Wash Center Service Details Page (5)

When cleaner clicked on delete icon at the top right corner of the service category card view, a dialog is pop out to confirm the deletion of service category as shown in figure 5-2-22-6. After cleaner click the “yes” button, the service category will be remove.

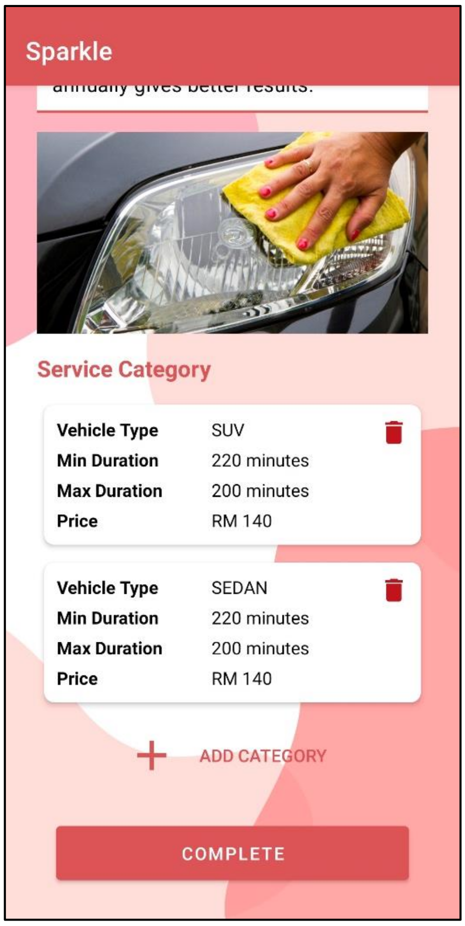


Service Name
Waxing / Polish

Short Description
Brightens dull paintwork

Long Description
A hot wax car wash creates a barrier between elements and your car's paint. It shields the car from minor physical abuse such as scraping by grime and grit during washing. In addition, it makes washing much easier. For dark colored cars, an additional waxing annually gives better results.

Figure 5-2-22-7: Add Car Wash Center Service Page (1)



Vehicle Type	SUV
Min Duration	220 minutes
Max Duration	200 minutes
Price	RM 140

Vehicle Type	SEDAN
Min Duration	220 minutes
Max Duration	200 minutes
Price	RM 140

Figure 5-2-22-8: Add Car Wash Center Service Page (2)

Figure 5-2-22-7 and figure 5-2-22-8 shows the screenshot of add car wash center service page. To add new service, cleaner have to fill in the details such as service name, short description, long description, and add the service category. After cleaner had complete the details, they can click “complete” button to add the service. The system will check the input and update the database.

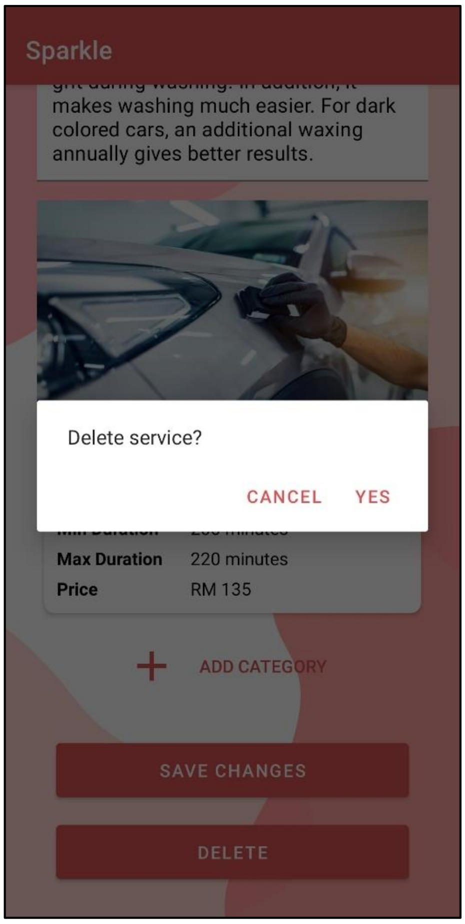

Sparkle
grit during washing. In addition, it makes washing much easier. For dark colored cars, an additional waxing annually gives better results.
Delete service?
CANCEL
YES
Min Duration 200 minutes
Max Duration 220 minutes
Price RM 135
ADD CATEGORY
SAVE CHANGES
DELETE

Figure 5-2-22-9: Delete Car Wash Center Service Alert Dialog

If the cleaner want to delete the service, they can click the “delete” button at the bottom of the car wash center service details page. An alert dialog is pop out to ask for the confirmation to delete the service. To delete the service, cleaner need to click “yes”. The system will delete the service from car wash center in Cloud Firestore.

5.2.23 Manage Car Wash Center Vehicle Type

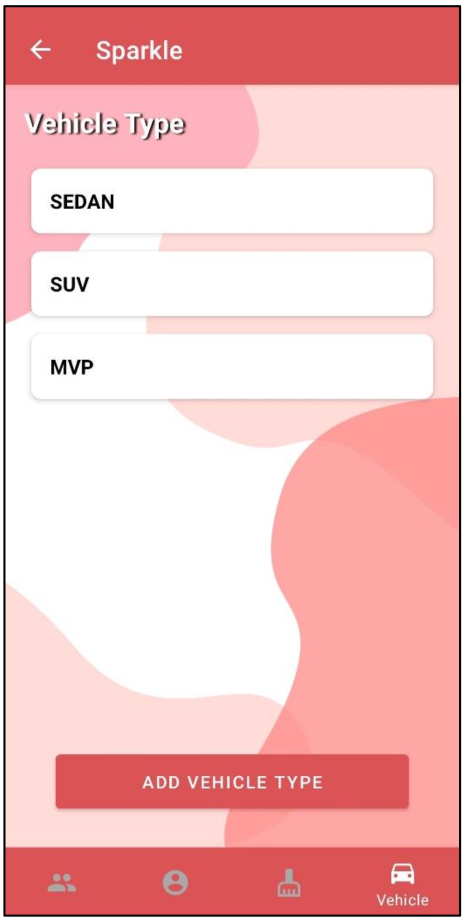


Figure 5-2-23-1: Car Wash Center Vehicle Type List Page

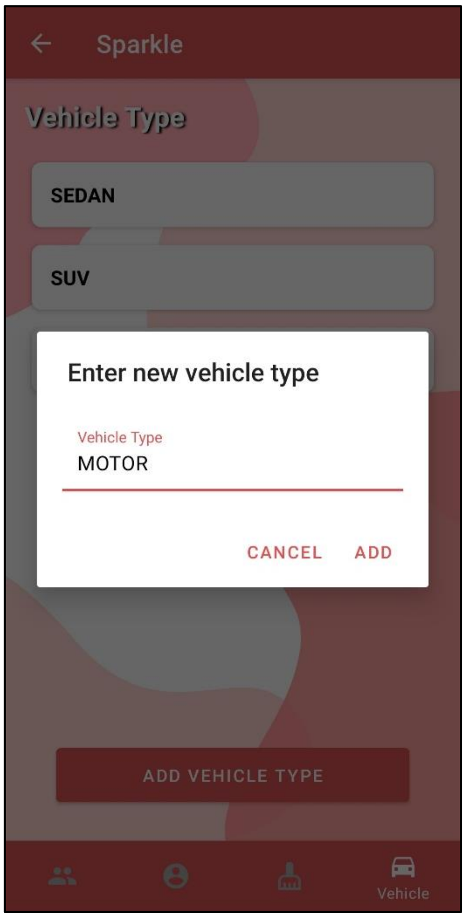
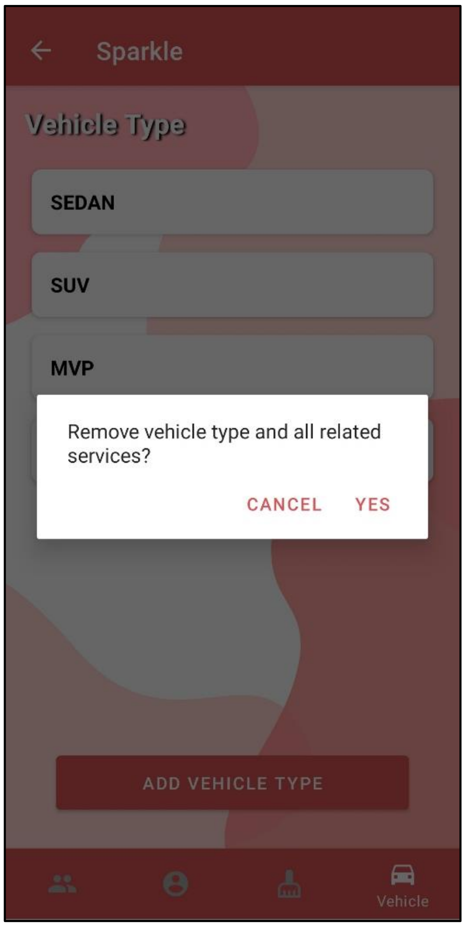


Figure 5-2-23-2: Add Car Wash Center Vehicle Type Page

Figure 5-2-23-1 shows the screenshot of the car wash center vehicle type page. All the vehicle type the car wash center serve is displayed in a list. Each card contains the vehicle type name. A “add vehicle type” button is placed at the bottom of the page. Cleaner can click the button if they want to add new vehicle type to the car wash center. When cleaner clicked on the button, a dialog is pop out and prompt for the new vehicle type name. cleaner enters the new vehicle type and clicked the “add” button. And finally, the vehicle type is added successfully. The system will also update the vehicle type of the car wash center in Cloud Firestore.

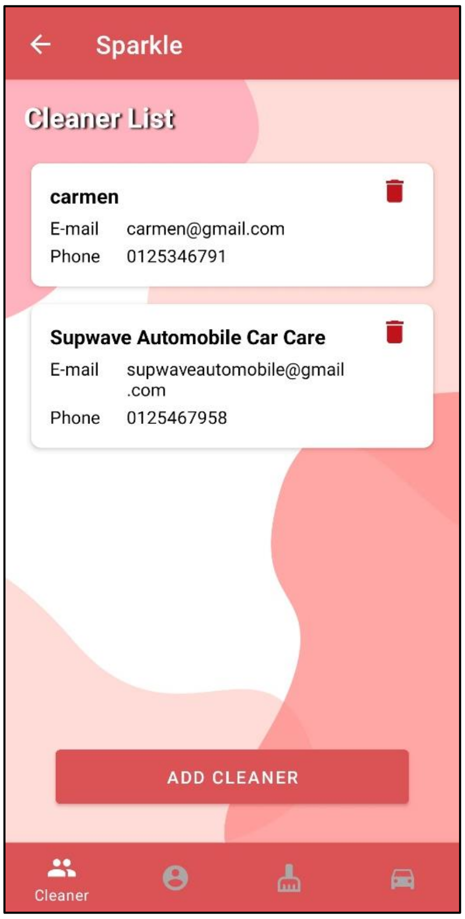


Remove vehicle type and all related services?

Figure 5-2-23-3: Delete Car Wash Center Vehicle Type

When cleaner want to delete the vehicle type, they just need to long click the vehicle type. Then, an alert dialog will pop out and ask for the confirmation to delete the vehicle type. If the cleaner clicked “yes”, the vehicle type and the service provided to the vehicle type will be removed.

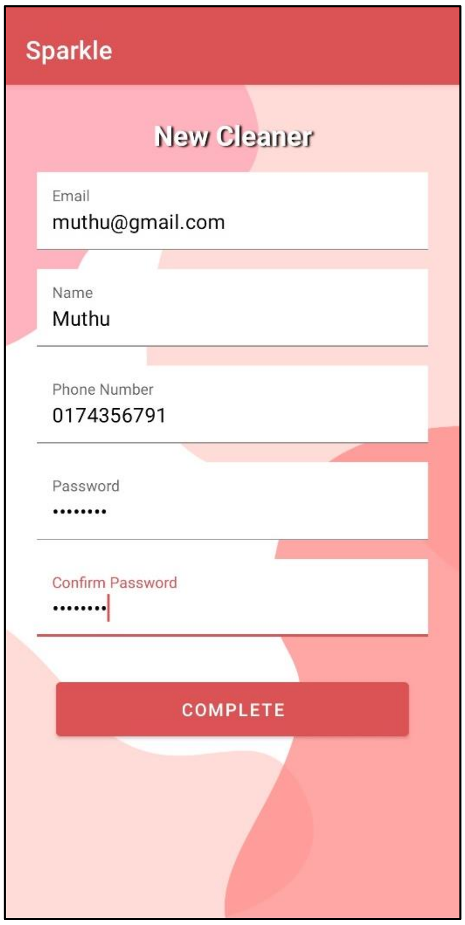
5.2.24 Manage Car Wash Center Cleaner



carmen
E-mail carmen@gmail.com
Phone 0125346791

Supwave Automobile Car Care
E-mail supwaveautomobile@gmail.com
Phone 0125467958

Figure 5-2-24-1: Car Wash Center Cleaner Page



Email: muthu@gmail.com
Name: Muthu
Phone Number: 0174356791
Password: ••••••••
Confirm Password: ••••••••

Figure 5-2-24-2: Add Car Wash Center Cleaner Page

Figure 5-2-24-1 shows the screenshot of the car wash center cleaner page. All the car wash center's cleaner is displayed in a list. Each card contains the cleaner name, email and phone number. A "add cleaner" button is placed at the bottom of the page. Cleaner can click the button if they want to add new cleaner to the car wash center. Then, the system will direct cleaner to the add car wash center cleaner page as shown in figure 5-2-24-2. In add cleaner page, the cleaner need to fill in the details which is the new cleaner's name, email, phone number, password and confirmation password. Email is needed in order to create an account and for log in purpose. The password and confirmation password is needed in order to verify the cleaner during the authenticate process next time when cleaner log in. The password and confirmation password must be the same. After cleaner had completed all the input fields, the cleaner can click the "complete" button in order to add the new cleaner.

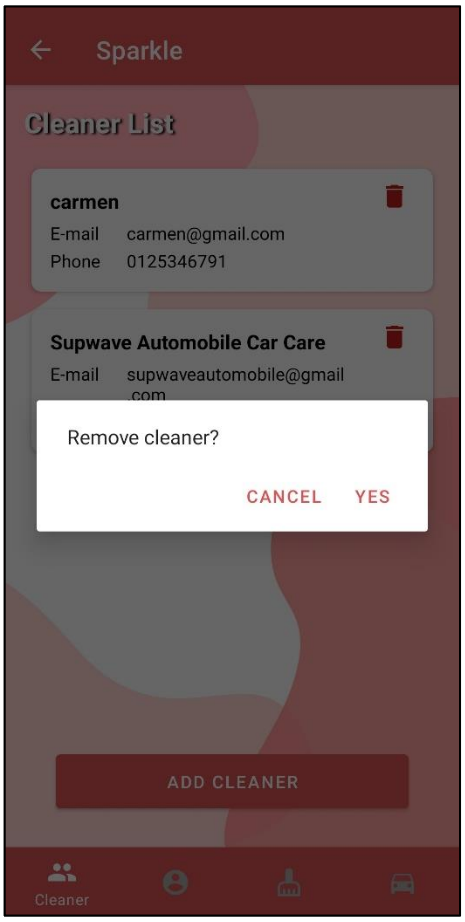
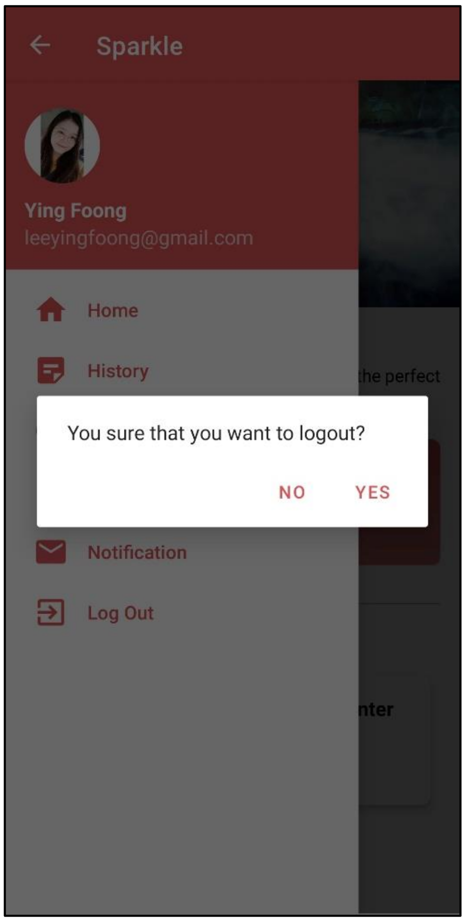


Figure 5-2-24-3: Remove Car Wash Center Cleaner

To remove the existing cleaner, cleaner can click the delete icon on the top right corner of the cleaner’s card view. After clicked the delete icon, an alert dialog is pop out and ask for the confirmation to delete the cleaner. If the cleaner clicked “yes”, the cleaner of the car wash center will be removed.

5.2.25 Log out



You sure that you want to logout?

NO YES

Figure 5-2-25-1: Log Out

To log out the account, user can click on the top left corner at the user's home page. A navigation bar is pop out and user need to click on the "log out" in the menu of the navigation bar. An alert dialog is pop out for confirmation to log out the account. User click "yes" to continue. Lastly, the system will direct user back to the log in page of the application.

5.3 Benchmark Performance between Scheduling Algorithm

In this section, the performance testing is conducted in order to compare 2 scheduling algorithm which is First Come First Serve (FCFS) and Least Laxity First (LLF) scheduling algorithm. Table below shown a task set of total 6 tasks to be scheduled. The attributes included are arrival time, execution time, worst case execution time and deadline of the task.

Task	Arrival Time (minutes)	Execution Time (minutes)	Worst case Execution Time (minutes)	Deadline (minutes)
P1	0	15	20	20
P2	0	20	35	35
P3	30	30	40	70
P4	30	15	20	50
P5	60	30	40	100
P6	60	15	20	80

Table 5-3-1: A task set of 6 tasks

5.3.1 First Come First Serve (FCFS) Scheduling Algorithm

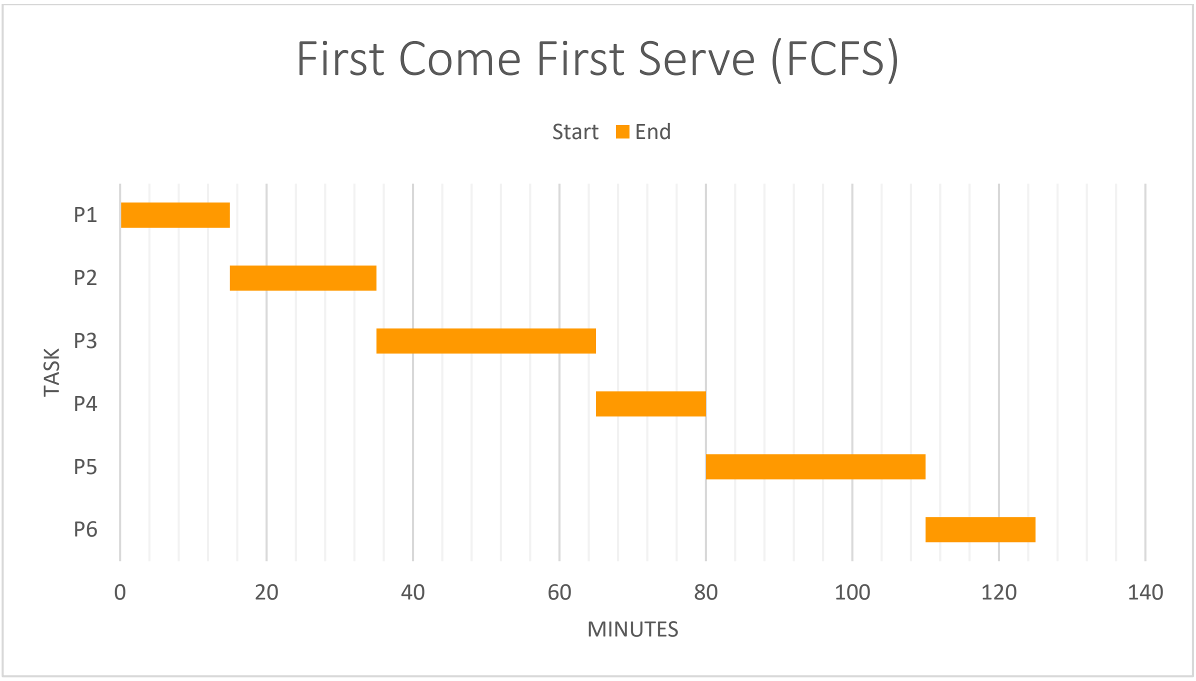


Figure 5-3-1-1: First Come First Serve (FCFS) scheduling algorithm

The order of task to be execute are as below:

P1 > P2 > P3 > P4 > P5 > P6

Given waiting time ≥ 0, the waiting time of the set of task is calculated by using the formula below:

Waiting Time = Start Time – Arrival Time

Hence, the average waiting time for First Come First Serve is calculated as below:

Average waiting time = (0 + (15-0) + (35-30) + (65-30) + (80-60) + (110- 60)) /6

= 125/6 = 20.833 minutes

Given time exceeding deadline ≥ 0, the time exceeding deadline of task is calculated by using the formula below:

Time Exceeding Deadline = End Time – Deadline

Hence, the average time exceeding deadline for First Come First Serve is calculated as below:

Average time exceeding deadline = (0 + 0 + 0 + (80-50) + (110-100) + (125-80)) /6

= 85/6

= 14.167 minutes

5.3.2 Least Laxity First (LLF) Scheduling Algorithm

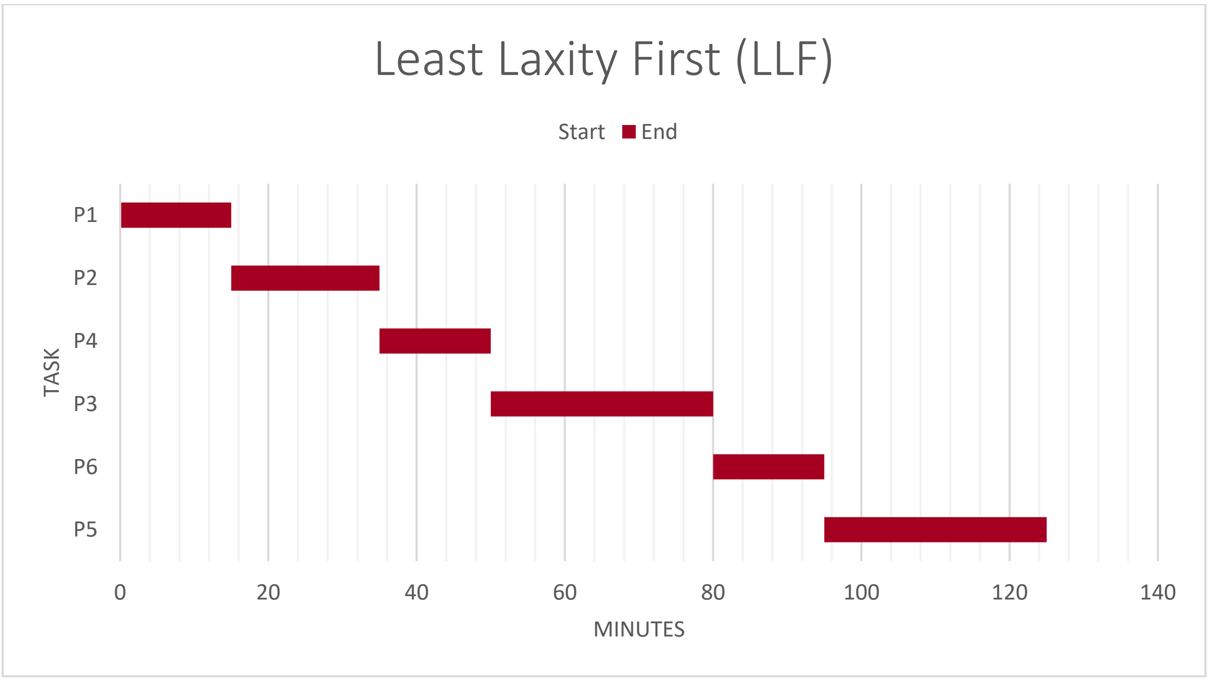


Figure 5-3-2-1: Least Laxity First (LLF) scheduling algorithm

The order of task to be execute are as below:

P1 > P2 > P4 > P3 > P6 > P5

Given waiting time ≥ 0, the waiting time of the set of task is calculated by using the formula below:

Waiting Time = Start Time – Arrival Time

Hence, the average waiting time for Least Laxity First scheduling algorithm is calculated as below:

Average waiting time = (0 + (15-0) + (35-30) + (50-30) + (80-60) + (95-60)) /6

= 95/6 = 15.833 minutes

Given time exceeding deadline ≥ 0, the time exceeding deadline of task is calculated by using the formula below:

Time Exceeding Deadline = End Time – Deadline

Hence, the average time exceeding deadline for Least Laxity First scheduling algorithm is calculated as below:

Average time exceeding deadline = (0 + 0 + 0 + (80-70) + (95-80) + (125-100)) /6

= 50/6

= 8.333 minutes

5.3.3 Discussion

The average waiting time for First Come First Serve (FCFS) Scheduling Algorithm is 20.833 minutes while the waiting time for Least Laxity First (LLF) scheduling algorithm is 15.833 minutes. Hence, the average waiting time for LLF scheduling algorithm is shorter than FCFS scheduling algorithm by 5 minutes.

Besides, the average time exceeding deadline for First Come First Serve (FCFS) Scheduling Algorithm is 14.167 minutes while the average time exceeding deadline for Least Laxity First (LLF) scheduling algorithm is 8.333 minutes. Hence, the average time exceeding deadline for LLF scheduling algorithm is shorter than FCFS scheduling algorithm by 5.834 minutes.

In conclusion, Least Laxity First (LLF) scheduling algorithm is chosen to implement in this project in order to increase the performance of the application.

5.4 User Testing and Feedback

When the application is completed, Android Application Package (APK) file of this application is generated. The APK file is then distributed to 10 friends as respondent for testing. All of the respondents have the same characteristic which is they do go for car wash service at least once per month. All the respondents need to download the APK and install the application in their device. A google feedback form is created and sent to them after they finished testing and explore the application. Google feedback form is used to collect respondent's feedback regarding to the application tested. In this section, the feedback of the respondents is collected and analyzed in order to have best possible solution to improve the performance of application.

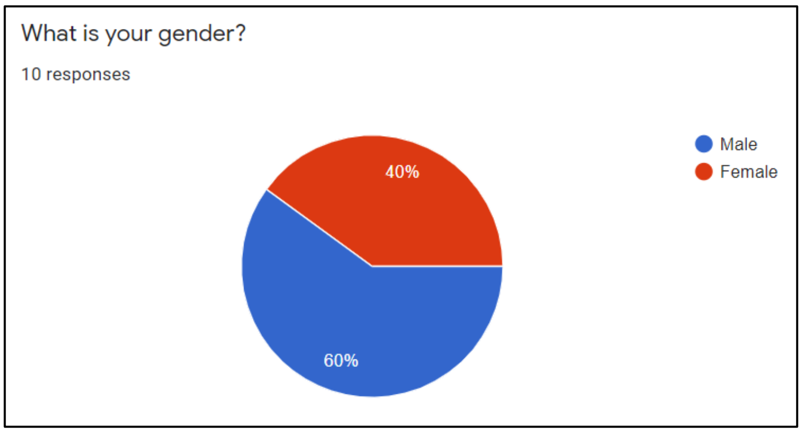


Figure 5-4-1: Respondent Gender

The first question asked is the gender of the respondents as shown in figure 5-4-1. Among 10 respondents, 60% of them are male and 40% of them are female. As shown in figure 5-4-1, the number of male respondent is higher than female respondents. Different gender may have different opinion towards the application due to the differ in usage habits.

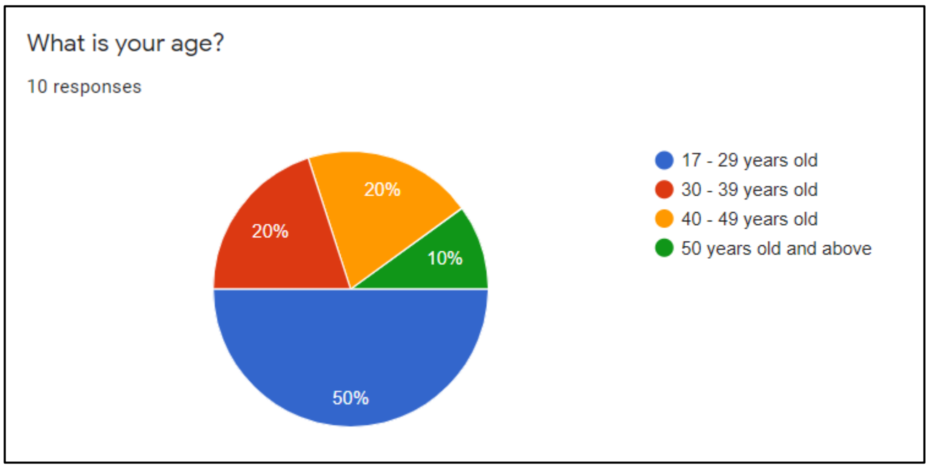


Figure 5-4-2: Respondent Age

The second question asked about the respondent's age. There are 4 age range for respondents to select such as 17 to 29 years old, 30 to 39 years old, 40 to 49 years old and 50 years old and above. Users in different age have different standards and demand towards an application. Hence, it is important to get responses from different age range. In figure 5-4-2 shows the respondents' response. 50% of them are 17 to 29 years old, 20% of them are 30 to 39 years old, 20% of them are 40 to 49 years old and 10% of them are 50 years old and above.

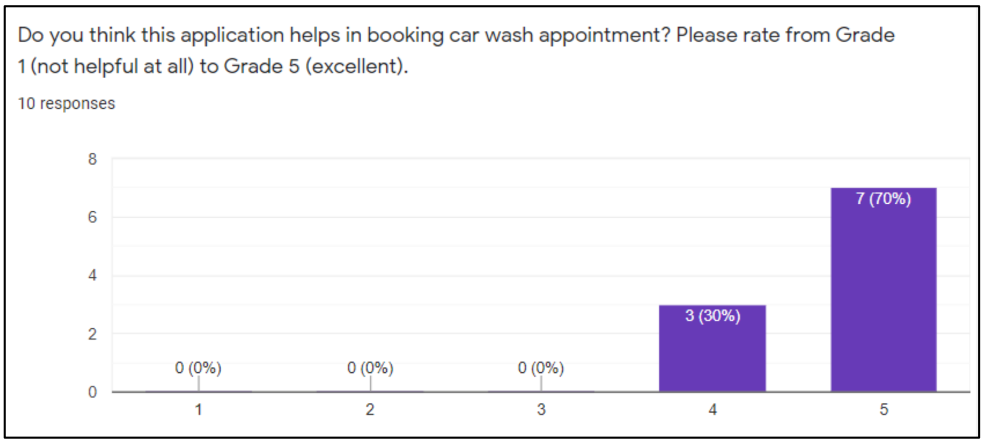


Figure 5-4-3: Rating of the Application Helps in Booking Car Wash Appointment

For the third question of the feedback form, it is to ask the respondents to rate whether the application helps in booking car wash center appointment. Respondents need to rate from grade 1 (not helpful at all) to grade 5 (excellent). If respondent felt that this application saves time and helps them a lot in making a car wash appointment, they can rate for grade 5. Else, if the user thinks that this application cannot help them in booking an appointment, they can rate for grade 1. Results had shown in figure 5-4-3 that all of the respondent rate for grade 4 and above. Majority of the respondent which is 70% of the respondent rate for grade 5 while 30% of the respondent rate for grade 4. In short, majority of the respondent agree that this application helps in booking car wash appointment.

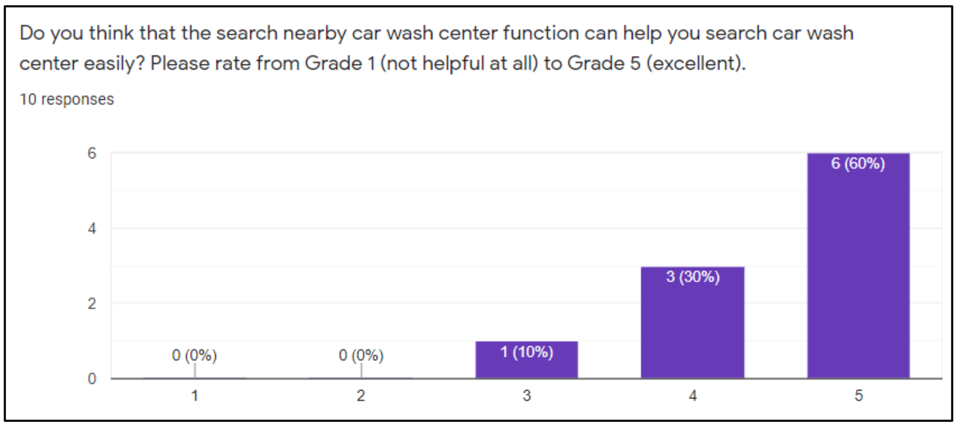


Figure 5-4-4: Rating of Search Nearby Car Wash Center Function

For the fourth question of the feedback form, it is to ask the respondents to rate whether the view nearby car wash center function in the application helps them to make appointment more easily. Respondents need to rate from grade 1 (not helpful at all) to grade 5 (excellent). If respondent felt that this function is useful, they can rate for grade 5 which is excellent. If they felt that this function does not help much, they can rate for grade 1 which is not helpful at all. The response from respondent is shown in figure 5-4-4. Majority of the respondent felt that the function helps them while searching car wash center for booking car wash appointment. 60% of them rate for grade 5, 30% of respondent rate for grade 4 and 10% of respondent rate for grade 3. Overall, the respondents felt that view car wash center nearby function are helpful and more easy for them to find car wash center.

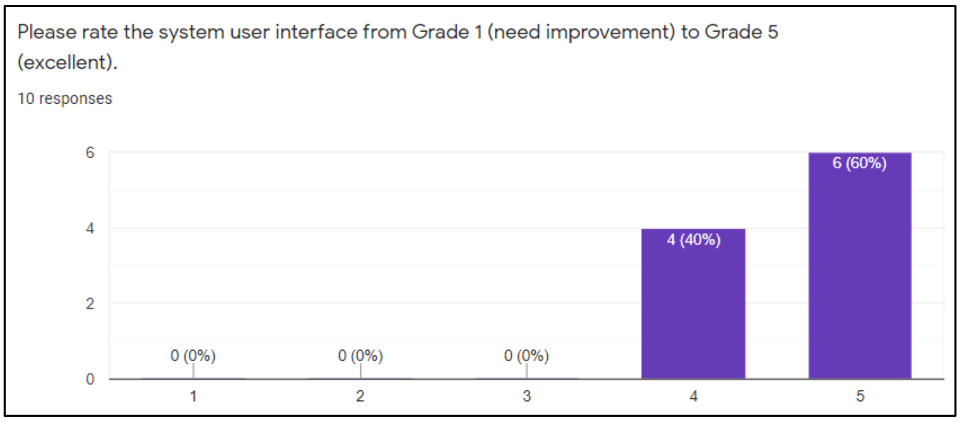


Figure 5-4-5: Rating of System User Interface

The fifth question is asking the respondent to rate the system user interface from grade 1 (need improvement) to grade 5 (excellent). A good system user interface is important for an application because it can gain more potential customers and maximize the functionality, efficiency and responsiveness of the application. As shown in figure 5-4-5, all of the respondent rate for grade 4 and grade 5 which the grade is above average. 60% of the respondent rate for grade 5 while 40% of the respondent rate for grade 4. In short, all the respondents are satisfied with the system user interface.

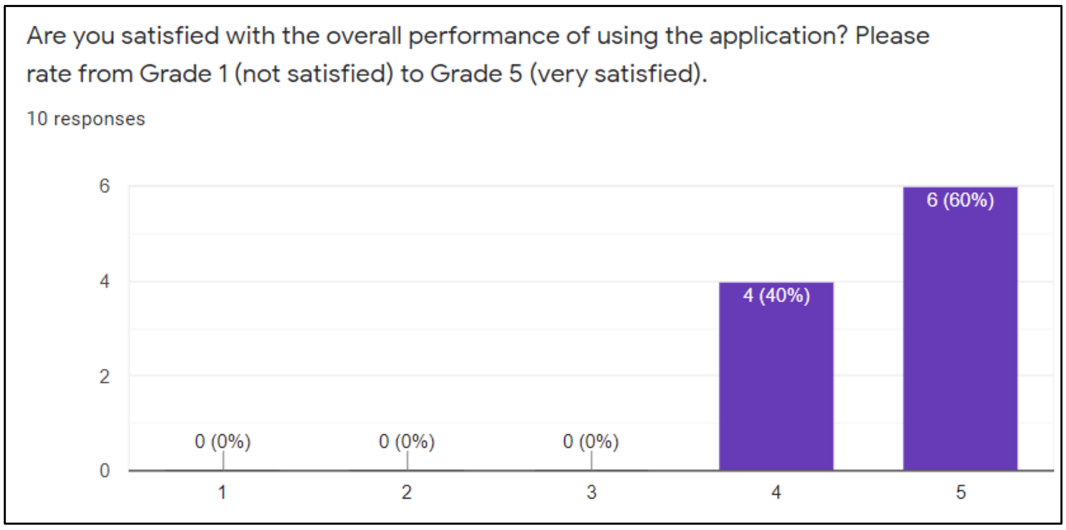
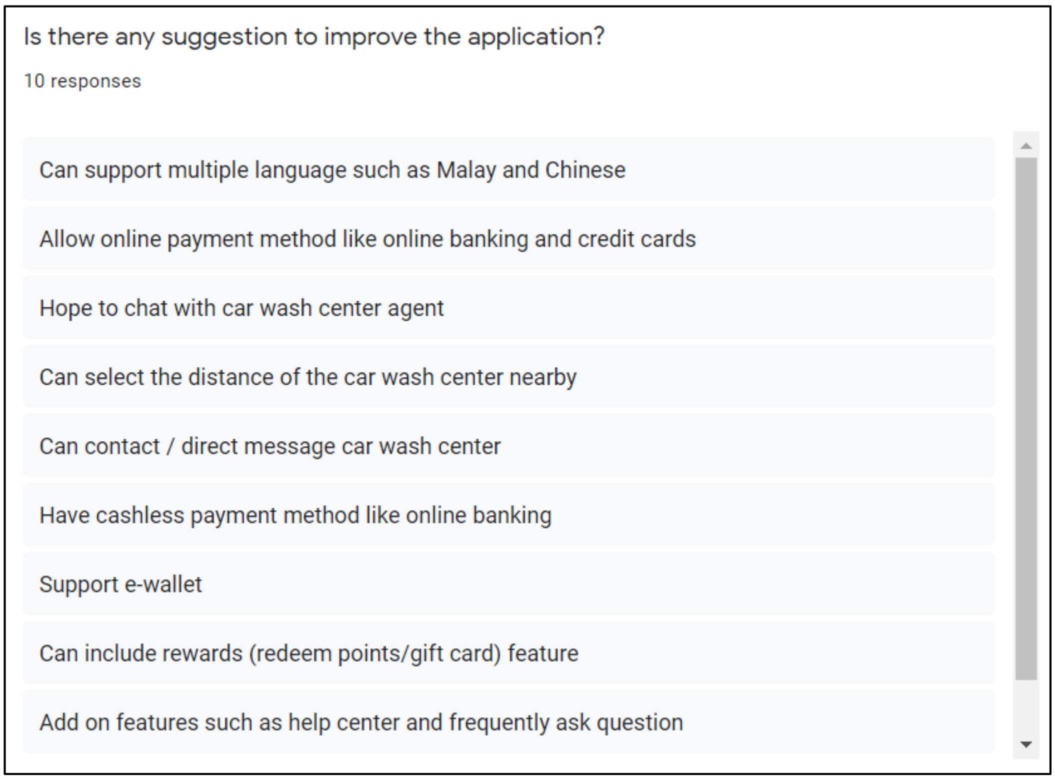


Figure 5-4-6: Rating of System Overall Performance

The sixth question of the feedback form is asking whether the respondents are satisfied with the overall performance of the system. The question requires respondent to rate the system's overall performance from grade 1 (not satisfied) to grade 5 (very satisfied). If the respondent is not satisfied with the overall performance of the system, they can rate for grade 1. If the respondent is very satisfied with the system's overall performance, they can rate for grade 5. In figure 5-4-6 had shown that all the respondent had rate for grade 4 and grade 5 which the overall responses are above average. 60% of the respondents rate for grade 5 and 40% of the respondents rate for grade 4. Overall, the respondents are still satisfied with the system performance.


Is there any suggestion to improve the application?

10 responses

Can support multiple language such as Malay and Chinese

Allow online payment method like online banking and credit cards

Hope to chat with car wash center agent

Can select the distance of the car wash center nearby

Can contact / direct message car wash center

Have cashless payment method like online banking

Support e-wallet

Can include rewards (redeem points/gift card) feature

Add on features such as help center and frequently ask question

Figure 5-4-7: Respondent Suggestion to Improve Application

The last question of the feedback form is an open-ended question. The respondents are being asked to give suggestion in order to improve the application. To improve the application, user's feedback and suggestion is important. The developer will get to know what feature is lacking or need improvement according to the user's feedback. Figure 5-4-7 shows the suggestions and feedbacks from the respondents. The suggestion had separate into a few parts.

Firstly, respondent suggested that the system can support online payment method. Respondent review that it is more convenient if customers can make payment online as cashless society is growing now. With online payment such as credit cards, debit cards, online banking and e-wallet, the speed of transaction is fast and it can be automatic. Next, the respondent also suggested that they hope to chat with the car wash center agent and help center of this application. There might be emergency or special incident that need customers to contact with the car wash center. Besides, when customer faced problems while using the application, they can contact the help center of the application for help. Hence, respondent wish to have chat feature in this application. It is convenient for customer to get in touch with the car wash center and help center. In addition, respondent also proposed to include the reward feature in the application. Respondent wish to have free coupons, gift cards or points given when they finish an appointment using this application. Reward feature not only can prompt customer to make more appointment using this application, it also can attract new customers to use this application. And lastly, the respondent proposed that the application can support different language such as Malay and Chinese. By adding this support, customers may feel more comfortable with their native language.

5.5 Summary

In conclusion, all the function and feature proposed in this project is built successfully. The result of development via the application screenshot is inserted and the functionality carried out is explained in this chapter. Besides, the benchmark performance of different algorithm is carried out. It had proven that Least laxity time scheduling algorithm is more suitable to be applied in this project. In addition, user feedback from 10 respondents is gathered and analyzed. In conclusion, the user feedback is overall positive. There are also many feasible suggestions from users and can be accepted to enhance the application in the future works.

CHAPTER 6 CONCLUSION

6.1 Project Review, Discussion and Conclusion

Car wash service market now are growing annually and the customers' demand are increasing every year. Nowadays, car wash service can be found anywhere. For example, standalone car wash, petrol station and elsewhere. Society now are living in a fast pace of live and many of them even does not have time to rest. Car wash center provides car wash services to people who wants to save time and save energy. Car wash center has make may people life more convenient indeed.

Since the number of consumers in car wash service market increases year by year, there are some problems faced by customers and car wash center agency. Customers has to wait for a long time as there is long queue in the car wash center. Most of the customers are busy and they do not have many free time to wait for the queue. For the worst case, customers may need to wait for at least a few hours until their turn to get a car wash service. Next, some customers need to have car wash service even they are not at their hometown. For example, they might be outstation, went to travel or went to other places which customers are not familiar with. In addition, the poor attendance of car wash appointment is also one of the problems faced by the car wash center. In the old days, people make appointment by calling to the car wash center or some of them need to go for miles to make an appointment. Thus, it is not convenient to customers as it is so troublesome.

All the problem mentioned above had motivate the development of this project. At the end of this project, an application named "Sparkle" has been delivered successfully. Throw-away prototype methodology is used to develop the mobile application in this project. The objective of this project is to integrate appointment scheduling system in the car wash service mobile application in order to increase the attendance of customers. Customers just have to download the application and they can make appointment using the application. It is much more convenient compared to making an appointment using calls. Moreover, a scheduling algorithm which is Least Laxity First (LLF) is used for scheduling appointments on real-time basis. Customers are able to manage their appointments as they preferred. Hence, it can enhance the user experience while using this application. Last but not least, the feature GPS location tracking is implemented. Customers not only able to view car wash center nearby, they can also view the slot available on real time basis. As a car wash service mobile application, features such as authentication, profile management, notification, history management and vehicle management is implemented to enhance the usability of this application.

For car wash center's cleaner, they are able to view the scheduled appointment and also manage the appointment in real time. When cleaner starts the car wash service, they can update the status of the appointment hence the customers are able to know their appointment status. Besides, features such as car wash center profile management, service management, cleaner management, cleaner profile management is also implemented in this. It is not only easy to use but also convenient and user friendly.

6.2 Novelties and Contributions

In this project, appointment scheduling system will be included in application in order to let the customers to book appointment for car wash service. By using appointment scheduling system, customers are able make appointment by just fill in the appointment's details in mobile application and submit it. They do not have to get the car wash center's phone number to call and discuss with the staff for the slot for appointment or drives for extra miles to car wash center to make appointment. It can also increase the attendance of customers to car wash company to get a car wash service as it is convenient to users. Customers still can make appointment when they are at anytime and anywhere.

Moreover, GPS location tracking for car wash center also allows customers to search for car wash center near by their location. Thus, this feature benefits users that wish to have car wash service at car wash center which is a short distance away. When customers went to another unfamiliar place, they can also search for car wash center easily which is just one click away. Customers can also monitor the car wash center's real-time status. The status of the car wash center is displayed to customers for example it will show whether the current time slot of car wash center is fully booked or booking is still available for that time slot.

In addition, Least Laxity First (LLF) scheduling algorithm is applied in scheduling appointments in real-time basis. Real-time scheduling allows customers to make new or cancel an appointment through this application. Besides, the system will update the time slot available on real-time basis. Users are not able to make appointment at the time slot which is fully booked. Thus, it will prevent overbooked of the time slot. Least Laxity First algorithm is used to schedule the sequence of the appointments to be execute. This scheduling algorithm will schedule the appointments according to the laxity of appointments which least laxity time will be execute first. Thus, schedule appointment using scheduling algorithm is more effective as it can utilize the cleaner to carry out the cleaning process and minimize the waiting time of users. Less waiting time not only increase the customer satisfaction to the car wash center, it will also earn car wash center good reputation. Hence, it not only increases the appointment and customer attendance, it also increases the profit of the car wash center.

In short, this application not only saves a lot of time and energy, it also efficient to all the users including customers and car wash centers due to the useful features implemented in this application.

6.3 Future Work

There is still some enhancement that Sparkle application could made. There is some improvement suggested by the users and can be done in future works. Firstly, the current application can only accept offline payment which is using cash to pay after the appointment. It is more convenient if customers can make payment online as cashless society is growing now. With online payment such as credit cards, debit cards, online banking and e-wallet, the speed of transaction is fast and it can be automatic. Hence, in future works, online payment feature can be added into the application.

Next, enhancement that can be done in this application is to chat with the car wash center agent and help center of this application. There might be emergency or special incident that need customers to contact with the car wash center. Besides, when customer faced problems while using the application, they can contact the help center of the application for help. Hence, chat feature can be insert in this application. It is convenient for customer to get in touch with the car wash center and help center.

In addition, the reward feature also may do in future work in the application. In user feedback form, some respondent wish to have free coupons, gift cards or points given when they finish an appointment using this application. Reward feature not only can prompt customer to make more appointment using this application, it also can attract new customers to use this application.

BIBLIOGRAPHY

Alina, 2019. The 37 Advantages and Disadvantages of Live Chat. Userlike Live Chat. Available at: https://www.userlike.com/en/blog/live-chat-advantages-disadvantages [Accessed August 10, 2020].

Anastasia Masters (2019) “6 ways to increase your appointment attendance rates,” JRNI, 10 December. Available at: https://www.jrni.com/blog/increase-appointment-attendance-rates (Accessed: August 27, 2020).

Anastasia Z., 2018. Why Are Push Notifications So Important? RubyGarage. Available at: https://rubygarage.org/blog/benefits-of-push-notifications [Accessed August 10, 2020].

Anon, 2017. Advantages and disadvantages of online payments. Small Business First. Available at: https://smallbusinessfirst.kochiesbusinessbuilders.com.au/advantages-and-disadvantages-of-online-payments/ [Accessed August 10, 2020].

Anon, Android Studio features : Android Developers. Android Developers. Available at: https://developer.android.com/studio/features [Accessed August 23, 2020].

Anon, Mobile Car Wash and Detailing. Washos. Available at: https://www.washos.com/mobile-app [Accessed August 23, 2020].

Baxter, 2020. My Honest Review of Get Spiffy Mobile Car Wash & Detailing. Carwash Country. Available at: https://www.carwashcountry.com/get-spiffy-mobile-car-wash-detailing-review/ [Accessed August 23, 2020].

Gaille, B. (2017) “21 Car Wash Industry Statistics and Trends,” *BrandonGaille.com*, 23 May. Available at: https://brandongaille.com/19-car-wash-industry-statistics-and-trends/ (Accessed: August 28, 2020).

Haq, S., 2016. 5 Benefits of GPS Tracking - Copy. Teletrac Navman. Available at: https://www.teletracnavman.com/resources/blog/5-benefits-of-gps-tracking-copy [Accessed August 10, 2020].

Harvsey, L., 2019. Top 10 Benefits of an Appointment Scheduler. eCommerce Blog. Available at: https://blog.3dcart.com/appointment-scheduler-benefits [Accessed August 10, 2020].

McCabe, E., 2019. Top 3 Benefits of an Automated Emergency Notification System. Marketing Automation Software - SMS, Fax, Voice, Email Marketing. Available at: https://www.simplycast.com/blog/top-3-benefits-of-an-automated-emergency-notification-system/ [Accessed August 10, 2020].

Staff, A.F., 2019. 5 Ways GPS Tracking Can Benefit Your Business. Telematics - Automotive Fleet. Available at: https://www.automotive-fleet.com/334347/5-ways-gps-tracking-can-benefit-your-business [Accessed August 10, 2020].

Stevenson, D., 2018. What is Firebase? The complete story, abridged. Medium. Available at: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0 [Accessed August 23, 2020].

User, G., 2016. The Biggest Benefits of Real Time Scheduling. TimeTap Online Scheduling. Available at: https://blog.timetap.com/blog/2016/2/16/the-biggest-benefits-of-real-time-scheduling [Accessed August 10, 2020].

Zhang, W., Teng, S., Zhu, Z., Fu, X., & Zhu, H. (2007). An Improved Least-Laxity-First Scheduling Algorithm of Variable Time Slice for Periodic Tasks. *6th IEEE International Conference on Cognitive Informatics*. doi:10.1109/coginf.2007.4341935

WEEKLY LOG

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	**Study week no.: 1, 2**
Student Name & ID: LEE YING FOONG 17ACB02770	
Supervisor: MR. YECK YIN PING	
Project Title: Appointment Scheduling System For Car Wash Service	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

1. **Implementation of Google Map**
   - 1.1 Users are able to locate their current location
     Application will ask users' permission to access to the device's location.
   - 1.2 Users are able to view the car wash center nearby their current location (within 5km) in Google Map

     The location of car wash center that is nearby users' location is retrieved from database and marker(s) is added to the Google Map.
     The data of the car wash center is retrieved from database and the overview
     of car wash center is shown after users clicked the marker.
   - 1.3 Users are able to view the car wash center nearby their current location (within 5km) in a list
     The overview of all car wash center nearby is displayed in a list.
   - 1.4 Users are able to search their preferable car wash center
     Users can search car wash center by entering the name of car wash center.
2. **Homepage**
   - 2.1 Front-end is designed and coded.
3. **Make appointment module**
   - 3.1 Front-end of user making appointment is developing.

2. WORK TO BE DONE **1. Make appointment module** 1.1 Front-end of user making appointment. 1.2 Back-end of user making an appointment. 1.3 Update database when user successfully made appointment.
3. PROBLEMS ENCOUNTERED Some bugs and errors are found when implementing Google Map to the application. I also found that it is challenging when managing data in database.
4. SELF EVALUATION OF THE PROGRESS The goal proposed for this 2 week is achieved. Although there are problems encountered during my developing phase, I am able to find out the solution and solve the problems successfully.

[illegible] ying

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	**Study week no.: 3,4**
Student Name & ID: LEE YING FOONG 17ACB02770	
Supervisor: MR. YECK YIN PING	
Project Title: Appointment Scheduling System For Car Wash Service	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

1. **Make Appointment Module**

   1.1 Users are able to select a car wash center

   1.2 Users are able to select the vehicle type and fill in the vehicle details

   1.3 Users are able to select their preferable time slot

   The availability of each time slot is checked and users are not able to select

   the time slot that is fully booked or unavailable.

   1.4 Users are able to select their preferable car wash service

   All services that provided by the car wash center will be retrieved from database. Then, users are able to view and select the service.

   1.5 Users are also able to view the details of the service selected

   The details of the service for example the price and description is retrieved from database and display for users.

   1.6 Users are able to have an overview of the appointment

   Users are able to view all the details for example car wash center, time slot, price and service selected. When users confirm the appointment, the database is updated.

2. **Home Page**

   2.1 Active appointment is listed in the homepage

2. WORK TO BE DONE

1. **Auto-generated notification**

1.1 Notification is sent automatically once they booked appointment successfully

2. **Implement scheduling algorithm to schedule the appointment**

3. PROBLEMS ENCOUNTERED

It is very challenging while designing interface and coding the layout. It also took me some time to check the availability of the time slot.

4. SELF EVALUATION OF THE PROGRESS

The goal proposed for this 2 week is achieved. Although there are problems encountered during my developing phase, I am able to find out the solution and solve the problems successfully.

[illegible]
Supervisor's signature

ying
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	**Study week no.: 5 , 6**
Student Name & ID: LEE YING FOONG 17ACB02770	
Supervisor: MR. YECK YIN PING	
Project Title: Appointment Scheduling System For Car Wash Service	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

1. **Notification function**
   1.1 Users will receive notification when they booked the appointment successfully.
   1.2 Users will receive notification when they cancel the appointment successfully.

2. **Implement scheduling algorithm to schedule the appointment**
   2.1 All the appointment on current day will be retrieved from database and scheduled using Least Laxity First algorithm.
   2.2 The scheduled appointment is then assigned to each of the washing slot accordingly.

2. WORK TO BE DONE

1. Car wash center can customize the vehicle type
2. Car wash center can customize the opening and closing hour
3. Car wash center can manage their own service and cleaner

3. PROBLEMS ENCOUNTERED

Faced some problems when implementing notification function. However, I am able to find solutions and now I had successfully solved the problems.

4. SELF EVALUATION OF THE PROGRESS

The goal proposed for this 2 week is achieved. Schedule for this 2 weeks is tight but I had learnt how to distribute time and assigned tasks properly in this 2 weeks.

[illegible]	ying
Supervisor’s signature	Student’s signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	**Study week no.: 7, 8**
Student Name & ID: LEE YING FOONG 17ACB02770	
Supervisor: MR. YECK YIN PING	
Project Title: Appointment Scheduling System For Car Wash Service	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

1. **Car Wash Center**
   - 1.1 Car Wash Center can manage their own profile. For example, they can customize their own business hour, address, phone number and etc.
   - 1.2 Car Wash Center can manage their cleaner. They can add a new cleaner or delete a cleaner.
   - 1.3 Car Wash Center can manage their service. They can view the details of the service, add new service, update service, and delete service. All the changes will be updated in the database.
   - 1.4 Car Wash Center can manage the vehicle type.
2. **Admin**
   - 2.1 Home page.
   - 2.2 Admin can add new car wash center.

2. WORK TO BE DONE

1. Adjust some of the user interface for better user experience.
2. Car wash center can customize their own car wash center's picture and service's picture.

3. PROBLEMS ENCOUNTERED

No problem encountered.

4. SELF EVALUATION OF THE PROGRESS

The goal proposed in this 2 weeks is achieved.

[illegible]

Supervisor's signature

ying

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	**Study week no.: 9, 10**
Student Name & ID: LEE YING FOONG 17ACB02770	
Supervisor: MR. YECK YIN PING	
Project Title: Appointment Scheduling System For Car Wash Service	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

1. Upload image
   1.1 Car wash center are able to add their car wash center image. The image path stored in storage is also stored in database. Car wash center can also change their car wash center image.
   1.2 Car wash center are able to customize the image of their service.
2. User interface adjustment
   Alignment, font size, button size is adjusted for better user experience
3. Performance of algorithm is tested

2. WORK TO BE DONE

1. Documentation and writing report

3. PROBLEMS ENCOUNTERED

No problem encountered.

4. SELF EVALUATION OF THE PROGRESS

The goal proposed in this 2 weeks is achieved.

[illegible]

Supervisor's signature

ying

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	**Study week no.: 11,12**
Student Name & ID: LEE YING FOONG 17ACB02770	
Supervisor: MR. YECK YIN PING	
Project Title: Appointment Scheduling System For Car Wash Service	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

1. Finished writing chapter 1 of FYP 2 report
2. Finished writing chapter 2 of FYP 2 report
3. Finished writing chapter 3 of FYP 2 report

   3.1 Use case diagram

   3.2 Use case description

   3.3 Activity diagram

   3.4 Coding explanation
4. Finished writing chapter 4 of FYP 2 report

2. WORK TO BE DONE

1. Complete Chapter 5 and Chapter 6 of FYP 2 report
2. Design poster
3. Combine report and check report format

3. PROBLEMS ENCOUNTERED

No problem encountered.

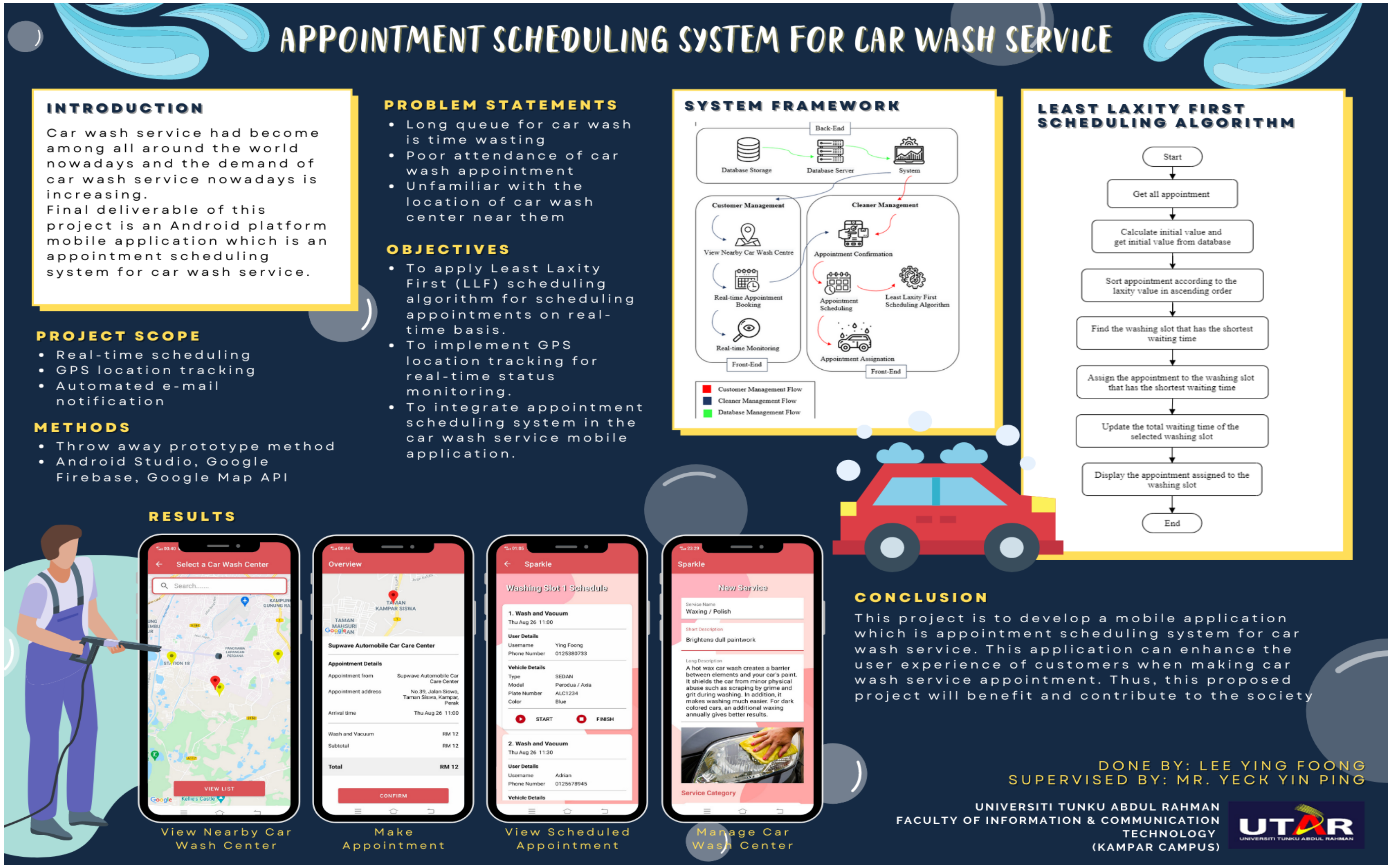
4. SELF EVALUATION OF THE PROGRESS

The goal proposed in this 2 weeks is achieved.

[illegible] ying

Supervisor's signature Student's signature

POSTER



APPOINTMENT SCHEDULING SYSTEM FOR CAR WASH SERVICE

INTRODUCTION

Car wash service had become among all around the world nowadays and the demand of car wash service nowadays is increasing.
Final deliverable of this project is an Android platform mobile application which is an appointment scheduling system for car wash service.

PROBLEM STATEMENTS

- Long queue for car wash is time wasting
- Poor attendance of car wash appointment
- Unfamiliar with the location of car wash center near them

OBJECTIVES

- To apply Least Laxity First (LLF) scheduling algorithm for scheduling appointments on real-time basis.
- To implement GPS location tracking for real-time status monitoring.
- To integrate appointment scheduling system in the car wash service mobile application.

PROJECT SCOPE

- Real-time scheduling
- GPS location tracking
- Automated e-mail notification

METHODS

- Throw away prototype method
- Android Studio, Google Firebase, Google Map API

SYSTEM FRAMEWORK

LEAST LAXITY FIRST SCHEDULING ALGORITHM

Start
Get all appointment
Calculate initial value and get initial value from database
Sort appointment according to the laxity value in ascending order
Find the washing slot that has the shortest waiting time
Assign the appointment to the washing slot that has the shortest waiting time
Update the total waiting time of the selected washing slot
Display the appointment assigned to the washing slot
End

RESULTS

View Nearby Car Wash Center

Make Appointment

View Scheduled Appointment

Manage Car Wash Center

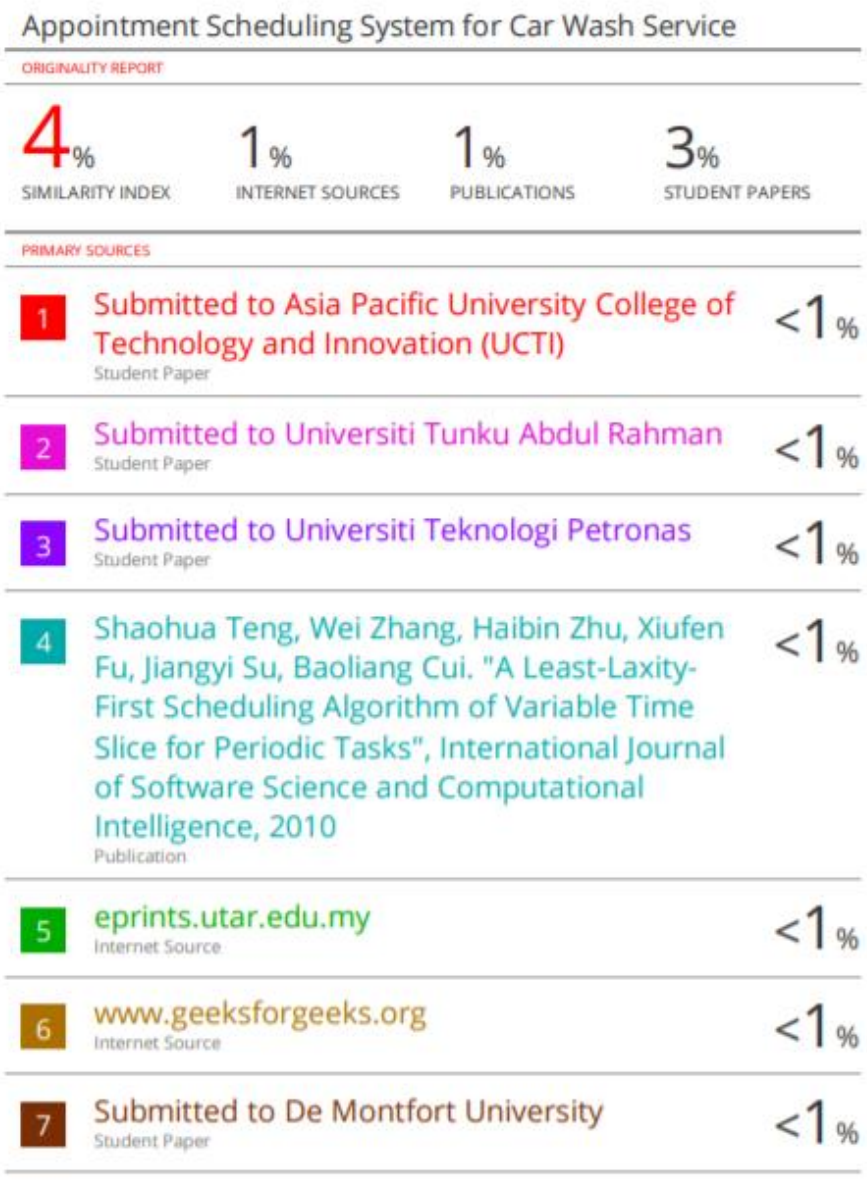
CONCLUSION

This project is to develop a mobile application which is appointment scheduling system for car wash service. This application can enhance the user experience of customers when making car wash service appointment. Thus, this proposed project will benefit and contribute to the society

DONE BY: LEE YING FOONG
SUPERVISED BY: MR. YECK YIN PING

UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

PLAGLARISM CHECK RESULT



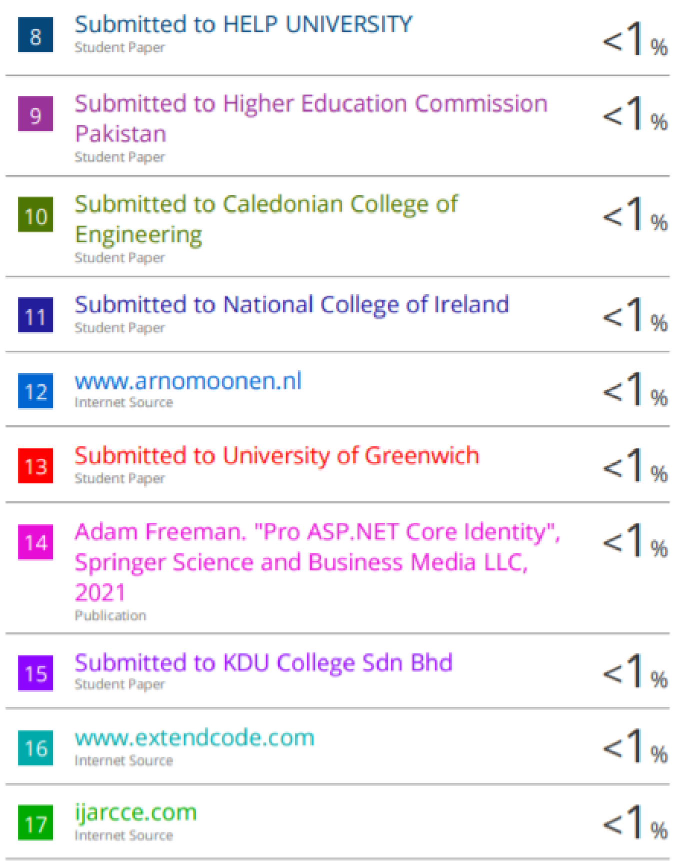
Appointment Scheduling System for Car Wash Service

ORIGINALITY REPORT

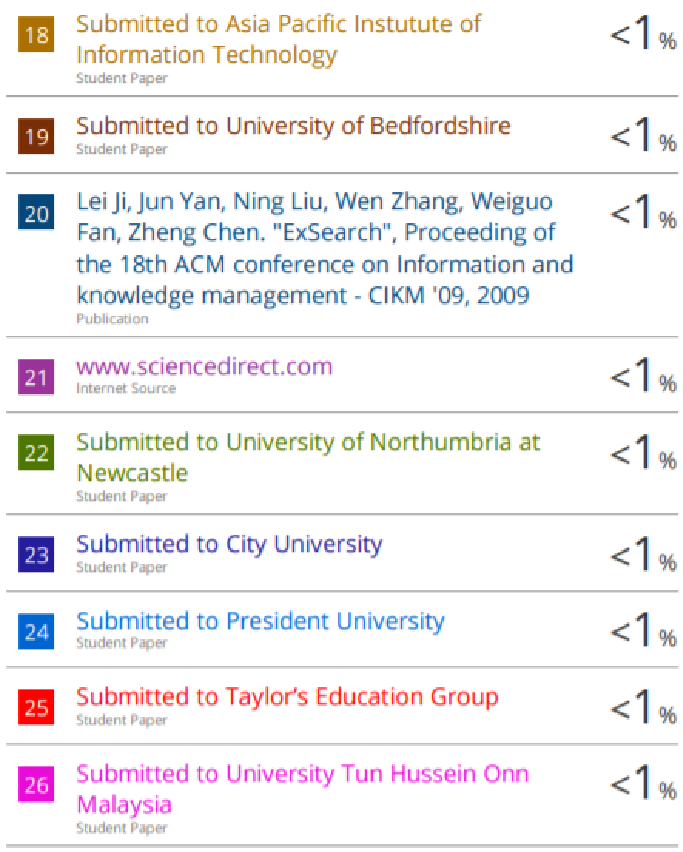
4%	1%	1%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
2	Submitted to Universiti Tunku Abdul Rahman Student Paper	<1%
3	Submitted to Universiti Teknologi Petronas Student Paper	<1%
4	Shaohua Teng, Wei Zhang, Haibin Zhu, Xiufen Fu, Jiangyi Su, Baoliang Cui. "A Least-Laxity-First Scheduling Algorithm of Variable Time Slice for Periodic Tasks", International Journal of Software Science and Computational Intelligence, 2010 Publication	<1%
5	eprints.utar.edu.my Internet Source	<1%
6	www.geeksforgeeks.org Internet Source	<1%
7	Submitted to De Montfort University Student Paper	<1%

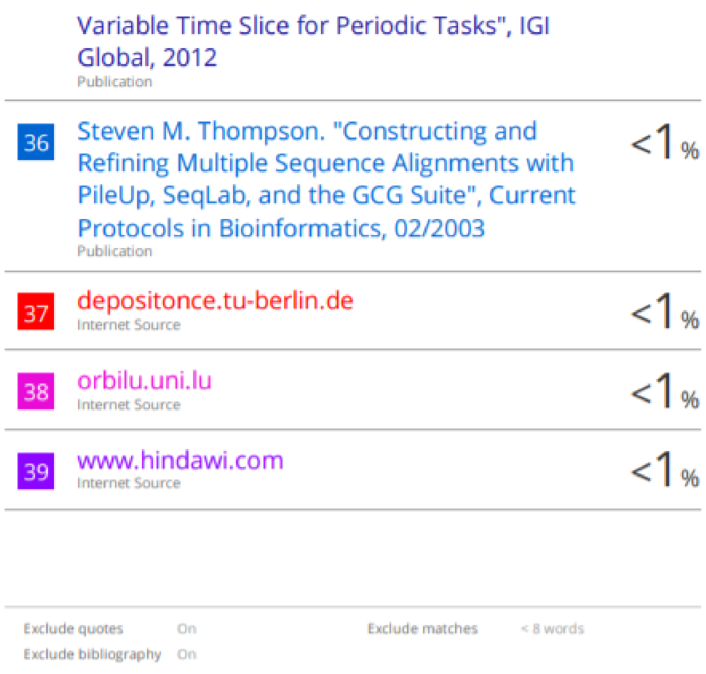


8	Submitted to HELP UNIVERSITY Student Paper	<1 %
9	Submitted to Higher Education Commission Pakistan Student Paper	<1 %
10	Submitted to Caledonian College of Engineering Student Paper	<1 %
11	Submitted to National College of Ireland Student Paper	<1 %
12	www.arnomoonen.nl Internet Source	<1 %
13	Submitted to University of Greenwich Student Paper	<1 %
14	Adam Freeman. "Pro ASP.NET Core Identity", Springer Science and Business Media LLC, 2021 Publication	<1 %
15	Submitted to KDU College Sdn Bhd Student Paper	<1 %
16	www.extendcode.com Internet Source	<1 %
17	ijarcce.com Internet Source	<1 %



18	Submitted to Asia Pacific Instutute of Information Technology Student Paper	<1 %
19	Submitted to University of Bedfordshire Student Paper	<1 %
20	Lei Ji, Jun Yan, Ning Liu, Wen Zhang, Weiguo Fan, Zheng Chen. "ExSearch", Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09, 2009 Publication	<1 %
21	www.sciencedirect.com Internet Source	<1 %
22	Submitted to University of Northumbria at Newcastle Student Paper	<1 %
23	Submitted to City University Student Paper	<1 %
24	Submitted to President University Student Paper	<1 %
25	Submitted to Taylor's Education Group Student Paper	<1 %
26	Submitted to University Tun Hussein Onn Malaysia Student Paper	<1 %

27	Submitted to University of London External System Student Paper	<1 %
28	Submitted to Singapore Institute of Technology Student Paper	<1 %
29	Submitted to Sunway Education Group Student Paper	<1 %
30	Submitted to University of Portsmouth Student Paper	<1 %
31	docplayer.net Internet Source	<1 %
32	Amit Pandey, Pawan Singh, Nirayo H. Gebreegziabher, Abdella Kemal. "Chronically Evaluated Highest Instantaneous Priority Next: A Novel Algorithm for Processor Scheduling", Journal of Computer and Communications, 2016 Publication	<1 %
33	gpwsirsa.edu.in Internet Source	<1 %
34	www.igi-global.com Internet Source	<1 %
35	Shaohua Teng, Wei Zhang, Haibin Zhu, Xiufen Fu, Jiangyi Su, Baoliang Cui. "chapter 17 A Least-Laxity-First Scheduling Algorithm of Variable Time Slice for Periodic Tasks", IGI Global, 2012 Publication	<1 %
36	Steven M. Thompson. "Constructing and Refining Multiple Sequence Alignments with PileUp, SeqLab, and the GCG Suite", Current Protocols in Bioinformatics, 02/2003 Publication	<1 %
37	depositonce.tu-berlin.de Internet Source	<1 %
38	orbilu.uni.lu Internet Source	<1 %
39	www.hindawi.com Internet Source	<1 %



Exclude quotes	On	Exclude matches	< 8 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)	LEE YING FOONG
ID Number(s)	17ACB02770
Programme / Course	COMPUTER SCIENCE (CS)
Title of Final Year Project	Appointment Scheduling System for Car Wash Service

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 4 **%** **Similarity by source** Internet Sources: 1 % Publications: 1 % Student Papers: 3 %	
Number of individual sources listed of more than 3% similarity: 0	

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.

[illegible]

Signature of Supervisor

Name: Yeck Yin Ping

Date: 3/9/2021

Signature of Co-Supervisor

Name: ____________

Date: ____________

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	17ACB02770
Student Name	LEE YING FOONG
Supervisor Name	MR. YECK YIN PING

TICK (√)	**DOCUMENT ITEMS** 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 Plastic Cover (for hardcopy)
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	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)
√	Weekly Log
√	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.	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.
ying	[illegible]
(Signature of Student) Date: 3/9/2021	(Signature of Supervisor) Date: 3/9/2021