

**WEB-BASED CAR RENTAL AND CAB  
SERVICE SYSTEM FOR UMP (UMPCAB)**

**NUR HASYA BINTI MOHD NORDIN**

**Bachelor of Computer Science (Software  
Engineering) with Honors**

**UNIVERSITI MALAYSIA PAHANG**

## UNIVERSITI MALAYSIA PAHANG

### DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : NUR HASYA BINTI MOHD NORDIN

Date of Birth : 20<sup>TH</sup> JANUARY 2000

Title : WEB-BASED CAR RENTAL AND CAB SERVICE SYSTEM  
FOR UMP (UMPCAB)

Academic Session : SEM 2 2021/2022

I declare that this thesis is classified as:

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

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

\_\_\_\_\_  
(Student's Signature)

\_\_\_\_\_  
(Supervisor's Signature)

\_\_\_\_\_  
New IC/Passport Number  
Date: 09/02/2023

TS. DR. AZLEE BIN ZABIDI  
\_\_\_\_\_  
Name of Supervisor  
Date: 09/02/2023

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

## THESIS DECLARATION LETTER (OPTIONAL)

Librarian,  
*Perpustakaan Universiti Malaysia Pahang,*  
Universiti Malaysia Pahang,  
Lebuhraya Tun Razak,  
26300, Gambang, Kuantan.

Dear Sir,

### CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name  
Thesis Title

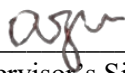
Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,



(Supervisor's Signature)

Date: 09/02/2023

Stamp: **TS. DR. AZLEE BIN ZABIDI**  
**SENIOR LECTURER**  
FACULTY OF COMPUTING  
COLLEGE OF COMPUTING & APPLIED SCIENCES  
UNIVERSITI MALAYSIA PAHANG  
26600 PEKAN, PAHANG DARUL MAKMUR  
TEL : 09-424 4638 FAX : 09-424 4666

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.



### SUPERVISOR'S DECLARATION

I/We\* hereby declare that I/We\* have checked this thesis/project\* and in my/our\* opinion, this thesis/project\* is adequate in terms of scope and quality for the award of the degree of \*Doctor of Philosophy/ Master of Engineering/ Master of Science in .....

(Supervisor's Signature)

Full Name : **TS. DR. AZLEE BIN ZABIDI**  
                  : **SENIOR LECTURER**  
Position : **FACULTY OF COMPUTING**  
                  : **COLLEGE OF COMPUTING & APPLIED SCIENCES**  
                  : **UNIVERSITI MALAYSIA PAHANG**  
Date : **26600 PEKAN, PAHANG DARUL MAKMUR**  
                  : **TEL : 09-424 4638 FAX : 09-424 4666**

(Co-supervisor's Signature)

Full Name :  
Position :  
Date :



## STUDENT'S DECLARATION

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

A handwritten signature in black ink, appearing to be 'Nur Hasya Binti Mohd Nordin', is written above a horizontal line.

(Student's Signature)

Full Name : NUR HASYA BINTI MOHD NORDIN

ID Number : CB19079

Date : 09/02/2023

WEB-BASED CAR RENTAL AND CAB SERVICE SYSTEM FOR UMP  
STUDENTS (UMPCAB)

NUR HASYA BINTI MOHD NORDIN

Thesis submitted in fulfillment of the requirements  
for the award of the degree of  
Bachelor of Computer Science (Software Engineering)

Faculty of Computing  
UNIVERSITI MALAYSIA PAHANG

JUNE 2022

## **ACKNOWLEDGEMENTS**

All form of praise and thanks is due to Allah, the creator of mankind, the most merciful and gracious for his blessings, protections, courage and guidance.

I am grateful and would like to express my sincere gratitude to several individuals and organizations for supporting me throughout my undergraduate studies. First, I would like to express my gratitude to my supervisor, Ts. Dr Azlee bin Zabidi for his helpful guidance, insightful feedbacks and practical advice that has tremendously helped me throughout my research and writing of this thesis. His immense knowledge, profound experience and professional expertise has enabled me to complete this project successfully. Without his support and guidance, this project would not have been possible.

I also wish to express my sincere thanks to my parents for their endless love and support. I am deeply indebted to them for all the sacrifices they have made for my sake. I am really thankful for their patience and understanding to make this work possible.

Lastly, I would like to give my sincere thanks to everyone who have made contributions for my final year project both directly and indirectly. I would like to acknowledge their comments and feedbacks which was crucial to the completion of this project.

## ABSTRAK

Sistem berasaskan web yang juga dikenali sebagai aplikasi web adalah aplikasi yang digunakan dengan pelayar Web melalui Internet. Ia disimpan pada pelayar jauh dan dihantar melalui Internet melalui antara muka pelayar. Hanya dalam satu dekad, Web telah berkembang daripada menjadi repositori halaman yang digunakan terutamanya untuk mengakses maklumat statik, kebanyakannya saintifik, kepada platform yang kuat untuk pembangunan dan penggunaan aplikasi. Berbanding dengan aplikasi mudah alih, aplikasi web lebih mudah digunakan. Ini kerana aplikasi mudah alih biasanya memerlukan pengguna mempunyai sejumlah simpanan untuk mereka dapat menggunakan aplikasi tersebut. Aplikasi web sebaliknya tidak memerlukan storan pengguna berfungsi kerana ia disimpan pada pelayan jauh dan dihantar melalui Internet. Selain itu, tidak semua orang mempunyai kapasiti penyimpanan untuk menyimpan banyak aplikasi di telefon pintar mereka. Dengan menggunakan aplikasi web, kedua-duanya boleh diakses melalui telefon pintar dan juga peranti lain melalui penyemak imbas. Aplikasi ini terdiri daripada dua modul iaitu Perkhidmatan Kereta Sewa dan Teksi. Tiga aplikasi kereta sewa sedia ada telah dikaji semula untuk menghasilkan versi aplikasi kereta sewa yang lebih baik. Model RAD telah dipilih untuk pembangunan projek ini. Terdapat lima fasa yang terlibat dalam model untuk memastikan bahawa aplikasi yang dibangunkan mencapai objektif yang dicadangkan. Aplikasi ini telah dinilai dan diuji oleh pelajar UMP untuk menyiasat keberkesanan dan persepsi pengguna terhadap aplikasi tersebut. Hasilnya menunjukkan bahawa pelajar memberikan maklum balas positif dan mereka bersetuju bahawa aplikasi ini dapat membantu mereka mengalami cara yang lebih baik untuk menyewa kereta dan menempah perkhidmatan teksi.



## **ABSTRACT**

A web-based system also known as a web application is an application that is invoked with a Web browser over the Internet. It is stored on a remote server and delivered over the Internet through a browser interface. In just one decade, the Web has evolved from being a repository of pages used primarily for accessing static, mostly scientific, information to a powerful platform for application development and deployment. Compared to mobile applications, web applications are much easier to use. This is because mobile application usually requires user to have a certain amount of storage for them to be able to use the application. Web application on the other hand does not require users' storage to function because it is stored on a remote server and delivered over the Internet. Besides, not everyone has the storage capacity to store multiple applications in their smartphones. By using web application, it can both be accessed through the smartphone and also other devices through the browser. This application consists of two modules which are Car Rental and Cab Service. Three existing application of car rental has been reviewed to produce a better version of car rental application. RAD model has been chosen for the development of this project. There are five phases involved in the model to make sure that the application developed achieves the proposed objectives. The application was evaluated and tested by the UMP students to investigate its effectiveness and user perception towards the application. The result showed that the students give positive feedbacks and they agree that this application can help them experience a better way of renting for cars and booking for cab services.

## TABLE OF CONTENT

<b>DECLARATION</b>	
<b>TITLE PAGE</b>	
<b>ACKNOWLEDGEMENTS</b>	<b>ii</b>
<b>ABSTRAK</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>TABLE OF CONTENT</b>	<b>v</b>
<b>LIST OF TABLES</b>	<b>x</b>
<b>LIST OF FIGURES</b>	<b>xi</b>
<b>LIST OF SYMBOLS</b>	<b>xiv</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xv</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 INTRODUCTION	1
1.2 PROBLEM STATEMENTS	2
1.3 OBJECTIVES	4
1.4 SCOPE OF PROJECT	5
1.5 SIGNIFICANCE OF PROJECT	6
1.6 THESIS ORGANIZATION	6
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>7</b>
2.1 INTRODUCTION	7
2.2 LITERATURE ANALYSIS OF RELATED WORKS	7
2.3 WEB-BASED APPLICATION USING LARAVEL FRAMEWORK	15

2.4	MAPPING APIs AND LOCATION SERVICES USING ARCGIS LOCATION SERVICES	16
2.5	ACCESSING ARCGIS LOCATION SERVICES USING ESRI LEAFLET	17
2.6	CAR-RENTAL SYSTEM	18
2.7	CAR POOLING SYSTEM (CAB SERVICE)	18
2.8	REVIEW OF EXISTING CAR RENTAL AND CARPOOL SYSTEM IN UMP	19
2.9	REVIEW OF EXISTING SYSTEMS	21
2.1.1	SOCAR	21
2.1.2	Rentalcars.com	22
2.1.3	GoSo Rent A Car Malaysia	23
2.10	COMPARISONS OF THREE EXISTING SYSTEMS	24
2.11	CONCLUSION	27
	<b>CHAPTER 3 METHODOLOGY</b>	<b>28</b>
3.1	RAPID APPLICATION DEVELOPMENT (RAD)	28
3.1.1	Analysis and Quick Design phase	29
3.1.2	Prototyping	30
3.1.3	Testing	31
3.1.4	Deployment	32
3.1.5	Project Management Framework	32
3.2	APPLICATION OF MAPPING APIS AND LOCATION SERVICES USING ARCGIS LOCATION SERVICES	33
3.3	REVERSE GEOCODING LOCATION USING ESRI LEAFLET	33
3.4	PROJECT REQUIREMENTS	34
3.4.1	Functional Requirements	34
3.4.2	Non-Functional Requirements	35

3.4.3	Constraints	35
3.4.4	Limitations	36
3.5	PROPOSED DESIGN	37
3.5.1	Context Diagram	37
3.5.2	Use Case Diagram	40
3.5.3	Use Case Description	43
3.5.4	Activity Diagram	57
3.6	DATA DESIGN	58
3.6.1	ERD	58
3.6.2	Data Dictionary	59
3.7	PROOF OF INITIAL CONCEPT	67
3.7.1	Register and Login	67
3.7.2	Mainpage	69
3.7.3	Dashboard	69
3.7.4	My Booking	70
3.7.5	Driver Dashboard	71
3.7.6	Report	73
3.7.7	Car Registration	75
3.7.8	Car Rental Booking	75
3.7.9	Cab Registration	78
3.7.10	Cab Booking	79
3.7.11	Car Rental Review	80
3.7.12	Cab Review	81
3.8	TESTING PLAN	82
3.9	POTENTIAL USE OF PROPOSED SOLUTION	86
3.10	GANTT CHART	87

<b>CHAPTER 4</b>	<b>88</b>
<b>IMPLEMENTATION, RESULT AND DISCUSSION</b>	<b>88</b>
4.1 INTRODUCTION	88
4.2 IMPLEMENTATION PROCESS	88
4.2.1 LARAVEL INSTALLATION AND SETUP	88
4.2.2 Database Environment and Connection Setup (MySQL and XAMPP)	93
4.2.3 Database Setup	94
4.3 DESIGNING USER INTERFACES AND CODE IMPLEMENTATION	97
4.3.1 Laravel Jetstream Components	97
4.3.2 Manage Car Rental	101
4.3.3 Manage Cab Service	105
4.3.4 Manage Car Review	108
4.3.5 Manage Users	110
4.3.6 Manage Report	111
4.4 TESTING RESULTS AND DISCUSSION	113
<b>CHAPTER 5 CONCLUSION</b>	<b>114</b>
5.1 INTRODUCTION	114
5.2 DISCUSSION ON USER ACCEPTANCE	114
5.3 PROJECT CONSTRAINTS	115
5.4 FUTURE WORK	116
REFERENCES	117
<b>APPENDIX A – SOFTWARE REQUIREMENT SPECIFICATION (SRS)</b>	
<b>APPENDIX B – SOFTWARE DESIGN DOCUMENT (SDD)</b>	

**APPENDIX C – USER ACCEPTANCE TESTING (UAT)**

**APPENDIX D – USABILITY TESTING (GOOGLE FORM)**

**APPENDIX E – SUS SCORE CALCULATION**

## LIST OF TABLES

Table 2.7.1.1 Summary of problems faced by students in renting for cars and cab services.	3
Table 2.7.1.1 Reviewing Table	9
Table 2.7.3.1 Comparison of existing systems	24
Table 3.5.1.1 Functional Requirements	34
Table 3.5.2.1 Non-functional requirements	35
Table 3.5.3.1 Constraints of the system	36
Table 3.5.4.1 Limitations of the system	36
Table 3.6.1 Modules Description with Actors Involved	40
Table 3.6.2 Manage Car Rental Use Case Description	43
Table 3.6.3 Manage Cab Booking Use Case Description	46
Table 3.6.4 Manage Car Review Use Case Description	50
Table 3.6.5 Manage Users Use Case Description	52
Table 3.6.6 Manage Report Use Case Description	55
Table 3.7.2 User Data Dictionary	59
Table 3.7.5 Car Rental Data Dictionary	60
Table 3.7.6 Cab Service Data Dictionary	61
Table 3.7.7 Car Data Dictionary	63
Table 3.9.1 User Acceptance Testing Form	82
Table 1 User Acceptance Testing Test Results	113

## LIST OF FIGURES

Figure 2.3.1 Laravel Logo	15
Figure 2.7.1 SOCAR website interface	22
Figure 2.7.2 Website homepage of Rentalcars.com	22
Figure 2.7.3 Homepage for GoSo Rent A Car Malaysia website	23
Figure 3.1.1 Phases of the RAD model	28
Figure 3.6.1.3 Context Diagram	37
Figure 3.6.2 Activity Diagram	57
Figure 3.7.1 Entity Relationship Diagram (ERD)	58
Figure 3.8.1 Register Interface	67
Figure 3.8.2 Login Interface	68
Figure 3.8.3 Forgot Password Interface	69
Figure 3.8.4 Mainpage Interface	69
Figure 3.8.5 Dashboard Interface	70
Figure 3.8.6 My Booking Interface	71
Figure 3.8.8 Driver Dashboard Interface	71
Figure 3.8.9 Registered Cars Interface	72
Figure 3.8.10 Cab Service Registered Interface	73
Figure 3.8.11 Report Interface	73
Figure 3.8.12 User List Interface	74
Figure 3.8.13 Car Rental List Interface	74
Figure 3.8.14 Cab Service List Interface	75
Figure 3.8.15 Register Car for Rental Interface	75
Figure 3.8.16 Car Rental Homepage Interface	76
Figure 3.8.17 Search Result Interface	76
Figure 3.8.18 Car Details Interface	77
Figure 3.8.19 Driver Details Interface	78
Figure 3.8.20 Rental Details Interface	78
Figure 3.8.22 Confirm Rental Interface	78
Figure 3.8.23 Register Car for Cab Service Interface	79
Figure 3.8.24 Cab Service Homepage Interface	80
Figure 3.8.25 Cab Details Interface	80
Figure 3.8.27 Car Rental Review Interface	81
Figure 3.8.28 Cab Review Interface	81



Figure 3.11.1 Gantt Chart	87
Figure 4.2.1 Composer download website	89
Figure 4.2.2 Create new laravel project command	89
Figure 4.2.3 Installing Laravel Jetstream command	90
Figure 4.2.4 npm install artisan command	91
Figure 4.2.5 npm run dev artisan command	91
Figure 4.2.6 Running the project on localhost	92
Figure 4.2.7 Controllers inside laravel	92
Figure 4.2.8 Models and migration files in Laravel	93
Figure 4.2.9 XAMPP Control Panel	94
Figure 4.2.10 XAMPP Control Panel	95
Figure 4.2.11 UMPCab database inside phpMyAdmin	95
Figure 4.2.12 .env file inside UMPCab project	95
Figure 4.2.13 Migrating database tables	96
Figure 4.2.14 Updating migration files	96
Figure 4.3.1 UMPCab landing page	98
Figure 4.3.2 UMPCab landing page code snippet	98
Figure 4.3.3 UMPCab Login Page	99
Figure 4.3.4 UMPCab Register Page	99
Figure 4.3.5 UMPCab Login and Register UI Page Code Snippet	100
Figure 4.3.6 Create new user code snippet (User Registration)	100
Figure 4.3.7 Profile Page	101
Figure 4.3.8 Profile Page Code Snippet	101
Figure 4.3.9 Book Car Rental Interface	104
Figure 4.3.10 Driver Dashboard User Interface	102
Figure 4.3.11 Driver Dashboard User Interface (Car Registration Form)	102
Figure 4.3.12 Driver Dashboard User Interface (Register for Rental)	103
Figure 4.3.14 Cab Service Homepage User Interface (Search Map)	105
Figure 4.3.16 Cab Service Homepage User Interface (Marking the Map)	106
Figure 4.3.17 Cab Service Homepage Code Snippet	107
Figure 4.3.19 My Bookings User Interface	109
Figure 4.3.20 Cab Review User Interface	110
Figure 4.3.21 Admin Dashboard User Interface	110
Figure 4.3.22 User List User Interfaces	111
Figure 4.3.23 Admin Dashboard User Interface	112

## LIST OF SYMBOLS

SBPWM	Simple Boost Pulse Width Modulation
ZSI	Z source inverter

## LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ERD	Entity Relationship Diagram
PHP	Hypertext Preprocessor
UML	Unified Modelling Language
IDE	Integrated Development Environment
SQL	Structured Query Language
UMP	Universiti Malaysia Pahang
HTML	Hypertext Mark-Up Language
CSS	Cascading Style Sheet
MVC	Model View Controller
DFD	Data Flow Diagram
SRS	Software Requirement Specification
SDD	Software Design Specification
3D	Three-dimensional
GPS	Global Positioning System
GUI	Graphical User Interface
OS	Operating System
IOS	iPhone Operating System
RAD	Rapid Application Development
SDLC	Software Development Lifecycle
UAT	User Acceptance Testing
MVC	Model-View-Controller

## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 INTRODUCTION**

Car rental systems are systems that rent cars or automobiles for a certain period of time to Passengers. In Malaysia, car rental service is increasingly becoming a preferred option for most people especially among students in higher education institutes. However, car rental systems in UMP are still implemented manually to manage the operations. In this project, a web-based car rental system will be developed. This application has the potential to ease students in renting for cars and booking for cab services.

After a few years of open-distance learning (ODL), higher education institutes all over Malaysia is starting to reimplement face-to-face class for their students. This requires students to attend classes at the university. Unfortunately, not all students are able to afford their own vehicle to travel to class especially if their class is far away from their residential college or their rented houses. The distance between their residential college and their class is so far away especially in Pekan Campus that students are usually left out of breath every time they reach their class. Students have no other choice but to travel to class on foot every day. Even though the distance is still reachable by foot, the evolution of today's technology should be able to ease up the burdens of those who are in need. Therefore, a car rental and cab service system is an essential system for students especially to those who does not own any vehicle in order to make it easier for them to search through a list of cars as well as helping them make some side income in order to accommodate their daily lives as students.

The car rental and cab service system allow students who own cars to publish their cars for rental and they can also do cab services to anywhere around Pahang. This system also allows students who wants to rent cars or book cab service to do so all in one application.

Since this system will be developed as a web-based system, users will be able to access this system anytime and anywhere if they have Internet connection.

Web-based systems are programmes that are kept on a distant server and distributed via the Internet using a browser interface. When organisations began to establish a presence in the online world, web application development became more important. As individuals began to utilise the Internet on a more frequent basis, more browsers and development platforms emerged, particularly after the covid-19 pandemic trapped them inside their homes. As organisations and corporations began to provide apps that needed the usage of a web browser, web applications grew in popularity. Web applications are useful for a variety of reasons, and they may be utilised by anybody, from organisations to people. Webmail, calculators, and e-commerce stores are examples of common web apps. Webmail, calculators, and e-commerce stores are examples of common web apps. Most web applications can be accessed regardless of the browser although some require specific browsers.

## **1.2 PROBLEM STATEMENTS**

One of the problems with UMP's existing car rental and cab service is that it is implemented manually. At Universiti Malaysia Pahang, students are required to attend face-to-face class and in order to do so, they are required to travel to their respective faculty to attend class. However, not a lot of students have their own vehicle to travel to class or any of their desired locations. Thus, these students will be needing transports to go to class. The existing way on how the students book a car and cab service is only through social medias such as Whatsapp. There will sometimes be announcements about car rental and cab services on the E-community platform and also Whatsapp. Students are not always alert about the announcements and they may have already missed it multiple times even when they are in urgent need of transportation. This is not really a structured way for UMP students to book cars or rent cab service as the existing services are scattered all over the place.

Besides that, another problem is that students are vulnerable to scammer attacks mainly because the manually implemented car rental and cab service can allow scammers anywhere with Internet access to take advantage of students' intentions to rent a car. The

students are naïve enough to fall under the scammers lies and will end up paying the scammer a good amount of money. There have been cases where scammers pretend to be one of the car rental car owners. Scam cases have been on the rise ever since the covid-19 outbreak. In Malaysia, the commercial crime investigation department deputy director Muhammed Hasbullah Ali found that cheating cases had increased by 60.6% over the last 10 years. This is due to the public not exposed enough about the dangers of scammer.

The next problem is that car owners of the car rental service will have a hard time keeping track of all the Passengers who have rented their cars. Since the existing way on how the car rental system is implemented in UMP, all transactions are done through Whatsapp and Telegram which is not really a systematic and efficient way. When the car owners get a lot of Passengers, it will be more difficult for them to see their past transactions, and this may actually cause them some unwanted loss. This may also cause multiple Passengers to have overlapping rent dates as the car owners may have wrongly overlooked their car status as not yet rented. Therefore, instead of manual implementation of the car rental and cab service system, we can implement web-based system for the car rental and cab service. *Table 1.2.1* summarizes the problems faced by students in renting for cars and cab services.

**Table 2.1.1.1 Summary of problems faced by students in renting for cars and cab services.**

No	Problem	Description	Effect
1	Manually implemented car rental system is not efficient	The process of renting for cars and cab service using the traditionally manually way such as through Whatsapp, Telegram and sometimes announcement on E-community is not effective. This is because students are not always alert about the announcements and may have	Students will have difficulty in looking for car rental services especially when they are in urgent need of it.

		already missed it multiple times.	
2	Easy for scammers to scam students	Since the car rental and cab service is manually implemented, scammers from anywhere with Internet access can freely take advantage of students' intentions to rent a car.	Students can be at a loss and lose a certain amount of money.
3	Difficult for car owners to keep track of all the Passengers who have rent their cars.	Car owners may have a lot of trouble in trying to keep track of the students that have rented their cars and if anything ever happen to their cars, it is difficult to keep track on where their vehicles have been.	Car owners may be at a loss because Passengers may be doing anything they want with the rented cars.

### 1.3 OBJECTIVES

Based on the problem statements, the objectives of the system are:

- i. To study the existing car rental and cab service system for UMP.
- ii. To develop a secure web-based system that can store bookings and reservations information to help the car owners keep track of all transaction records.
- iii. To validate the car rental and cab service system by using user satisfaction test.

## **1.4 SCOPE OF PROJECT**

The scope of the project are:

User Scope:

- i. UMP students at Pekan Campus
- ii. UMP students at Gambang Campus

System Scope:

- i. Consists of five modules which are Manage Car Rental, Manage Cab Service, Manage Car Review, Manage Users and Manage Report.

Development Scope:

- i. This system will be developed using Visual Studio Code, Laravel framework and MySQL as database server.
- ii. This system contains a Leaflet map.



## **1.5 SIGNIFICANCE OF PROJECT**

### **i. Students (Passengers)**

Students can rent a car and book a cab service in a more efficient and systematic way.

### **ii. Students (Car Owners)**

Car owners can publish their car rental or cab service in a more efficient and systematic way. They can also keep track of all their Passenger and transaction records.

## **1.6 THESIS ORGANIZATION**

This thesis consists of five chapters. Chapter 1 discusses on the introduction to the project which are the introduction, problem Statements, objectives, the scope and significance of the project as well as the thesis organization.

Chapter 2 briefly explains the literature review; historical overview of the system and some related works and achievements made by other researchers towards enhancing car rental systems.

Chapter 3 covers the analysis and design of the system, which are noted after intensively and carefully chosen the development methodology and how the methodology would be approached. This project implements RAD methodology. The stages used in this project are Analysis and Quick Design, Prototyping, Testing, and Deployment.

Chapter 4 comprises the implementation of the system, the programming language used, results and discussion based on the development and testing of the system. All of the results and outputs were briefly discussed in this chapter. These include software development, application testing, data collection, and project results.

The final chapter of this project wraps up and summarises the findings. This chapter discusses the project's limitations and constraints, as well as the project's future efforts.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 INTRODUCTION**

Chapter 2 is about the review of existing car rental and cab service system. Three existing car rental and cab service system were explained in detail through their Graphical User Interface (GUI), supported platform, GPS implementation, target audience of the system, car rental history, and the advantages and disadvantages of the existing system. This chapter also covers the introduction to web-based application using Laravel framework, mapping APIs and location services using ArcGIS location services, accessing ArcGIS location services using Esri Leaflet, car rental system and carpooling system. All related materials were taken into account and revised in collecting information about the topic.

#### **2.2 LITERATURE ANALYSIS OF RELATED WORKS**

Works in [1-3] applied the use of GPS tracking system on their projects. Papers [1-3] explained how the GPS tracking system can track and identify the whereabouts of objects on the map. The suggested system in [1] includes a GPS device that receives Haji's current location (latitude and longitude) and transmits it to a database server, allowing the Haji's position on the map to appear as a point on the website's Google map. The system in [3] used the GPS-based tracking device as a location tracking mechanism and the location is reported for every 1 second to the cloud database. [4] presents an accurate and reliable real time tracking system using GPS and GSM (Global System for Mobile Communication) services that permits localization of a portable tracked unit and transmitting the position to the tracking centre.

The works in [4-6] highlight the importance of carpooling and ridesharing. The suggested system in [9] depicts the notion of ridesharing for an organisation that is an IOS application in real-time that will help students and teachers accept their requests and

connect them with the driver in a matter of seconds. The carpooling system in [5] incorporates a matching algorithm, dynamic web pages and a database system that is smoothly connected with Google Maps API. The carpooling system in [5] will also match the passengers and drivers depending on their travel paths. Paper [6] explains how the GrabShare algorithm was created from a data standpoint, as well as how different approaches to the problem can have varying effects on the Grab community.

Papers [7-8] highlights the importance and advantages of using Laravel framework for a web-based application. The works in [7-8] emphasizes how using Laravel frameworks can help in the development of web-based application especially in terms of time consumption. Paper [7] also presents the security techniques that are present in Laravel framework and how it can protect the web applications from attacks based on the ten vulnerabilities listed in the OWASP Top Ten. [8] concludes what advantages Laravel had on other frameworks and how it is different to work on frameworks.

The use of car rental systems is displayed in papers [9-12]. These papers highlight how car rental systems can help the public with their daily transportation needs. The car rental system in [9] helps users to rent cars when needed while also giving their idle car in rent which will give them extra income. The car rental system in [10] is implemented with a GPS tracking that will help Passengers to track the whereabouts of their rented cars. In paper [11], the challenges encountered in the general automobile rental system for the search mechanism was compared and the rental car's value was determined.

The concept of using dynamic maps was depicted in paper [5]. Paper [5] presents a system that is inspired from the concept of dynamic maps that collects data from user vehicles and maps them out by location and time, keeping a history of all recorded information. Then, the performance of the system module in charge of localization and tracking of users' devices, and present and discuss the obtained results. This paper also introduced a different approach for dynamic maps implementation. This approach is aimed to keep a log of traffic data and allow users to efficiently filter and access the recorded vehicle video streams and the information that has been extracted from them.

Table 2.2.1 below shows the reviewing table.

**Table 2.1.1.1 Reviewing Table**

No	Author	Database published	Approaches	Outcome	Year published	Title
1	Md. Ehtesham UI Hossain Khan Nowshin Anjum Fouzia Arida Haque Mohammad Monirujjaman Khan	IEEE	<ul style="list-style-type: none"> <li>- Using GPS Antenna, GSM module SIM808 and Arduino UNO R3.</li> <li>- Using PHP, JavaScript and AJAX to construct the server side.</li> <li>- Using MySQL as a database server.</li> </ul>	<ul style="list-style-type: none"> <li>- Using an adaptable and cost-effective tracking device.</li> <li>- System implemented with maximum level of location precision possible.</li> <li>- Being able to see the whereabouts of the Hajjs.</li> </ul>	2021	Hajji Tracker: Development of Web-Based GPS Tracking System for Pilgrims
2	Sarah Aimi Saad Amirah 'Aisha Badrul Hisham	IEEE	<ul style="list-style-type: none"> <li>- Using GPS receiver and ESPresso Lite V2.0 for tracking device.</li> </ul>	<ul style="list-style-type: none"> <li>- Providing an estimated time of arrival (ETA) based on real-time information on the bus speed and the</li> </ul>	2018	Real-time on-campus public transportation monitoring system

	<p>Mohamad Hafis Izran Ishak</p> <p>Mohd Husaini Mohd Fauzi</p> <p>Muhammad Ariff Baharudin</p> <p>Nurul Hawani Idris</p>		<ul style="list-style-type: none"> <li>- Using MySQL and MariaDB as database servers.</li> <li>- Using Apache as web server.</li> <li>- Using FileZilla as file management tool.</li> <li>- Using PHP, JavaScript, JSON, Ajax and HTML as programming languages.</li> </ul>	<p>remaining distance and the bus and the target bus stops.</p>		
3	<p>Hind Abdalsalam Abdallah Dafallah</p>	IEEE	<ul style="list-style-type: none"> <li>- Using PHP and JavaScript as programming language.</li> <li>- Using MySQL as database server.</li> <li>- Using Garmin 18-5 HZ GPS receiver as a tracker.</li> </ul>	<ul style="list-style-type: none"> <li>- Provide a reliable real time tracking system using GPS.</li> </ul>	2014	Design and implementation of an accurate real time GPS tracking system
4	<p>Abber Javed Syed</p> <p>Sara Saba</p>	IEEE	<ul style="list-style-type: none"> <li>- Using PHP Web Servers to connect WordPress and Servers.</li> </ul>	<ul style="list-style-type: none"> <li>- Front-end is devised on</li> </ul>	2020	CABTAB – A Factual Analysis Concerned with Travelling

	Zain-ul-Abideen Muhammad Noman Adnan Talib		- Using MySQL and Firebase as database.	WordPress and Servers.		Issues, Specifically for An Organization
5	Fu-Shiung Hsieh	IEEE	- Using JAVA as programming language for Android application. - Using Google Maps API for map services.	- Using Android application as front-end. - Provide results that significant reduction in total distance can be achieved through car pooling based on the proposed car pooling algorithm.	2017	Car Pooling Based on Trajectories of Drivers and Requirements of Passengers
6	Muchen Tang Serene Ow Wenqing Chen	IEEE	- Using simulator to investigate performances of different markets.	- Demonstrated how the effective use of data was beneficial in the design, deployment and subsequent	2017	The Data and Science behind GrabShare Carpooling

	Yang Cao Kong-wei Lye Yaozhang Pan			improvement of the product.		
7	Igor Vanderlei Jean Araujo Rodrigo Rocha Gabriel Silva Felipe Pacheco Jamilson Dantas	IEEE	<ul style="list-style-type: none"> <li>- Using PHP as programming language.</li> <li>- Using MySQL as database server.</li> </ul>	<ul style="list-style-type: none"> <li>- Using MVC architecture approach.</li> </ul>	2021	Analysis of Laravel Framework Security Techniques against Web Application Attacks
8	Neha Yadav Dharmveer Singh Rajpoot Shri Krishna Dhakad	IEEE	<ul style="list-style-type: none"> <li>- Using PHP &gt;= 7.1.3</li> <li>- Using openssl PHP extension.</li> <li>- Using PDO PHP extension.</li> <li>- Using XML PHP extension.</li> <li>- Using JSON PHP extension.</li> </ul>	<ul style="list-style-type: none"> <li>- Using MVC architecture.</li> </ul>	2019	LARAVEL: A PHP Framework for E-Commerce Website

			<ul style="list-style-type: none"> <li>- Using Ctype PHP extension.</li> <li>- Using Composer to manage dependencies.</li> </ul>			
9	<p>Shakhawat Hossain Mahi</p> <p>Umme Habiba Maliha</p> <p>Sadman Sakib</p>	IEEE	<ul style="list-style-type: none"> <li>- Using JAVA as programming language.</li> <li>- Using Android Studio for Android app development.</li> <li>- Using Firebase as database.</li> </ul>	<ul style="list-style-type: none"> <li>- Provide a car rental system to users.</li> </ul>	2020	Development of Web and Mobile Application Based Online Buy, Sell and Rent Car System
10	<p>Gurram Venkata Bhargavi</p> <p>Jonnalagadda Pavan Vitesh</p> <p>Tadipaneni Gopi Chand</p> <p>Bethu Srilekha</p>	IEEE	<ul style="list-style-type: none"> <li>- Using PHP, HTML, CSS and JavaScript as programming languages.</li> <li>- Using ESP8266 SoC as GPS module.</li> </ul>	<ul style="list-style-type: none"> <li>- Provide a rental system with GPS tracking functionality for harvesters.</li> </ul>	2021	Practical Rental System for Harvesters with GPS Tracking



	Chakravarthy Gunturu					
11	Falah Y H Ahmed  Eizwan Bin Hazlan  Muhammad Irsyad Abdulla	IEEE	<ul style="list-style-type: none"> <li>- Using Agile Methodology for development of mobile application.</li> </ul>	<ul style="list-style-type: none"> <li>- Boost existing rental system for vehicles using EZGO application.</li> </ul>	2021	Enhancement of Mobile-Based Application for Vehicle Rental
12	Mariem Maiouak  Tarik Taleb	IEEE	<ul style="list-style-type: none"> <li>- Using Tensorflow-GPU with CUDA 9.0 and OpenCV for object detection service.</li> <li>- Using RTMP as streaming servers.</li> <li>- Using Google Maps API and Google Places API for mapping module.</li> </ul>	<ul style="list-style-type: none"> <li>- Provide a dynamic mapping system for vehicles tracking.</li> </ul>	2019	A Dynamic Map-based Framework for Real-Time Mapping of Vehicles and their Surroundings

### 2.3 WEB-BASED APPLICATION USING LARAVEL FRAMEWORK

A web-based application is an application that is accessed through HTTP. Web-based applications runs in a web browser. Web applications are increasingly present in our daily lives to help people in carrying out their tasks (Igor, Jean, Rodrigo, Gabriel, Felipe & Jamilson, 2021). A web-based application can be accessed by anyone through a browser as long as they have internet connections.

For a web application to be developed, one must have the knowledge of PHP language and other supporting tools like CSS and javaScript. Building a website in PHP is time consuming and it requires a lot of technical understanding. Sometimes, the same codes are written many times for different projects. In order to overcome this problem, frameworks are built (Neha, Dharmveer & Shri, 2019). Laravel is classified among one of the successful PHP frameworks because it is a widely used open-source PHP framework which is used to build the comprehensive and efficient web application. The *Figure 2.3.1* shows the logo of the widely used PHP framework.

**Figure 2.3 Laravel Logo**

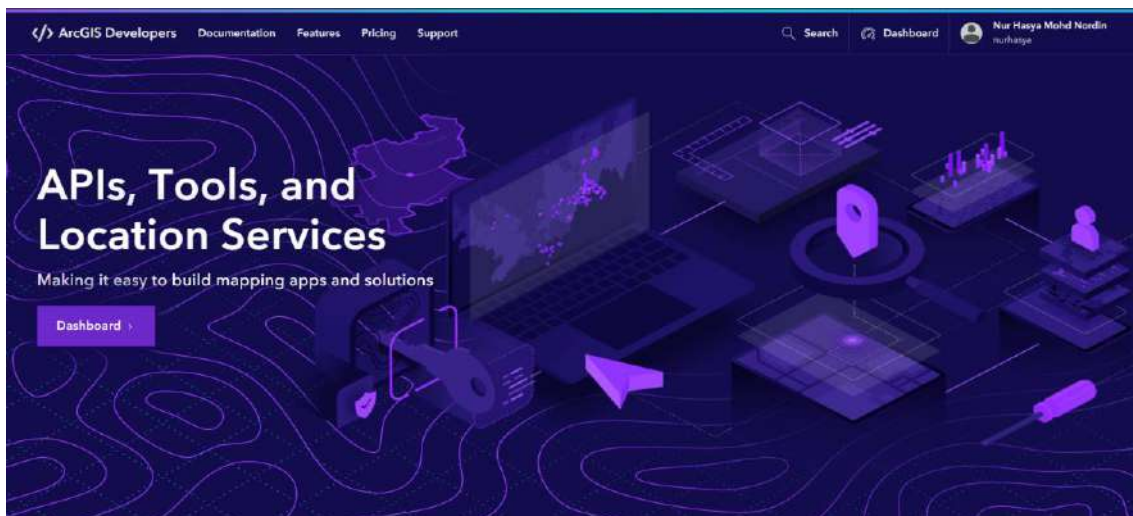


Web designing with Laravel framework is very easy and simple but one should know the basics of PHP to understand the framework structures (Neha, Dharmveer & Shri, 2019).

## 2.4 MAPPING APIs AND LOCATION SERVICES USING ARCGIS LOCATION SERVICES

A Map API also known as Mapping API provides location intelligence for software developers creating location-based products and services. It is the base building block for location-aware applications, feature-rich maps and the retrieval of geographical-related data. It allows the creation of location-aware infographics, mind maps and visualization aids. A typical Mapping API includes features for geocoding, reverse geocoding, geolocation, directions and navigations and is not limited to displaying different types of maps. One of the location services that can be used for Mapping APIs is ArcGIS location services.

ArcGIS location services are a complete set of services for building mapping and spatial analysis applications. The services can be accessed with different APIs to display maps, style layers, search for places, geocode addresses, finding routes, or performing advanced operations such as data enrichment or spatial analysis. The services and developer tools offered can also be used to host and manage data in the cloud.



**Figure 2.4 ArcGIS Developer website**

## **2.5 ACCESSING ARCGIS LOCATION SERVICES USING ESRI LEAFLET**

ArcGIS location services can be accessed using Esri Leaflet. Esri Leaflet is a lightweight, open-source Leaflet plugin for accessing ArcGIS location services and ArcGIS Enterprise services. The API can be used to display interactive maps and data, and to access services to perform operations such as geocoding, routing, and spatial analysis.

## **2.6 CAR-RENTAL SYSTEM**

A car rental system is a system that allows users to find for available cars to be rented and also have their own cars up for rental services. Car rental is a method of using a vehicle that can be used lawfully with a fee for a period of time. Car rental systems are system that provides services to the people who want to rent a car (Shakhawat, Umme & Sadman, 2020). It helps people to drive around to their destination even without owning their own vehicle (Falah, Eizwan & Muhammad, 2021). Traditionally, anyone who wants to locate a car rental service can call the rental service manager by phone or go to the store to search for available vehicles for rental directly (Falah, Eizwan & Muhammad, 2021). An online car rental system would make it much easier for vehicle owners to manage their vehicles.

## **2.7 CAR POOLING SYSTEM (CAB SERVICE)**

Carpooling or also know as cab service is the concept of sharing cars to accommodate more than one person at a time, eliminating the need for riders to drive themselves in separate vehicles. Carpooling is a collective transportation system based on a shared use of vehicles, whose objective is to reduce the number of cars in use by grouping people (Fu-Shiung Hsieh, 2017). The main incentive of a carpooling system is to eliminate the hectic and expense of travelling as it has been remained essential during each phase of life (Abber, Sara, Zain-ul-Abideen, Muhammad & Adnan Talib).

## 2.8 REVIEW OF EXISTING CAR RENTAL AND CARPOOL SYSTEM IN UMP

The existing car rental and carpool system in UMP is only done manually through social medias such as Whatsapp and Telegram. The car rental and carpool posts are posted in group chats where students need to browse through thousands of other messages from other students. The announcement for available car rentals can also sometimes be found on E-community which is a site for UMP Students to view UMP related stuff. Section 2.8.1 shows the existing car rental system for UMP and section 2.8.2 shows the existing carpool system for UMP.

### 2.8.1 Existing Car Rental System in UMP

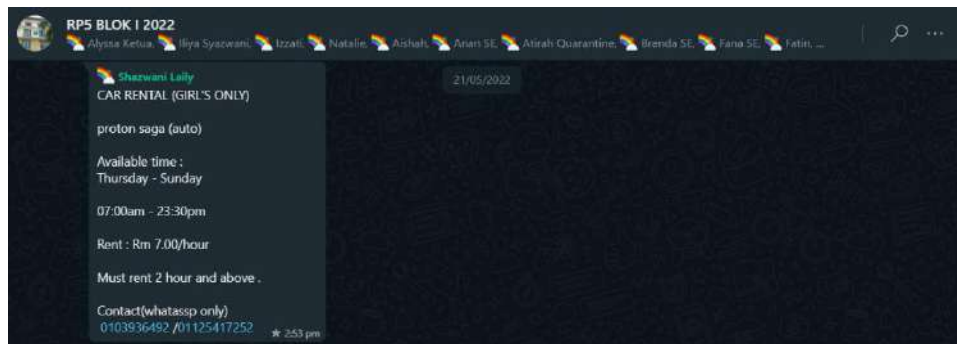


Figure 2.8.1.1 Car Rental Announcement Through Whatsapp

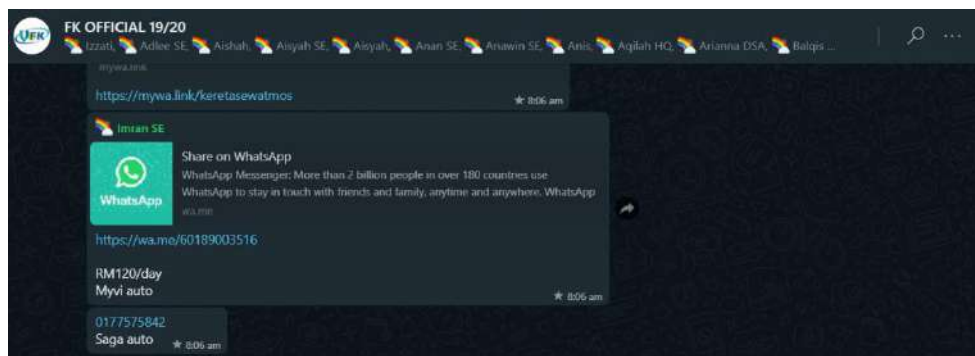


Figure 2.8.1.2 Car Rental Announcement Through Whatsapp (2)

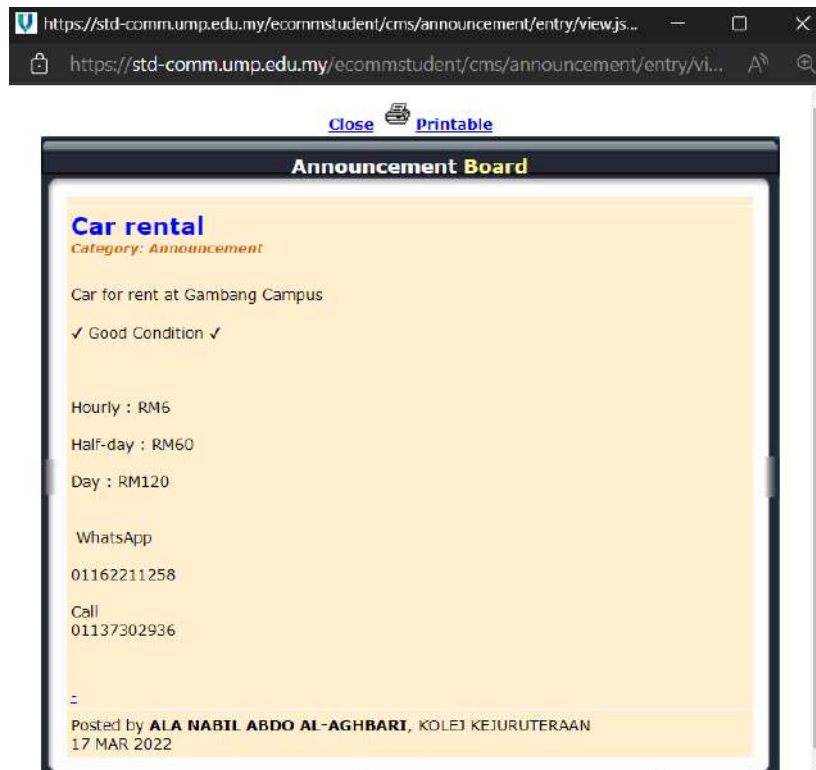


Figure 2.8.1.3 Car Rental Announcement Through E-Community

## 2.8.2 Existing Carpool System in UMP



Figure 1.8.2.1 Carpool Announcement Through Whatsapp



Figure 2.8.2.2 Carpool Announcement Through E-Community

## 2.9 REVIEW OF EXISTING SYSTEMS

This section reviews the three existing car rental and cab service systems. The systems that will be reviewed are SOCAR, Rentalcars.com and KKIA Car Rental.

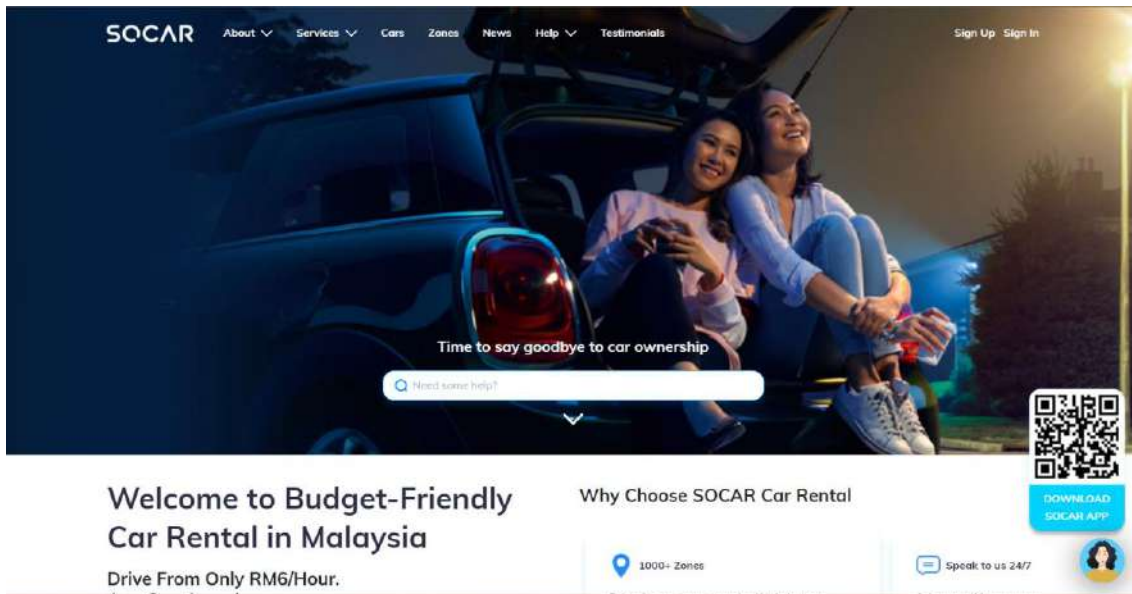
### 2.1.1 SOCAR

SOCAR is a car rental system that is available both as an android application and also the web. This system provides car-sharing services which allows users to rent cars hourly, daily, weekly, or even monthly. The mission of this system is to minimise car ownership through accessible mobility and bring impact to a greener society in every part of the country and beyond.

In order to use this system, users can either download the SOCAR app from Google Play and the App Store or they can access SOCAR's website through a browser. *Figure 2.7.1.1* below shows the interface of SOCAR.



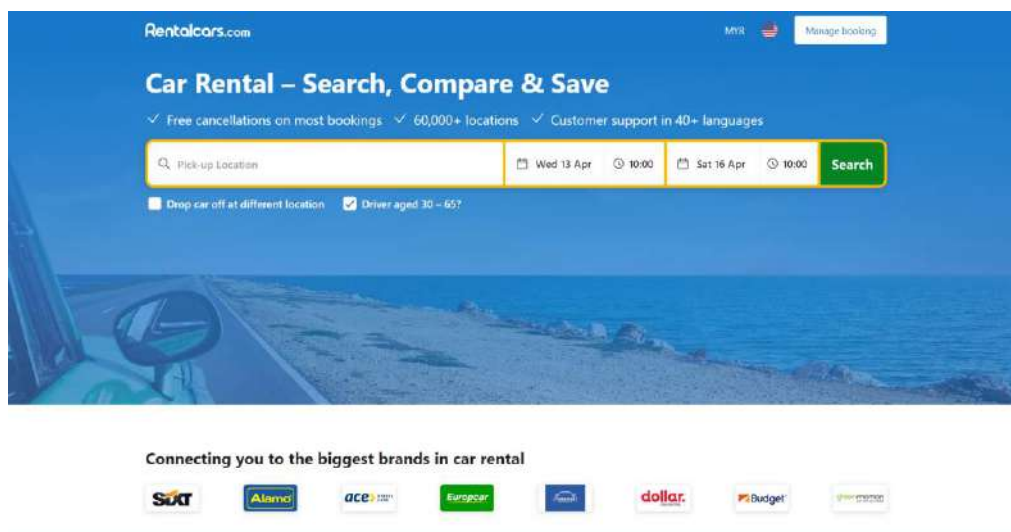
**Figure 2.1.1 SOCAR website interface**



### 2.1.2 Rentalcars.com

Rentalcars.com is a web-based car rental system that allow users to rent cars directly from their website. This website allows users to search, compare and book the car that they want. In order to book a car, the users must first provide the system with their pick-up location and date before they can search for the cars they want to rent. *Figure 2.7.2.1* below shows the website homepage of Rentalcars.com.

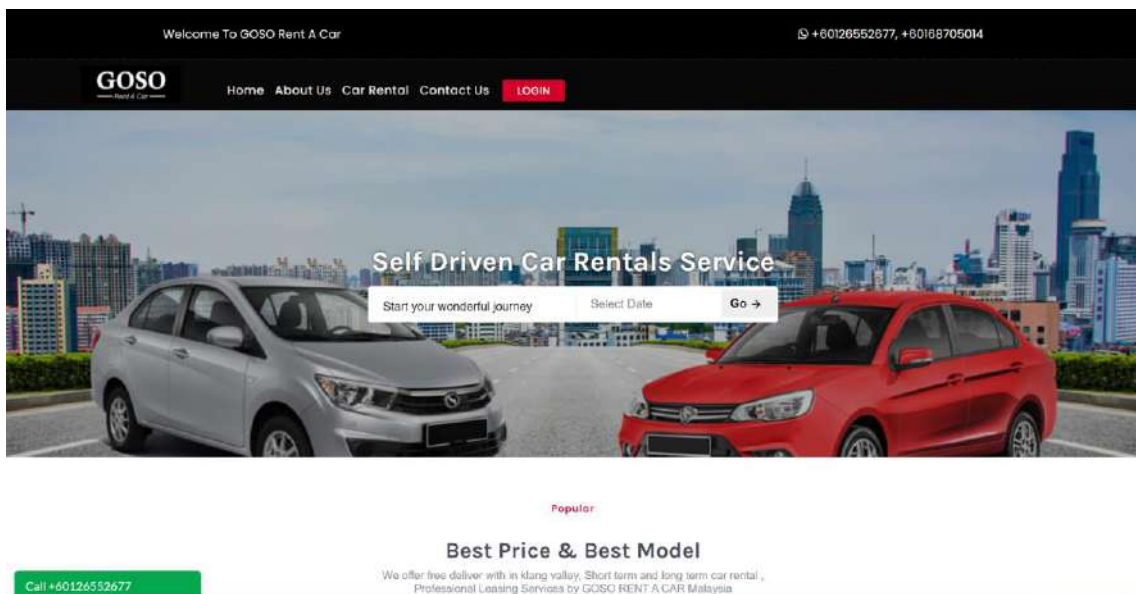
**Figure 2.1.2 Website homepage of Rentalcars.com**



### 2.1.3 GoSo Rent A Car Malaysia

GoSo Rent A Car Malaysia is a web-based car rental system that provide users with available services with the best economy cars in Malaysia. This website allows users to search for available cars by providing the date or by browsing through a list of available cars. To book a car, users need to finalize their final drive and then choose a pickup date and time, return date and time, adding any additional preferences and then proceed on booking and register and checkout. *Figure 2.7.3.1* below shows the homepage of the GoSo Rent A Car Malaysia website.

**Figure 2.1.3 Homepage for GoSo Rent A Car Malaysia website**



## 2.10 COMPARISONS OF THREE EXISTING SYSTEMS

From the comparisons of the three existing systems, each application has its own advantages and drawbacks. In terms of the Graphical User Interface (GUI), SOCAR is the most attractive compared to Rentalcars.com and GoSo Rent A Car Malaysia that has much simpler design. All three of the applications can be supported on web browsers but SOCAR can also be supported on other platforms such as android and IOS. None of the applications has GPS implementation and they have the same audience which are adults. All three of the applications does not show the history of the previously booked cars. Out of the three reviewed applications, only SOCAR allow users to pinpoint location directly on the map.

*Table 2.8.1* below shows the comparison between SOCAR, Rentalcars.com and GoSo Rent A Car Malaysia in terms of Graphical User Interface (GUI), supported platforms, GPS implementation, target audience, ability to view car rental history, ability to pinpoint location directly on the map, advantages and the disadvantages.

**Table 2.1.3.1 Comparison of existing systems**

<b>Application Name</b>	SOCAR	Rentalcars.com	GoSo Rent A Car Malaysia
<b>Graphical User Interface</b>	The interface is beautiful and is easy to navigate through.	The interface is simple.	The interface is simple.

<b>Supported Platforms</b>	Android, IOS and web browsers.	Web browsers.	Web browsers.
<b>GPS Implementation</b>	None	None	None
<b>Audience</b>	Adults	Adults	Adults
<b>Car Rental History</b>	Not showed	Not showed	Not showed
<b>Pinpoint Location</b>	Yes	No	No
<b>Advantages</b>	<ul style="list-style-type: none"> <li>• Users can access this application through multiple platforms such as web browsers, android and IOS.</li> <li>• A chat box is available for users to ask for assistance.</li> <li>• Users can pinpoint their pick-up location on the map.</li> </ul>	<ul style="list-style-type: none"> <li>• Users can filter through many car criteria.</li> <li>• Users can directly book a car through the website.</li> </ul>	<ul style="list-style-type: none"> <li>• Users can directly book cars through the website.</li> </ul>
<b>Disadvantages</b>	<ul style="list-style-type: none"> <li>• Users cannot track the real-time location of the car.</li> <li>• Users can only book cars through the SOCAR mobile app.</li> </ul>	<ul style="list-style-type: none"> <li>• Users cannot track the real-time location of the car.</li> </ul>	<ul style="list-style-type: none"> <li>• Users cannot track the real-time location of the car.</li> </ul>

		<ul style="list-style-type: none"><li>• No chat box is available if users require urgent assistance.</li></ul>	<ul style="list-style-type: none"><li>• No chat box is available if users require urgent assistance.</li></ul>
--	--	--	--

Based on the above table, the features that will be added into the proposed system are:

- I. Provide reverse geocoding functionality to the system using ArcGIS location services.
- II. Ability for users to view their car rental history.

## **2.11 CONCLUSION**

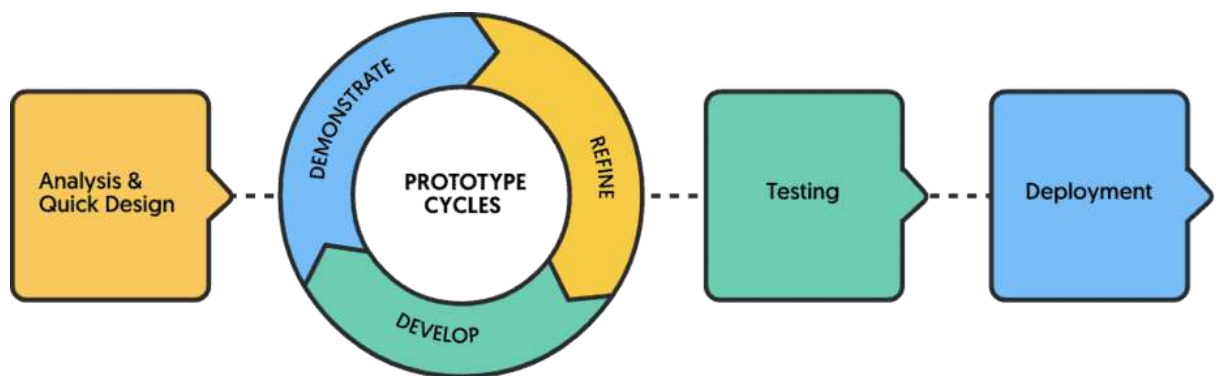
The literature review provides broad knowledge about the important material necessary to comprehend the field of the proposed project. The literature review was undertaken utilising articles, journals, websites and other academic resources to establish the most appropriate marker tracking methods. The chosen method was by using Pusher Channels because it would be compatible with web-based application developed using Laravel framework. The reason why a Laravel framework is preferred for the development of this web-based application was also discussed.

## CHAPTER 3

### METHODOLOGY

#### 3.1 RAPID APPLICATION DEVELOPMENT (RAD)

Rapid Application Development, or RAD for short, is a flexible software development method. During the development process, a RAD method is based on flexibility and the ability to react to new knowledge as well as the emergence of new requirements. RAD projects typically take a shorter time to complete compared to other SDLC models. Since this project will be developed in a short amount of time and only require one person as a developer, a RAD approach is the most suitable model for this project. RAD consists of four fundamental steps which are analysis and quick design, prototyping, testing and deployment. *Figure 3.1.1* shows the four phases of the Rapid Application Development model.



A typical RAD cycle

Figure 3.1 Phases of the RAD model

### **3.1.1 Analysis and Quick Design phase**

The first phase of the Rapid Application Development model is the analysis and quick design phase. Requirements elicitation occurs during this phase, in which stakeholders speak with one another to define project requirements, goals, and expectations, as well as current and potential difficulties that will need to be handled throughout construction. The client will submit product insights, and research will be undertaken in partnership with other stakeholders to fulfil the requirements. This is to make sure that every stakeholder is on the same page as each other in the early stage of the development cycle to avoid miscommunications. This phase is an important phase because requirements in RAD can change at any point in the development cycle.

For this project, this phase involves planning, brainstorming, and gathering requirements for the car rental and cab service system. The stakeholders involved were UMP students, the PSM supervisor and the developer. The objectives of this project were determined based on the problem statements identified from the existing car rental and cab service system for UMP. A Gantt chart was also created to make sure that the proposed system is successfully developed within the time frame. The modules that will be implemented which are the in the project was determined by considering the needs of UMP students in renting for cars and booking for cab services around campus.



### **3.1.2 Prototyping**

The system's initial models and prototypes are created during this phase. Users collaborate with developers to design and produce prototypes that meet the system's needs. Users will interact with the prototype and provide feedback based on what worked and what did not. The bugs reported by the users are worked out in an iterative process. This phase is continuous and will be repeated multiple times until users are satisfied with the prototype.

For this project, the design of the system was determined as well as the development of the prototype. The user interfaces developed in the prototype was shown to the users for feedbacks and is designed based on feedbacks from the users. The user inputs received will be used to improve the prototypes so that it is inline with the requirements.

### **3.1.3 Testing**

The testing phase is also known as the rapid construction phase where the prototypes are converted into working models. The testing phase involves unit testing, integration testing as well as system testing. This phase also involves user inputs which then will be used by developers to make sure everything is working smoothly and that the end result satisfies the client's expectations and objectives.

For this project, the prototype for the UMPCab was developed and tested by both the users and developers.

### 3.1.4 Deployment

In the deployment phase, the finished product is implemented and deployed. Before the product is deployed, approval upon the product is required. Once the product is properly assessed for factors like usability and stability, it is ready to be delivered. For this project, the UMPCab will be deployed and delivered after being tested and approved.

### 3.1.5 Project Management Framework

The Project Management Framework is the table below is derived from the RAD model for each phases along with the activities and deliverables.

PHASES	ACTIVITIES INVOLVED	DELIVERABLES
Analysis and Quick Design Phase	<ul style="list-style-type: none"><li>• Planning, brainstorming and gathering of requirements for the car rental and sab service system.</li><li>• Identifying problem statement.</li><li>• Identifying objective for the system based on the problem statement.</li></ul>	<ul style="list-style-type: none"><li>• Software Requirement Specification (SRS)</li></ul>
Prototyping	<ul style="list-style-type: none"><li>• Designing the architecture of the system using wireframe.</li><li>• Designing the user interface based on the system architecture identified which is the Model-View-Controller (MVC) architecture.</li></ul>	<ul style="list-style-type: none"><li>• Software Design Document (SDD)</li><li>• MVC Architecture</li><li>• User Interface of the system</li><li>• System prototype</li></ul>
Testing	<ul style="list-style-type: none"><li>• Transforming user interface from the wireframe into source codes.</li><li>• Performing testing such as unit testing, integration testing and system testing.</li></ul>	<ul style="list-style-type: none"><li>• System prototype</li><li>• Software testing result</li></ul>
Deployment	<ul style="list-style-type: none"><li>• Performing user acceptance testing (UAT) to make sure that the developed system can perform designed tasks based on the requirements.</li><li>• Releasing the system into the production environment.</li></ul>	<ul style="list-style-type: none"><li>• Google form for user acceptance testing</li><li>• Deployed system</li></ul>

### **Table 3.1.5 Project Management Framework phases**

#### **3.2 APPLICATION OF MAPPING APIS AND LOCATION SERVICES USING ARCGIS LOCATION SERVICES**

For the Cab Service in Manage Cab module, ArcGIS location services will be used to allow users to directly choose their pick-up and drop-off points on the map. The ArcGIS location services will be using services such as to display map, to search for places, to geocode address as well as performing reverse-geocoding on the locations.

#### **3.3 REVERSE GEOCODING LOCATION USING ESRI LEAFLET**

Reverse geocoding is the process of converting a location to an address or place. To reverse geocode, the geocoding service and the reverseGeocode operation is required. This operation requires an initial location and returns an address with attributes such as place name and location. For this, Esri Leaflet will be used as it provides a geocoder to access the geocoding services.

Reverse geocoding will be applied in the Cab Service of the Manage Cab modules where users can choose their pick-up and drop-off locations directly on the map which will then be reverse geocoded into readable addresses instead of latitude and longitude.

### 3.4 PROJECT REQUIREMENTS

#### 3.4.1 Functional Requirements

Table 3.4.1.1 below shows the functional requirements for each module in the proposed system.

**Table 3.4.1.1 Functional Requirements**

Function Name	Description
Manage Car Rental	<ul style="list-style-type: none"><li>- The system must allow students to use the search and filter functions to find available cars for rental.</li><li>- The system must allow students who own cars to make rental service.</li><li>- The system must calculate the car rental fare based on the different types of car.</li></ul>
Manage Cab Service	<ul style="list-style-type: none"><li>- The system must allow students to see available cab service hotspots on the map and clicks on it to book cab services.</li><li>- The system must measure the distance for the carpool ride from the map.</li><li>- The system must calculate the cab fare based on the distance of the ride.</li></ul>
Manage Car Review	<ul style="list-style-type: none"><li>- The system must allow users to leave ratings and reviews on car rentals and cabs they have rented and ride.</li></ul>
Manage Users	<ul style="list-style-type: none"><li>- The system must allow the admin to modify or remove information about users and cars.</li></ul>

	- The system must allow the admin to approve the car registration request of users who have registered into the system.
Manage Report	- The system must generate report for total users, total car rental and total cab services.

### 3.4.2 Non-Functional Requirements

Table 3.4.2.1 below shows the non-functional requirements for the proposed system.

**Table 3.4.2.1 Non-functional requirements**

Non-Functional Requirements	Description
Performance	The system must handle a large number of users at a time to avoid network traffic.
Security	The system must remain resilient in the event of an attack on the system to ensure the integrity of the students' account information.
Maintainability	The system must have a long lifespan.
Usability	The system must be easy to navigate through.
Localization	The system must have features that can match the students' geographical location.

### 3.4.3 Constraints

The UMPCab consists of a few constraints that limit users' actions when using the system. Constraints are the requirement of the system that restricts the way the system should be developed. It is important as it can help the development team to speed up the

development of the system by preventing challenges during the development process. *Table 3.4.3.1* below shows the constraints in the development of UMPCab.

**Table 3.4.3.1 Constraints of the system**

Type of Constraints	Descriptions
Time	Users can access the system anytime except during maintenance.
Culture	The system must not contain symbols or graphics that could offend any culture.
Security	The system must only allow access of authorized users.
Maintainability	The system must only be maintained outside of office hours.
Scalability	The system must be adapted with the Internet.
Usability	The maximum response time for the system must not exceed 3 seconds.

#### 3.4.4 Limitations

When developing a system, there are some limitations that restricts the system from functioning to its' full potential. *Table 3.4.4.1* shows the limitations of UMPCab.

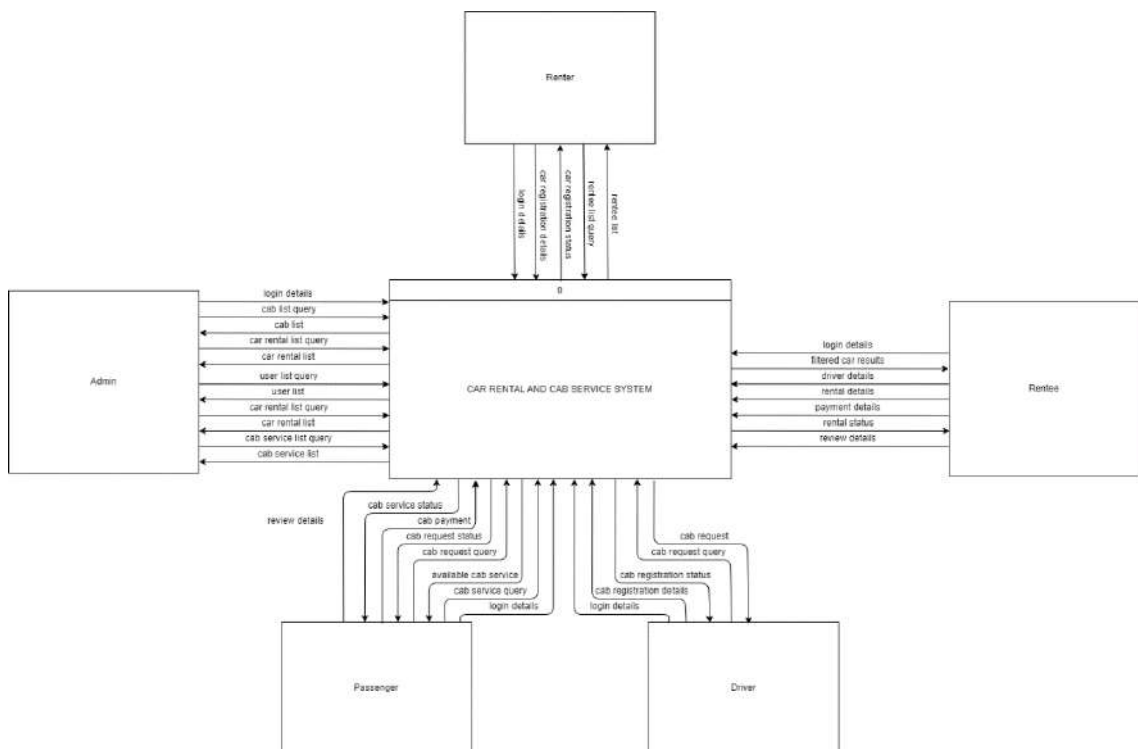
**Table 3.4.4.1 Limitations of the system**

Limitations	Descriptions
Server downtime	The system may be facing server downtime sometimes and this may affect user experiences especially when they are in urgent need to use the system.
Internet connectivity	Since this system is web-based application, users can only access it only with Internet connection.

Browser support	Some browsers may not be able to support this system.
Device support	Some of the system features (user interface design) may be misaligned depending on the device the user is accessing it through.

### 3.5 PROPOSED DESIGN

#### 3.5.1 Context Diagram



**Figure 3.5.1 Context Diagram**

The Car Rental and Cab Service System for UMP Students (UMPCab) is a web-based system developed using Laravel framework. The main purpose for the development of this project is to ease UMP Students in looking for available cars for rental around college. Students can also look for available cab services that are offered around the campus. The stakeholders involved in this system are UMP students and the administrator of the system.



UMP students are divided into 4 which are Renter, Rentee, Driver and Passenger. Renter is UMP students who wants to register their car up for rental in the system. Rentee are UMP students who book for car rental services offered in the system. Driver are UMP students who wants to offer cab service to other students through the system. Passengers are UMP students who book for cab services offered in the system.

This system can help UMP students to reduce their time in looking for available transport to travel around the campus and also around Pahang, mainly in the Pekan and Gambang area. This system will also help students to earn some side incomes by allowing students who own cars to offer their own car rental or cab service. There will be a total of five modules for this system which Manage Car Rental, Manage Car Review, Manage Cab Service, Manage Users and Manage Report.

First of all, both users which are the administrator, and the UMP students are required to login into the system and sign up if they are not yet registered into the system. They will be required to provide their login credentials in order to access the system.

For the Admin, they can view the user list, car rental list and cab service list and make changes to the respective lists. The Admin can also view the car registration list of users who have registered into the system and either approve or reject their car registration request.

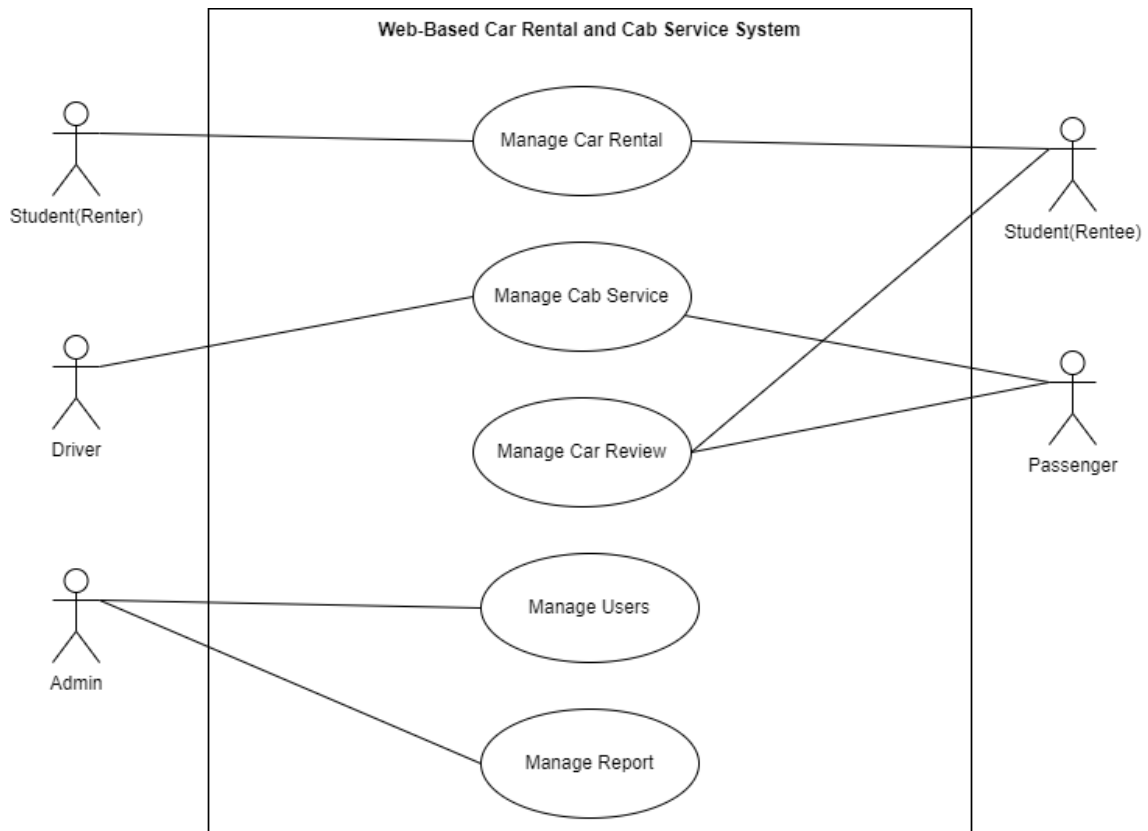
For the Renter, they can register their car up for rental in the system by providing car registration details to the system. Renter is also able to view list of Rentee who have booked their car for rental.

For the Rentee, they can browse through the car rental list by sorting the car rental list based on the car rental fare to the system. The system will then return a list of available car rentals based on their request. The Rentee need to provide the system with renter details and rental details in order to rent any cars in the system. The system will then provide the Rentee with their car rental status. Rentee can also leave reviews on the cars they have rented by providing appropriate review details.

For the Driver, they can register their car up for cab services in the system by providing cab registration details to the system. Driver can also view list of Passengers request to ride a cab based on their pickup and dropoff location and choose which cab ride request they want to accept.

For the Passenger, they can browse through the map for the places they want to go. Passengers can also provide cab service query such as their pickup and drop-off location and then the system will save their journey request to the database. Their cab ride request will then be accepted by any available Drivers nearby. Passengers can also leave reviews on the cabs they have booked and ride by providing appropriate review details.

### 3.5.2 Use Case Diagram



**Figure 3.5.2 Use Case Diagram**

This system consists of five modules which Manage Car Rental, Manage Cab Service, Manage Car Review, Manage Users and Manage Report. The table below shows the description and actors involved for each module.

**Table 3.5.2 Modules Description with Actors Involved**

Modules	Description	Actors Involved
Manage Car Rental	This module allows UMP students (Renter) who own cars to register their car up for rental in the system.  This module also allows Rentee to update and delete their registered car.	Rentee and Renter

		This module also allows UMP students (Rentee) to book for car rental offered in the system.	
Manage Service	Cab	<p>This module allows UMP students (Driver) to register their car for cab services in the system.</p> <p>This module also allows Driver to update and delete their registered cars.</p> <p>This module also allows UMP students (Passenger) to book for cab services offered in the system.</p>	Driver and Passenger
Manage Review	Car	This module allows UMP students (Rentee) to provide reviews and feedbacks for the car rental services they have booked. This module also allows UMP students (Passenger) to provide reviews and feedbacks for the cab services they have ride and completed.	Rentee and Passenger
Manage Users		<p>This module allows the Administrator of the system to monitor the users of the system and also the cars registered into the system.</p> <p>This module also allows the Administrator to view, modify and delete information from the system.</p> <p>This module also allows the Administrator to view a list of car registration request and approve or reject their requests.</p>	Admin
Manage Report		This module allows the Administrator of the system to view the report of the	Admin

	system such as the total number of users in the system and the total number of cars registered into the system.	
--	---	--

### 3.5.3 Use Case Description

#### 3.5.3.1 Manage Car Rental

**Table 3.5.3.1 Manage Car Rental Use Case Description**

<b>Use Case Name</b>	Manage Car Rental
<b>Use Case ID</b>	UMPCab-001
<b>Brief Description</b>	This use case allows the Renter to register their car for rental into the system. This use case also allow Rentee to search and look for available cars for rental.
<b>Actor</b>	Renter and Rentee.
<b>Pre-condition</b>	The users must already be logged in into the system.
<b>Basic Flow</b>	<ol style="list-style-type: none"><li>1) The use case begins when the Renter and Rentee is redirected to the Dashboard after log in.</li><li>2) The Rentee can:<ol style="list-style-type: none"><li>a. Register Cars [<b>Register Car</b>]<ol style="list-style-type: none"><li>1) The Rentee clicks the &lt;&lt;<b>My Car</b>&gt;&gt; button on the sidebar.</li><li>2) The system displays the My Car page.</li><li>3) The Rentee clicks the &lt;&lt;<b>Register Car</b>&gt;&gt; button.</li><li>4) The system displays a modal with Car Registration form.</li><li>5) The Rentee fills in the car registration details and click &lt;&lt;<b>Register</b>&gt;&gt; button.</li><li>6) The system saves the information into the database.</li><li>7) The system displays the car registration status.</li></ol></li></ol></li></ol>

	<p>8) The system displays the registered car in the My Car page.</p> <p>b. Register Cars for Rental [<b>Register Car for Rental</b>]</p> <ol style="list-style-type: none"> <li>1) The Rentee clicks the &lt;&lt;<b>Register for rental</b>&gt;&gt; button.</li> <li>2) The system prompts user to confirm their register car for rental request.</li> <li>3) The Rentee clicks the &lt;&lt;<b>OK</b>&gt;&gt; button.</li> <li>4) The system displays the car rental registration status.</li> </ol> <p>c. View Car Rental Request history [<b>View Rental History</b>]</p> <ol style="list-style-type: none"> <li>1) The Rentee clicks the &lt;&lt;<b>User Request</b>&gt;&gt; button in the sidebar.</li> <li>2) The system retrieves data from the database.</li> <li>3) The system displays the User Request page.</li> <li>4) The Rentee clicks the &lt;&lt;<b>History</b>&gt;&gt; button.</li> <li>5) The system retrieves data from the database.</li> <li>6) The system displays a modal with the list of users who have rented their cars.</li> </ol> <p>d. Edit Car [<b>Edit Car</b>]</p> <ol style="list-style-type: none"> <li>1) The Rentee clicks the &lt;&lt;<b>Edit</b>&gt;&gt; button.</li> <li>2) The system retrieves car data from database.</li> <li>3) The system displays the car details in the Update Car Details page.</li> <li>4) The Rentee edit the existing details of the car.</li> <li>5) The Rentee clicks the &lt;&lt;<b>Update</b>&gt;&gt; button.</li> <li>6) The system verifies request and details.</li> <li>7) The system saves and update data to database.</li> <li>8) The system displays car update status.</li> </ol> <p>e. Delete Car [<b>Delete Car</b>]</p> <ol style="list-style-type: none"> <li>1) The Rentee clicks the &lt;&lt;<b>Delete</b>&gt;&gt; button.</li> <li>2) The system prompts user to confirm delete car request.</li> </ol>
--	---

	<ol style="list-style-type: none"> <li>3) The Rentee clicks the &lt;&lt;<b>OK</b>&gt;&gt; button.</li> <li>4) The system verifies delete request.</li> <li>5) The system delete data from database.</li> <li>6) The system displays car deletion status.</li> </ol> <p>3) The Rentee can:</p> <ol style="list-style-type: none"> <li>a. Browse and book for car rental [<b>Rent Cars</b>] <ol style="list-style-type: none"> <li>1) The Rentee clicks on the &lt;&lt;<b>Book Car for Rental</b>&gt;&gt; button.</li> <li>2) The system retrieves a list of available cars for rental from the database.</li> <li>3) The system displays the list of available cars for rental.</li> <li>4) The Rentee clicks on the &lt;&lt;<b>View</b>&gt;&gt; button on any car of their choice.</li> <li>5) The system retrieves car details from the database.</li> <li>6) The system displays the car details.</li> <li>7) The system displays car rental booking form.</li> <li>8) The Renter fills in rental and renter details.</li> <li>9) The Rentee clicks on the &lt;&lt;<b>Book This Car</b>&gt;&gt; button.</li> <li>10) The system saves the information into the database and displays car rental status.</li> <li>11) The system displays car rental booking status.</li> <li>12) The system displays the My Bookings list.</li> </ol> </li> <li>4) The use case ends.</li> </ol>
<b>Alternative Flow</b>	None
<b>Exception Flow</b>	None



<b>Rules</b>	None
<b>Constraints</b>	None
<b>Post Condition</b>	The system successfully displays the car registration status and car rental booking status.

### 3.5.3.2 Manage Cab Service

**Table 3.5.3.2 Manage Cab Service Use Case Description**

<b>Use Case Name</b>	Manage Cab Service
<b>Use Case ID</b>	UMPCab-002
<b>Brief Description</b>	This use case allows the Driver to register their cars for cab service in the system. This use case also allows Passenger to look for available cab services.
<b>Actor</b>	Driver and Passenger.
<b>Pre-condition</b>	The users must already be logged in into the system.
<b>Basic Flow</b>	<ol style="list-style-type: none"> <li>1) The use case begins when the Driver and Passenger is redirected to the Dashboard after log in.</li> <li>2) The Driver can: <ol style="list-style-type: none"> <li>a. Register cars [<b>Register Car</b>] <ol style="list-style-type: none"> <li>1) The Driver clicks the &lt;&lt;My Car&gt;&gt; button on the sidebar.</li> <li>2) The system displays the My Car Page.</li> <li>3) The Driver clicks the &lt;&lt;Register Car&gt;&gt; button.</li> </ol> </li> </ol> </li> </ol>

	<ol style="list-style-type: none"> <li>4) The system displays a modal with Car Registration form.</li> <li>5) The Driver fills in the car registration details and click &lt;&lt;<b>Register</b>&gt;&gt; button.</li> <li>6) The system saves the information into the dataset.</li> <li>7) The system displays the car registration status.</li> <li>8) The system displays the registered car in the My Car page.</li> </ol> <p>b. Register Cars for Cab Service [<b>Register Car for Cab Service</b>]</p> <ol style="list-style-type: none"> <li>1) The Driver clicks the &lt;&lt;<b>Register for carpool</b>&gt;&gt; button.</li> <li>2) The system prompts user to confirm their register car for carpool request.</li> <li>3) The Driver clicks the &lt;&lt;<b>OK</b>&gt;&gt; button.</li> <li>4) The system displays the car rental registration status.</li> </ol> <p>c. Accept Cab Request [<b>Accept Cab Request</b>]</p> <ol style="list-style-type: none"> <li>1) The Driver clicks the &lt;&lt;<b>User Requests</b>&gt;&gt; button in the sidebar.</li> <li>2) The system retrieves car requests from database.</li> <li>3) The system displays cab request list in the User Request page.</li> <li>4) The Driver clicks the &lt;&lt;<b>View Request</b>&gt;&gt; button.</li> <li>5) The system retrieves cab request details from the database.</li> <li>6) The system displays a modal with cab request details of chosen cab request.</li> <li>7) The Driver clicks the &lt;&lt;<b>Accept Request</b>&gt;&gt; button.</li> <li>8) The system updates and save data of status of cab request to the database.</li> <li>9) The system displays the cab request page.</li> </ol>
--	---

	<p>10) The Driver clicks the &lt;&lt;<b>Ride Completed</b>&gt;&gt; button.</p> <p>11) The system updates and save the status of the cab ride to the database.</p> <p>12) The system displays status of cab ride.</p> <p>13) The system displays the User Request page.</p> <p>d. <b>Edit Car [Edit Car]</b></p> <ol style="list-style-type: none"> <li>1) The Driver clicks the &lt;&lt;<b>Edit</b>&gt;&gt; button.</li> <li>2) The system retrieves car data from database.</li> <li>3) The system displays the car details in the Update Car Details page.</li> <li>4) The Driver edits the existing details of the car.</li> <li>5) The Driver clicks the &lt;&lt;<b>Update</b>&gt;&gt; button.</li> <li>6) The system verifies request and details.</li> <li>7) The system saves and update data to database.</li> <li>8) The system displays car update status.</li> </ol> <p>e. <b>Delete Car [Delete Car]</b></p> <ol style="list-style-type: none"> <li>1) The Driver clicks the &lt;&lt;<b>Delete</b>&gt;&gt; button.</li> <li>2) The system prompts user to confirm delete car request.</li> <li>3) The Driver clicks the &lt;&lt;<b>OK</b>&gt;&gt; button.</li> <li>4) The system verifies delete request.</li> <li>5) The system delete data from database.</li> <li>6) The system displays car deletion status.</li> </ol> <p>f. <b>View Cab Request History [View Cab History]</b></p> <ol style="list-style-type: none"> <li>1) The Driver clicks the &lt;&lt;<b>User Request</b>&gt;&gt; button in the sidebar.</li> <li>2) The system retrieves data from the database.</li> <li>3) The system displays the User Request page.</li> <li>4) The Driver clicks the &lt;&lt;<b>History</b>&gt;&gt; button.</li> <li>5) The system retrieves data from the database.</li> <li>6) The system displays a modal with the a list of cab request they have completed.</li> </ol> <p>3) The Passenger can:</p>
--	---

	<p>a. Book Cab Service [<b>Book Cab</b>]</p> <ol style="list-style-type: none"> <li>1) The Passenger clicks on the &lt;&lt;<b>Book a carpool</b>&gt;&gt; button.</li> <li>2) The system displays a map with cab service request form.</li> <li>3) The Passenger fill in the cab service request form. [<b>A1: Passenger search location</b>]</li> <li>4) The Passenger clicks on &lt;&lt;<b>Submit</b>&gt;&gt; button.</li> <li>5) The system saves the information into the database.</li> <li>6) The system displays cab request status.</li> <li>7) The system displays the My Bookings List.</li> </ol> <p>4) The use case ends.</p>
<b>Alternative Flow</b>	<p><b>A1: Passenger enter location</b></p> <ol style="list-style-type: none"> <li>1) The Passenger fills in the location details and clicks the &lt;&lt;<b>Search</b>&gt;&gt; button.</li> <li>2) The system displays a marker based on the search results.</li> <li>3) The use case continues to Basic Flow step (3)(a)(4).</li> </ol>
<b>Exception Flow</b>	None
<b>Rules</b>	None
<b>Constraints</b>	None
<b>Post Condition</b>	The system successfully displays the cab registration and cab booking status.

### 3.5.3.3 Manage Car Review

**Table 3.5.3.3 Manage Car Review Use Case Description**

<b>Use Case Name</b>	Manage Car Review
<b>Use Case ID</b>	UMPCab-003
<b>Brief Description</b>	This use case allows Passenger to give review on the cab service they have ride on. This use case also allows Rentee to give review on the car they have rented.
<b>Actor</b>	Passenger and Rentee
<b>Pre-condition</b>	The users must already be logged in into the system.
<b>Basic Flow</b>	<ol style="list-style-type: none"> <li>1) The use case begins when the Passenger and Rentee clicks on the &lt;&lt;<b>My Booking</b>&gt;&gt; button in the sidebar.</li> <li>2) The system retrieves a list of car rental and cab service bookings from the database.</li> <li>3) The system displays a list of car rental and cab service bookings.</li> <li>4) The Rentee can: <ol style="list-style-type: none"> <li>a. Give Review for Car Rental [<b>Review Car Rental</b>] <ol style="list-style-type: none"> <li>1) The Rentee clicks on the &lt;&lt;<b>Ongoing</b>&gt;&gt; button in the bookings list with the &lt;&lt;<b>Car Rental</b>&gt;&gt; booking type.</li> <li>2) The system retrieves car rental details from the database.</li> <li>3) The system displays the car rental details with a form for review details.</li> <li>4) The Rentee fills in the car rental review form and clicks the &lt;&lt;<b>Complete Review</b>&gt;&gt; button.</li> <li>5) The system verifies user request and shows pop-up for confirmation of car rental completion status.</li> </ol> </li> </ol> </li> </ol>

	<ol style="list-style-type: none"> <li>6) The Rentee clicks on the &lt;&lt;<b>Confirm</b>&gt;&gt; button.</li> <li>7) The system updates the completion status to the database.</li> <li>8) The system displays car rental completion status.</li> </ol> <p>5) The Passenger can:</p> <ol style="list-style-type: none"> <li>a. Give Review for Cab Service [<b>Review Cab Service</b>] <ol style="list-style-type: none"> <li>1) The Passenger clicks on the &lt;&lt;<b>Rate Car Ride</b>&gt;&gt; button in the bookings list with the &lt;&lt;<b>Cab Service</b>&gt;&gt; booking type.</li> <li>2) The system retrieves cab service details from the database.</li> <li>3) The system displays the cab service details with a form for review details.</li> <li>4) The Passenger fills in the cab service review form and clicks the &lt;&lt;<b>Complete Review</b>&gt;&gt; button.</li> <li>5) The system verifies user request and shows pop-up for confirmation of cab service completion status.</li> <li>6) The Passenger clicks on the &lt;&lt;<b>Confirm</b>&gt;&gt; button.</li> <li>7) The system updates the cab service completion status to the database.</li> <li>8) The system displays cab service completion status.</li> </ol> </li> </ol> <p>6) The use case ends.</p>
<b>Alternative Flow</b>	None
<b>Exception Flow</b>	None
<b>Rules</b>	None

<b>Constraints</b>	None
<b>Post Condition</b>	The system successfully saved users review and updated car rental and cab service completion status to the database.

### 3.5.3.4 Manage Users

**Table 3.5.3.4 Manage Users Use Case Description**

<b>Use Case Name</b>	Manage Users
<b>Use Case ID</b>	UMPCab-004
<b>Brief Description</b>	This use case allows Admin to monitor the users of the system by giving them access to view, modify and delete information about the users and the cars.
<b>Actor</b>	Admin
<b>Pre-condition</b>	The user must already be logged in into the system.
<b>Basic Flow</b>	<ol style="list-style-type: none"> <li>1) The use case begins when the Admin is redirected to the dashboard after login.</li> <li>2) The Admin can: <ol style="list-style-type: none"> <li>a. Edit User List [<b>Edit User List</b>] <ol style="list-style-type: none"> <li>1) The Admin clicks on the &lt;&lt;<b>View User List</b>&gt;&gt; button.</li> <li>2) The system retrieves data from database.</li> <li>3) The system displays a list of users.</li> <li>4) The Admin chooses any existing record in the list and clicks &lt;&lt;<b>Update</b>&gt;&gt; button.</li> <li>5) The system retrieves data from the database.</li> </ol> </li> </ol> </li> </ol>

	<ol style="list-style-type: none"><li>6) The system displays user details.</li><li>7) The Admin edits the details and clicks on &lt;&lt;<b>Update</b>&gt;&gt; button.</li><li>8) The system updates the information to the database.</li><li>9) The system displays update status.</li></ol> <p>b. Delete User [<b>Delete User</b>]</p> <ol style="list-style-type: none"><li>1) The Admin clicks on the &lt;&lt;<b>View User List</b>&gt;&gt; button.</li><li>2) The system displays a list of users.</li><li>3) The Admin chooses any existing record in the list and clicks &lt;&lt;<b>Delete</b>&gt;&gt; button.</li><li>4) The system verifies user request to delete data.</li><li>5) The system remove the data from the database.</li><li>6) The system displays delete status.</li></ol> <p>c. Edit Car List [<b>Edit Car</b>]</p> <ol style="list-style-type: none"><li>1) The Admin clicks on the &lt;&lt;<b>View Car Rental List</b>&gt;&gt; button.</li><li>2) The system retrieves data from the database.</li><li>3) The system displays a list of car rentals registered into the system.</li><li>4) The Admin chooses any existing record in the list and clicks &lt;&lt;<b>Update</b>&gt;&gt; button.</li><li>5) The system retrieves data from the database.</li><li>6) The system displays car rental details.</li></ol>
--	--



- 7) The Admin edits the details and clicks on <<**Update**>> button.
- 8) The system updates the information to the database.
- 9) The system displays update status.

d. Delete Car [**Delete Car**]

- 1) The Admin clicks on the <<**View cars**>> button.
- 2) The system displays a list of cars registered into the system.
- 3) The Admin chooses any existing record in the list and clicks <<**Delete**>> button.
- 4) The system verifies user request to delete data.
- 5) The system remove the data from the database.
- 6) The system displays delete status.

e. Approve car registration request [**Manage Car**]

- 1) The Admin clicks the <<**Manage Car**>> button in the sidebar.
- 2) The system retrieves car registration request from database.
- 3) The system displays car registration request list in the Car Registration List page.
- 4) The Admin clicks the <<**Approve**>> button.  
**[A1: Admin Rejects Car Registration Request]**
- 5) The system verifies approval and saves data inside database.
- 6) The system displays car registration approval status.

	3) The use case ends.
<b>Alternative Flow</b>	<p><b>A1: Admin Rejects Car Registration Request [Reject Car]</b></p> <ol style="list-style-type: none"> <li>1) The Admin clicks the &lt;&lt;<b>Reject</b>&gt;&gt; button.</li> <li>2) The system prompts Admin to confirm car registration rejection.</li> <li>3) The Admin clicks the &lt;&lt;<b>OK</b>&gt;&gt; button.</li> <li>4) The system verifies car rejection request.</li> <li>5) The system saves data to the database and display car rejection status.</li> <li>6) The use case continues to Basic Flow step (2)(e)(3).</li> </ol>
<b>Exception Flow</b>	None
<b>Rules</b>	None
<b>Constraints</b>	None
<b>Post Condition</b>	The system successfully saved and updated the user details, car details and car registration status into the database.

### 3.5.3.5 Manage Report

**Table 3.5.3.5 Manage Report Use Case Description**

<b>Use Case Name</b>	Manage Report
<b>Use Case ID</b>	UMPCab-005
<b>Brief Description</b>	This use case allows Admin to view the report about the system.
<b>Actor</b>	Admin

<b>Pre-condition</b>	The user must already be logged in into the system.
<b>Basic Flow</b>	<ol style="list-style-type: none"> <li>1) The use case begins when the Admin is redirected to the dashboard after login.</li> <li>2) The system displays the Report page.</li> <li>3) The system retrieves data from the database.</li> <li>4) The system displays the report for total users and total cars registered into the system.</li> <li>5) The use case ends.</li> </ol>
<b>Alternative Flow</b>	None
<b>Exception Flow</b>	None
<b>Rules</b>	None
<b>Constraints</b>	None
<b>Post Condition</b>	The system successfully displays the total users and total cars registered into the system.



## 3.6 DATA DESIGN

### 3.6.1 ERD



Figure 3.6.1 Entity Relationship Diagram (ERD)

### 3.6.2 Data Dictionary

#### User Table

**Table 3.6.2.1 User Data Dictionary**

<b>Data Name</b>	<b>Description</b>	<b>Data Type</b>	<b>Constraint</b>
id	ID for user	INT	PK
name	Users name	STRING	
dob	Users date of birth	STRING	
gender	Users gender	STRING	
email	Users email	STRING	
phone_num	Users phone number	STRING	
usertype	Type of user either 0 or 1 where:  0 => Regular user  1 => Administrator	STRING	
password	Users password	STRING	

## CarRental Table

Table 3.6.2.2 Car Rental Data Dictionary

Data Name	Description	Data Type	Constraint
id	ID for car rental	INT	PK
car_id	ID for car	INT	FK
user_id	ID for user	INT	FK
drivers_license	Renter's drivers license	STRING	
ic	Renter's ic	STRING	
renter_name	Renter's name	STRING	
renter_email	Renter's email	STRING	
renter_phone_num	The renter's phone number	STRING	
rental_status	The status of the car rental either <b>"Ongoing"</b> or <b>"Completed"</b>	STRING	
rental_duration	The time duration between the start date and end date of the car rental	INT	

start_date	The start date for the car rental booking	DATETIME	
end_date	The end date for the car rental booking	DATETIME	
rental_amount	The total rental fare	DECIMAL	
created_at	The timestamps where the car rental booking is made	TIMESTAMPS	

## Cab Table

**Table 3.6.2.3 Cab Service Data Dictionary**

<b>Data Name</b>	<b>Description</b>	<b>Data Type</b>	<b>Constraint</b>
id	ID for cab	INT	PK
car_id	ID for car	INT	FK
user_id	ID for user	INT	FK
passenger_name	Passenger's name	STRING	
passenger_email	Passenger's email	STRING	
passenger_phone_num	Passenger's phone number	STRING	



pickup_location	Pick-up location for cab ride	STRING	
pickuflatlng	Latitude and longitude for pick-up location	STRING	
dropoff_location	Drop-off location for cab ride	STRING	
dropofflatlng	Latitude and longitude for drop-off location	STRING	
status	<p>Cab booking status which are <b>“Waiting”</b>, <b>“Accepted”</b> and <b>“Completed”</b>.</p> <p><b>Waiting</b> =&gt; When the passenger has made a cab ride request and is waiting for a driver to accept their request</p> <p><b>Accepted</b> =&gt; When a driver has accepted a passenger’s cab ride request</p> <p><b>Completed</b> =&gt; When the driver has completed the cab ride request</p>	STRING	

total_distance	The total distance between the pick-up and drop-off location in KM	DOUBLE	
total_cab_fare	The total fare for the cab ride	DOUBLE	
created_at	The timestamps when the cab booking is made.	TIMESTAMPS	

### Car Table

**Table 3.6.2.4 Car Data Dictionary**

<b>Data Name</b>	<b>Description</b>	<b>Data Type</b>	<b>Constraint</b>
id	ID for car	INT	PK
user_id	ID for user	INT	FK
drivers_license	User's (car owner) drivers license	STRING	
ic	User's (car owner) ic	STRING	
car_color	Car's color	STRING	
car_model	Car's model	STRING	

cab_fare	Car's fare per KM (if registered for carpool)	DOUBLE	
rental_fare	Car's fare per hour (if registered for rental)	DOUBLE	
rental	Car's rental registration status which can either be <i>NULL</i> or " <b>Car Rental</b> "	STRING	
cab	Car's carpool registration status which can either be <i>NULL</i> or " <b>Cab Service</b> "	STRING	
car_image	Car's image	STRING	
plate_num	Car's plate number	STRING	
reg_status	Car's registration status which are "Waiting for Approval", "Approved" and "Rejected"	STRING	
status	Car's status if they are booked for rental or not booked. The status are:	STRING	

	<p><b>“Booked”</b> =&gt; If the car is either booked for rental or carpool</p> <p><b>“Vacant”</b> =&gt; If the car is not booked for rental nor carpool</p>		
created_at	The timestamps when the car is registered	TIMESTAMPS	

### Rating Table

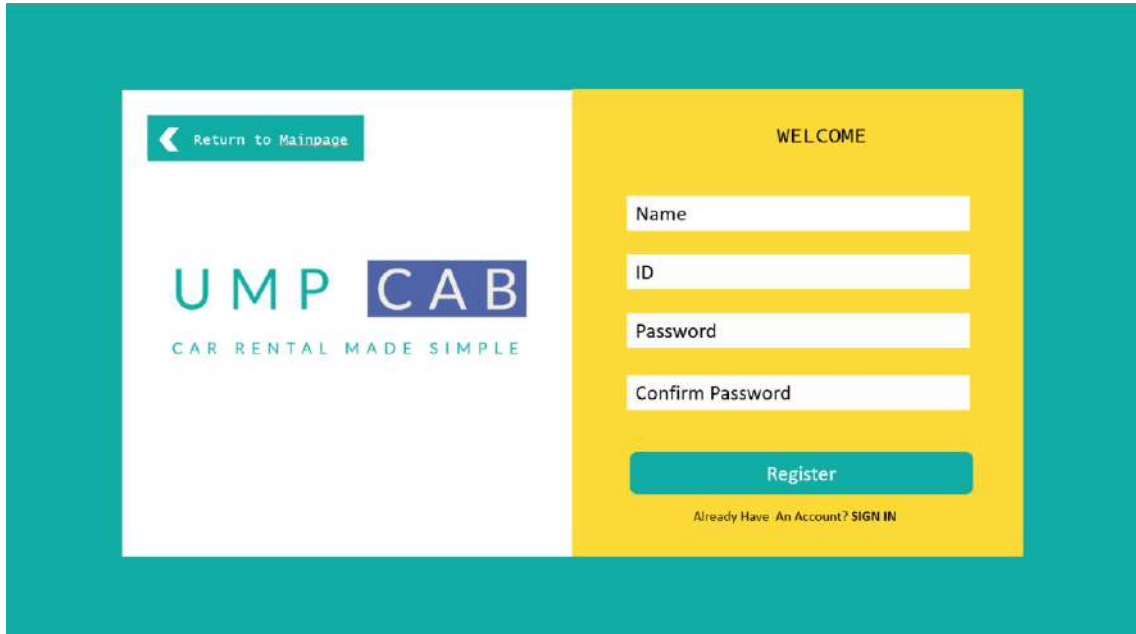
**Table 3.6.2.5 Rating Data Dictionary**

<b>Data Name</b>	<b>Description</b>	<b>Data Type</b>	<b>Constraint</b>
id	ID for rating	INT	PK
car_id	ID for car	INT	FK
user_id	ID for user	INT	FK
rating	Star rating given to cars	STRING	
feedback	Feedbacks given to cars	LONGTEXT	
created_at	The timestamp when the rating and feedback is given	TIMESTAMPS	

## 3.7 PROOF OF INITIAL CONCEPT

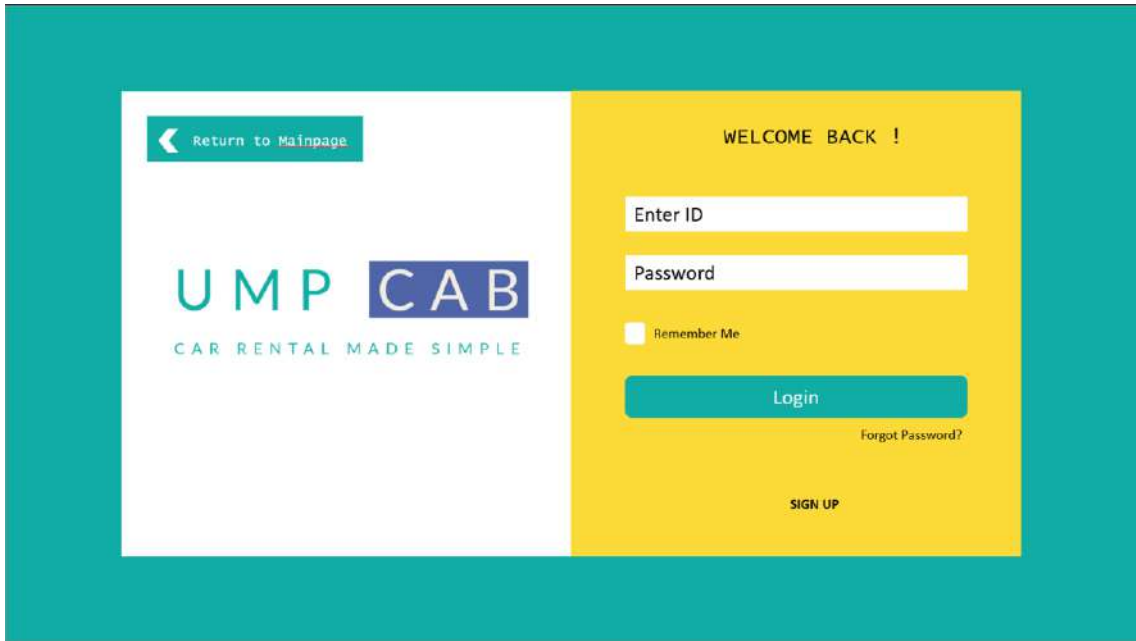
### 3.7.1 Register and Login

The Register interface is for new users who wants to use the system. The users will be required to enter their credentials such as Name, ID, Password and Confirm Password to register into the system. The figure below shows the Register interface.

The image shows a web interface for registering a new user. On the left, there is a white panel with a teal header containing a back arrow and the text "Return to Mainpage". Below this is the logo for "UMP CAB" with the tagline "CAR RENTAL MADE SIMPLE". On the right, there is a yellow panel with the heading "WELCOME". It contains four input fields labeled "Name", "ID", "Password", and "Confirm Password". Below these fields is a teal "Register" button. At the bottom of the yellow panel, there is a link that says "Already Have An Account? SIGN IN".

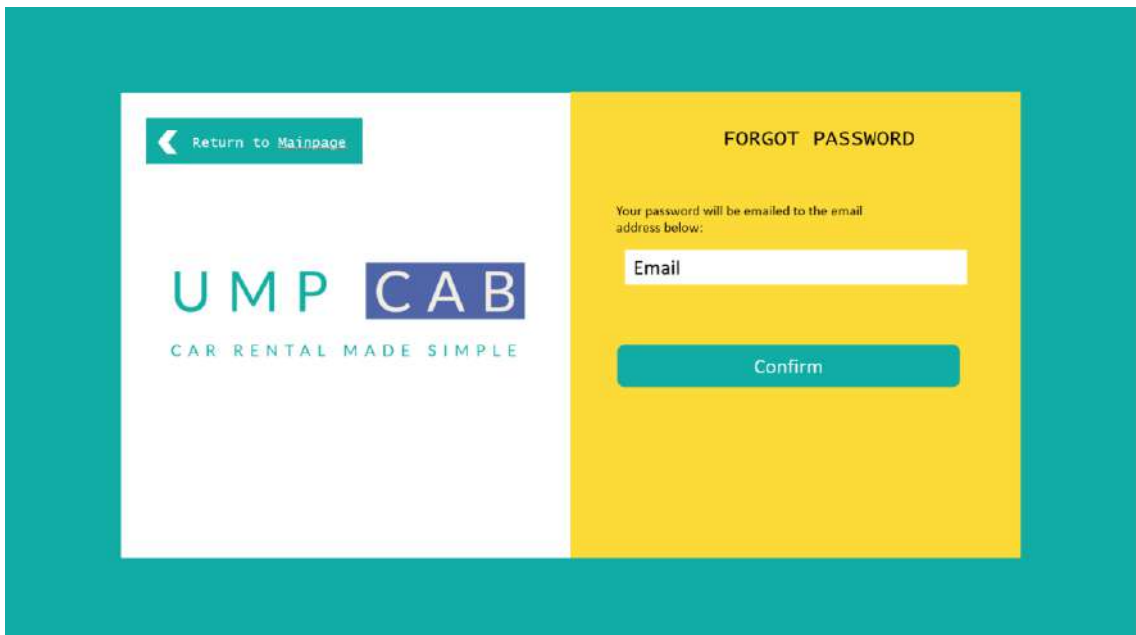
**Figure 3.7.1.1 Register Interface**

The Login interface is for existing system to login into the system to access it. Users are required to enter credentials such as their ID and password. The figure below shows the Login interface.



**Figure 0.1 Login Interface**

In any case the users forget their password for the system, they can always access the Forgot Password interface through the Login interface. Users will be prompted to enter their email address in order to reset their password. The figure below shows the Forgot Password interface.



## Figure 0.2 Forgot Password Interface

### 3.7.2 Mainpage

The Mainpage is the first page that users will see when they access the system. The Mainpage consists of buttons to allows users to Register Car, Browse for Car Rental services, Register Cab and Browse for Cab Services. The figure below shows the Mainpage of the system.

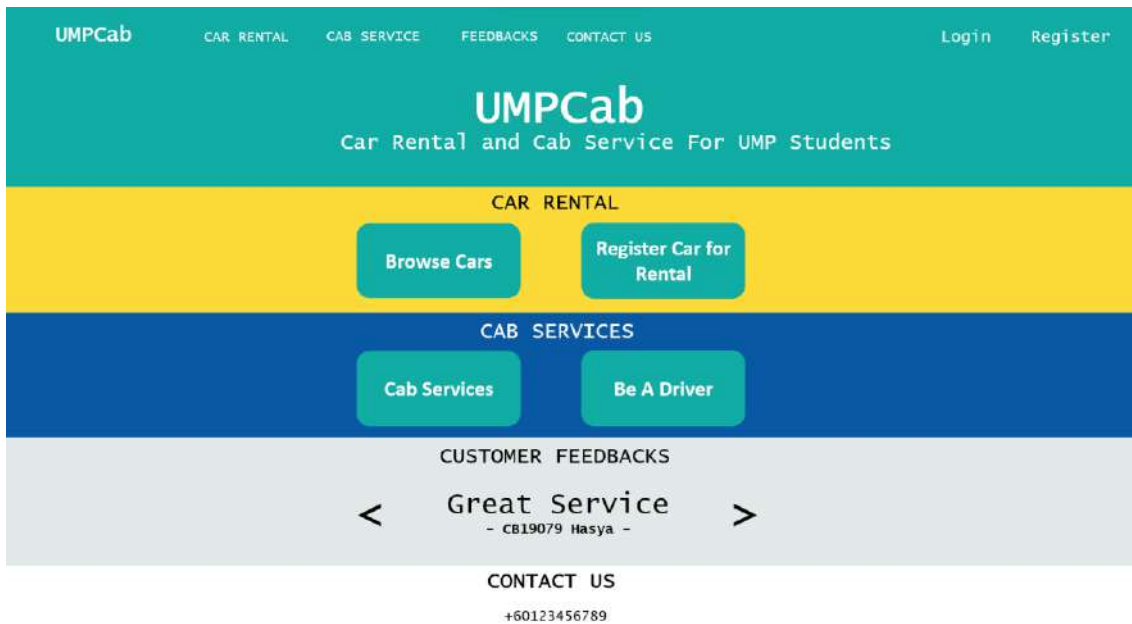
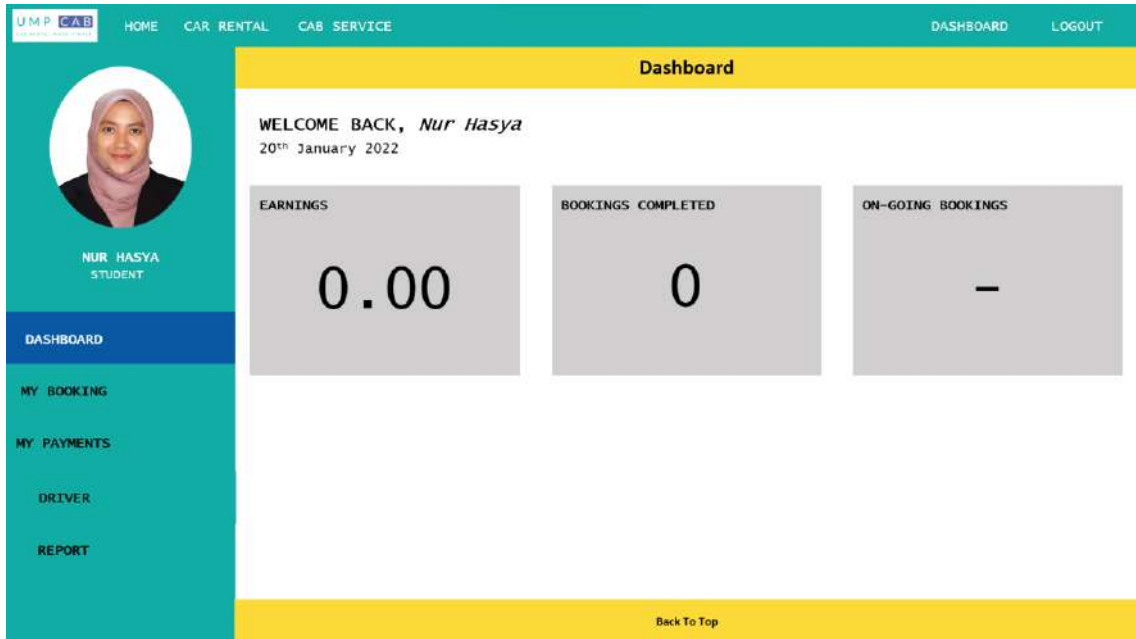


Figure 3.7.2.1 Mainpage Interface

### 3.7.3 Dashboard

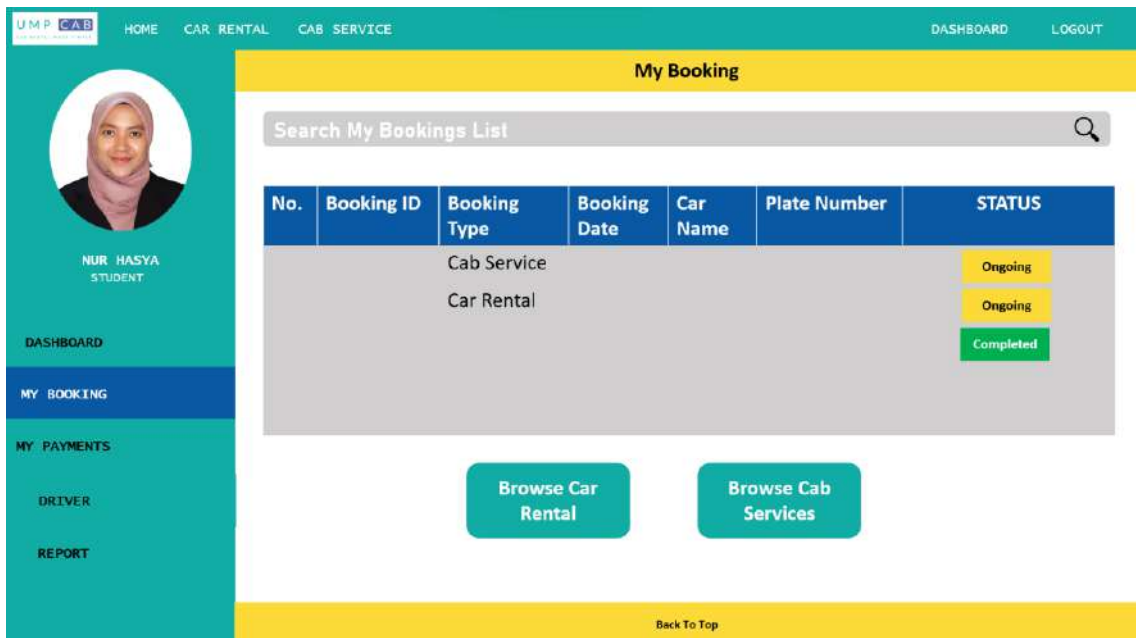
After login and registration, all users will be redirected to the Dashboard interface as shown in the figure below. In the Dashboard interface, users can view their earnings (if they have registered cars or cab in the system), their total number of bookings completed (if they have registered cars or cab in the system) and their on-going bookings (if they have registered cars or cab in the system).



**Figure 3.7.3.1 Dashboard Interface**

### 3.7.4 My Booking

For the My Booking interface, users can view a list of their completed and ongoing bookings for both car rental and cabs. The figure below shows the My Booking interface.

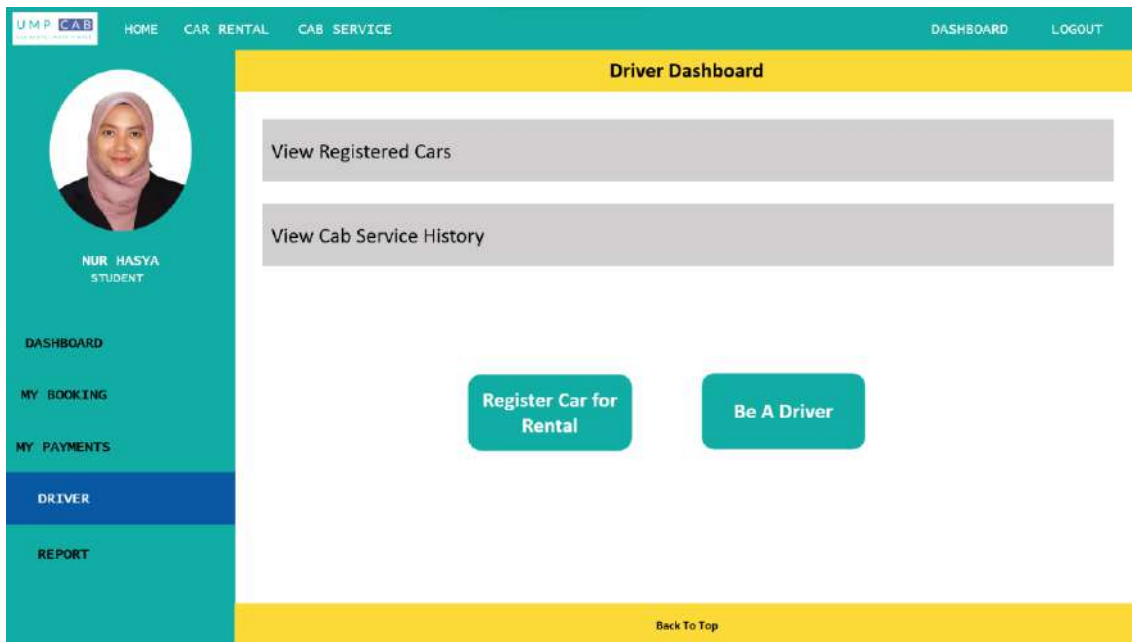




### Figure 3.7.4.1 My Booking Interface

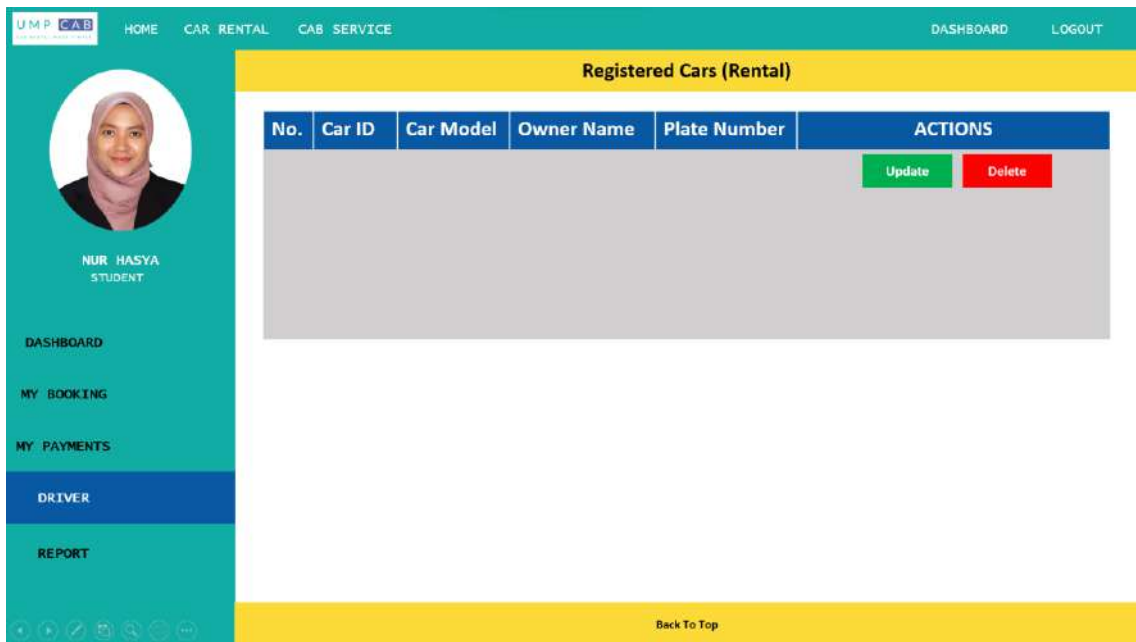
#### 3.7.5 Driver Dashboard

The Driver Dashboard are for users who have cars or cabs registered into the system. In this interface, both Renters and Drivers can view the car and cab they have registered into the system. The figure below shows the Driver Dashboard interface.



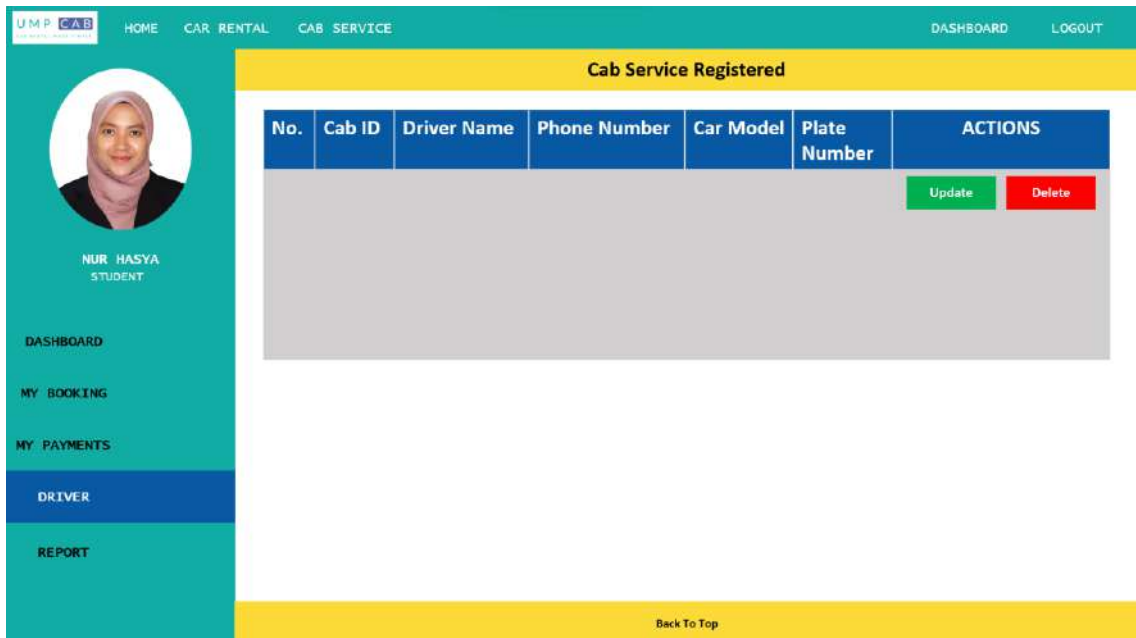
#### Figure 3.7.5.1 Driver Dashboard Interface

When the Driver/Renter clicks on the View Registered Cars, they will be redirected to the Registered Cars interface. The Registered Cars interface is as shown in the figure below.



**Figure 3.7.5.2 Registered Cars Interface**

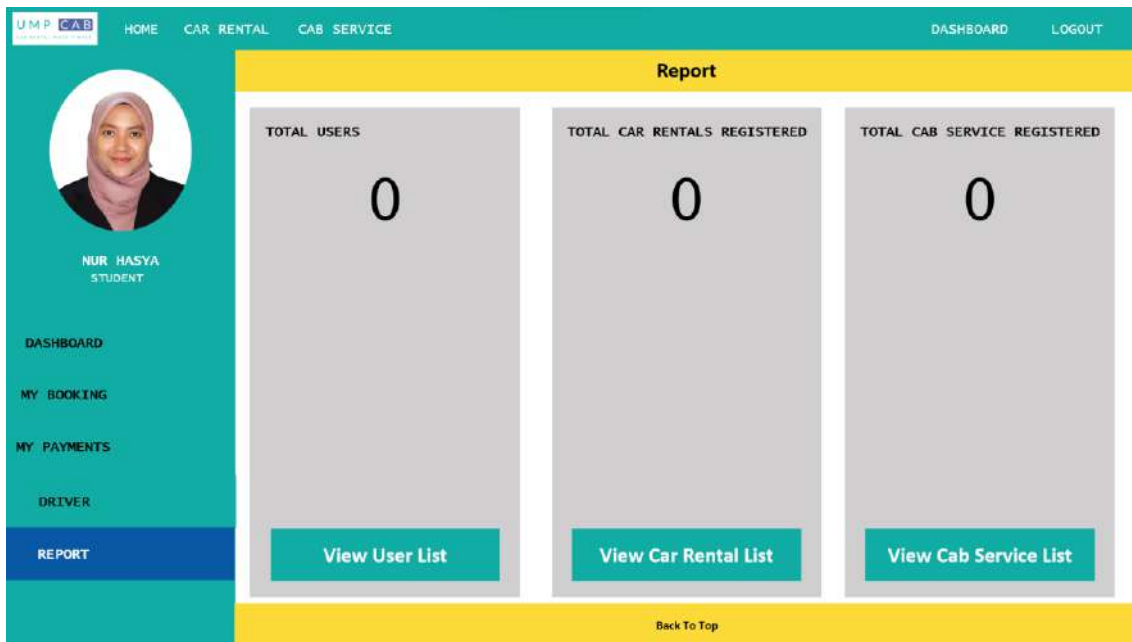
When the Driver/Renter clicks on the View Cab Service History, they will be redirected to the Cab Service Registered interface. The Cab Service Registered interface is as shown in the figure below.



**Figure 3.7.5.3 Cab Service Registered Interface**

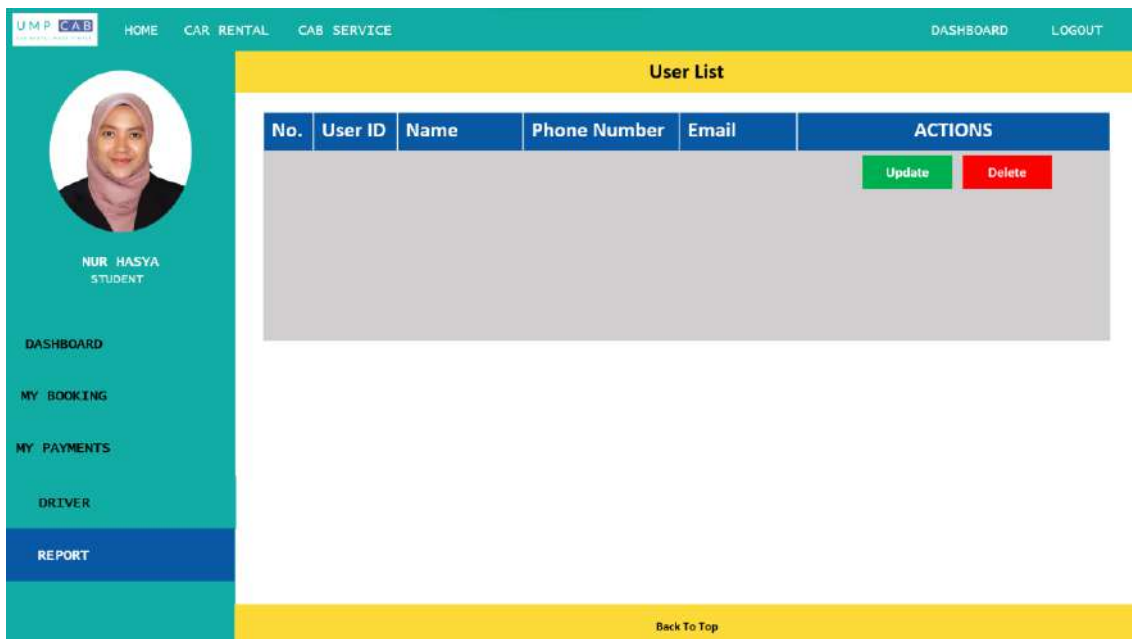
### 3.7.6 Report

For the Report interface, only Admin can access it to view the report about the system and also manage the users, car rentals and cabs registered into the system. The figure below shows the Report interface.



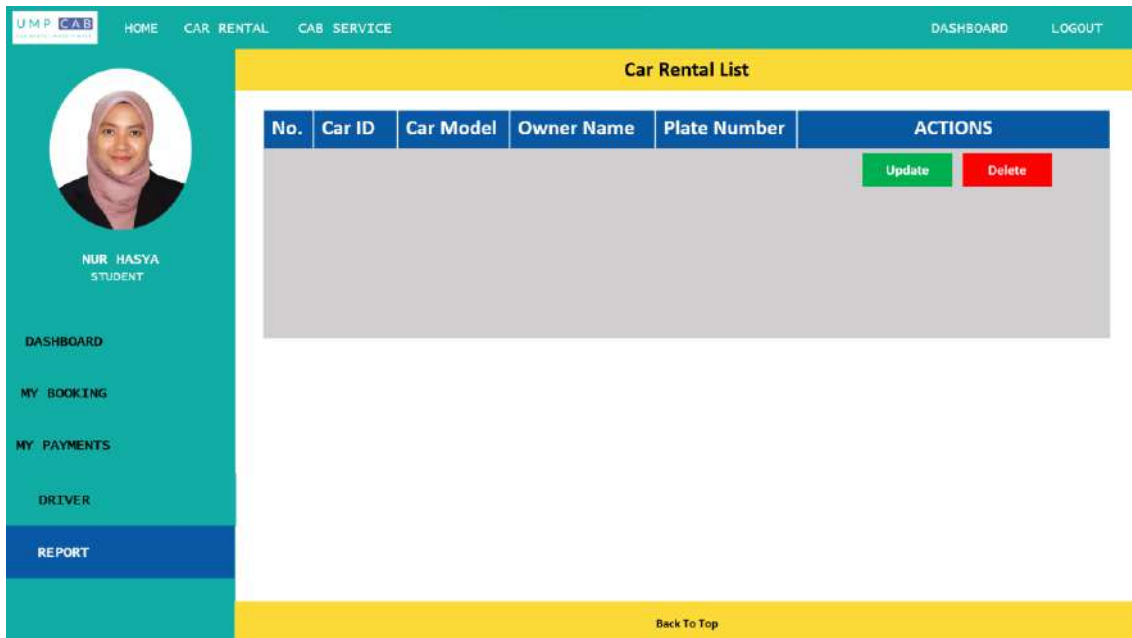
**Figure 3.7.6.1 Report Interface**

When the Admin clicks on the View User List, they will be redirected to the User List interface. The User List interface is as shown in the figure below.



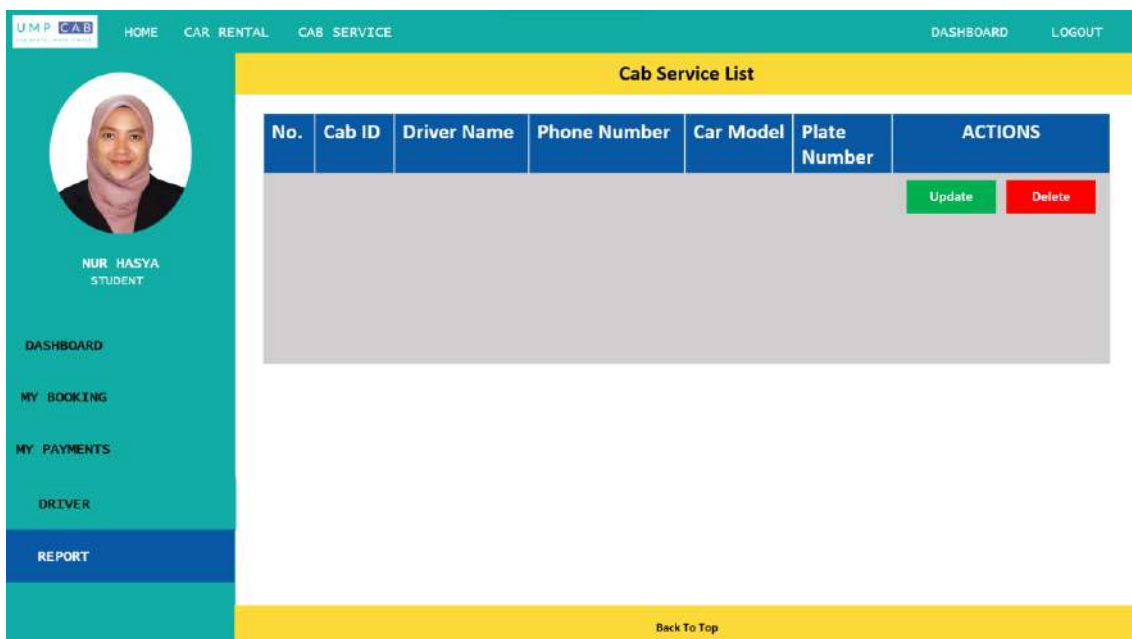
### Figure 3.7.6.1 User List Interface

When the Admin clicks on the View Car Rental List, they will be redirected to the Car Rental List interface. The Car Rental List interface is as shown in the figure below.



### Figure 3.7.6.3 Car Rental List Interface

When the Admin clicks on the View Cab Service List, they will be redirected to the Cab Service List interface. The Cab Service List interface is as shown in the figure below.



### Figure 3.7.6.4 Cab Service List Interface

#### 3.7.7 Car Registration

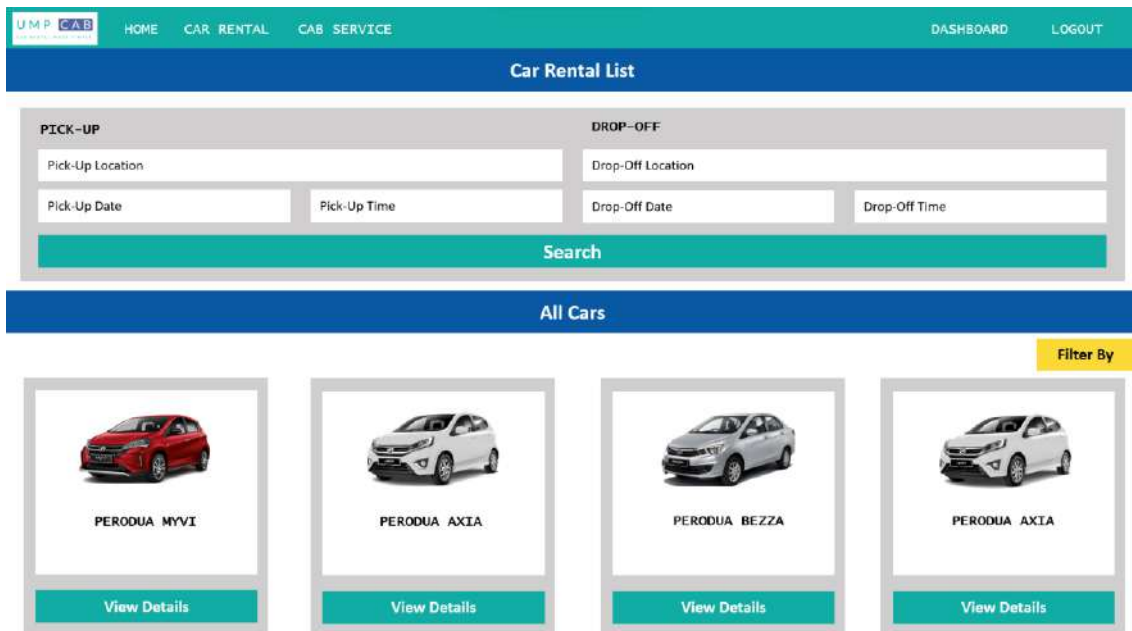
For users who want to register their car for rental, they will be redirected to the Register Car for Rental interface. The users will be required to provide their credentials for the car rental registration. The Register Car for Rental interface is as shown in the figure below.

The screenshot shows a web interface for registering a car for rental. At the top, there is a teal navigation bar with the 'U.M.P. CAB' logo on the left and links for 'HOME', 'CAR RENTAL', 'CAB SERVICE', 'DASHBOARD', and 'LOGOUT' on the right. Below this is a yellow banner with the text 'Register Car for Rental'. The main content area has a light gray background and is titled 'CAR DETAILS'. It contains a form with several input fields: 'DRIVERS LICENSE' with an 'Upload Drivers License' button, 'OWNER NAME', 'CAR MODEL' with a dropdown menu showing 'Choose Car Model', 'CAR IMAGE', 'PLATE NUMBER', and 'PAYMENT RATE PER HOUR'. A blue 'REGISTER' button is positioned at the bottom right of the form area.

Figure 3.7.7.1 Register Car for Rental Interface

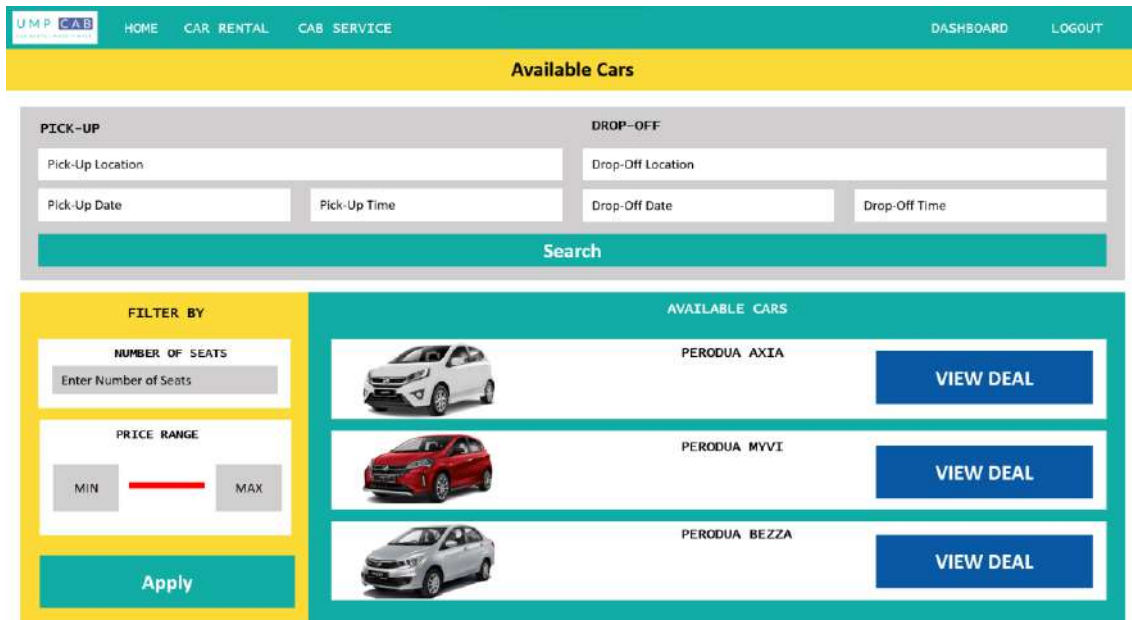
#### 3.7.8 Car Rental Booking

For the car rental booking, users will first browse through a list of available cars for rental in the Car Rental Homepage interface. In this interface, users can also search for car rental services based on the search details they enter. The Car Rental Homepage interface is as shown in the figure below.



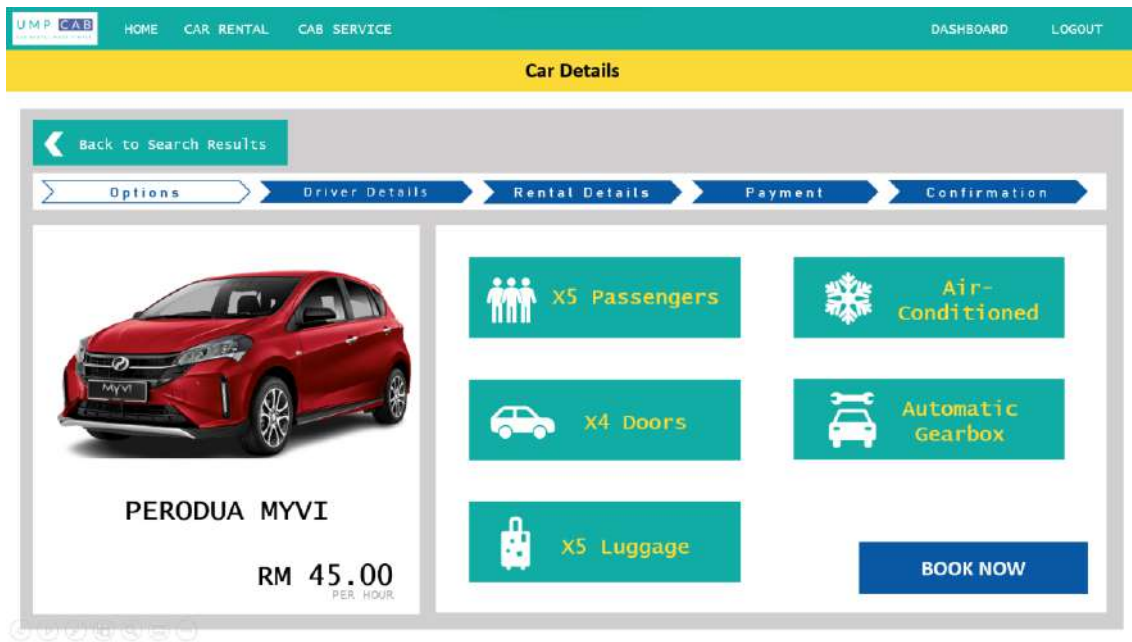
**Figure 3.7.8.1 Car Rental Homepage Interface**

When the user clicks on the Search button, the system will display a list of available cars for rental based on their search results. Users can also filter the search results according to their own preferences. The figure below shows the Search Result interface.



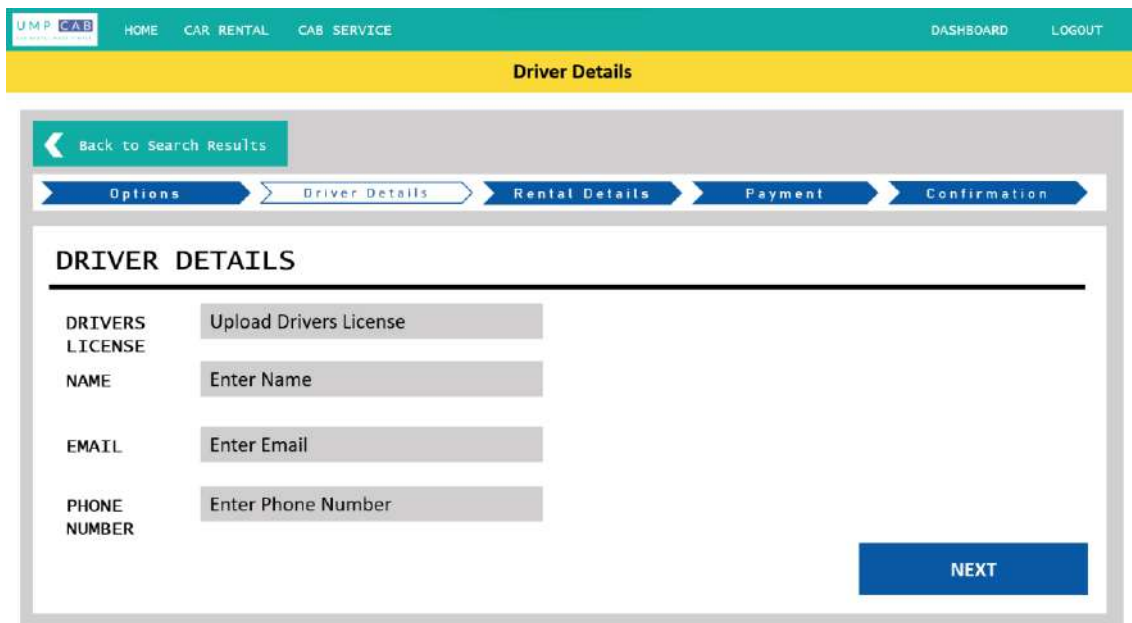
**Figure 3.7.8.2 Search Result Interface**

Users will then click on the View Deal button, and they will be redirected to the Car Details interface as shown below.

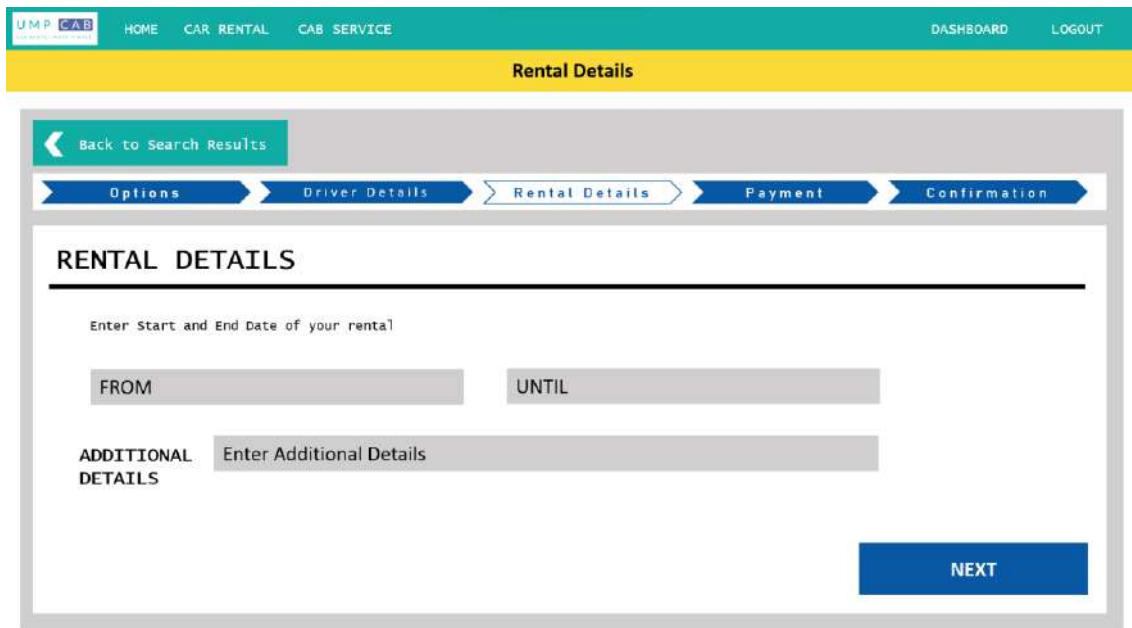


**Figure 3.7.8.3 Car Details Interface**

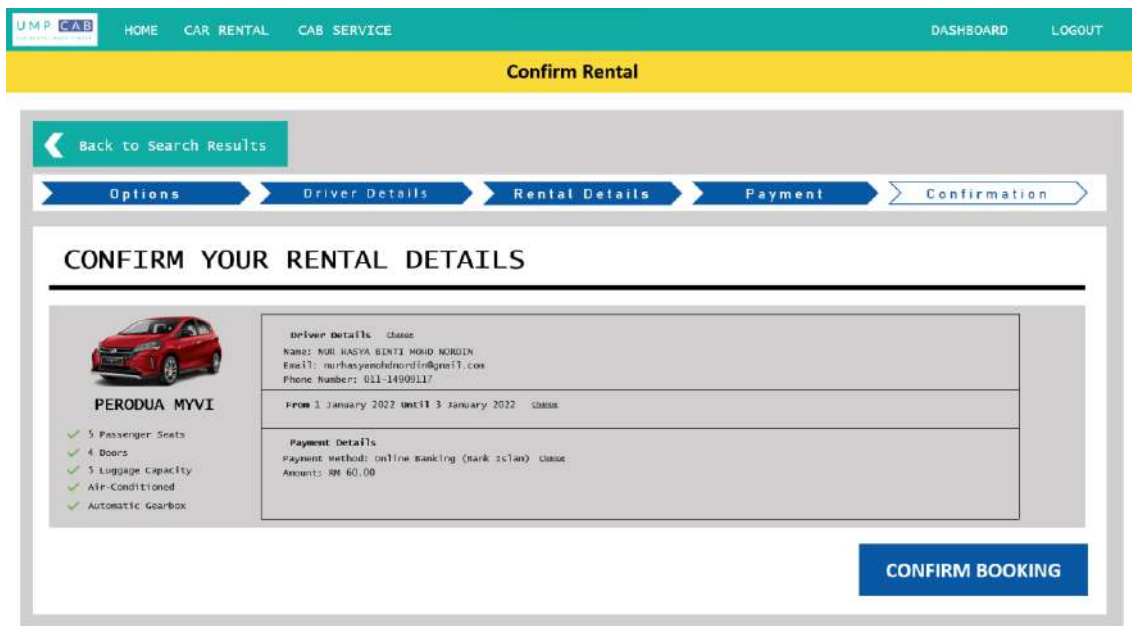
Users can then proceed to provide their details for the car rental such as Driver Details, Rental Details,. The figures below show the interfaces for the respective car rental details.



**Figure 3.7.8.4 Driver Details Interface**



**Figure 3.7.8.5 Rental Details Interface**



**Figure 3.7.8.6 Confirm Rental Interface**

### 3.7.9 Cab Registration

For users who want to register their car up for cab service, they will be redirected to the Register Car for Cab Service interface where they are required to provide appropriate credentials. The Register Car for Cab Service interface is as shown in the figure below.



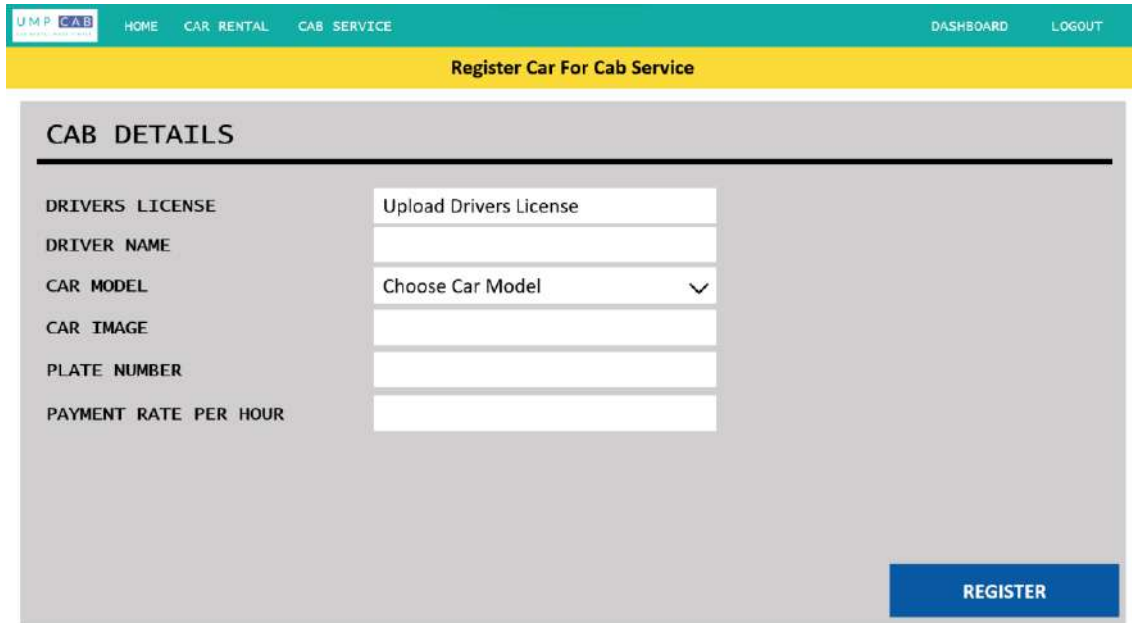
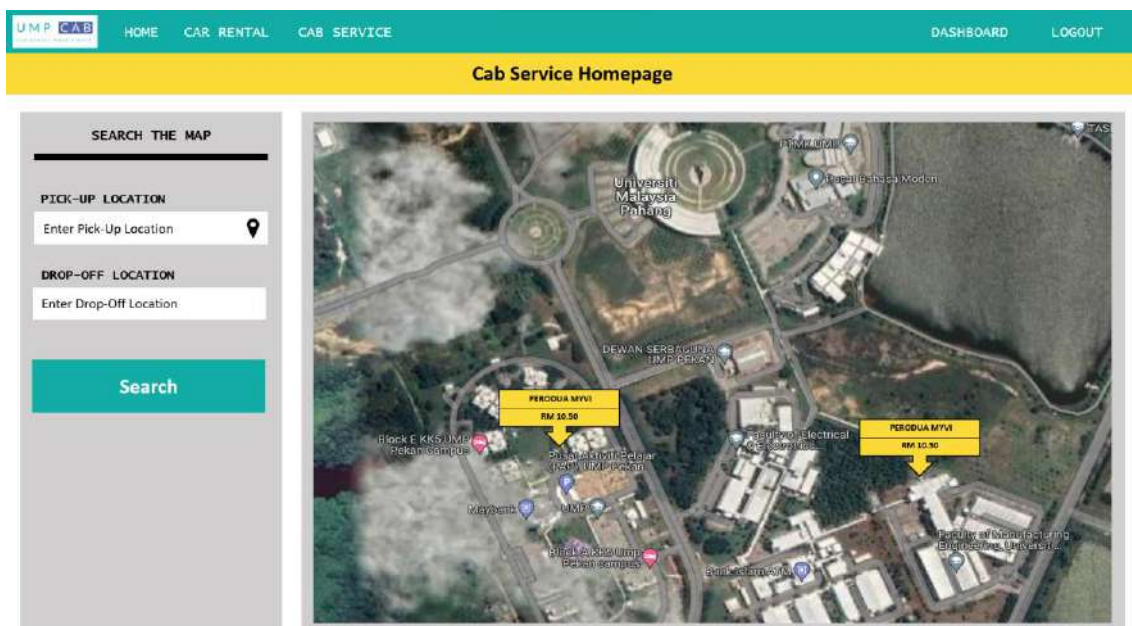


Figure 3.7.9.1 Register Car for Cab Service Interface

### 3.7.10 Cab Booking

For cab booking, users will first browse through available cab services on a map in the Cab Service Homepage interface. In this interface, users can also search the map based on the location they want by entering their location details in the Search corner. The Cab Service Homepage interface is shown in the figure below.



### Figure 3.7.10.1 Cab Service Homepage Interface

To proceed booking a cab, users need to click on any available cab services on the map. They will then be redirected to the Cab Details interface as shown in the figure below.

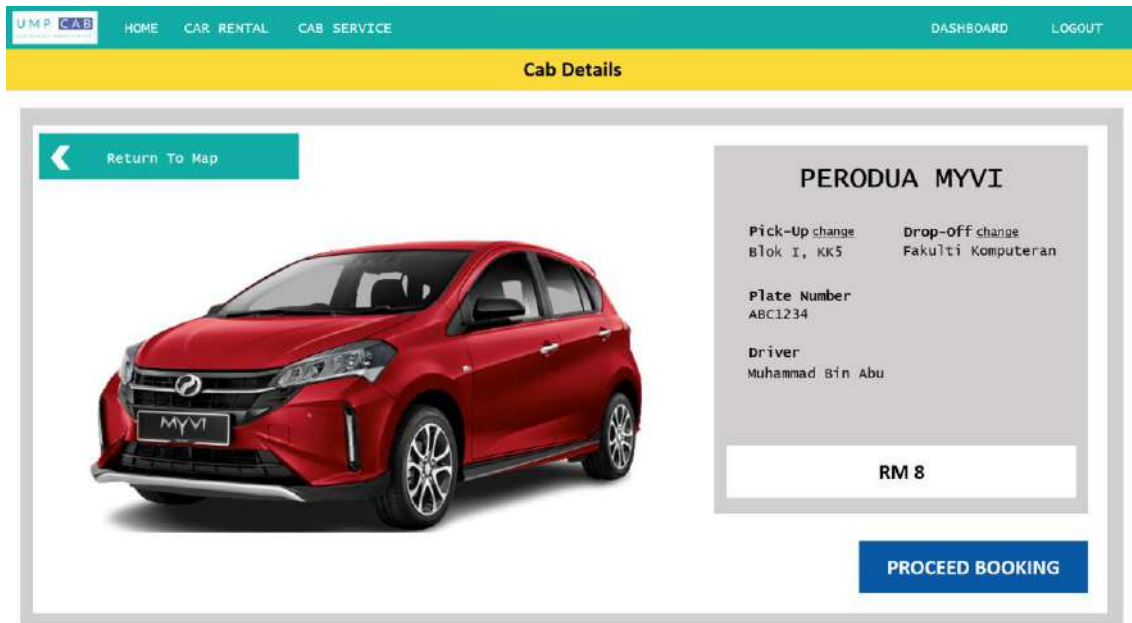
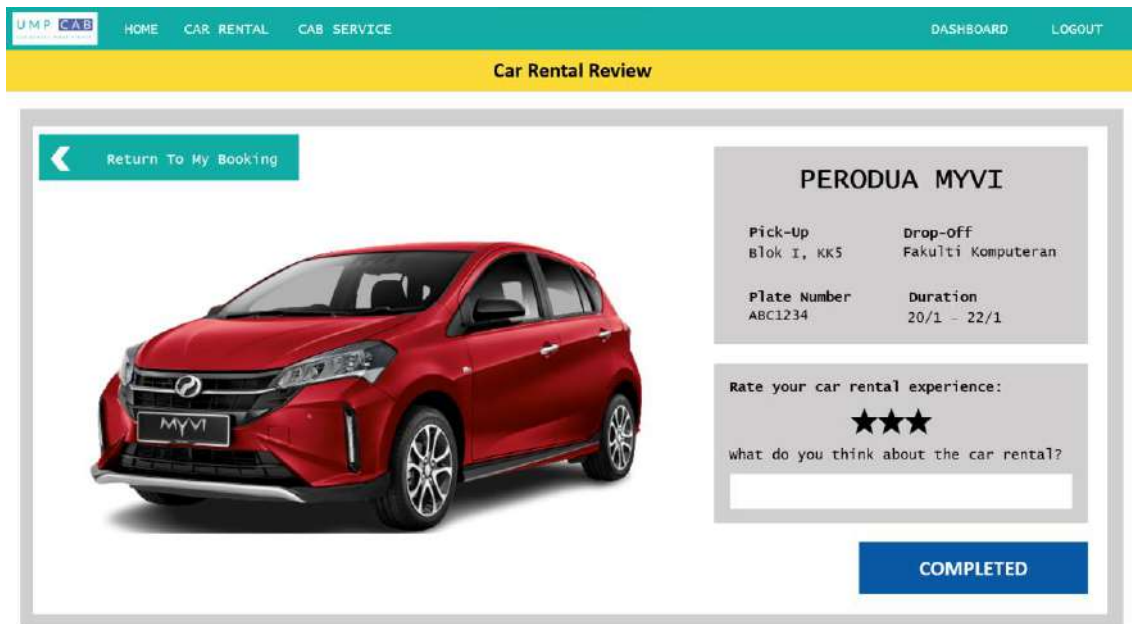


Figure 3.7.10.2 Cab Details Interface

### 3.7.11 Car Rental Review

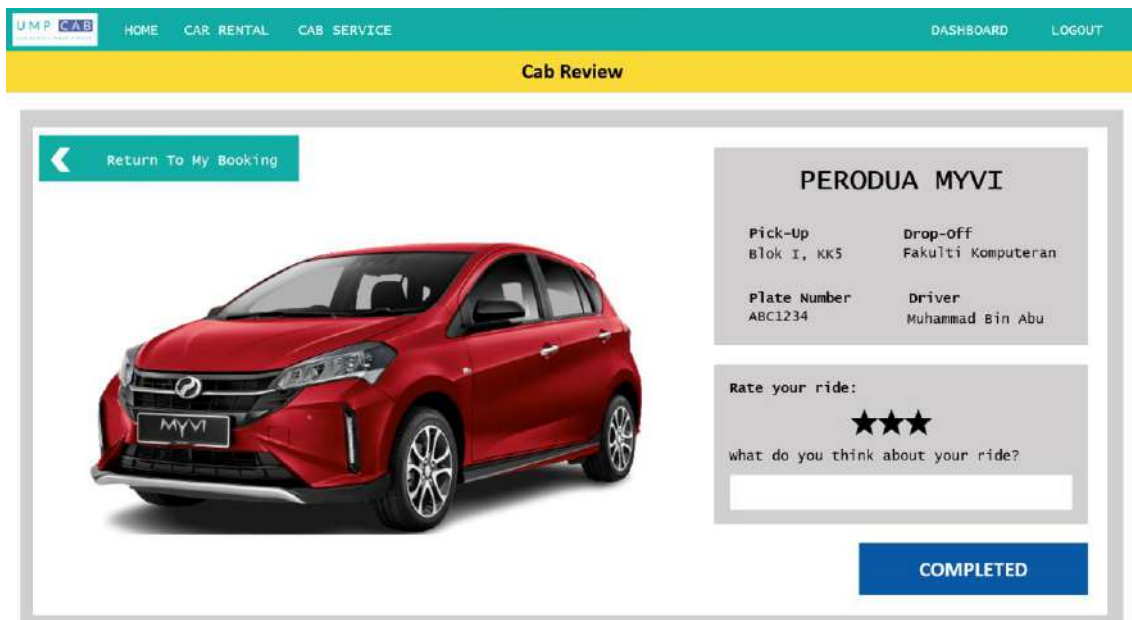
Users are required to rate and review the car rental after they have completed the rental period. For that, they will be redirected to the Car Rental Review interface as shown in the figure below.



**Figure 3.7.11.1 Car Rental Review Interface**

### 3.7.12 Cab Review

Users are required to rate and review the cab service after they have completed the cab ride. For that, they will be redirected to the Cab Review interface as shown in the figure below.



**Figure 3.7.12.1 Cab Review Interface**

### 3.8 TESTING PLAN

The testing that will be performed is the User Acceptance Testing (UAT). This type of testing is performed by the end users of the system to verify the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing are done.

Table 3.8.1 shows the User Acceptance Testing Form

**Table 3.8.1 User Acceptance Testing Form**

No.	Module	Function	Input	Expected Output	Status		Comments (Actual Results)
					Yes	No	
1	Manage Car Rental	Allow Rentee to register car into the system [Register Car]	Car registration details and Register button	Please wait for your car to be Approved before making any car rental or cab service			
		Allow Rentee to register car for rental [Register Car for Rental]	Register for rental button	Your car is registered for rental			
		Allow Rentee to view car rental history [View Rental History]	History button	Car Rental History List			
		Allow Rentee to edit registered car [Edit Car]	Car details and Update button	Car Details Successfully Updated			

		Allow Rentee to delete registered car [Delete Car]	Delete button	Car Successfully Deleted			
		Allow Renter to browse and book for car rental services [Rent Cars]	Renter details and Rental details	You have successfully booked this car!			
2	Manage Cab Service	Allow Driver to register car into the system [Register Car]	Car registration details and Register button	Please wait for your car to be Approved before making any car rental or cab service			
		Allows Driver to register car for cab service [Register Car for Cab Service]	Register for carpool button	Your car is registered for cab service			
		Allow Drivers to accept cab requests [Accept Cab Request]	Accept Request button	Do not forget to mark this ride as completed as soon as you have dropped off passenger at their destination.			
		Allow Drivers to edit registered car [Edit Car]	Car details and Update button	Car Details Successfully Updated			

		Allow Drivers to delete registered car [Delete Car]	Delete button	Car Successfully Deleted			
		Allow Driver to view cab request history [View Cab History]	History button	Cab Request History List			
		Allow Passenger to book cab services [Book Cab]	Destination details	Please wait for a driver to accept your request			
3	Manage Car Review	Allow Rentee to leave rating and review on cars they have rent [Review Car Rental]	Car Review and Rating	Thank you for rating this car!			
		Allow Passenger to leave rating and reviews on cabs they have ride [Review Cab Services]	Cab Review and Rating	Thank you for rating this car!			
4	Manage Users	Allow Admin to edit existing details of users registered into the system [Edit User List]	User details	User Details Updated Successfully			
		Allow Admin to delete existing users from the system [Delete User]	Delete button	User Successfully Deleted			

		Allow Admin to edit details of existing cars registered into the system [Edit Car]	Car details	Car Details Updated Successfully			
		Allow Admin to delete existing cars from the system [Delete Car]	Delete button	Car Successfully Deleted			
		Allow Admin to approve car registration request [Manage Car]	Approve button	Car Registration Approved!			
		Allow Admin to reject car registration [Reject Car]	Reject button	Car Registration Rejected			
5	Manage Report	Allow Admin to view report of the system [View Report]	Report button	Total number of users, Total number of cars registered			

### **3.9 POTENTIAL USE OF PROPOSED SOLUTION**

With Malaysia's change in the covid-19 phase of pandemic to an endemic phase, a lot of higher education institutions are starting to change the mode of learning from online to hybrid and some even conduct fully face-to-face class. Because of this change in phase, students especially university students are required to return to their respective campus. In order to attend face-to-face classes, students require transportation especially if the distance of their class from their rooms are not a walkable distance. However, not all students own cars, motorcycles or any forms of vehicles and it is not easy to look for available transportation especially during the last minute.

This Web-Based Car Rental and Cab Service system does exactly that by providing students with the ability to search for available car rental and carpools around campus. Students can also earn some extra income through the system by registering their cars up for either rental or carpool in the system. This way, students can both make money while also searching for other car rental and carpool services.

Because of Universiti Malaysia Pahang's large campus especially in the Pekan Campus, transportation is a must to ease their daily transportation needs. For the car rental function, students are able to search any nearby car rental services which is also offered by other UMP students. For the carpool function, students can directly find available cabs by scrolling through the map.



### 3.10 GANTT CHART

The Gantt chart below shows the project phases from starting until the project is completed based on the methodology of Rapid Application Development (RAD). The phases include Analysis and Quick Design, Prototyping, Testing and Deployment. Figure 3.10.1 below shows the Gantt chart for the whole project

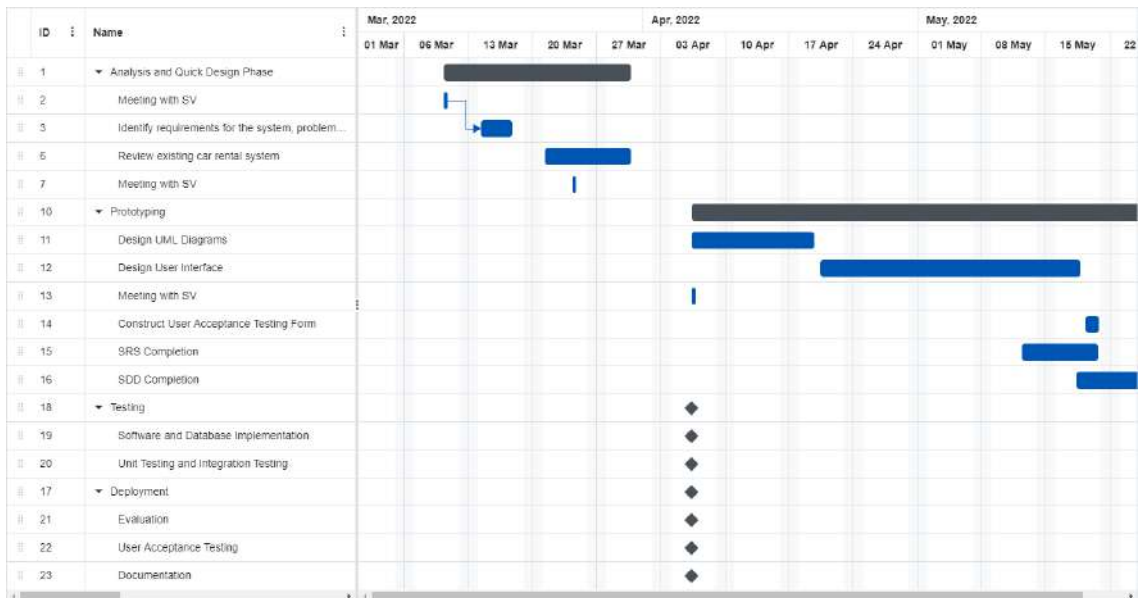


Figure 3.10.1 Gantt Chart

## **CHAPTER 4**

### **IMPLEMENTATION, RESULT AND DISCUSSION**

#### **4.1 INTRODUCTION**

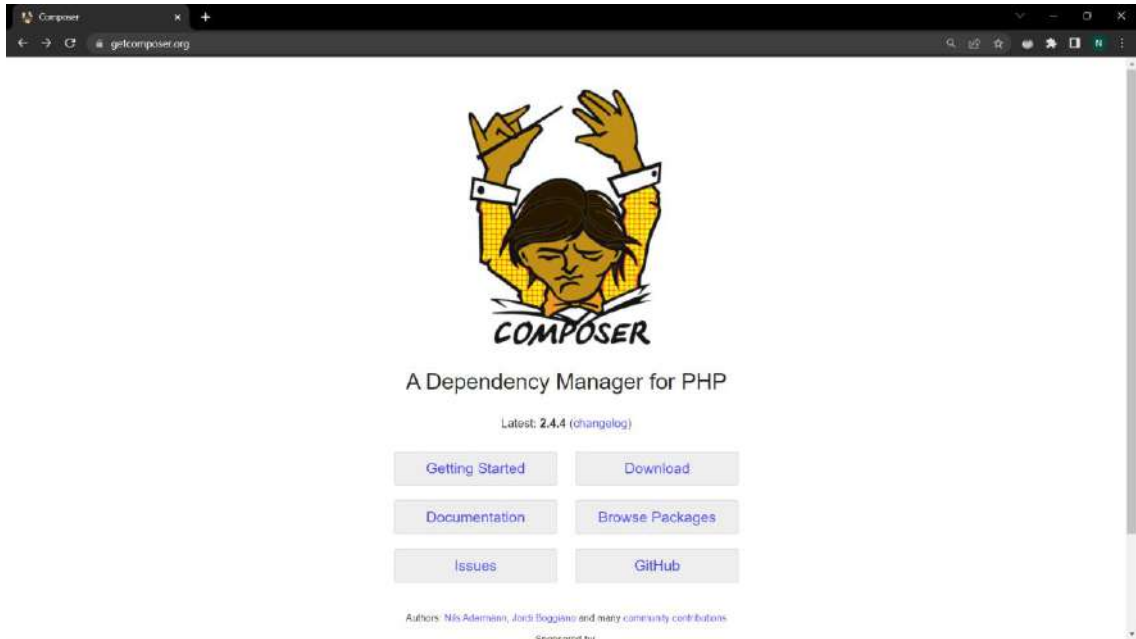
Chapter 4 will discuss about the development, implementation and testing of Web-Based Car Rental and Cab Service System (UMPCab) for UMP Students. We will also look at the results and discuss on the system.

#### **4.2 IMPLEMENTATION PROCESS**

This section shows the development process of the Web-Based Car Rental and Cab Service System (UMPCab) for UMP Students. We will first go through the set-up and installation of Laravel and Laravel Jetstream. Then, we will proceed to the process of coding and creating graphical user interfaces for each module.

##### **4.2.1 LARAVEL INSTALLATION AND SETUP**

For this Web-Based Car Rental and Cab Services System (UMPCab) for UMP Students, I used Laravel. Laravel Jetstream is also used for the features such as login, register, and profile management. In order to use Laravel, the installation of composer is required. Composer is a dependency management tool in PHP. The Composer was installed from the website as shown in the figure below:



**Figure 4.2.1 Composer download website**

After the installation of Composer is successful, open the project file where I want to create my Laravel project and create a new Laravel project using the command line as shown below:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER\Desktop\LaravelPro>laravel new umpcab
```

**Figure 4.2.2 Create new laravel project command**

Next, the installation of Laravel Jetstream on my project is done as shown in the figure below:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER\Desktop\LaravelPro\umpcab>composer require laravel/jetstream_
```

### **Figure 4.2.3 Installing Laravel Jetstream command**

Next, I opened the created project file inside Visual Studio Code to finalize the installation of Laravel Jetstream using the terminal.

```
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\USER\Desktop\LaravelPro\umpcab> npm install

up to date, audited 470 packages in 1s

28 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 low, 6 high)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

**Figure 4.2.4 npm install artisan command**

```
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

PS C:\Users\USER\Desktop\LaravelPro\umpcab> npm run dev

> dev
> vite

VITE v3.1.0 ready in 580 ms
  → Local: http://127.0.0.1:5173/
  → Network: use --host to expose

LARAVEL v9.38.0 plugin v0.6.0

  → APP_URL: http://127.0.0.1:8000
```

**Figure 4.2.5 npm run dev artisan command**

Finally, I used the built in PHP server to serve the UMPCab application with the serve artisan command which will give us the localhost server at <http://127.0.0.1:8000>

```
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\USER\Desktop\LaravelPro\umpcab> php artisan serve

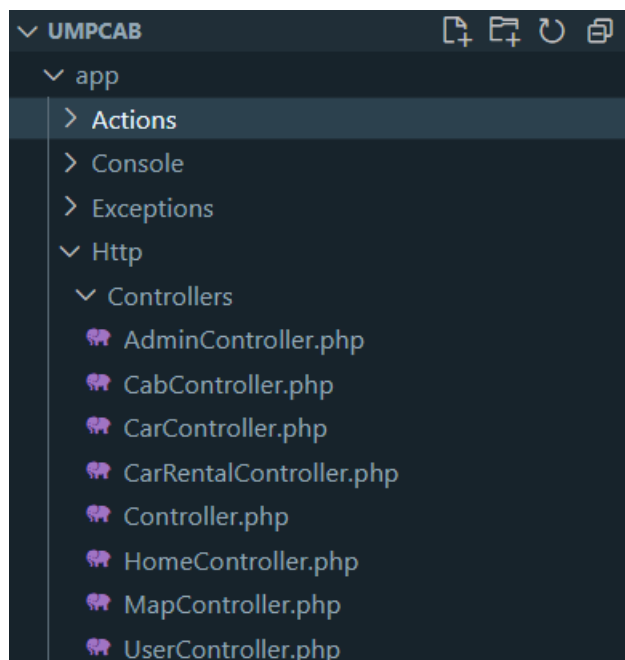
[INFO] Server running on [http://127.0.0.1:8000].

Press Ctrl+C to stop the server
```

**Figure 4.2.6 Running the project on localhost**

### Creating Controllers

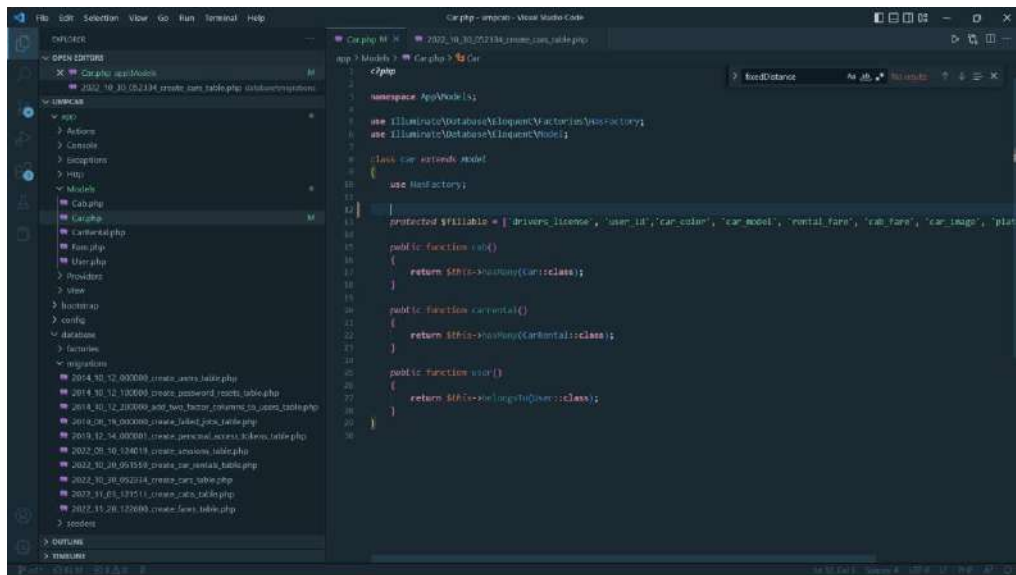
To create a controller, the artisan command of *php artisan make:controller CarController* is used. For this, the CarController.php file can be found inside App/Http/Controllers/CarController



**Figure 4.2.7 Controllers inside laravel**

### Creating Models and Migrations

To create Models and Migrations, the artisan command of *php artisan make:model Car -m* can be used which will create the Car model along with its migration file. The Car model can be found inside the App/Models/Car and the migration file can be found inside the App/Database/Migrations.



**Figure 4.2.8 Models and migration files in Laravel**

## 4.2.2 Database Environment and Connection Setup (MySQL and XAMPP)

Before the project is able to be run and viewed inside localhost, the connection with APACHE and MYSQL must first be setup using XAMPP. For the users to be able to use the functionality of the system, connection to the database is required. For the database, MySQL is used as well as PHPMysqlAdmin to monitor the data involved in the system. To achieve connection from the localhost to the database, XAMPP is required to run as shown below.

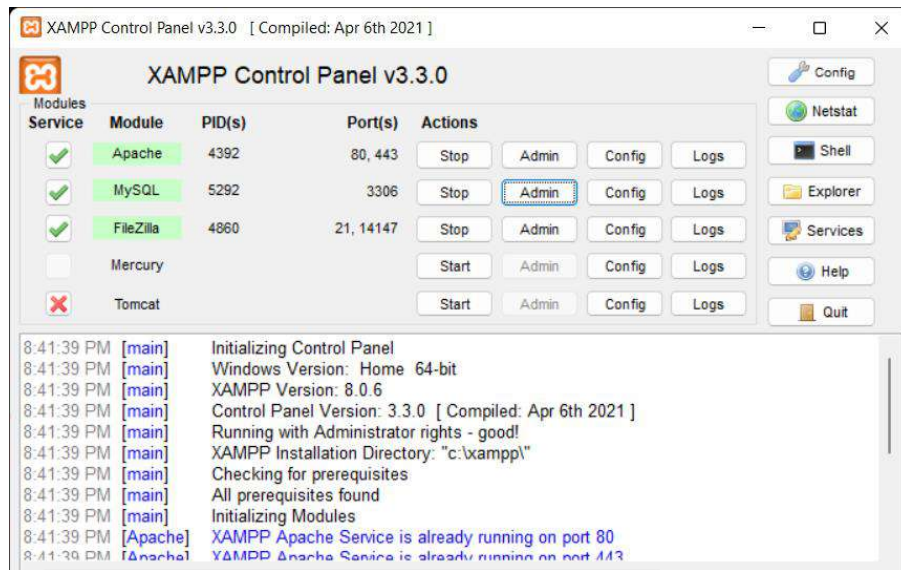
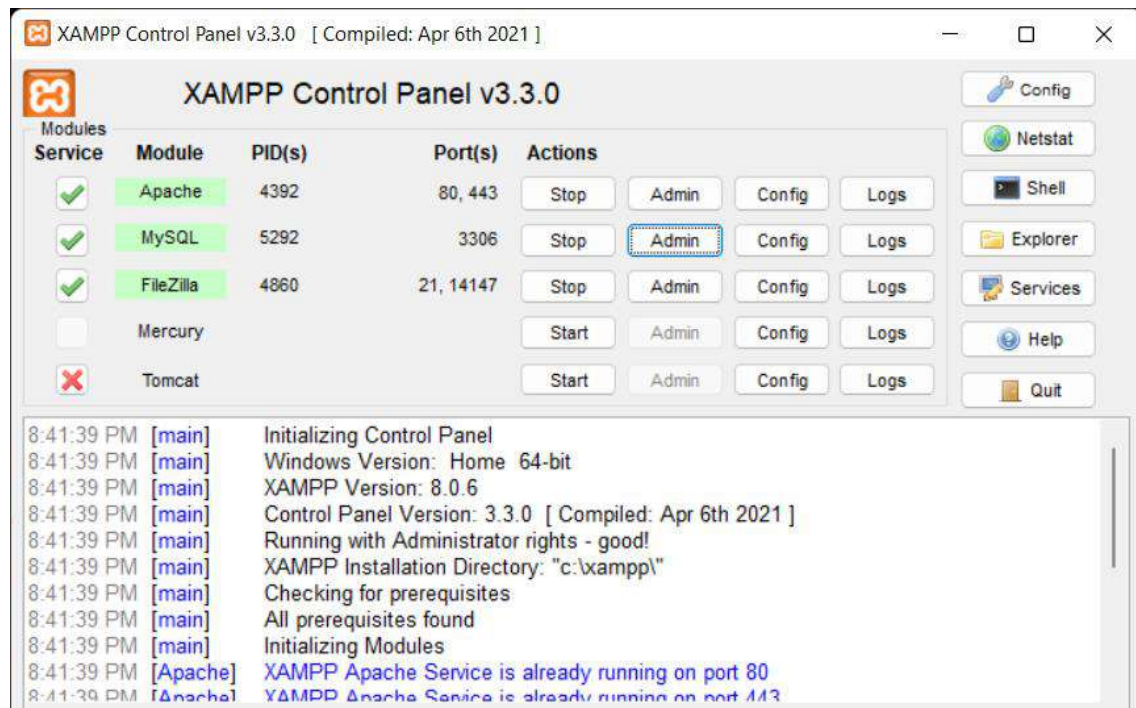


Figure 4.2.9 XAMPP Control Panel

### 4.2.3 Database Setup

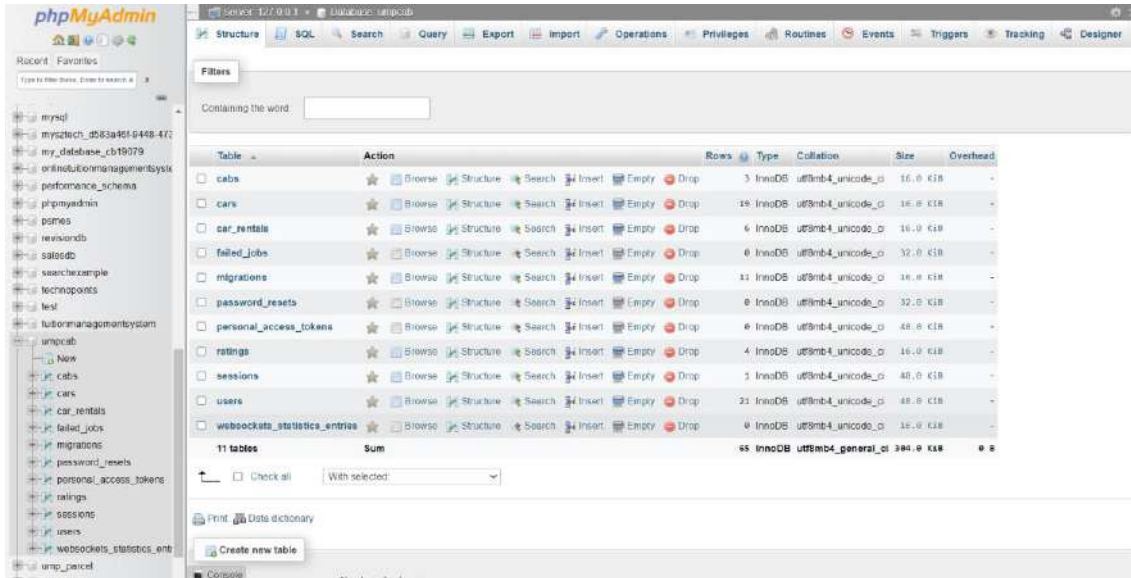
In this section, the database setup is explained. The database used for this project is MySQL. The MySQL database can be viewed from the phpMyAdmin which can be accessed by clicking the *Admin* button in the XAMPP Control Panel.





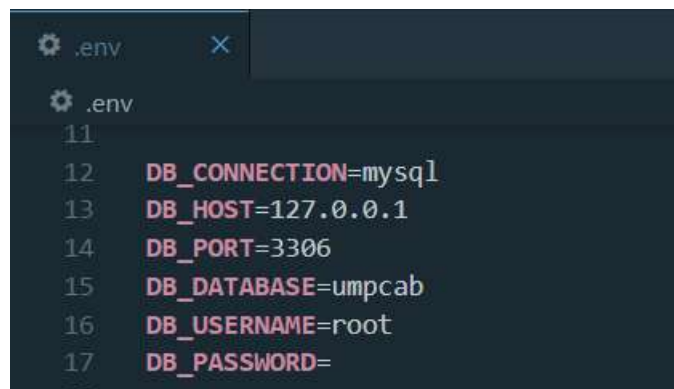
**Figure 4.2.10 XAMPP Control Panel**

Inside the phpMyAdmin, the database for the UMPCab project is created which is the umpcab database.



**Figure 4.2.11 UMPCab database inside phpMyAdmin**

In order to connect the database with the UMPCab project, the .env file inside the project must be modified as shown in the figure below.



**Figure 4.2.12 .env file inside UMPCab project**

In order to create the tables inside the database in phpMyAdmin, the migration command is required to be run inside the project's terminal. The artisan command for the migration is *php artisan migrate* which will migrate all the tables for the UMPCab project inside the database.

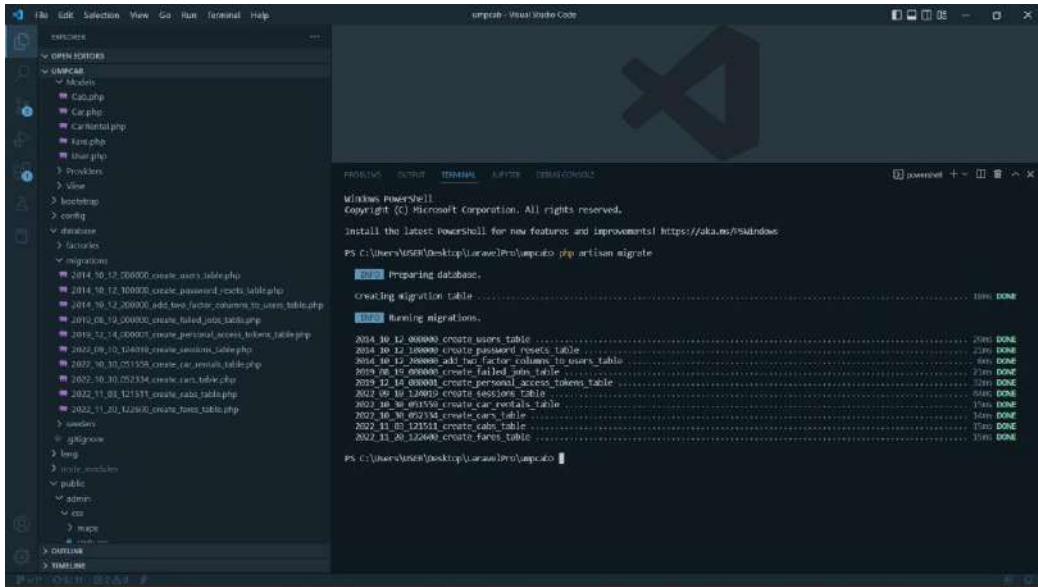


Figure 4.2.13 Migrating database tables

Whenever changes are done inside any of the migration files, the artisan command *php artisan migrate* should be used instead in order to refresh the modified source codes inside the migration files.

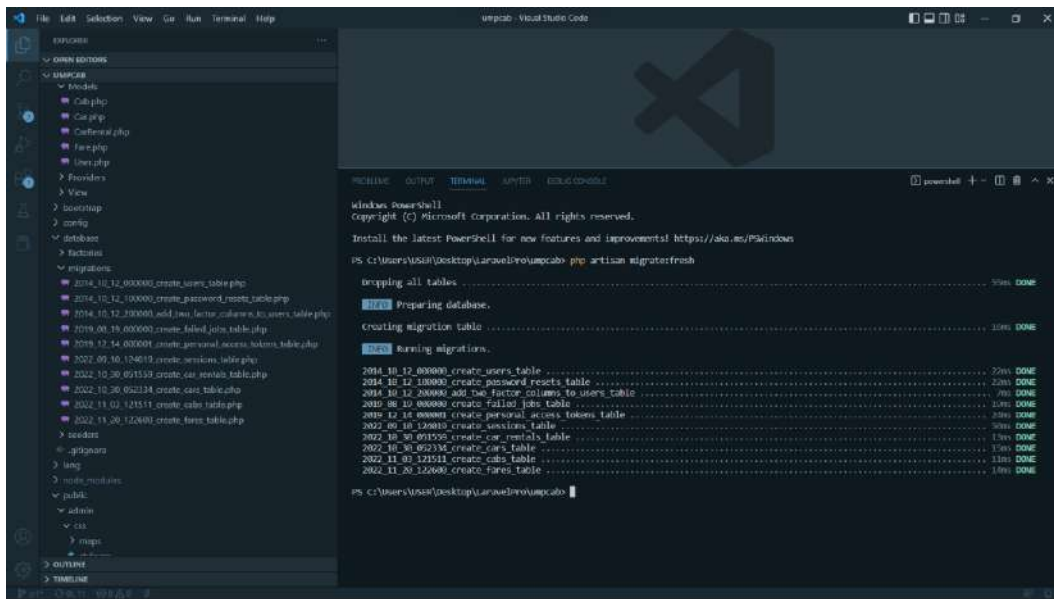


Figure 4.2.14 Updating migration files

## **4.3 DESIGNING USER INTERFACES AND CODE IMPLEMENTATION**

### **4.3.1 Laravel Jetstream Components**

The user interface of the system is designed and developed using IDE which is Visual Studio Code. The landing page of the system is designed and developed using Tailwind CSS and Bootstrap CSS. Figures below shows the landing page, and the code snippets for creating the landing page.

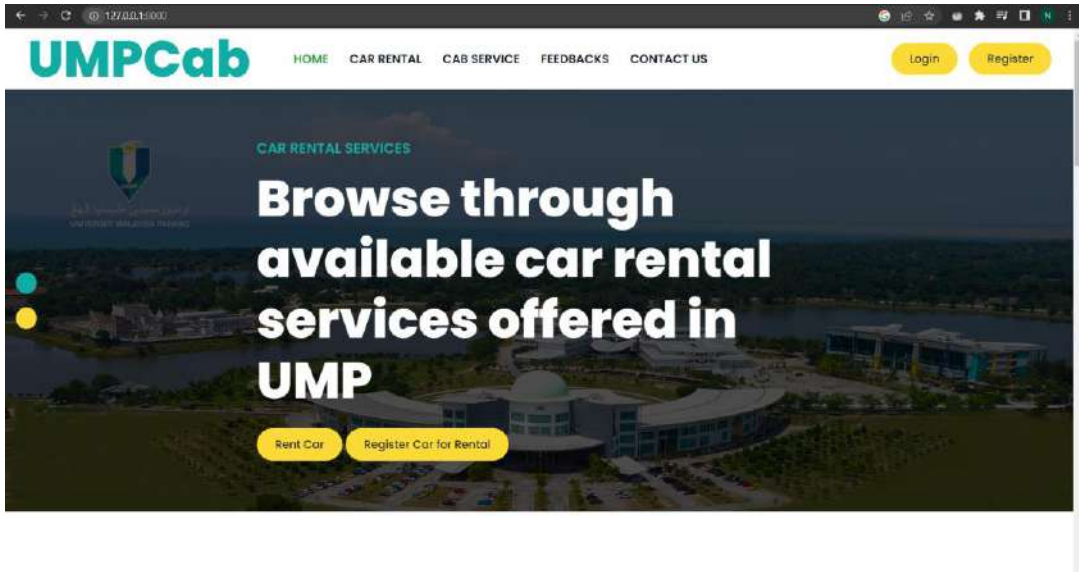


Figure 4.3.1 UMPCab landing page

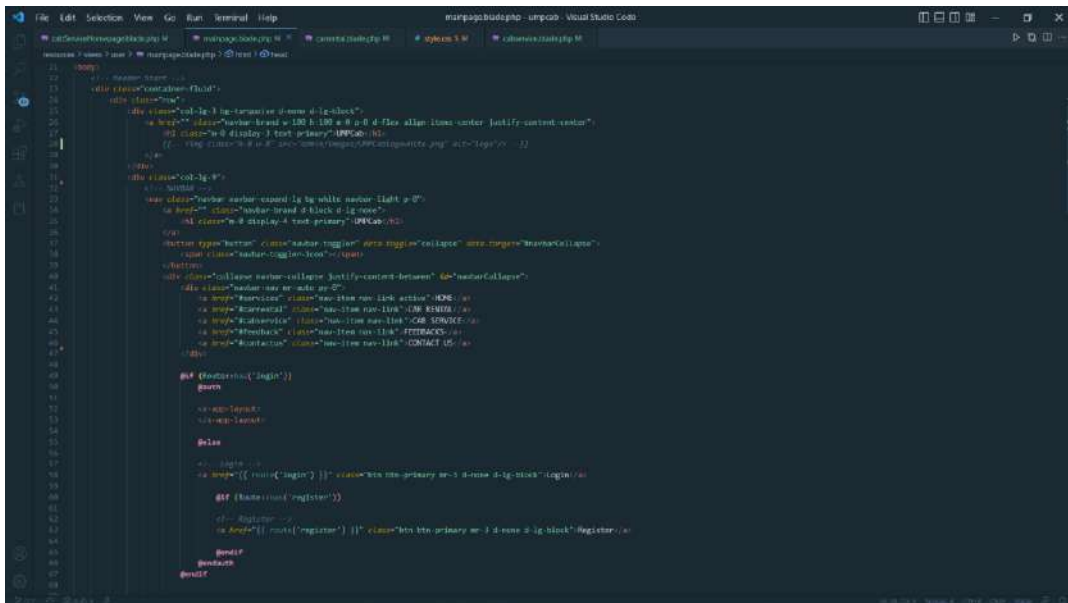


Figure 4.3.2 UMPCab landing page code snippet

After designing the landing page of the system, log in and register process is required for the users of the system to get full functionality of the system. Laravel Jetstream is used for the login and register page as well as the profile page of the users that logged in into the system. The figures below shows the login page, register page and also the profile page of the system.

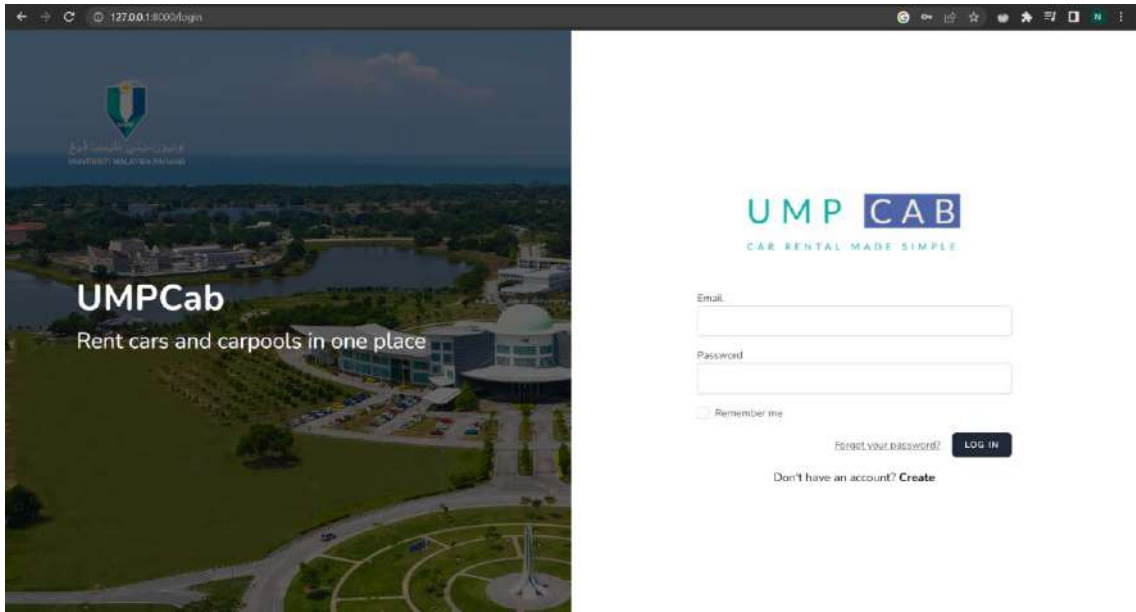


Figure 4.3.3 UMPCab Login Page

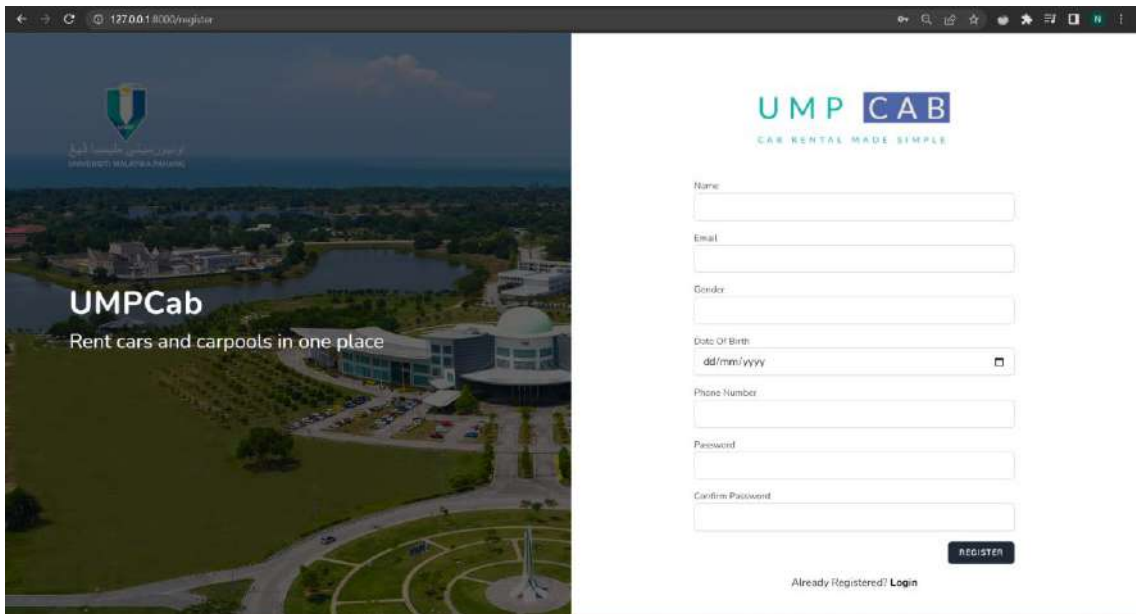


Figure 4.3.4 UMPCab Register Page

```

login.blade.php - umpcb - Visual Studio Code
resources/views/auth/login.blade.php
<script src="/assets/images/UMPCabLogoWhite.png" alt="logo"/>
</script>
<div class="row" style="border: 1px solid #ccc; padding: 10px; margin: 10px auto; width: 80%;>
<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px; text-align: center; font-weight: bold; color: #007bff;">
    Register
    </div>
<div style="display: flex; justify-content: space-between;">
<div style="width: 45%;">
<div style="margin-bottom: 10px;">
<input type="text" class="form-control" value="{{ old('name') }}" />
</div>
<div style="margin-bottom: 10px;">
<input type="text" class="form-control" value="{{ old('email') }}" />
</div>
<div style="margin-bottom: 10px;">
<input type="text" class="form-control" value="{{ old('password') }}" />
</div>
<div style="margin-bottom: 10px;">
<input type="text" class="form-control" value="{{ old('confirm_password') }}" />
</div>
<div style="margin-bottom: 10px;">
<input type="text" class="form-control" value="{{ old('dob') }}" />
</div>
<div style="margin-bottom: 10px;">
<input type="text" class="form-control" value="{{ old('gender') }}" />
</div>
<div style="margin-bottom: 10px;">
<input type="text" class="form-control" value="{{ old('phone_num') }}" />
</div>
</div>
<div style="width: 45%; text-align: right;">
<div style="margin-bottom: 10px;">
<input type="checkbox" class="checkbox" /> Remember me
</div>
<div style="margin-bottom: 10px;">
<input type="button" value="Register" class="btn btn-primary" />
</div>
</div>
</div>
</div>

```

Figure 4.3.5 UMPCab Login and Register UI Page Code Snippet

For the creation of new users, the *CreateNewUser.php* file is modified according to the details that the users of the UMPCab system will have as shown in the figure below.

```

CreateNewUser.php - umpcb - Visual Studio Code
namespace App\Actions\Vortify;
use App\Models\User;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Laravel\Fortify\Contracts\CreateNewUsers;
use Laravel\Fortify\Streams;

class CreateNewUser implements CreateNewUsers
{
    use PasswordValidationRules;

    /**
     * Validate and create a newly registered user.
     *
     * @param array $input
     * @return App\Models\User
     */
    public function create(array $input)
    {
        $validator = Validator::make($input, [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'phone_num' => ['required', 'string'],
            'dob' => ['required', 'string', 'max:255'],
            'gender' => ['required', 'string', 'max:255'],
            'password' => $this->passwordRules(),
            'confirm_password' => $this->confirmPasswordRules(),
        ]);

        if ($validator->fails()) {
            return $validator->errors();
        }

        $user = User::create([
            'name' => $input['name'],
            'email' => $input['email'],
            'password' => Hash::make($input['password']),
            'dob' => $input['dob'],
            'gender' => $input['gender'],
            'phone_num' => $input['phone_num'],
        ]);

        return $user;
    }
}

```

Figure 4.3.6 Create new user code snippet (User Registration)



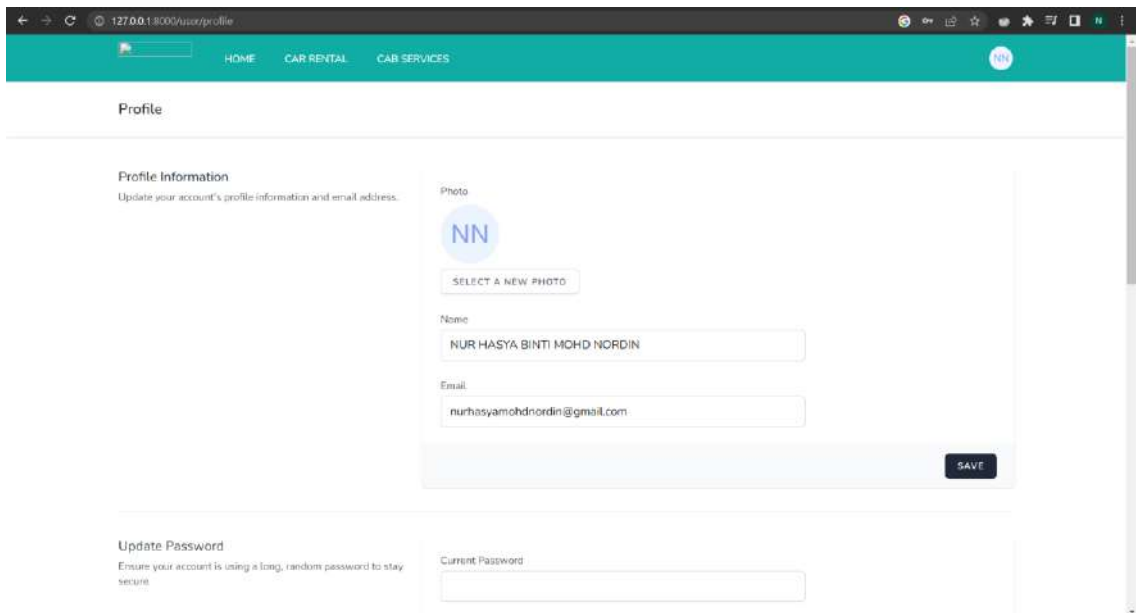


Figure 4.3.7 Profile Page

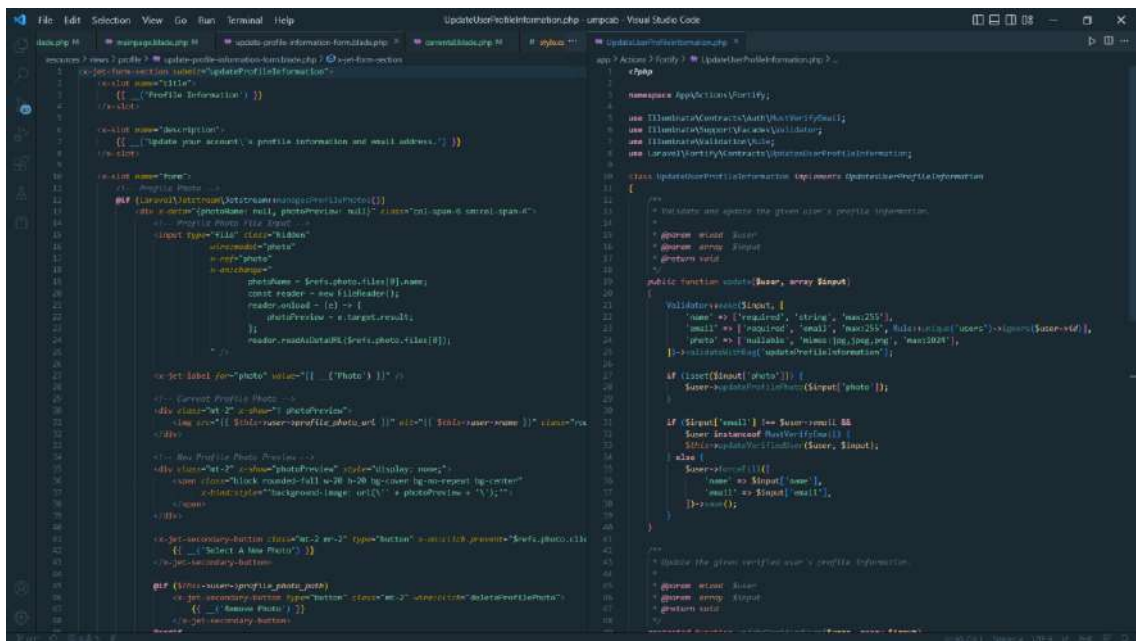
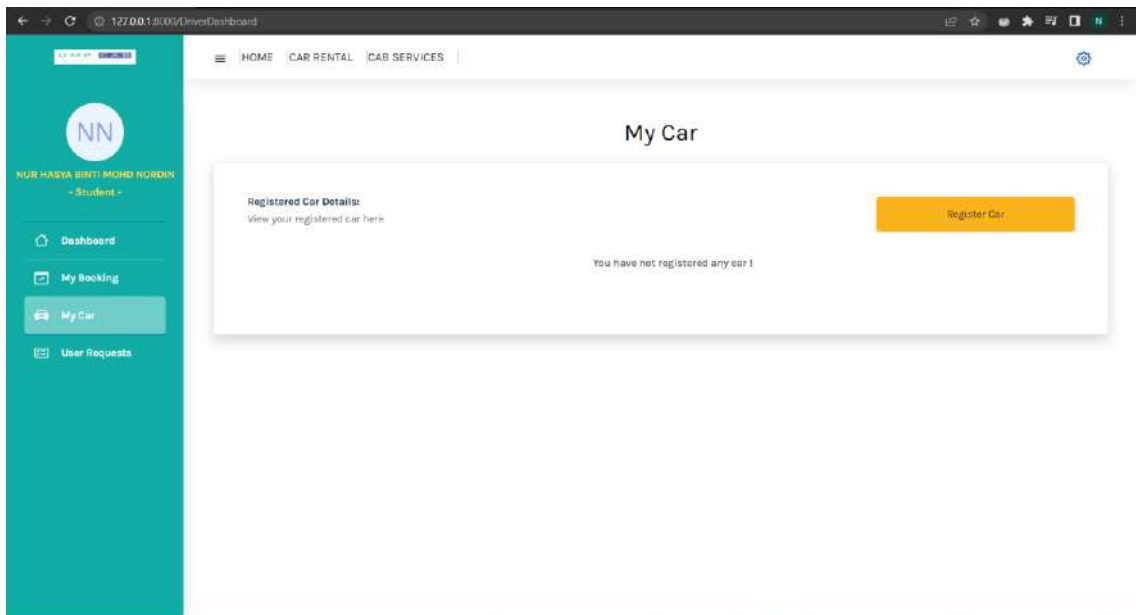


Figure 4.3.8 Profile Page Code Snippet

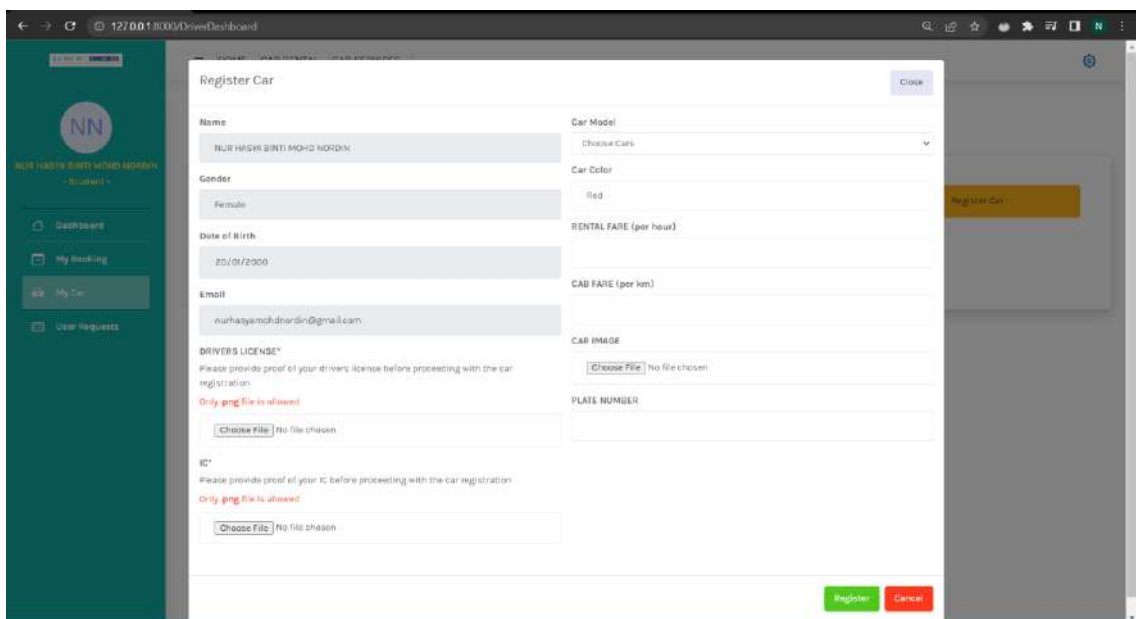
## 4.3.2 Manage Car Rental

*Allow Renteer to register car into the system [Register Car]*

For users who own cars and they want to register their car for rental, they can go to the Driver tab in their dashboard and click the **Register Car** button which will trigger a modal where they are required to fill in the details of their car.



**Figure 4.3.9 Driver Dashboard User Interface**

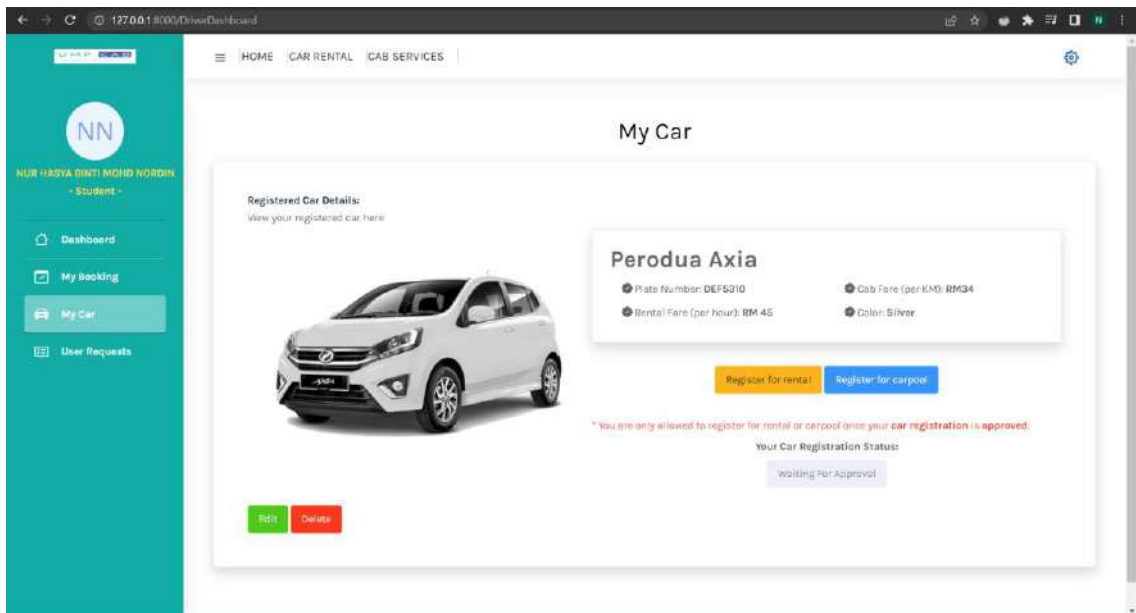


**Figure 4.3.10 Driver Dashboard User Interface (Car Registration Form)**

*Allow Rentee to register car for rental [Register Car for Rental]*

After they have registered their car, they can choose the **Register for Rental** button to make their car available for rental in the system.

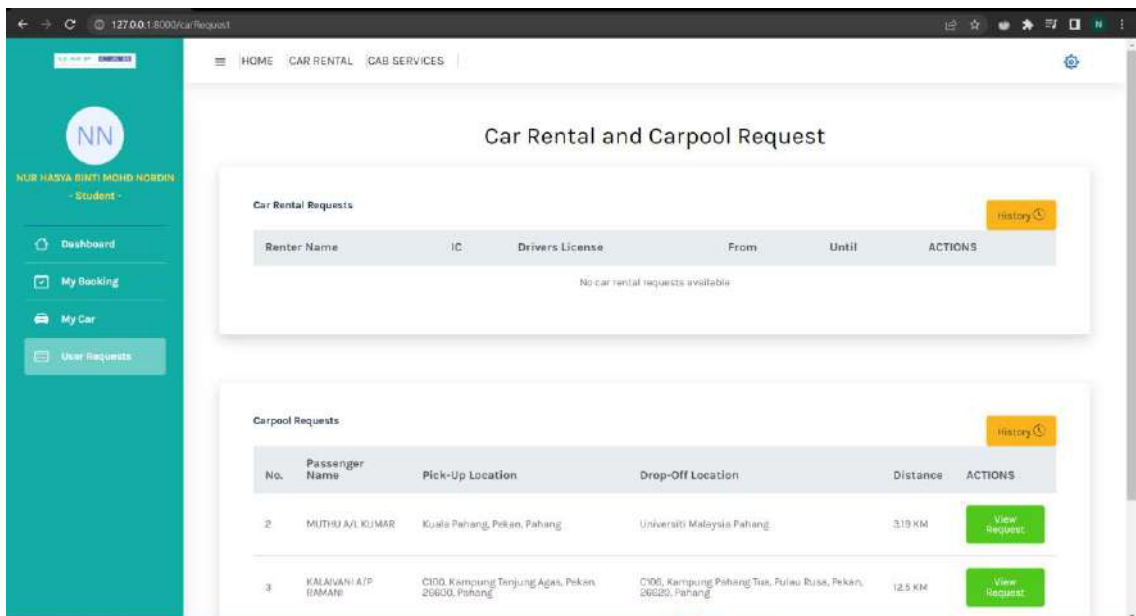


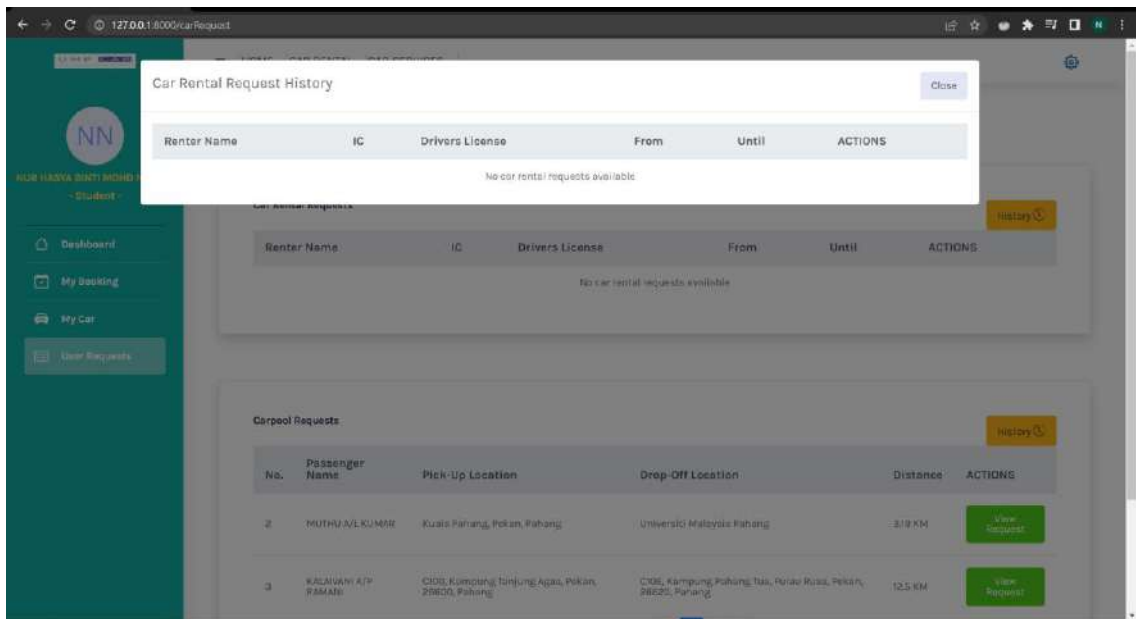


**Figure 4.3.11 Driver Dashboard User Interface (Register for Rental)**

Allow Rentee to view car rental history [View Rental History]

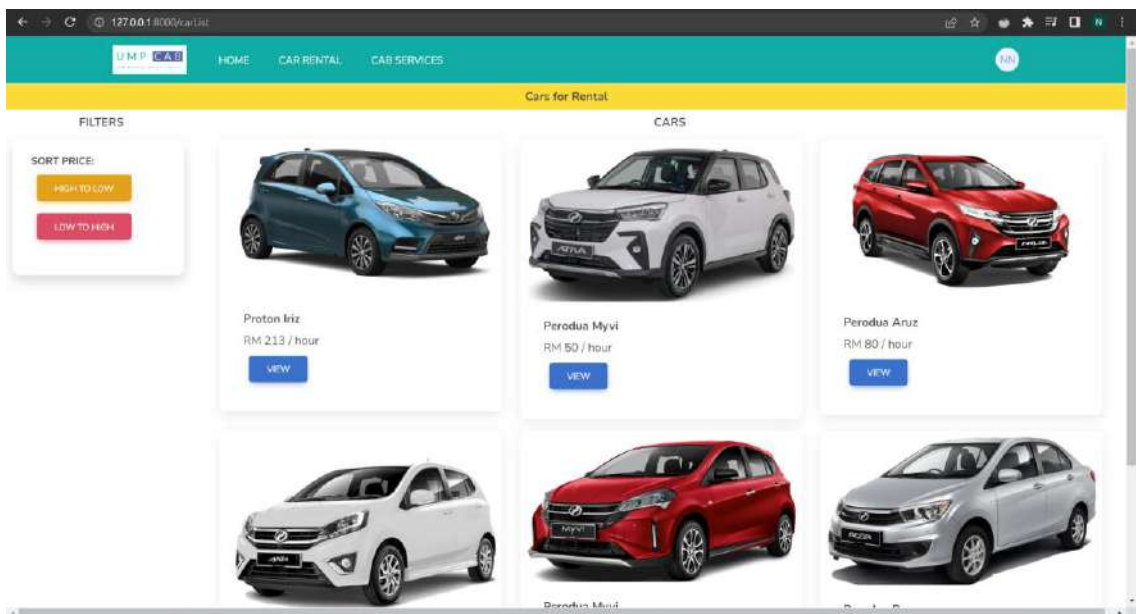
Rentee who wants to view their car rental history can go to the *User Request* tab in the sidebar and click the *History* button.





*Allow Renter to browse and book for car rental services [Rent Cars]*

The figure below shows the interface for the car rental booking. In this interface, users can search and book for any available cars that they want for rent.

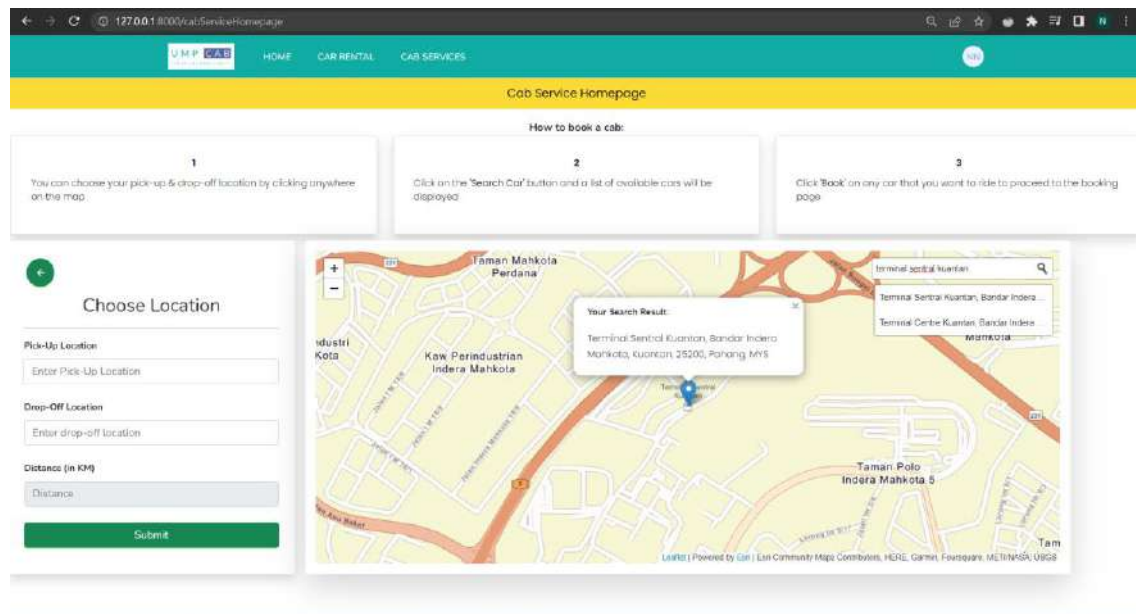


**Figure 4.3.12 Book Car Rental Interface**

Users can view the details of their desired car they want to rent in the car details interface.

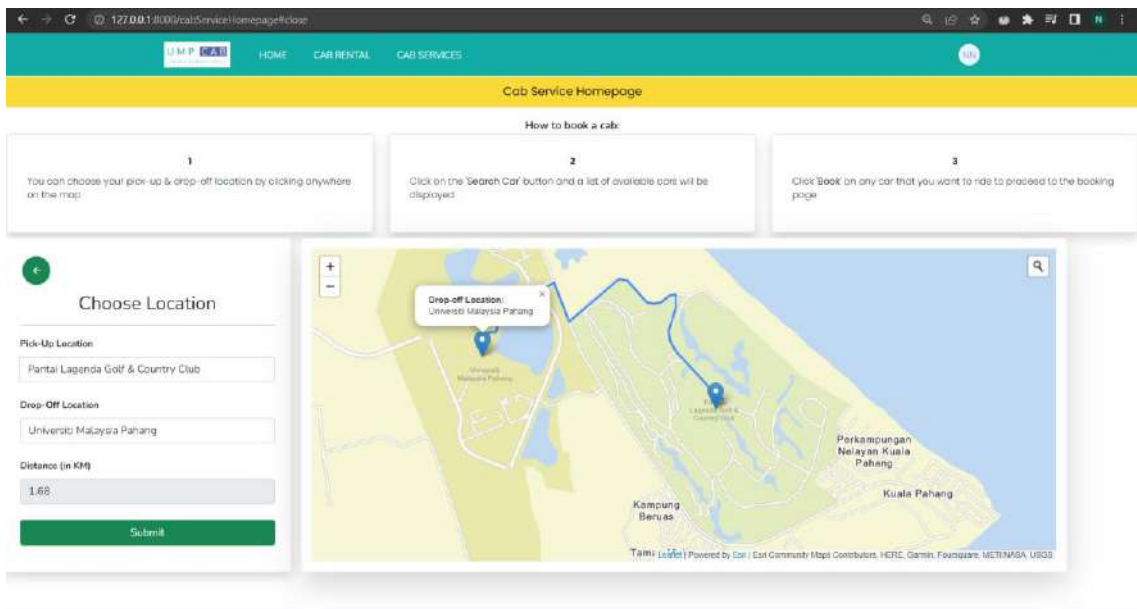
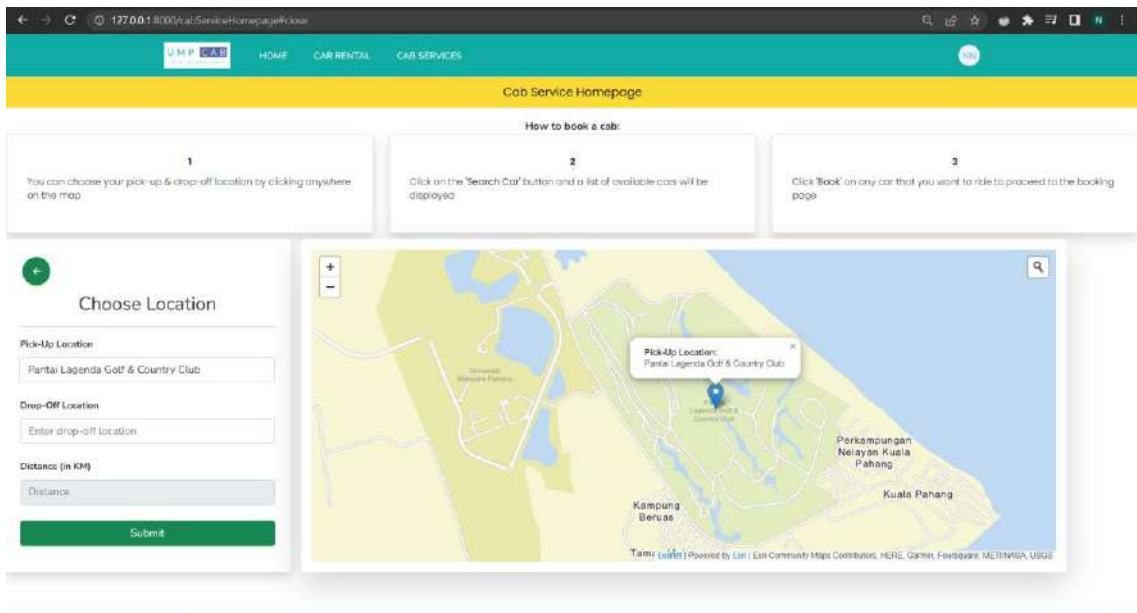
### 4.3.3 Manage Cab Service

Users can search anywhere on the map by clicking the *search* icon on the top right corner of the map and search for their desired location. The example is shown in the figure below.



**Figure 4.3.13 Cab Service Homepage User Interface (Search Map)**

When entering the pick-up and drop-off location, users can either manually fill in their pick-up and drop-off location or they can mark two locations on the map and the addresses of the two locations will be displayed inside the input form as well as the distance between the locations. For the map, I use Leaflet JS for the display of the map and also the markers that can be marked on the map. For converting latitude and longitude into readable address, I used reverse geocoding. The code snippet and the example is shown in the figure below.



**Figure 4.3.14 Cab Service Homepage User Interface (Marking the Map)**

```

File Edit Selection View Go Run Terminal Help
cabServiceHomepageBlock.php - umptob - Visual Studio Code

cabServiceHomepageBlock.php
// get current location
function getLocation() {
    // .....
}

// generate center
const searchControl = L.esri.Geocoding.geocodeSearch({
    position: "topright",
    placeholder: "Enter address (e.g. Terminal Sentral Kuantan)",
    autoComplete: true,
    providers: [
        L.esri.Geocoding.geocodeSearchProvider({
            apiKey: apiKey,
            nearby: {
                lat: -3.3333,
                lng: 103.2000
            }
        })
    ]
}).addTo(map);

// display search result on map
const results = L.layerGroup().addTo(map);

// search address search type from map and replace with new one
searchControl.on("result", (data) => {
    results.clearLayers();

    // add search result to marker
    for (let i = 0; data.results.length > 0; i++) {
        const marker = L.marker(data.results[i].latlng);

        // create address of search result
        const lnglatString = ` ${Math.round(data.results[i].latlng.lng * 100000).toFixed(5)}, `;
        Math.round(data.results[i].latlng.lat * 100000).toFixed(5) ` `;

        // create address of search result
        const lnglatString = ` ${Math.round(data.results[i].latlng.lng * 100000).toFixed(5)}, `;
        Math.round(data.results[i].latlng.lat * 100000).toFixed(5) ` `;

        // create address of search result
        const lnglatString = ` ${Math.round(data.results[i].latlng.lng * 100000).toFixed(5)}, `;
        Math.round(data.results[i].latlng.lat * 100000).toFixed(5) ` `;

        results.addToLayer(marker);
    }
});

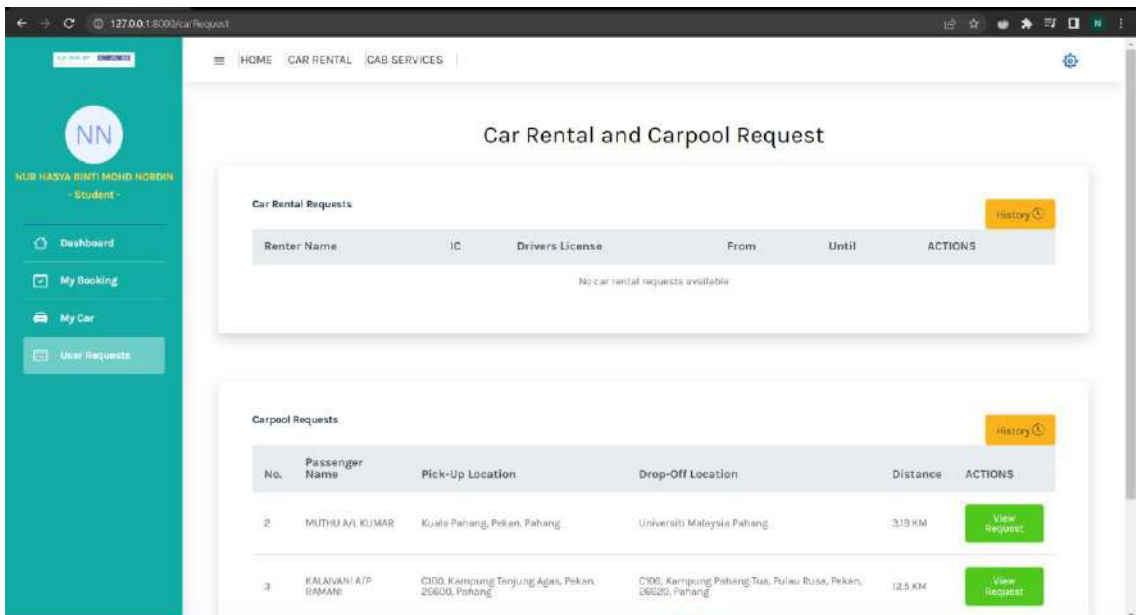
```

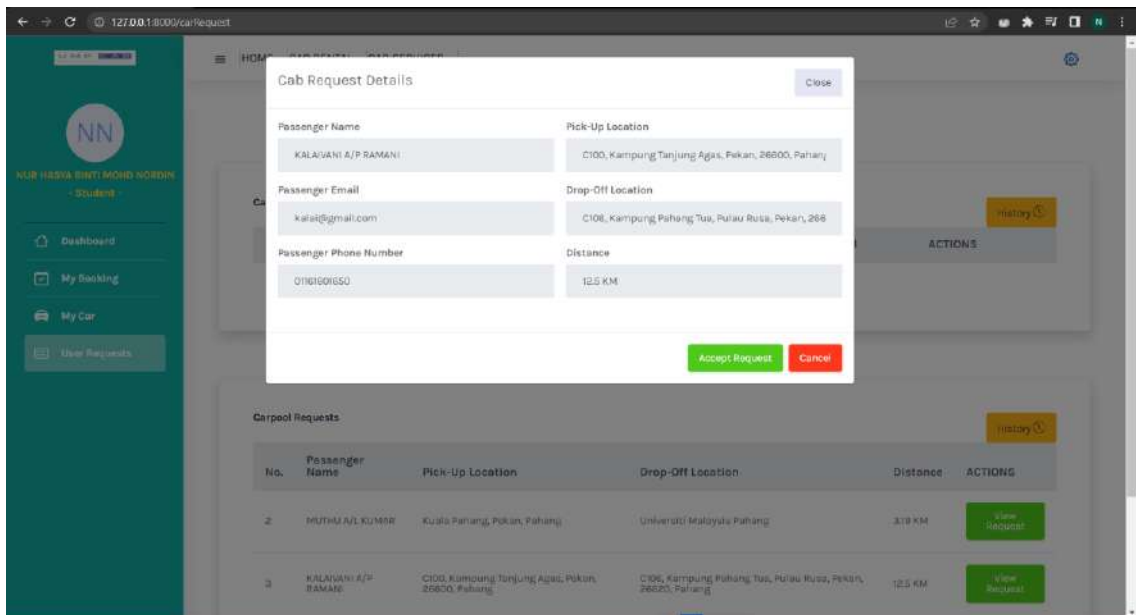
**Figure 4.3.15 Cab Service Homepage Code Snippet**

For users who wants to make their car available for carpool service in the system, they can click the **Register for Carpool** button in their registered car section of their dashboard.

*Allow Drivers to accept cab requests [Accept Cab Request]*

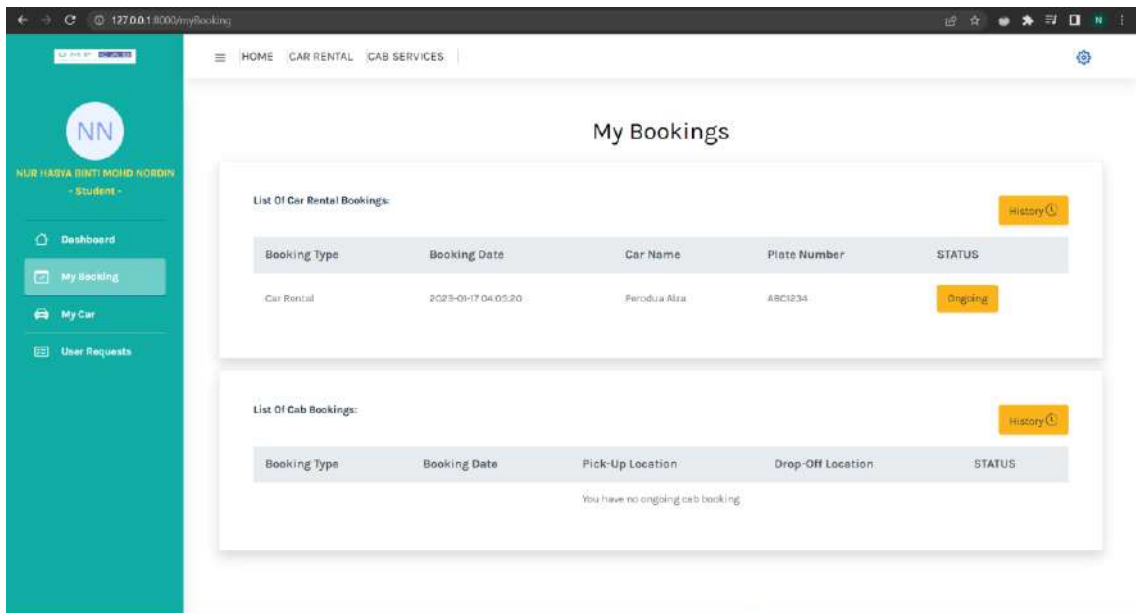
Drivers can go to the **User Request** tab in the sidebar and view the list of available cab ride requests and click the **View Request** button to view the details.





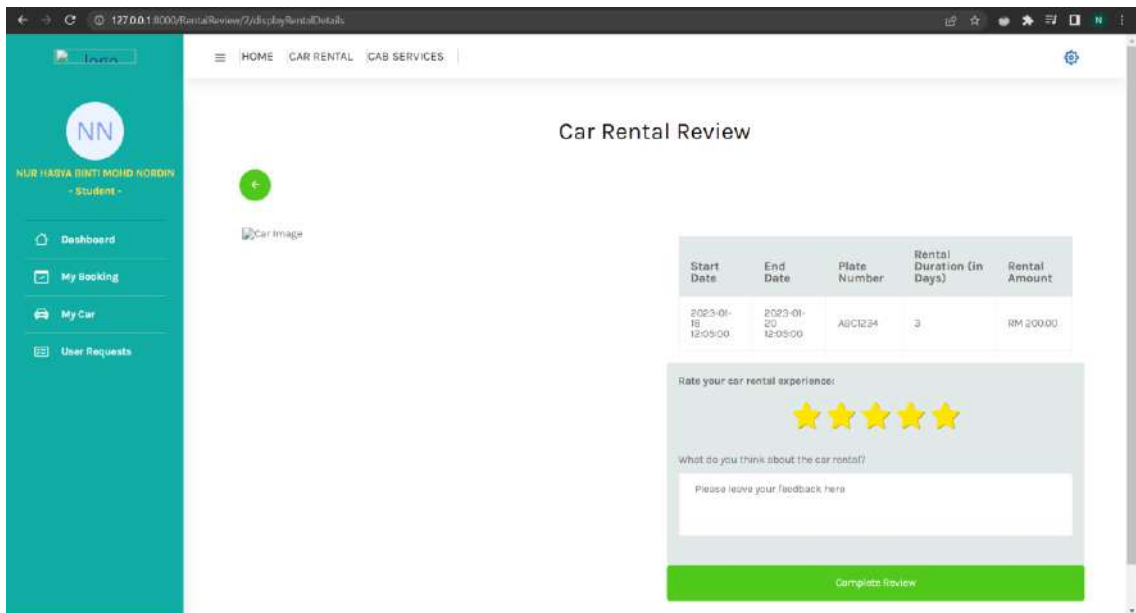
#### 4.3.4 Manage Car Review

After users have completed their car rental or carpool services, they can give ratings and also reviews to the car owners by going to the **My Booking** tab in their dashboard and clicking on the **Ongoing** button as shown in the figure below.



**Figure 4.3.16 My Bookings User Interface**

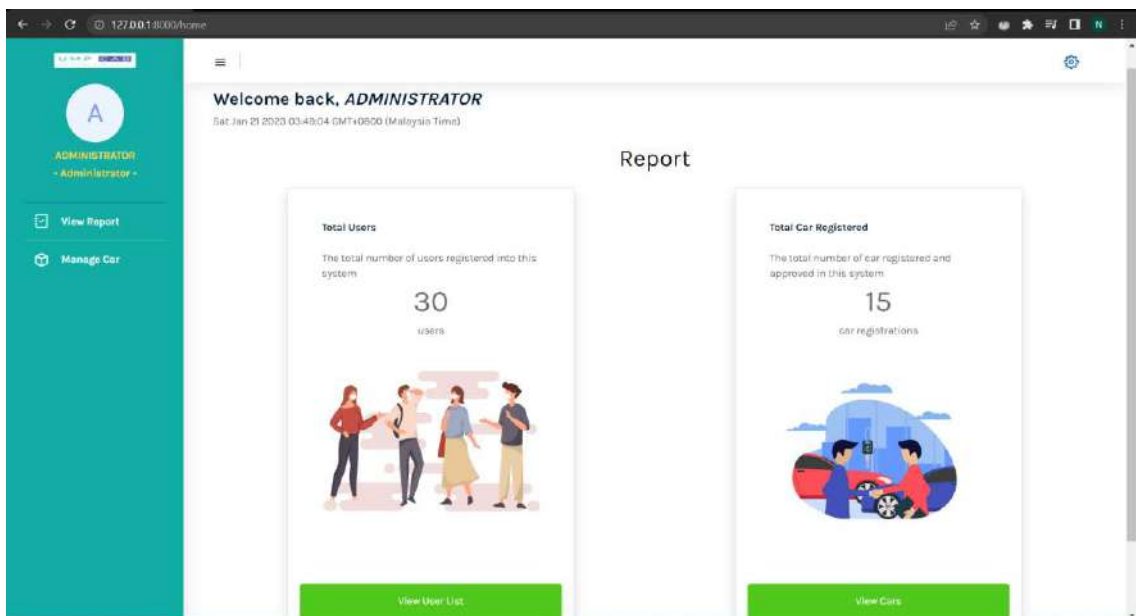
Users will then be directed to the Cab or Car Review page where they can rate and review the car as shown in the figure below.



**Figure 4.3.17 Car Rental Review User Interface**

### 4.3.5 Manage Users

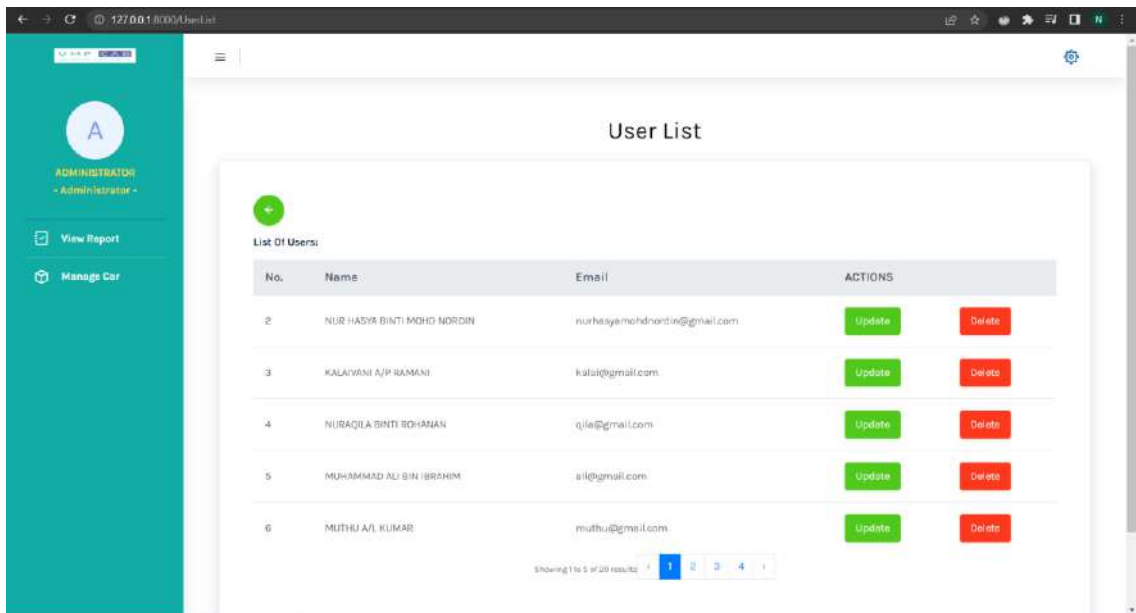
The Admin of the system can manage the users of the system which are the users and the cars registered into the system. These actions can be done by clicking *View User List*, *View Car Rental List* and the *View Cab Service List* button on the Admin Dashboard.



**Figure 4.3.18 Admin Dashboard User Interface**

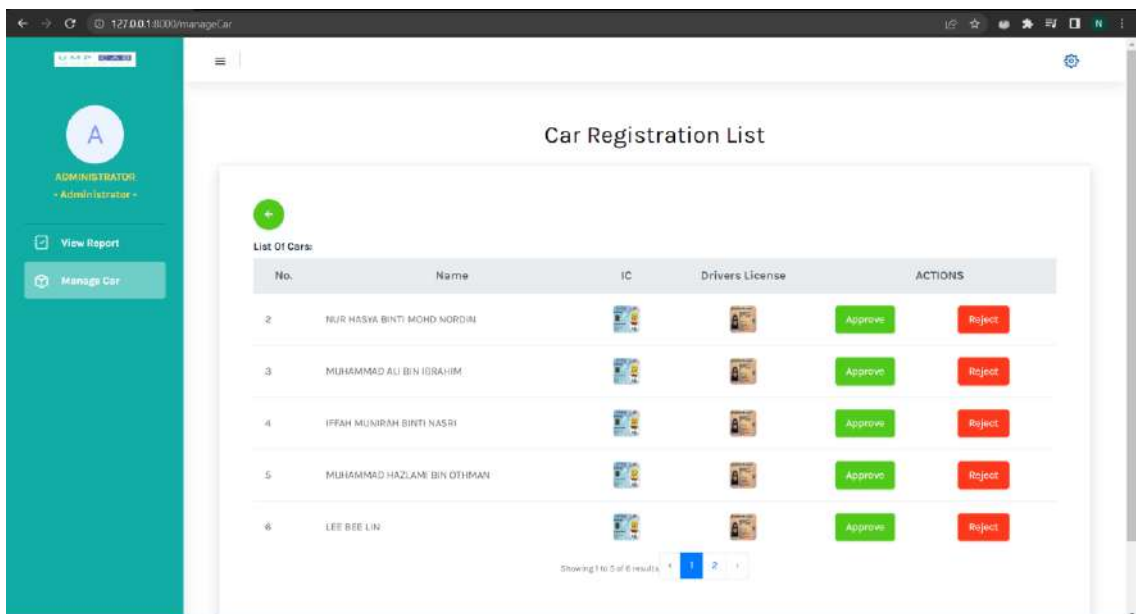
For example, the Admin wants to manage the users so they need to click on the *View Users List* button and they will be redirected to the User List page as shown below.





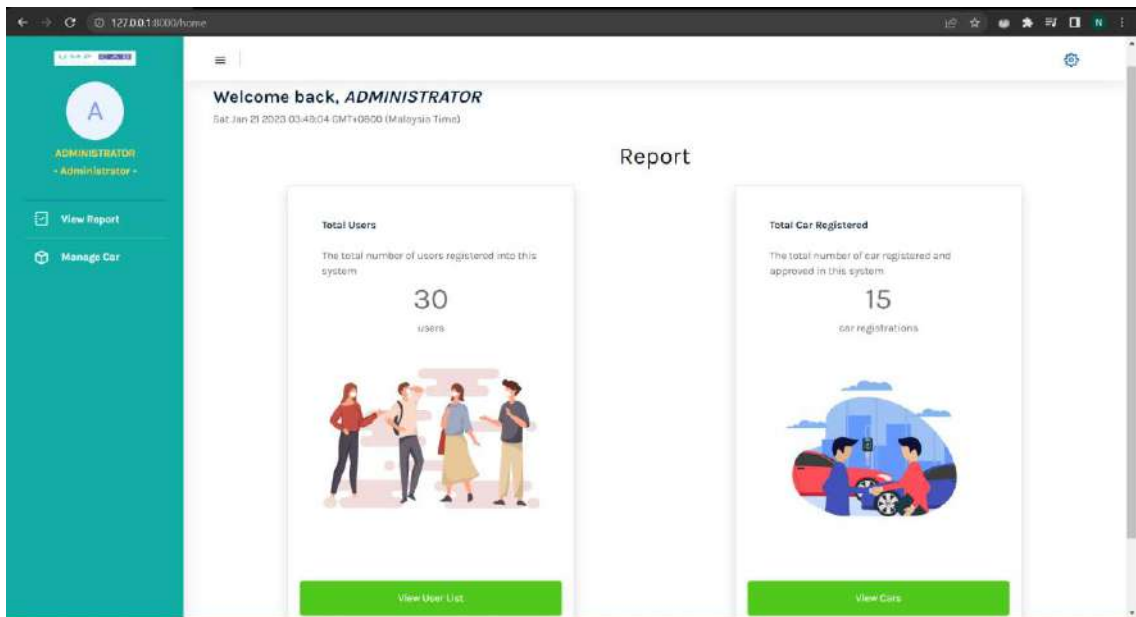
**Figure 4.3.19 User List User Interfaces**

The Admin can also manage the car registration request by going to the *Manage Car* button in the sidebar which will redirect them to the *Car Registration List* page.



### 4.3.6 Manage Report

The Admin of the system can view the overall report of the system which are the total number of users registered into the system, the total number of car rentals registered into the system and also the carpool services registered into the system. The report can be viewed on the Admin's dashboard as shown in the figure below.



**Figure 4.3.20 Admin Dashboard User Interface**

#### 4.4 TESTING RESULTS AND DISCUSSION

After the development process is complete, the testing process is applied to evaluate the usability and effectiveness of the application. User Acceptance Testing (UAT) is used to determine the applications' functionalities, in which the user tests all the system's functionalities to see if they are working as it should. To ascertain whether the system complies with the requirements is the main goal of this phase of system testing. There are numerous testing levels that can be utilised in software testing to look at behaviour and performance. These testing levels serve as a bridge between the various stages of the development lifecycle to fill in any gaps that may exist. User Acceptance Testing (UAT) is a testing procedure used to gather input and feedback from the system's users.

The results of the UAT indicate that the system's functionalities are operational and functioning properly. The UAT findings can be found in *APPENDIX C* of this document. Based on the UAT, the testing results indicates that the system's success rate surpasses the system's failure rate which is the success rate is 95.45% while the failure rate is 4.55%. Based on the results, this system successfully follows the requirements.

While testing and reviewing the usability and effectiveness of the system, users are provided with a questionnaire through a Google Form (refer to *APPENDIX D*) once they have gone through the system. The questionnaire was answered by 20 users which are the UMP students in which the questionnaire consists of 10 questions. The questions are using the System Usability Scale (SUS) survey. The SUS survey is a 10-item questionnaire designed to measure users' perceived usability of a product or system. These questions must be answered by the users from a scale of strongly disagree, disagree, neutral, agree, and strongly agree.

The SUS score can be calculated by subtracting the scale position from 1 on all oddly numbered items, and subtract 5 from the scale position on all evenly numbered items, then multiply the sum of all items by 2.5 to get an overall SUS score that ranges from 0-100. Based on the responses obtained from the survey, the SUS score calculation is as shown in *APPENDIX E*. Based on the SUS Score obtained in *APPENDIX E* which is 83.13, it can be concluded that this system's usability is excellent which means that the users of this system can use this system with ease.

## **CHAPTER 5 CONCLUSION**

### **5.1 INTRODUCTION**

The results of developing the Web-Based Car Rental and Cab Service system (UMPCab) to accomplish the goals and resolve the problems indicated in the problem description previously in Chapter 1 will be summarised in Chapter 5.

After a few years of open-distance learning (ODL), higher education institutes all over Malaysia is starting to reimplement face-to-face class for their students. This requires students to attend classes at the university. Unfortunately, not all students are able to afford their own vehicle to travel to class especially if their class is far away from their residential college or their rented houses. This approach has potential to be an excellent tool for UMP students. The Laravel Framework which employs PHP and HTML as programming languages, was utilized to create this system. Visual Studio Code is used to create the system, and GitLab is used to host the repository. To evaluate the system's usability and functionality, UMP students have used this system. The evaluation procedure shows that students offer encouraging feedback and that the application achieves its goals.

### **5.2 DISCUSSION ON USER ACCEPTANCE**

Following the completion of the development phase, the implementation and evaluation processes are carried out to test the functionality, usability, and effectiveness of this system. User Acceptance Testing (UAT) is carried out to test the functionality of the system and users must test if all the functionality is working as intended or not. APPENDIX C contains the findings of the UAT. According to the overall results, all of the accessible functions in the system is functioning properly.

For the usability and effectiveness of this system, users are provided with a questionnaire through Google form after they have gone through the system module by module. Scale-based inquiries are included in the survey. The majority of people scored highly agree and agree, per the analysis of the scale question. Therefore, it can be said that this system garnered favourable user reviews.

### **5.3 PROJECT CONSTRAINTS**

Project constraints are the factors that limit a system's ability to expand and evolve. The elements that restrict the development process are project limitations. They may be physical or immaterial. Budgetary restrictions on the project or the need for specific hardware for coding can be considered material project limits. Non-material project limitations might include everything from customer happiness to schedule limits, which can hurt or impede the project's progress.

Throughout the project's development, the following are the project's constraints:

#### **i. Time Limitation**

The development of the system is greatly influenced by time. Due to timing constraints, this system has only five modules.

#### **ii. Programming Error**

For the development of this system, programming is needed. There are instances where coding errors occur. If there were problems, it would be impossible to test or use this system. For the project to be completed, the errors must be fixed beforehand.

#### **iii. Hardware Limitation**

This system's development has been significantly hampered by the hardware, which occasionally delays. This is because the hardware needed to be temporarily turned off.

## **5.4 FUTURE WORK**

There are various enhancements that may be used to improve the Web-Based Car Rental and Cab Service system (UMPCab) in the future.

- i. The developer can add a payment module to the system. For example, allowing the use of PayPal API for the payment process.
- ii. The developer can add a dynamic geotracking service to the Manage Cab module where the passenger can see the driver's movement in the map.
- iii. The developer can add a chatting module to allow the users in the system to communicate with each other directly in the system.
- iv. The developer can add multi-language support to the system to allow users of multi-racial backgrounds to use the system with ease.
- v. The developer can improve the system's customizability by adding a feature to allow users to customize the system according to their preferences.

## REFERENCES

- [1] M. E. U. H. Khan, N. Anjum, F. Arida, H. and M. M. Khan, "Hajji Tracker: Development of Web-Based GPS Tracking System for Pilgrims," in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2021.
- [2] S. A. Saad, A. ' . Badrul Hisham, M. H. I. Ishak, M. H. Mohd Fauzi, M. A. Baharudin and N. . H. Idris, "Real-time on-campus public transpotation monitoring system," in *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*, 2018.
- [3] H. A. Abdallah Dafallah, "Design and Implementation of an accurate real time GPS tracking system," in *The Third International Conference on e-Technologies and Networks for Development (ICeND2014)*, 2014.
- [4] A. J. Syed, S. Saba, Z.-u.-A. M. N. and A. Talib, "CABTAB - A Factual Analysis Concerned with Travelling Issues, Specifically for An Organization," in *2020 International Conference on Information Science and Communication Technology (ICISCT)*, 2020.
- [5] F. S. Hsieh, "Car Pooling Based on Trajectories of Drivers and Requirements of Passengers," in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, 2017.
- [6] M. Tang, S. Ow, W. Chen, K. W. Lye and Y. Pan, "The Data and Science behind GrabShare Carpooling," in *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2017.
- [7] I. Vanderlei, J. Araujo, R. Rocha, G. Silva, F. Pacheco and J. Dantas, "Analysis of Laravel Framework Security Techniques Against Web Application Attacks," in *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, 2021.
- [8] N. Yadav, D. S. Rajpoot and S. K. Dhakad, "LARAVEL: A PHP Framework for E-Commerce Website," in *2019 Fifth International Conference on Image Information Processing (ICIIP)*, 2019.
- [9] S. H. Mahi, U. H. Maliha and S. Sakib, "Development of Web and Mobile Application Based Online Buy, Sell and Rent Car System," in *2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA)*, 2020.
- [10] G. V. Bhargavi, J. P. Vitesh, T. G. Chand, B. Srilekha and C. Gunturu, "Practical Rental System for Harvesters with GPS Tracking," in *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, 2021.
- [11] F. Y. H. Ahmed, E. Hazlan and M. I. Abdulla, "Enhancement of Mobile-Based Application for Vehicle Rental," in *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2021.

- [12] M. Maiouak and T. Taleb, "A Dynamic Map-based Framework for Real-Time Mapping of Vehicles and their Surroundings," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019.



## **APPENDIX A**

### **SOFTWARE REQUIREMENT SPECIFICATION (SRS)**

2022

# SOFTWARE REQUIREMENT SPECIFICATION (SRS)

[WEB-BASED CAR RENTAL AND CAB  
SERVICE SYSTEM]



DOCUMENT APPROVAL

	Name	Date
<p><b>Authenticated by:</b></p>  <p>_____</p> <p>NUR HASYA BINTI MOHD NORDIN</p>	<p>NUR HASYA BINTI MOHD NORDIN</p>	
<p><b>Approved by:</b></p>  <p>_____</p> <p>Client</p>		

Software :

Archiving Place :

## TABLE OF CONTENT

CONTENT	PAGE
<b>DOCUMENT APPROVAL .....</b>	<b>ii</b>
<b>TABLE OF CONTENT.....</b>	<b>iii</b>
<b>LIST OF FIGURES .....</b>	<b>v</b>
<b>LIST OF TABLES .....</b>	<b>vii</b>
<b>LIST OF APPENDICES .....</b>	<b>viii</b>
<b>1.1 PROJECT DESCRIPTION.....</b>	<b>1</b>
<b>1.2 SYSTEM IDENTIFICATION .....</b>	<b>3</b>
<b>1.3 CONTEXT DIAGRAM .....</b>	<b>3</b>
<b>1.4 DATA FLOW DIAGRAM.....</b>	<b>6</b>
<b>2.1 USE CASE DIAGRAM AND DESCRIPTION .....</b>	<b>7</b>
2.1.1 Use Case Diagram .....	7
2.1.2 Use Case Description.....	8
2.1.2.1 Manage Car Rental .....	8
2.1.2.2 Manage Cab Service .....	11
2.1.2.3 Manage Car Review.....	14
2.1.2.4 Manage Users.....	16
2.1.2.5 Manage Report.....	20
<b>2.2 SEQUENCE DIAGRAM.....</b>	<b>21</b>
2.2.1 Manage Car Rental .....	21
2.2.2 Manage Cab Service .....	25
2.2.3 Manage Car Review.....	29
2.2.4 Manage Users.....	32
2.2.5 Manage Report.....	38
<b>3.1 INTERFACE DESIGN .....</b>	<b>39</b>
3.1.1 Register and Login.....	39
3.1.2 Mainpage .....	41
3.1.3 Dashboard.....	41
3.1.4 My Booking.....	42

3.1.5	Driver Dashboard .....	43
3.1.6	Report .....	45
3.1.7	Car Registration.....	47
3.1.8	Car Rental Booking .....	48
3.1.9	Cab Registration .....	51
3.1.10	Cab Booking .....	52
3.1.11	Car Rental Review .....	53
3.1.12	Cab Review.....	54
<b>3.2</b>	<b>HARDWARE AND SOFTWARE SPECIFICATION.....</b>	<b>55</b>
3.2.1	Hardware Specifications.....	55
3.2.2	Software Specifications .....	55

## LIST OF FIGURES

Figure 1.3 Context Diagram .....	<b>Error! Bookmark not defined.</b>
Figure 1.4 Data Flow Diagram .....	<b>Error! Bookmark not defined.</b>
Figure 2.1.2.1.1 Use Case Diagram .....	7
Figure 2.2.1.1 Manage Car Rental Basic Flow (Register Car) .....	21
Figure 2.2.1.2 Manage Car Rental Basic Flow (Rent Car).....	23
Figure 2.2.2.1 Manage Cab Service Basic Flow (Register Cab) .....	<b>Error! Bookmark not defined.</b>
Figure 2.2.2.2 Manage Cab Service Basic Flow (Book Cab).....	27
Figure 2.2.2.3 Manage Cab Service Alternative Flow (Customer Enter Location) .....	28
Figure 2.2.3.1 Manage Car Review Basic Flow (Review Car Rental).....	29
Figure 2.2.3.2 Manage Car Review Basic Flow (Review Cab Service).....	31
Figure 2.2.4.1 Manage Users Basic Flow (Edit User List).....	32
Figure 2.2.4.2 Manage Users Basic Flow (Delete User) .....	33
Figure 2.2.4.3 Manage Users Basic Flow (Edit Car Rental List) .....	34
Figure 2.2.4.4 Manage Users Basic Flow (Delete Car Rental).....	35
Figure 2.2.4.5 Manage Users Basic Flow (Edit Cab Service List) .....	<b>Error! Bookmark not defined.</b>
Figure 2.2.4.6 Manage Users Basic Flow (Delete Cab Service) .....	<b>Error! Bookmark not defined.</b>
Figure 2.2.5.1 Manage Report Basic Flow (View Report).....	38
Figure 2.2.5.1 Register Interface.....	39
Figure 2.2.5.2 Login Interface .....	40
Figure 2.2.5.3 Forgot Password Interface .....	40
Figure 2.2.5.1 Mainpage Interface .....	41
Figure 2.2.5.1 Dashboard Interface.....	42
Figure 2.2.5.1 My Booking Interface.....	42
Figure 2.2.5.1 Driver Dashboard Interface .....	43

Figure 2.2.5.2 Registered Cars Interface .....44

Figure 2.2.5.3 Cab Service Registered Interface .....44

Figure 3.1.7.1 Report Interface .....45

Figure 3.1.7.2 User List Interface .....46

Figure 3.1.7.3 Car Rental List Interface .....46

Figure 3.1.7.4 Cab Service List Interface .....47

Figure 3.1.8.1 Register Car for Rental Interface.....47

Figure 3.1.9.1 Car Rental Booking Interface .....48

Figure 3.1.9.2 Search Result Interface .....49

Figure 3.1.9.3 Car Details Interface .....49

Figure 3.1.9.4 Driver Details Interface .....50

Figure 3.1.9.5 Rental Details Interface .....50

Figure 3.1.9.7 Confirm Rental Interface .....51

Figure 3.1.10.1 Register Car for Cab Service Interface.....51

Figure 3.1.11.1 Cab Service Homepage Interface .....52

Figure 3.1.11.2 Cab Details Interface .....53

Figure 3.1.12.1 Car Rental Review Interface .....53

Figure 3.1.13.1 Cab Review Interface .....54

**LIST OF TABLES**

Table 1.1 Description and Actors Involved for each module ..... 1

Table 2.1.2.1 Use Case Description for Manage Car Rental ..... 8

Table 2.1.2.2 Use Case Description for Manage Cab Booking ..... 11

Table 2.1.2.3 Use Case Description for Manage Car Review ..... 14

Table 2.1.2.4 Use Case Description for Manage Users ..... 16

Table 2.1.2.5 Use Case Description for Manage Report ..... 20

Table 3.2.1 Hardware Specifications ..... 55

Table 3.2.2 Software Specifications ..... 55



**LIST OF APPENDICES**

## CHAPTER 1

### 1.1 PROJECT DESCRIPTION

This system consists of five modules which Manage Car Rental, Manage Cab Service, Manage Car Review, Manage Users and Manage Report. The table below shows the description and actors involved for each module.

Table 1.1 Description and Actors Involved for each module

<b>Modules</b>	<b>Description</b>	<b>Actors Involved</b>
Manage Car Rental	<p>This module allows UMP students (Renter) who own cars to register their car up for rental in the system.</p> <p>This module also allows Rentee to update and delete their registered car.</p> <p>This module also allows UMP students (Rentee) to book for car rental offered in the system.</p>	Rentee and Renter
Manage Cab Service	<p>This module allows UMP students (Driver) to register their car for cab services in the system.</p> <p>This module also allows Driver to update and delete their registered cars.</p> <p>This module also allows UMP students (Passenger) to book for cab services offered in the system.</p>	Driver and Passenger
Manage Car Review	<p>This module allows UMP students (Rentee) to provide reviews and feedbacks for the car rental services they have booked. This module also allows UMP students (Passenger) to provide reviews and</p>	Rentee and Passenger

	feedbacks for the cab services they have ride and completed.	
Manage Users	<p>This module allows the Administrator of the system to monitor the users of the system and also the cars registered into the system.</p> <p>This module also allows the Administrator to view, modify and delete information from the system.</p> <p>This module also allows the Administrator to view a list of car registration request and approve or reject their requests.</p>	Admin
Manage Report	This module allows the Administrator of the system to view the report of the system such as the total number of users in the system and the total number of cars registered into the system.	Admin

**1.2 SYSTEM IDENTIFICATION**

System Title: Web-Based Car Rental and Cab Service System for UMP Students

System Abbreviation: UMPCab

Year: 2022

Version: Version 1

System Identification Number: **SRS-UMPCab-2022-V1**

**1.3 CONTEXT DIAGRAM**

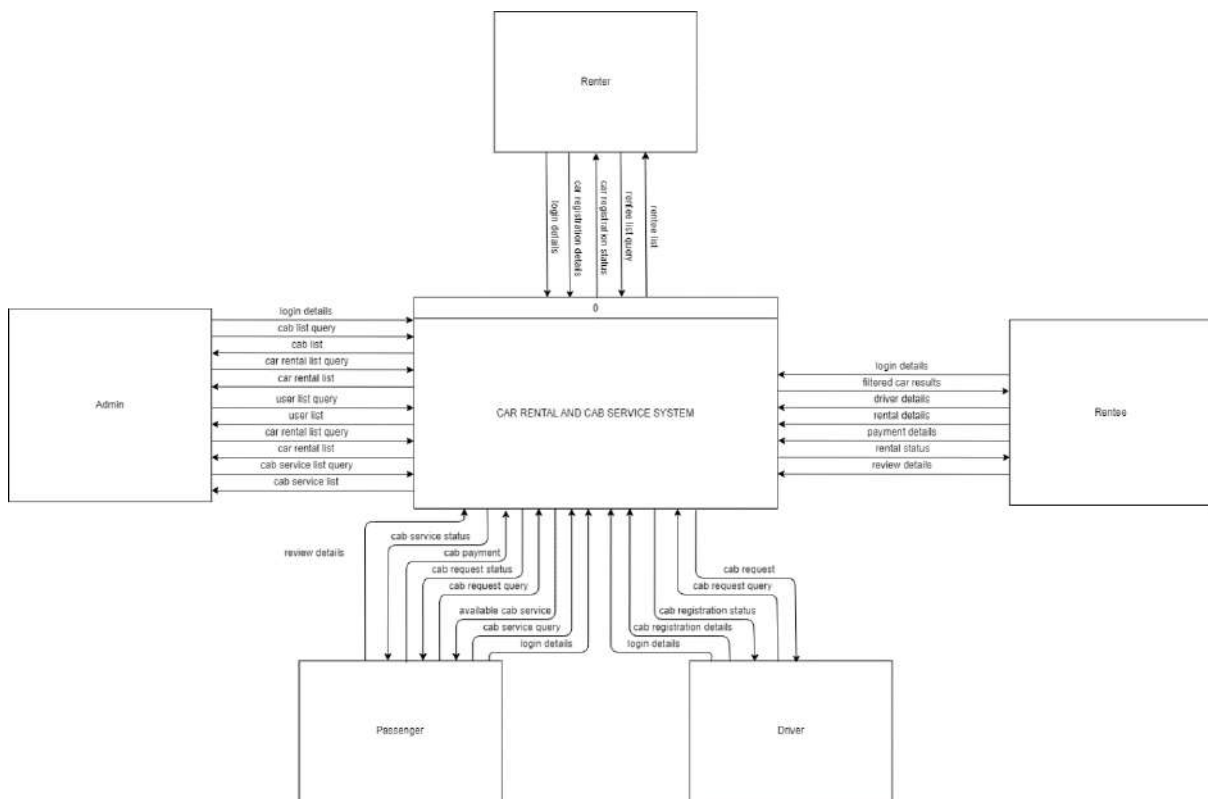


Figure 1.3 Context Diagram

The Car Rental and Cab Service System for UMP Students (UMPCab) is a web-based system developed using Laravel framework. The main purpose for the development of this project is to ease UMP Students in looking for available cars for rental around college. Students can also look for available cab services that are offered around the campus. The stakeholders involved in this system are UMP students and the administrator of the system.

UMP students are divided into 4 which are Renter, Rentee, Driver and Passenger. Renter is UMP students who wants to register their car up for rental in the system. Rentee are UMP

students who book for car rental services offered in the system. Driver are UMP students who wants to offer cab service to other students through the system. Passengers are UMP students who book for cab services offered in the system.

This system can help UMP students to reduce their time in looking for available transport to travel around the campus and also around Pahang, mainly in the Pekan and Gambang area. This system will also help students to earn some side incomes by allowing students who own cars to offer their own car rental or cab service. There will be a total of five modules for this system which Manage Car Rental, Manage Car Review, Manage Cab Service, Manage Users and Manage Report.

First of all, both users which are the administrator, and the UMP students are required to login into the system and sign up if they are not yet registered into the system. They will be required to provide their login credentials in order to access the system.

For the Admin, they can view the user list, car rental list and cab service list and make changes to the respective lists. The Admin can also view the car registration list of users who have registered into the system and either approve or reject their car registration request.

For the Renter, they can register their car up for rental in the system by providing car registration details to the system. Renter is also able to view list of Rentee who have booked their car for rental.

For the Rentee, they can browse through the car rental list by sorting the car rental list based on the car rental fare to the system. The system will then return a list of available car rentals based on their request. The Rentee need to provide the system with renter details and rental details in order to rent any cars in the system. The system will then provide the Rentee with their car rental status. Rentee can also leave reviews on the cars they have rented by providing appropriate review details.

For the Driver, they can register their car up for cab services in the system by providing cab registration details to the system. Driver can also view list of Passengers request to ride a cab based on their pickup and dropoff location and choose which cab ride request they want to accept.

For the Passenger, they can browse through the map for the places they want to go. Passengers can also provide cab service query such as their pickup and drop-off location and

then the system will save their journey request to the database. Their cab ride request will then be accepted by any available Drivers nearby. Passengers can also leave reviews on the cabs they have booked and ride by providing appropriate review details.

1.4 DATA FLOW DIAGRAM

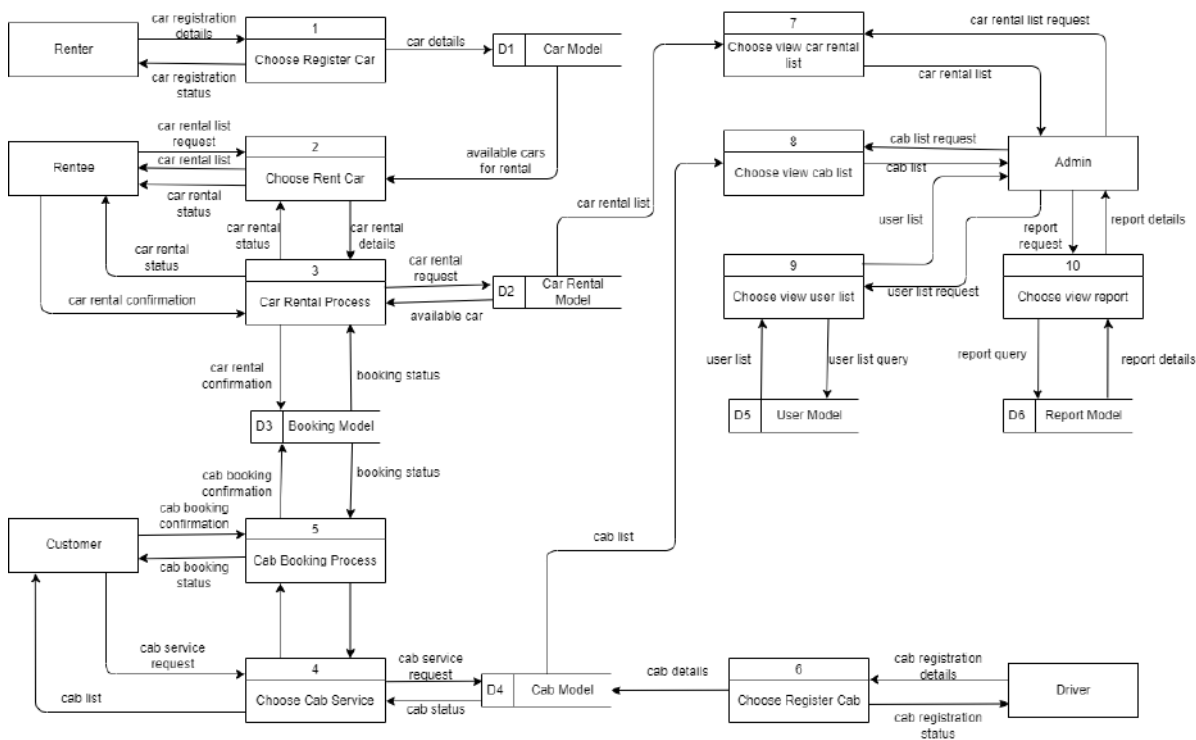


Figure 1.4.1 Data Flow Diagram

Figure 1.4.1 shows the data flow diagram of the car rental and cab service system for UMP students. The flows for each module are shown in the figure above.

## CHAPTER 2

### 2.1 USE CASE DIAGRAM AND DESCRIPTION

#### 2.1.1 Use Case Diagram

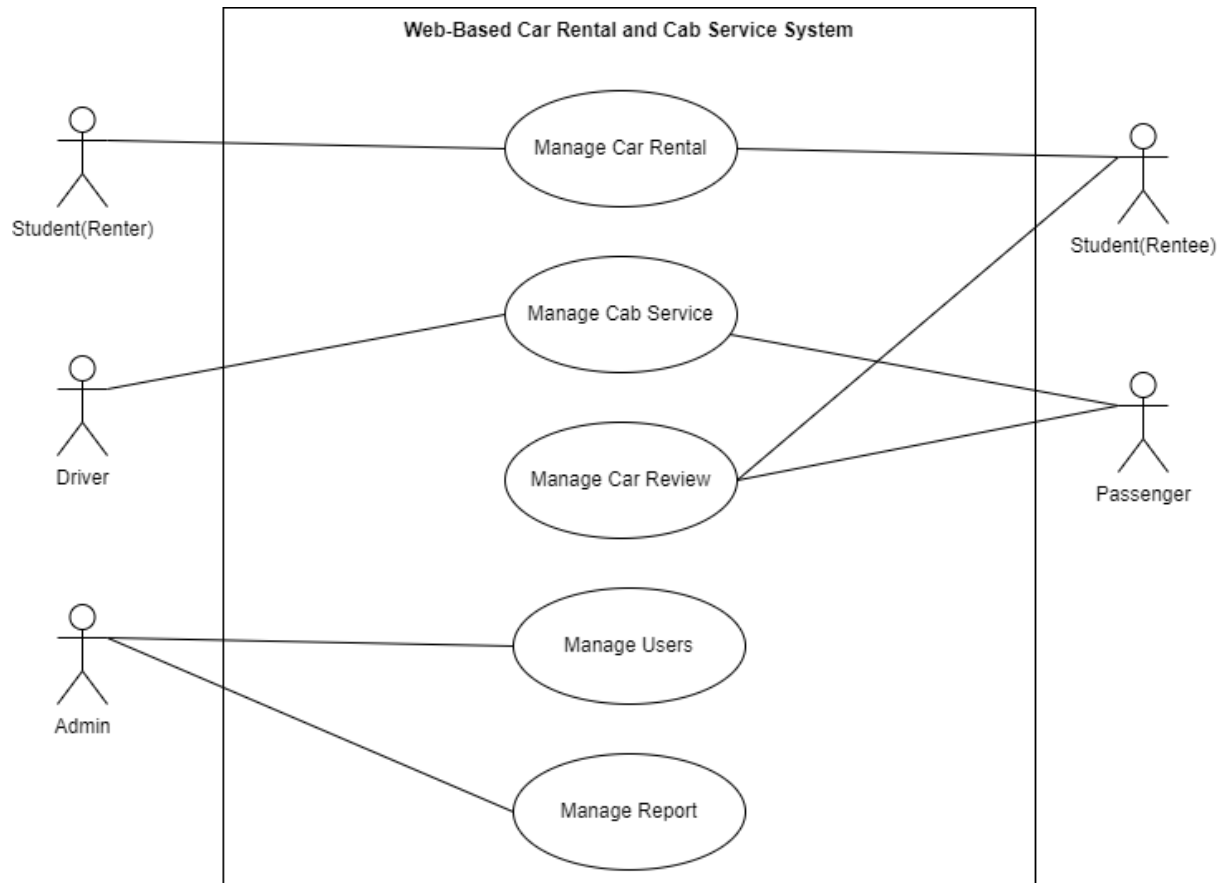


Figure 2.1.2.1.1 Use Case Diagram



## 2.1.2 Use Case Description

### 2.1.2.1 Manage Car Rental

Table 2.1.2.1 Use Case Description for Manage Car Rental

<b>Use Case Name</b>	Manage Car Rental
<b>Use Case ID</b>	UMPCab-001
<b>Brief Description</b>	This use case allows the Renter to register their car for rental into the system. This use case also allow Rentee to search and look for available cars for rental.
<b>Actor</b>	Renter and Rentee.
<b>Pre-condition</b>	The users must already be logged in into the system.
<b>Basic Flow</b>	<ol style="list-style-type: none"> <li>1) The use case begins when the Renter and Rentee is redirected to the Dashboard after log in.</li> <li>2) The Rentee can: <ol style="list-style-type: none"> <li>a. Register Cars [<b>Register Car</b>] <ol style="list-style-type: none"> <li>1) The Rentee clicks the &lt;&lt;<b>My Car</b>&gt;&gt; button on the sidebar.</li> <li>2) The system displays the My Car page.</li> <li>3) The Rentee clicks the &lt;&lt;<b>Register Car</b>&gt;&gt; button.</li> <li>4) The system displays a modal with Car Registration form.</li> <li>5) The Rentee fills in the car registration details and click &lt;&lt;<b>Register</b>&gt;&gt; button.</li> <li>6) The system saves the information into the database.</li> <li>7) The system displays the car registration status.</li> <li>8) The system displays the registered car in the My Car page.</li> </ol> </li> <li>b. Register Cars for Rental [<b>Register Car for Rental</b>] <ol style="list-style-type: none"> <li>1) The Rentee clicks the &lt;&lt;<b>Register for rental</b>&gt;&gt; button.</li> </ol> </li> </ol> </li> </ol>

	<ol style="list-style-type: none"><li>2) The system prompts user to confirm their register car for rental request.</li><li>3) The Rentee clicks the &lt;&lt;<b>OK</b>&gt;&gt; button.</li><li>4) The system displays the car rental registration status.</li></ol> <p>c. View Car Rental Request history [<b>View Rental History</b>]</p> <ol style="list-style-type: none"><li>1) The Rentee clicks the &lt;&lt;<b>User Request</b>&gt;&gt; button in the sidebar.</li><li>2) The system retrieves data from the database.</li><li>3) The system displays the User Request page.</li><li>4) The Rentee clicks the &lt;&lt;<b>History</b>&gt;&gt; button.</li><li>5) The system retrieves data from the database.</li><li>6) The system displays a modal with the list of users who have rented their cars.</li></ol> <p>d. Edit Car [<b>Edit Car</b>]</p> <ol style="list-style-type: none"><li>1) The Rentee clicks the &lt;&lt;<b>Edit</b>&gt;&gt; button.</li><li>2) The system retrieves car data from database.</li><li>3) The system displays the car details in the Update Car Details page.</li><li>4) The Rentee edit the existing details of the car.</li><li>5) The Rentee clicks the &lt;&lt;<b>Update</b>&gt;&gt; button.</li><li>6) The system verifies request and details.</li><li>7) The system saves and update data to database.</li><li>8) The system displays car update status.</li></ol> <p>e. Delete Car [<b>Delete Car</b>]</p> <ol style="list-style-type: none"><li>1) The Rentee clicks the &lt;&lt;<b>Delete</b>&gt;&gt; button.</li><li>2) The system prompts user to confirm delete car request.</li><li>3) The Rentee clicks the &lt;&lt;<b>OK</b>&gt;&gt; button.</li><li>4) The system verifies delete request.</li><li>5) The system delete data from database.</li><li>6) The system displays car deletion status.</li></ol> <p>3) The Rentee can:</p>
--	---

	<p>a. Browse and book for car rental [<b>Rent Cars</b>]</p> <ol style="list-style-type: none"> <li>1) The Rentee clicks on the &lt;&lt;<b>Book Car for Rental</b>&gt;&gt; button.</li> <li>2) The system retrieves a list of available cars for rental from the database.</li> <li>3) The system displays the list of available cars for rental.</li> <li>4) The Rentee clicks on the &lt;&lt;<b>View</b>&gt;&gt; button on any car of their choice.</li> <li>5) The system retrieves car details from the database.</li> <li>6) The system displays the car details.</li> <li>7) The system displays car rental booking form.</li> <li>8) The Renter fills in rental and renter details.</li> <li>9) The Rentee clicks on the &lt;&lt;<b>Book This Car</b>&gt;&gt; button.</li> <li>10) The system saves the information into the database and displays car rental status.</li> <li>11) The system displays car rental booking status.</li> <li>12) The system displays the My Bookings list.</li> </ol> <p>4) The use case ends.</p>
<b>Alternative Flow</b>	None
<b>Exception Flow</b>	None
<b>Rules</b>	None
<b>Constraints</b>	None
<b>Post Condition</b>	The system successfully displays the car registration status and car rental booking status.

## 2.1.2.2 Manage Cab Service

Table 2.1.2.2 Use Case Description for Manage Cab Booking

<b>Use Case Name</b>	Manage Cab Service
<b>Use Case ID</b>	UMPCab-002
<b>Brief Description</b>	This use case allows the Driver to register their cars for cab service in the system. This use case also allows Passenger to look for available cab services.
<b>Actor</b>	Driver and Passenger.
<b>Pre-condition</b>	The users must already be logged in into the system.
<b>Basic Flow</b>	<ol style="list-style-type: none"> <li>1) The use case begins when the Driver and Passenger is redirected to the Dashboard after log in.</li> <li>2) The Driver can: <ol style="list-style-type: none"> <li>a. Register cars [<b>Register Car</b>] <ol style="list-style-type: none"> <li>1) The Driver clicks the &lt;&lt;My Car&gt;&gt; button on the sidebar.</li> <li>2) The system displays the My Car Page.</li> <li>3) The Driver clicks the &lt;&lt;Register Car&gt;&gt; button.</li> <li>4) The system displays a modal with Car Registration form.</li> <li>5) The Driver fills in the car registration details and click &lt;&lt;Register&gt;&gt; button.</li> <li>6) The system saves the information into the dataset.</li> <li>7) The system displays the car registration status.</li> <li>8) The system displays the registered car in the My Car page.</li> </ol> </li> <li>b. Register Cars for Cab Service [<b>Register Car for Cab Service</b>] <ol style="list-style-type: none"> <li>1) The Driver clicks the &lt;&lt;Register for carpool&gt;&gt; button.</li> </ol> </li> </ol> </li> </ol>

	<ol style="list-style-type: none"><li>2) The system prompts user to confirm their register car for carpool request.</li><li>3) The Driver clicks the &lt;&lt;<b>OK</b>&gt;&gt; button.</li><li>4) The system displays the car rental registration status.</li></ol> <p>c. Accept Cab Request [<b>Accept Cab Request</b>]</p> <ol style="list-style-type: none"><li>1) The Driver clicks the &lt;&lt;<b>User Requests</b>&gt;&gt; button in the sidebar.</li><li>2) The system retrieves car requests from database.</li><li>3) The system displays cab request list in the User Request page.</li><li>4) The Driver clicks the &lt;&lt;<b>View Request</b>&gt;&gt; button.</li><li>5) The system retrieves cab request details from the database.</li><li>6) The system displays a modal with cab request details of chosen cab request.</li><li>7) The Driver clicks the &lt;&lt;<b>Accept Request</b>&gt;&gt; button.</li><li>8) The system updates and save data of status of cab request to the database.</li><li>9) The system displays the cab request page.</li><li>10) The Driver clicks the &lt;&lt;<b>Ride Completed</b>&gt;&gt; button.</li><li>11) The system updates and save the status of the cab ride to the database.</li><li>12) The system displays status of cab ride.</li><li>13) The system displays the User Request page.</li></ol> <p>d. Edit Car [<b>Edit Car</b>]</p> <ol style="list-style-type: none"><li>1) The Driver clicks the &lt;&lt;<b>Edit</b>&gt;&gt; button.</li><li>2) The system retrieves car data from database.</li><li>3) The system displays the car details in the Update Car Details page.</li><li>4) The Driver edits the existing details of the car.</li><li>5) The Driver clicks the &lt;&lt;<b>Update</b>&gt;&gt; button.</li><li>6) The system verifies request and details.</li><li>7) The system saves and update data to database.</li><li>8) The system displays car update status.</li></ol>
--	---

e. Delete Car [**Delete Car**]

- 1) The Driver clicks the <<**Delete**>> button.
- 2) The system prompts user to confirm delete car request.
- 3) The Driver clicks the <<**OK**>> button.
- 4) The system verifies delete request.
- 5) The system delete data from database.
- 6) The system displays car deletion status.

f. View Cab Request History [**View Cab History**]

- 1) The Driver clicks the <<**User Request**>> button in the sidebar.
- 2) The system retrieves data from the database.
- 3) The system displays the User Request page.
- 4) The Driver clicks the <<**History**>> button.
- 5) The system retrieves data from the database.
- 6) The system displays a modal with the a list of cab request they have completed.

## 3) The Passenger can:

a. Book Cab Service [**Book Cab**]

- 1) The Passenger clicks on the <<**Book a carpool**>> button.
- 2) The system displays a map with cab service request form.
- 3) The Passenger fill in the cab service request form. [**A1: Passenger search location**]
- 4) The Passenger clicks on <<**Submit**>> button.
- 5) The system saves the information into the database.
- 6) The system displays cab request status.
- 7) The system displays the My Bookings List.

## 4) The use case ends.

<b>Alternative Flow</b>	<b>A1: Passenger enter location</b> <ol style="list-style-type: none"> <li>1) The Passenger fills in the location details and clicks the &lt;&lt;<b>Search</b>&gt;&gt; button.</li> <li>2) The system displays a marker based on the search results.</li> <li>3) The use case continues to Basic Flow step (3)(a)(4).</li> </ol>
<b>Exception Flow</b>	None
<b>Rules</b>	None
<b>Constraints</b>	None
<b>Post Condition</b>	The system successfully displays the cab registration and cab booking status.

### 2.1.2.3 Manage Car Review

Table 2.1.2.3 Use Case Description for Manage Car Review

<b>Use Case Name</b>	Manage Car Review
<b>Use Case ID</b>	UMPCab-003
<b>Brief Description</b>	This use case allows Passenger to give review on the cab service they have ride on. This use case also allows Rentee to give review on the car they have rented.
<b>Actor</b>	Passenger and Rentee
<b>Pre-condition</b>	The users must already be logged in into the system.
<b>Basic Flow</b>	<ol style="list-style-type: none"> <li>1) The use case begins when the Passenger and Rentee clicks on the &lt;&lt;<b>My Booking</b>&gt;&gt; button in the sidebar.</li> <li>2) The system retrieves a list of car rental and cab service bookings from the database.</li> <li>3) The system displays a list of car rental and cab service bookings.</li> <li>4) The Rentee can: <ol style="list-style-type: none"> <li>a. Give Review for Car Rental [<b>Review Car Rental</b>]</li> </ol> </li> </ol>

	<ol style="list-style-type: none"><li>1) The Rentee clicks on the &lt;&lt;<b>Ongoing</b>&gt;&gt; button in the bookings list with the &lt;&lt;<b>Car Rental</b>&gt;&gt; booking type.</li><li>2) The system retrieves car rental details from the database.</li><li>3) The system displays the car rental details with a form for review details.</li><li>4) The Rentee fills in the car rental review form and clicks the &lt;&lt;<b>Complete Review</b>&gt;&gt; button.</li><li>5) The system verifies user request and shows pop-up for confirmation of car rental completion status.</li><li>6) The Rentee clicks on the &lt;&lt;<b>Confirm</b>&gt;&gt; button.</li><li>7) The system updates the completion status to the database.</li><li>8) The system displays car rental completion status.</li></ol> <p>5) The Passenger can:</p> <ol style="list-style-type: none"><li>a. Give Review for Cab Service [<b>Review Cab Service</b>]<ol style="list-style-type: none"><li>1) The Passenger clicks on the &lt;&lt;<b>Rate Car Ride</b>&gt;&gt; button in the bookings list with the &lt;&lt;<b>Cab Service</b>&gt;&gt; booking type.</li><li>2) The system retrieves cab service details from the database.</li><li>3) The system displays the cab service details with a form for review details.</li><li>4) The Passenger fills in the cab service review form and clicks the &lt;&lt;<b>Complete Review</b>&gt;&gt; button.</li><li>5) The system verifies user request and shows pop-up for confirmation of cab service completion status.</li><li>6) The Passenger clicks on the &lt;&lt;<b>Confirm</b>&gt;&gt; button.</li></ol></li></ol>
--	--



	<p>7) The system updates the cab service completion status to the database.</p> <p>8) The system displays cab service completion status.</p> <p>6) The use case ends.</p>
<b>Alternative Flow</b>	None
<b>Exception Flow</b>	None
<b>Rules</b>	None
<b>Constraints</b>	None
<b>Post Condition</b>	The system successfully saved users review and updated car rental and cab service completion status to the database.

#### 2.1.2.4 Manage Users

Table 2.1.2.4 Use Case Description for Manage Users

<b>Use Case Name</b>	Manage Users
<b>Use Case ID</b>	UMPCab-004
<b>Brief Description</b>	This use case allows Admin to monitor the users of the system by giving them access to view, modify and delete information about the users and the cars.
<b>Actor</b>	Admin
<b>Pre-condition</b>	The user must already be logged in into the system.
<b>Basic Flow</b>	<ol style="list-style-type: none"> <li>1) The use case begins when the Admin is redirected to the dashboard after login.</li> <li>2) The Admin can:             <ol style="list-style-type: none"> <li>a. Edit User List [<b>Edit User List</b>]                 <ol style="list-style-type: none"> <li>1) The Admin clicks on the &lt;&lt;<b>View User List</b>&gt;&gt; button.</li> <li>2) The system retrieves data from database.</li> <li>3) The system displays a list of users.</li> </ol> </li> </ol> </li> </ol>

	<ol style="list-style-type: none"><li>4) The Admin chooses any existing record in the list and clicks &lt;&lt;<b>Update</b>&gt;&gt; button.</li><li>5) The system retrieves data from the database.</li><li>6) The system displays user details.</li><li>7) The Admin edits the details and clicks on &lt;&lt;<b>Update</b>&gt;&gt; button.</li><li>8) The system updates the information to the database.</li><li>9) The system displays update status.</li></ol> <p>b. Delete User [<b>Delete User</b>]</p> <ol style="list-style-type: none"><li>1) The Admin clicks on the &lt;&lt;<b>View User List</b>&gt;&gt; button.</li><li>2) The system displays a list of users.</li><li>3) The Admin chooses any existing record in the list and clicks &lt;&lt;<b>Delete</b>&gt;&gt; button.</li><li>4) The system verifies user request to delete data.</li><li>5) The system remove the data from the database.</li><li>6) The system displays delete status.</li></ol> <p>c. Edit Car List [<b>Edit Car</b>]</p> <ol style="list-style-type: none"><li>1) The Admin clicks on the &lt;&lt;<b>View Car Rental List</b>&gt;&gt; button.</li><li>2) The system retrieves data from the database.</li><li>3) The system displays a list of car rentals registered into the system.</li><li>4) The Admin chooses any existing record in the list and clicks &lt;&lt;<b>Update</b>&gt;&gt; button.</li><li>5) The system retrieves data from the database.</li></ol>
--	---

	<ol style="list-style-type: none"><li>6) The system displays car rental details.</li><li>7) The Admin edits the details and clicks on &lt;&lt;<b>Update</b>&gt;&gt; button.</li><li>8) The system updates the information to the database.</li><li>9) The system displays update status.</li></ol> <p>d. Delete Car [<b>Delete Car</b>]</p> <ol style="list-style-type: none"><li>1) The Admin clicks on the &lt;&lt;<b>View cars</b>&gt;&gt; button.</li><li>2) The system displays a list of cars registered into the system.</li><li>3) The Admin chooses any existing record in the list and clicks &lt;&lt;<b>Delete</b>&gt;&gt; button.</li><li>4) The system verifies user request to delete data.</li><li>5) The system remove the data from the database.</li><li>6) The system displays delete status.</li></ol> <p>e. Approve car registration request [<b>Manage Car</b>]</p> <ol style="list-style-type: none"><li>1) The Admin clicks the &lt;&lt;<b>Manage Car</b>&gt;&gt; button in the sidebar.</li><li>2) The system retrieves car registration request from database.</li><li>3) The system displays car registration request list in the Car Registration List page.</li><li>4) The Admin clicks the &lt;&lt;<b>Approve</b>&gt;&gt; button. <b>[A1: Admin Rejects Car Registration Request]</b></li><li>5) The system verifies approval and saves data inside database.</li><li>6) The system displays car registration approval status.</li></ol>
--	--

	3) The use case ends.
<b>Alternative Flow</b>	<b>A1: Admin Rejects Car Registration Request [Reject Car]</b>  1) The Admin clicks the << <b>Reject</b> >> button. 2) The system prompts Admin to confirm car registration rejection. 3) The Admin clicks the << <b>OK</b> >> button. 4) The system verifies car rejection request. 5) The system saves data to the database and display car rejection status. 6) The use case continues to Basic Flow step (2)(e)(3).
<b>Exception Flow</b>	None
<b>Rules</b>	None
<b>Constraints</b>	None
<b>Post Condition</b>	The system successfully saved and updated the user details, car details and car registration status into the database.

### 2.1.2.5 Manage Report

Table 2.1.2.5 Use Case Description for Manage Report

<b>Use Case Name</b>	Manage Report
<b>Use Case ID</b>	UMPCab-005
<b>Brief Description</b>	This use case allows Admin to view the report about the system.
<b>Actor</b>	Admin
<b>Pre-condition</b>	The user must already be logged in into the system.
<b>Basic Flow</b>	<ol style="list-style-type: none"><li>1) The use case begins when the Admin is redirected to the dashboard after login.</li><li>2) The system displays the Report page.</li><li>3) The system retrieves data from the database.</li><li>4) The system displays the report for total users and total cars registered into the system.</li><li>5) The use case ends.</li></ol>
<b>Alternative Flow</b>	None
<b>Exception Flow</b>	None
<b>Rules</b>	None
<b>Constraints</b>	None
<b>Post Condition</b>	The system successfully displays the total users and total cars registered into the system.

## 2.2 SEQUENCE DIAGRAM

For all of the sequence diagram, users must first be logged in into the system before they can continue to rent cars, book cab, register car or register cab.

### 2.2.1 Manage Car Rental

#### 2.2.1.1 Register Car Basic Flow

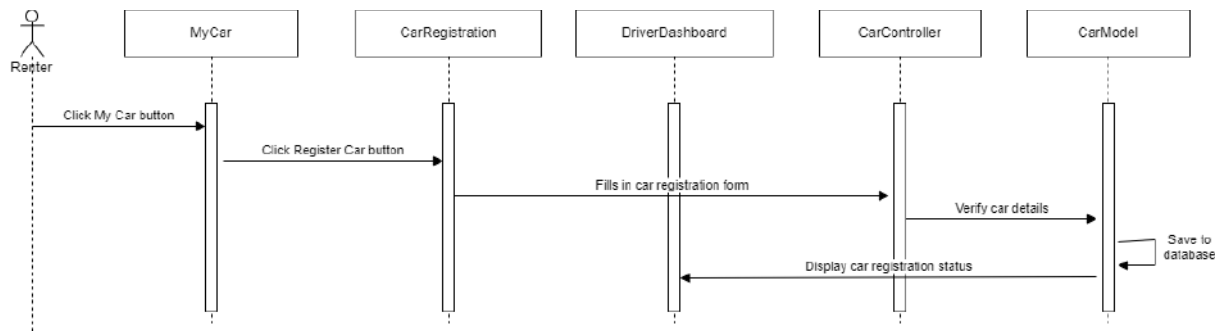


Figure 2.2.1.1 Manage Car Basic Flow (Register Car)

The Register Car Basic Flow is a sequence diagram for users who want to register their car in the system. It consists of three interfaces, one controller and one model. The interfaces are Home, CarRegistration and DriverDashboard. The Controller is the CarController and the model is the CarModel. This sequence diagram shows the basic flow for the Register Car function in the Manage Car Rental module in *2.1.2.1 Manage Car Rental Use Case Description*.

The Renter will first click on the Home button and they will be redirected to the Home interface. Then, in the Home interface, the Rentee clicks on the Register Car button and they will be required to fill in their car registration details. After they have filled in the details, their details will be verified by the CarController and their details will be saved to the database which is inside the CarModel. After that, the system will display their car registration status.

2.2.1.2 Register Car for Rental Basic Flow

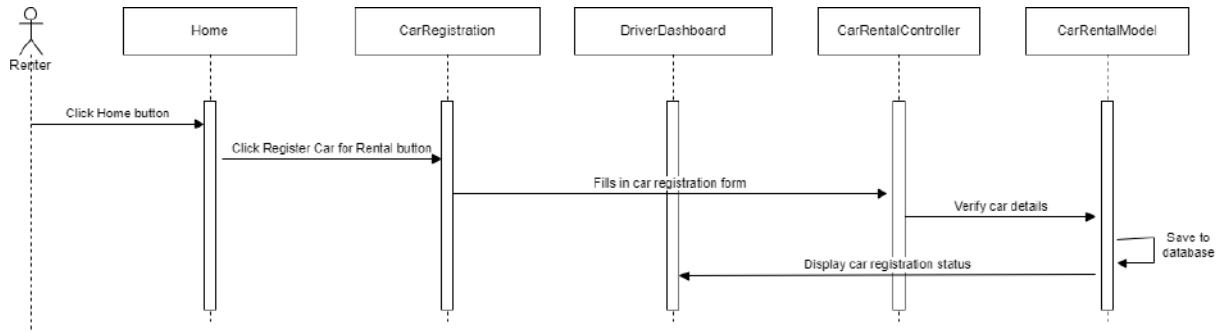


Figure 2.2.1.2 Manage Car Rental Basic Flow (Register Car for Rental)

2.2.1.3 View Rental History Basic Flow

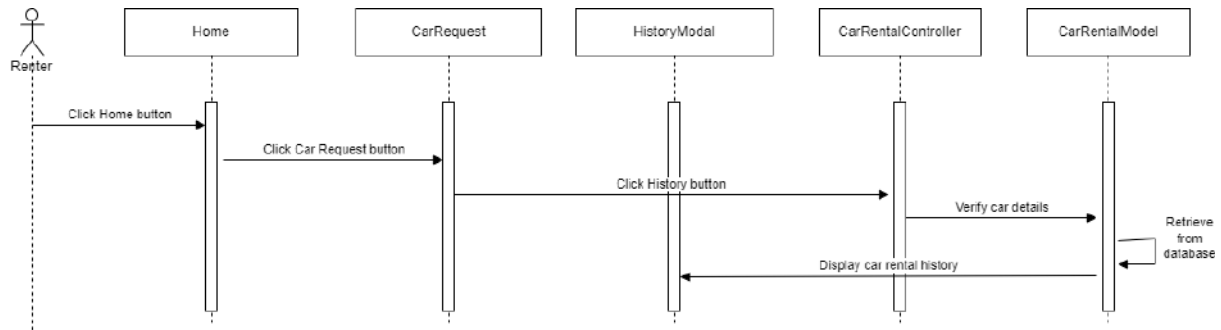


Figure 2.2.1.3 Manage Car Rental Basic Flow (View Rental History)

2.2.1.4 Edit Car Basic Flow

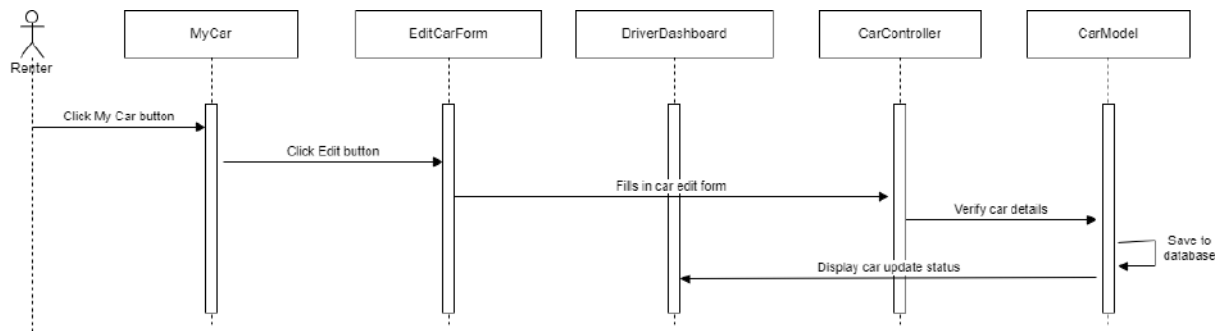


Figure 2.2.1.4 Manage Car Rental Basic Flow (Edit Car)

2.2.1.5 Delete Car Basic Flow

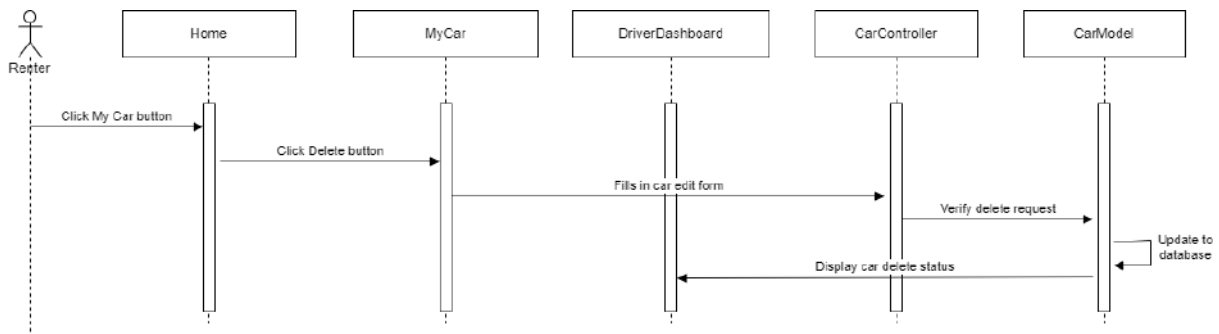


Figure 2.2.1.5 Manage Car Rental Basic Flow (Delete Car)

2.2.1.6 Rent Car Basic Flow

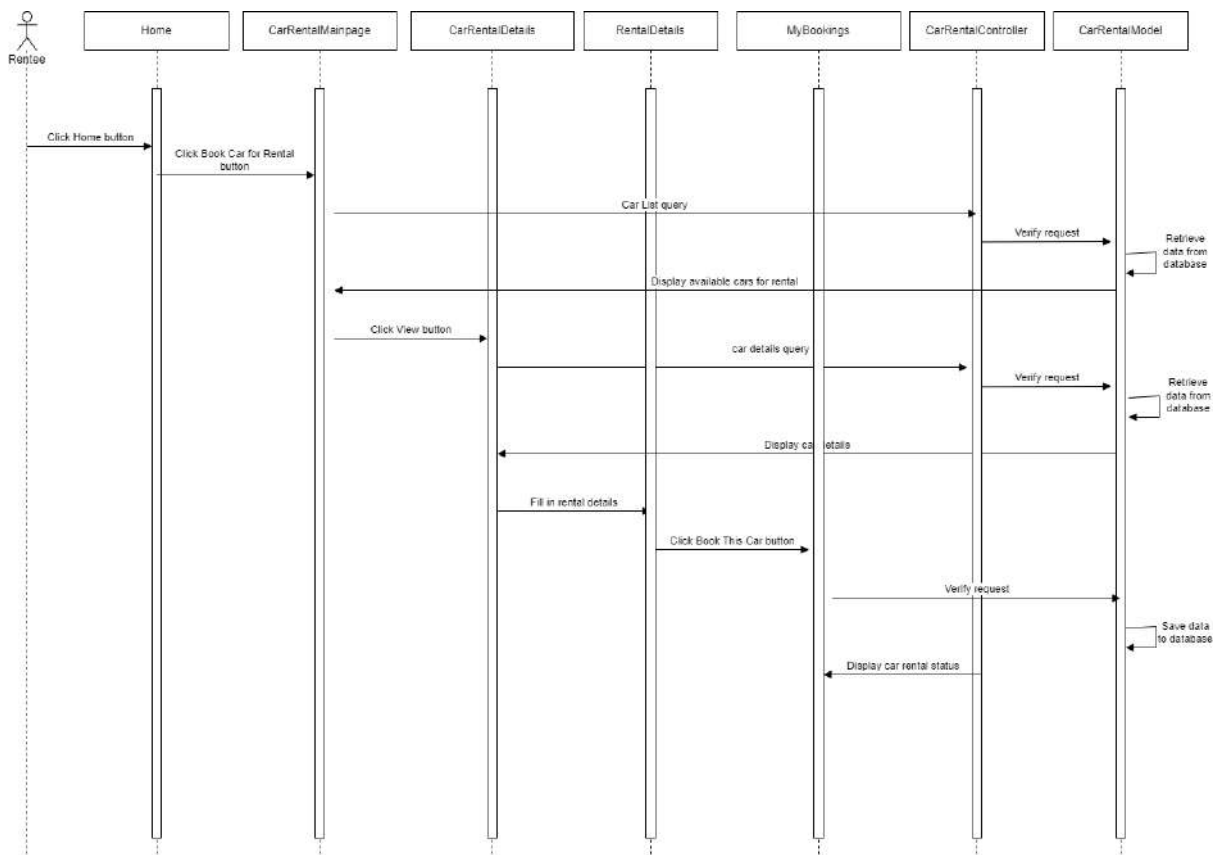


Figure 2.2.1.6 Manage Car Rental Basic Flow (Rent Car)

The Rent Car Basic Flow is a sequence diagram for users who want to browse for available car rental services and book them. It consists of eight interfaces, one controller and one model. The interfaces are Home, CarRentalMainpage, CarRentalDetails, DriverDetails, RentalDetails, ConfirmPage and MyBookings. The controller is the CarController and the model is the CarModel. This sequence diagram shows the basic flow for the Rent Car function in the Manage Car Rental module in 2.1.2.1 Manage Car Rental Use Case Description.



The Rentee will first click on the Home button which will redirect them to the Home interface. In the Home interface, they will click on the Browse Cars button in order to browse for available cars for rental. The system will then verify their request to view list of available cars for rental and retrieves data from the database. The system will then display the available cars for rental in the CarRentalMainpage interface.

Next, the Rentee will click on the View Details button to view the details of their desired car rental. The CarRentalController will verify the request and retrieves the car rental details from the CarRentalModel. The system will then display the car rental details in the CarRentalDetails interface.

In the CarRentalDetails interface, Rentee clicks on the Next button to proceed to the next page which is the RentalDetails page where they are required to fill in their rental details. Rentee will then click on the Book This Car button.

The CarRentalController will verify their booking request and save their car rental information to the database which is in the CarRentalModel. Then, the system will display their car rental status.

## 2.2.2 Manage Cab Service

### 2.2.2.1 Register Car Basic Flow

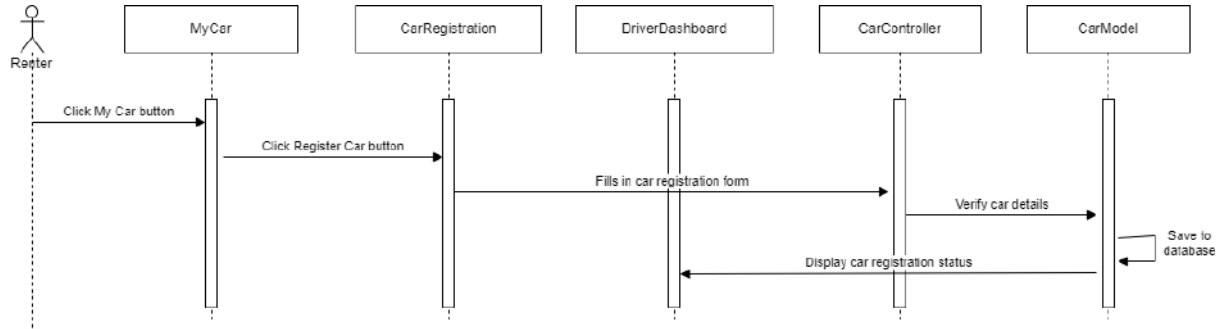


Figure 2.2.2.1 Manage Cab Service Basic Flow (Register Car)

### 2.2.2.2 Register Car for Cab Service Basic Flow

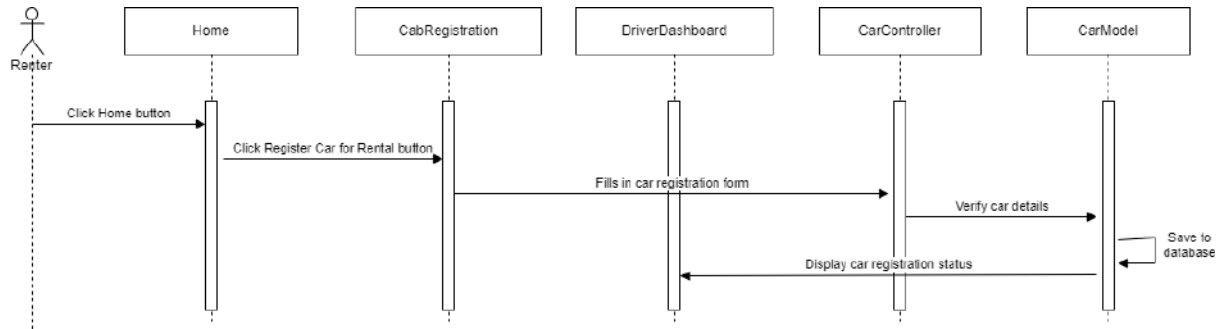


Figure 2.2.2.2 Manage Cab Service Basic Flow (Register Car for Cab Service)

### 2.2.2.3 Accept Cab Request Basic Flow

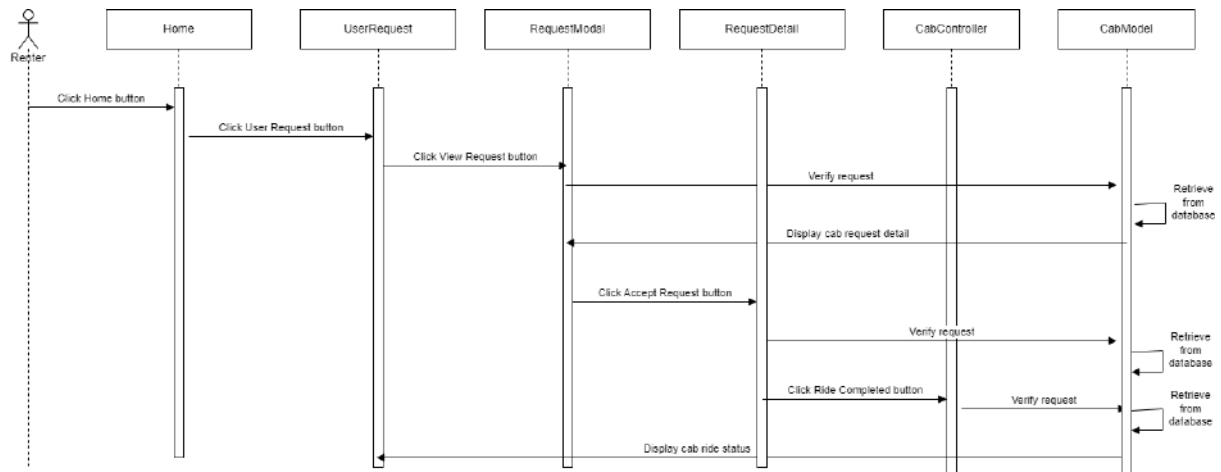


Figure 2.2.2.3 Manage Cab Service Basic Flow (Accept Cab Request)

2.2.2.4 Edit Car Basic Flow

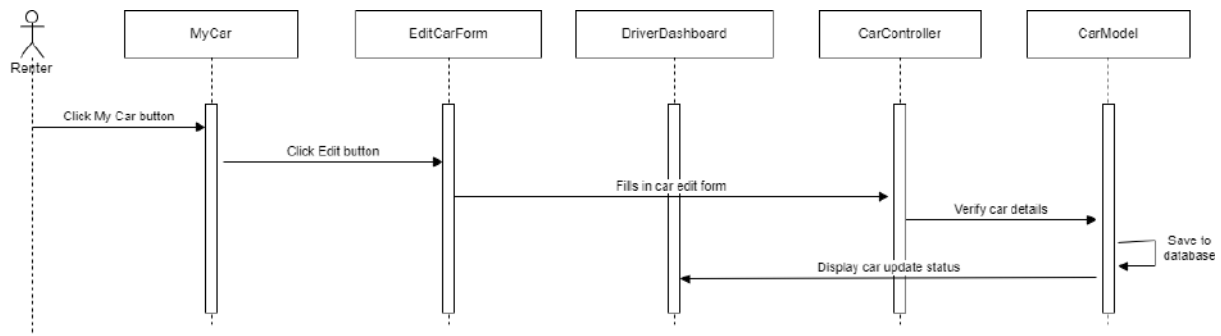


Figure 2.2.2.4 Manage Cab Service Basic Flow (Edit Car)

2.2.2.5 Delete Car Basic Flow

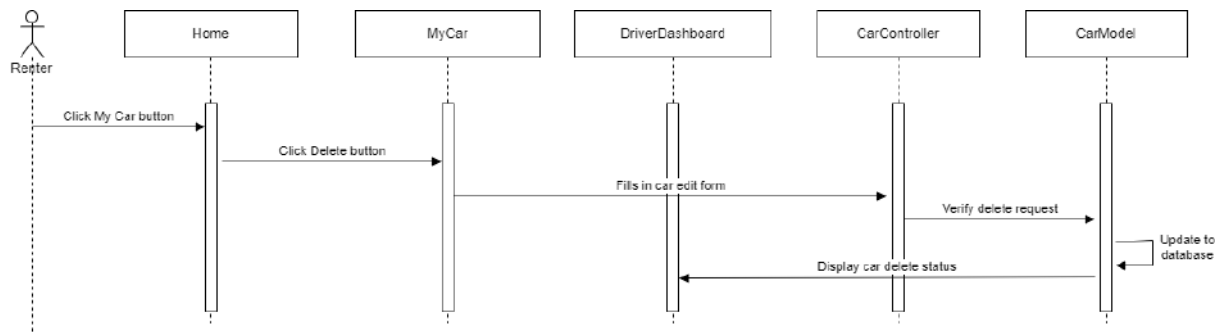


Figure 2.2.2.5 Manage Cab Service Basic Flow (Delete Car)

2.2.2.6 View Cab History Basic Flow

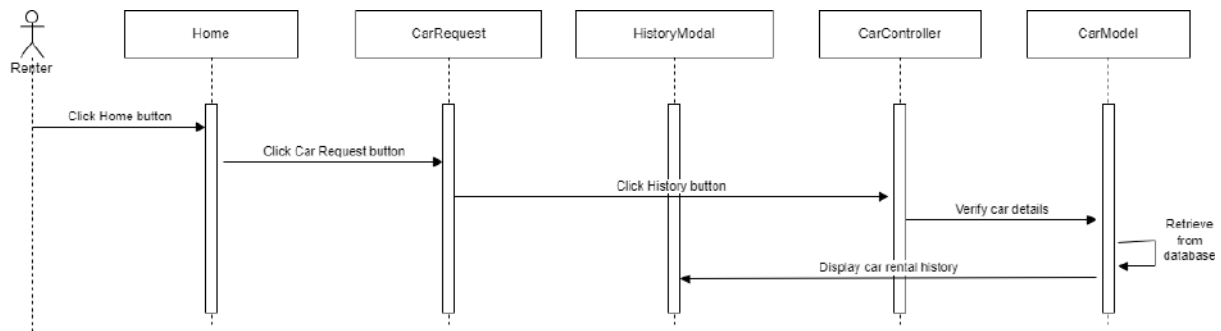


Figure 2.2.2.6 Manage Cab Service Basic Flow (View Cab History)

### 2.2.2.7 Book Cab Basic Flow

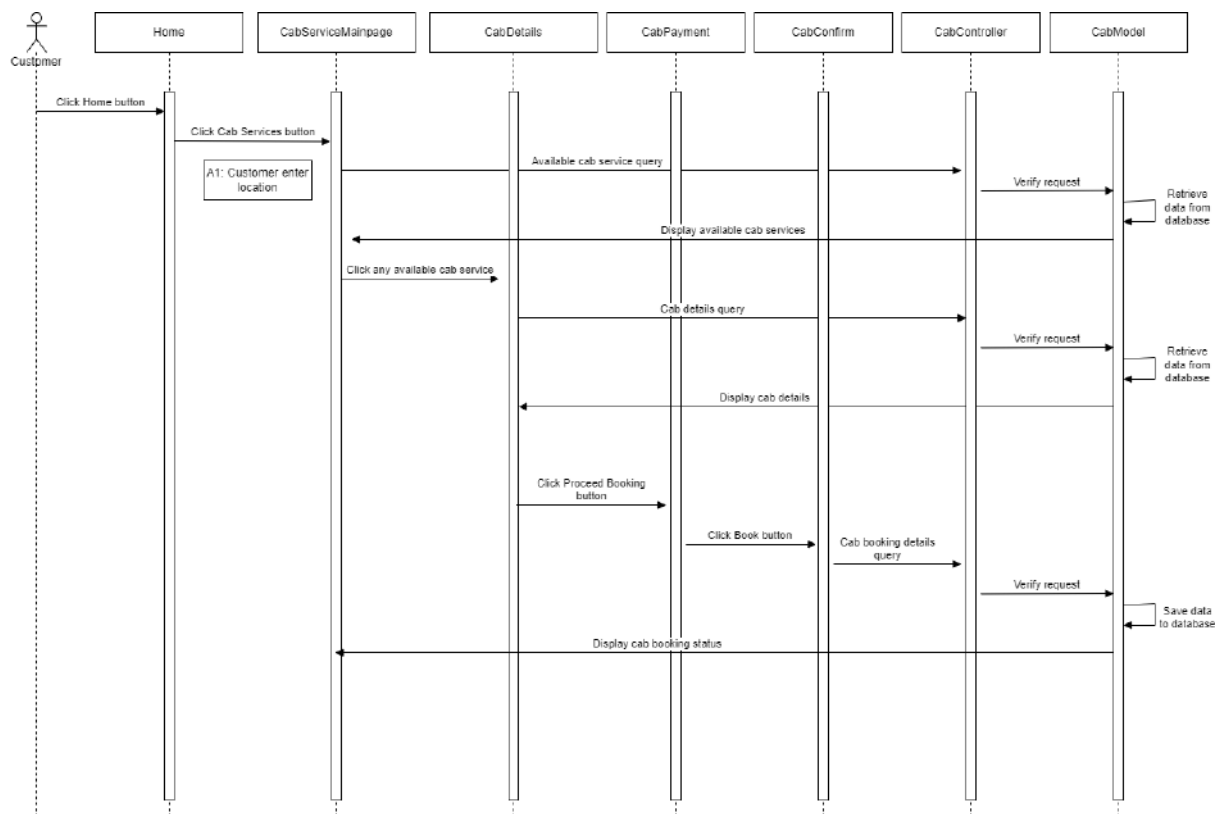


Figure 2.2.2.1 Manage Cab Service Basic Flow (Book Cab)

The Book Cab Basic Flow is a sequence diagram for users who want to book for available cab services. It consists of five interfaces, one controller, and one model. The interfaces are Home, CabServiceMainpage, CabDetails, and CabConfirm. This sequence diagram shows the basic flow for the Book Cab function in the Manage Cab Booking module in *2.1.2.2 Manage Cab Booking Use Case Description*.

The Customer will first click the Home button which will redirect them to the Home interface. In the Home interface, Customer will click on the Cab Services button and they will be redirected to the CabServiceMainpage interface. The CabController will verify user request and retrieve available cab services from the database which is the CabModel. The system will then display the available cab services in the map in the CabServiceMainpage interface.

Next, the Customer clicks on any available cab services on the map and they will be redirected to the CabDetails interface. The CabController will verify the request and retrieve the cab details from the database which is in the CabModel. The system will then display the cab details in the CabDetails interface.

Then, Customer clicks on the Book button which will redirect them to the CabConfirm interface. The CabController will then verify user request and save the cab booking details provided by the customer into the database in the CabModel. The system will then display the cab booking status.

### 2.2.2.8 Customer Enter Location Alternative Flow

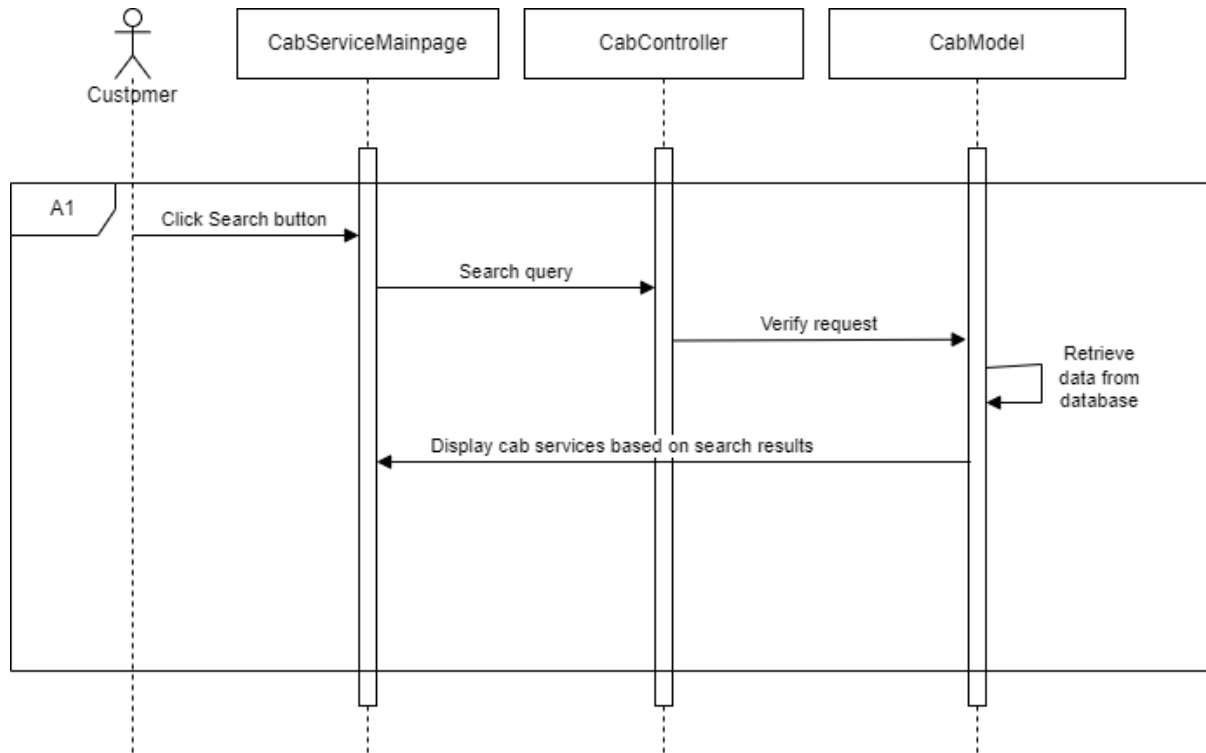


Figure 2.2.2.2 Manage Cab Service Alternative Flow (Customer Enter Location)

The Customer Enter Location Alternative Flow allow users to search for cab services based on the location they want. It consists of one interface which is the CabServiceMainpage, one controller which is the CabController and one model which is the CabModel. This sequence diagram shows the alternative flow for the Book Cab function in the Manage Cab Booking module in 2.1.2.2 *Manage Cab Booking Use Case Description*.

In the CabServiceMainpage, the Customer will first provide the location they want and then clicks the Search button. The CabController will then verify their request and then retrieves data from the database in the CabModel. The system will then display the map based on the Customer's search results.

### 2.2.3 Manage Car Review

#### 2.2.3.1 Review Car Rental Basic Flow

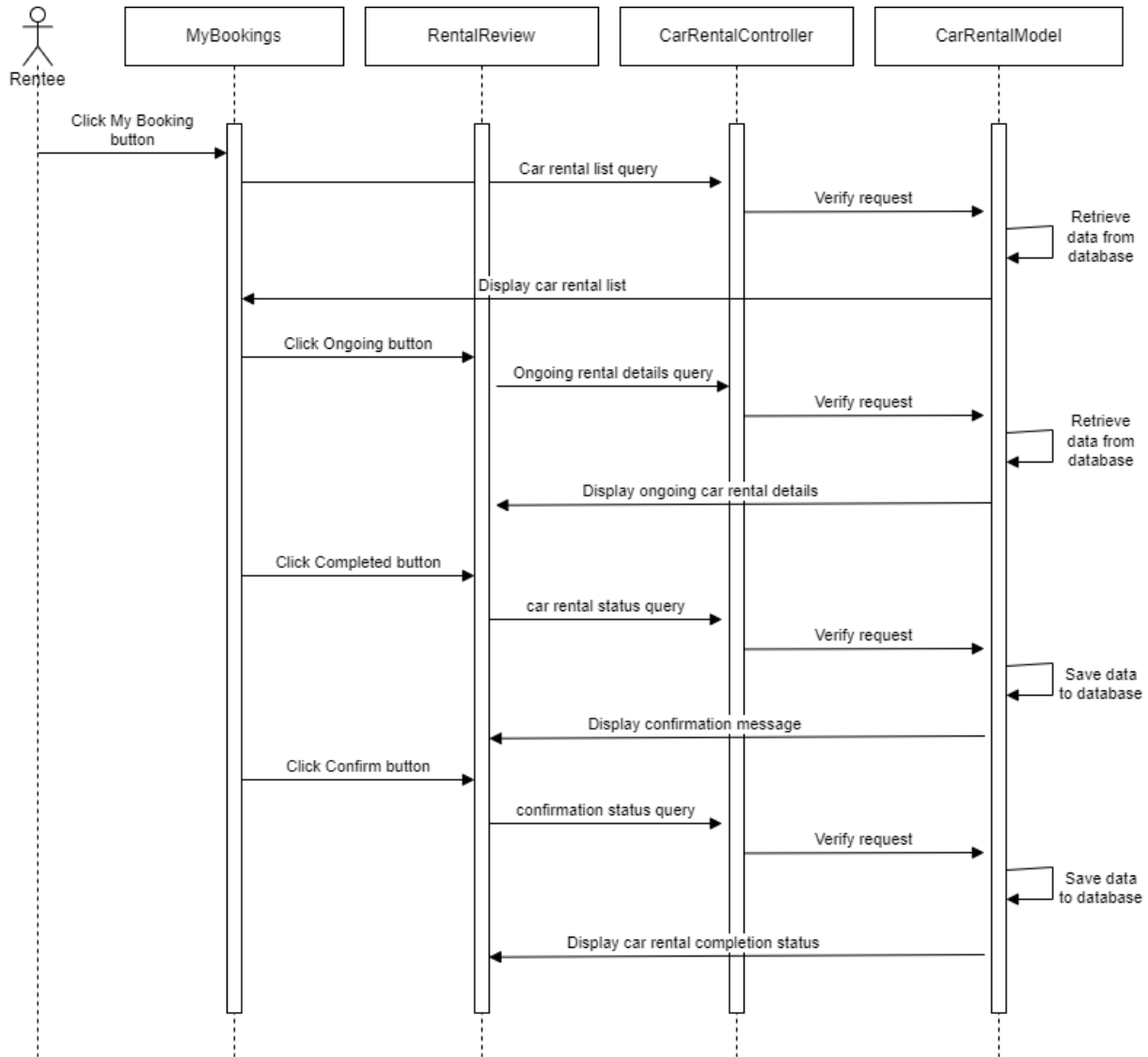


Figure 2.2.3.1 Manage Car Review Basic Flow (Review Car Rental)

The Review Car Rental Basic Flow is for users to give their reviews and feedbacks after they have successfully completed the rental period. It consists of two interfaces, one controller and one model. The interfaces are MyBookings and RentalReview. The controller is the CarRentalController and the model is the CarRentalModel. This sequence diagram shows the basic flow for the Review Car Rental function in the Manage Car Review module in *2.1.2.3 Manage Car Review Use Case Description*.

The Rentee will first click on the My Booking button which will redirect them to the MyBookings interface. The CarRentalController will then verify the request and retrieves data

from the database which is from the CarRentalModel. The system will then display the list of car rentals made by the particular user.

In the MyBookings interface, Rentee can click on the Ongoing button to view the ongoing bookings. The CarRentalController will then verify the request and retrieve data from the database in CarRentalModel. The system will then display the ongoing car rental details in the RentalReview interface. The Rentee fills in the review form and clicks on the Completed button and the system will prompt the user to confirm their car rental completion. The Rentee will then click on the Confirm button. The CarRentalController will verify the request and save the user's review to the database in CarRentalModel. The system will then display the car rental completion status.

**2.2.3.2 Review Cab Service Basic Flow**

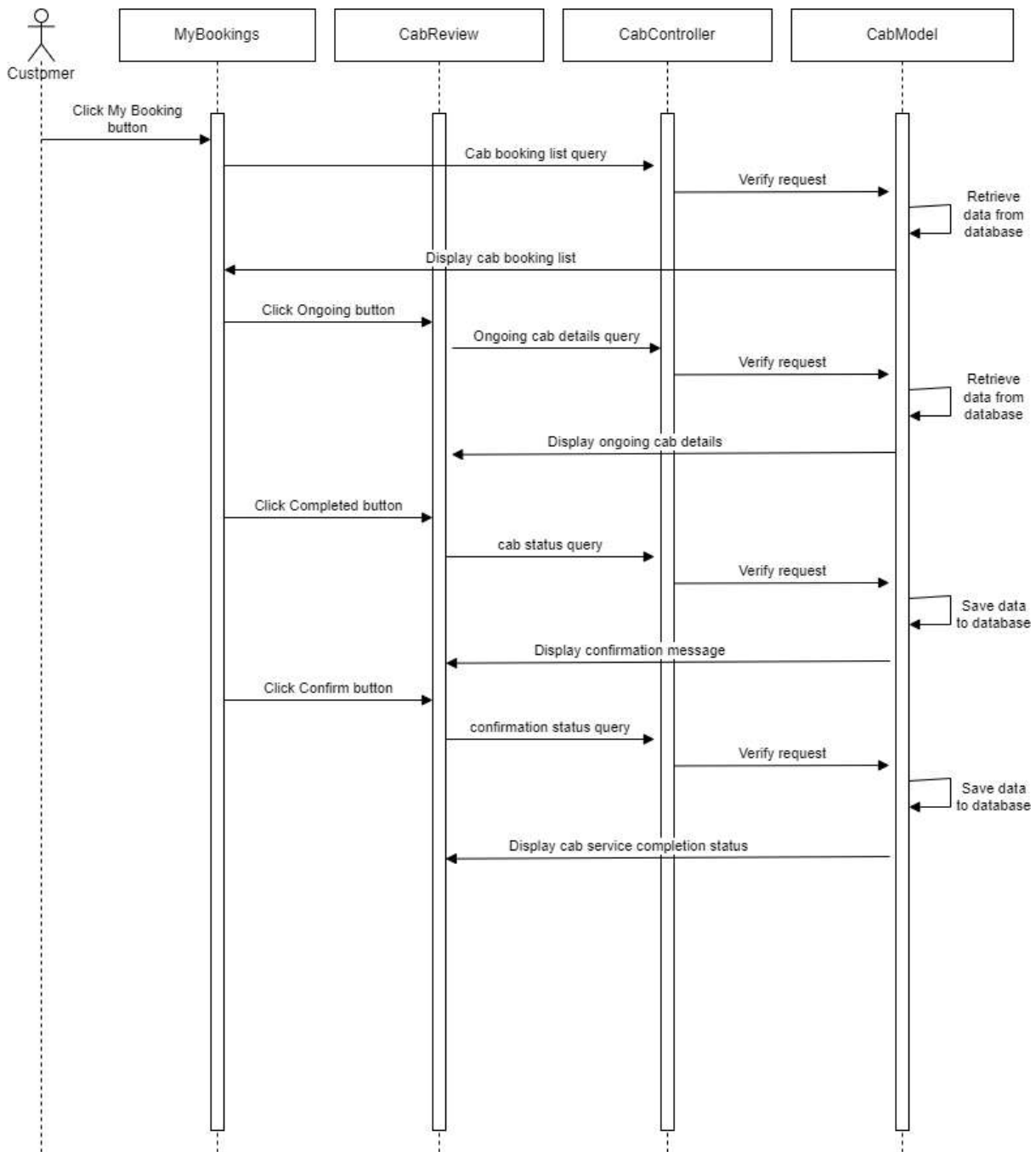


Figure 2.2.3.2 Manage Car Review Basic Flow (Review Cab Service)

The Review Cab Service Basic Flow is for users to give their reviews and feedbacks after they have successfully completed their cab ride. It consists of two interfaces, one controller and one model. The interfaces are MyBookings and CabReview. The controller is the CabController and the model is the CabModel. This sequence diagram shows the basic flow for the Review Cab Service function in the Manage Car Review module in *2.1.2.3 Manage Car Review Use Case Description*.



The Customer will first click on the My Booking button which will redirect them to the MyBookings interface. The CabController will then verify the request and retrieves data from the database which is from the CabModel. The system will then display the list of cab rides booked by the particular user.

In the MyBookings interface, Customer can click on the Ongoing button to view the ongoing bookings. The CabController will then verify the request and retrieve data from the database in CabModel. The system will then display the ongoing cab rides details in the CabReview interface. The Customer fills in the review form and clicks on the Completed button and the system will prompt the user to confirm their cab rides completion. The Customer will then click on the Confirm button. The CabController will verify the request and save the user’s review to the database in CabModel. The system will then display the cab ride completion status.

**2.2.4 Manage Users**

**2.2.4.1 Edit User List Basic Flow**

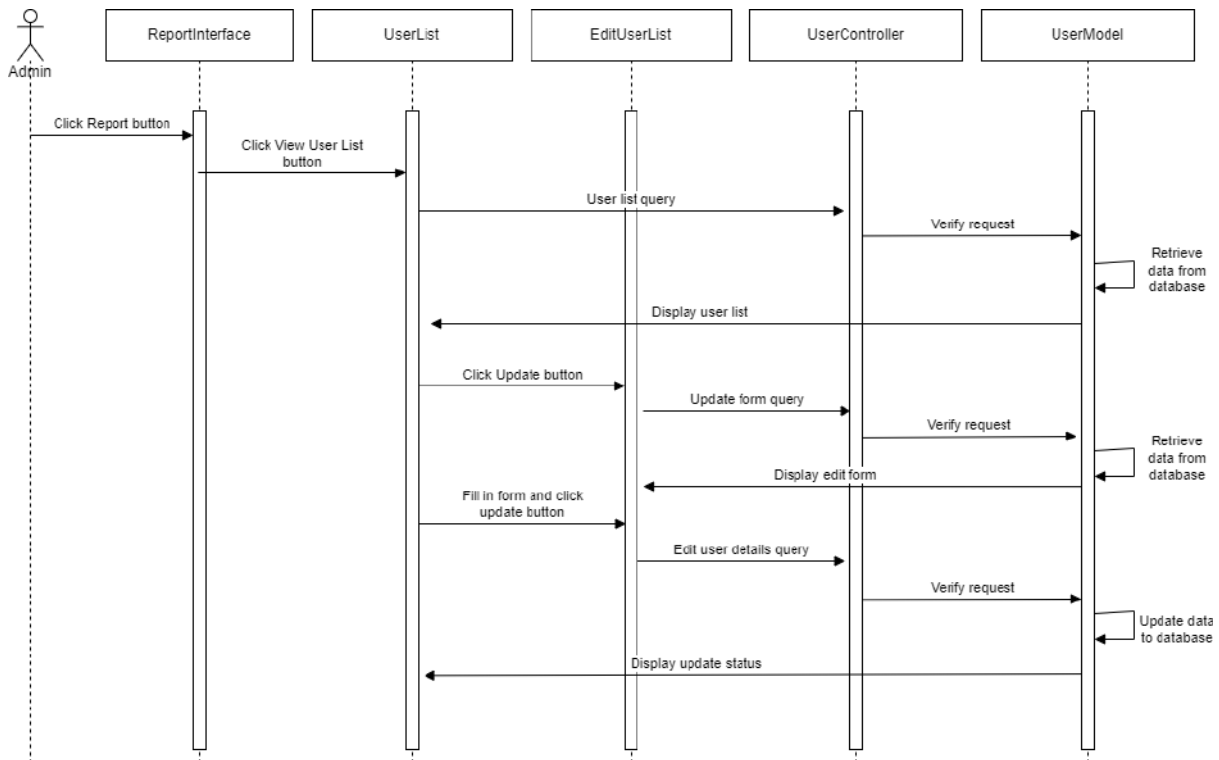


Figure 2.2.4.1 Manage Users Basic Flow (Edit User List)

The Edit User List Basic Flow is for Admin to view and modify the list of users in the system. It consists of three interfaces, one controller and one model. The interfaces are ReportInterface, UserList and EditUserList. The controller is the UserController and the model is the

UserModel. This sequence diagram shows the basic flow for the Edit User List function in the Manage Users module in *2.1.2.4 Manage Users Use Case Description*.

The Admin will first click on the Report button which will redirect them to the ReportInterface interface. The Admin then clicks on the View User List button and they will be redirected to the UserList interface. The UserController will then verify the request and retrieves data from the database in the UserModel. The system will then display the user list in the UserList interface.

Then, the Admin clicks on the Update button in the UserList interface. The UserController will then verify the request and retrieves details of the particular user from the database in the UserModel. The system will then display details of the user in the EditUserList interface. The Admin then edits the existing data of the user and clicks the Update button. The UserController will then verify the data and updates the data to the database. The system will then display the update status.

**2.2.4.2 Delete User Basic Flow**

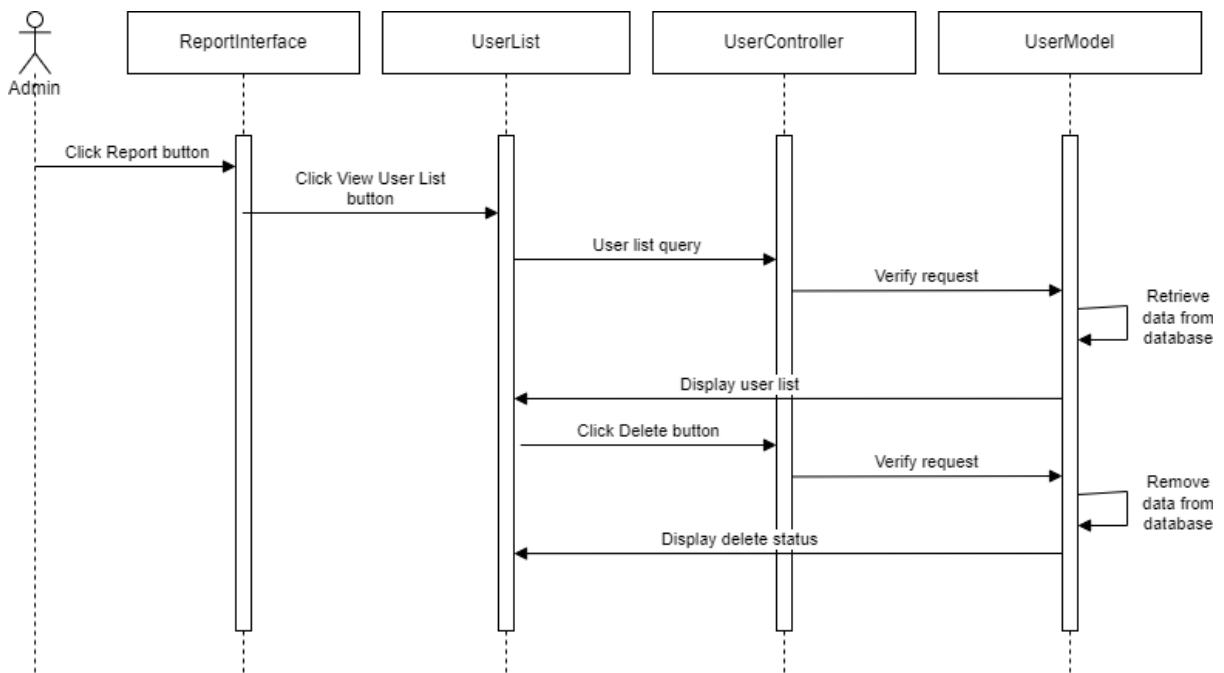


Figure 2.2.4.2 Manage Users Basic Flow (Delete User)

The Delete User Basic Flow is for Admin to delete any data from the list of users. It consists of two interfaces which are ReportInterface and UserList, one controller which is the UserController and one model which is UserModel. This sequence diagram shows the basic

flow for the Delete User function in the Manage Users module in 2.1.2.4 *Manage Users Use Case Description*.

The Admin will first click on the Report button which will redirect them to the ReportInterface interface. The Admin then clicks on the View User List button and they will be redirected to the UserList interface. The UserController will then verify the request and retrieves data from the database in the UserModel. The system will then display the user list in the UserList interface.

Then, the Admin clicks on the Delete button in the UserList interface. The UserController will then verify the delete request and removes data from the database in UserModel. The system will then display the delete status.

**2.2.4.3 Edit Car List Basic Flow**

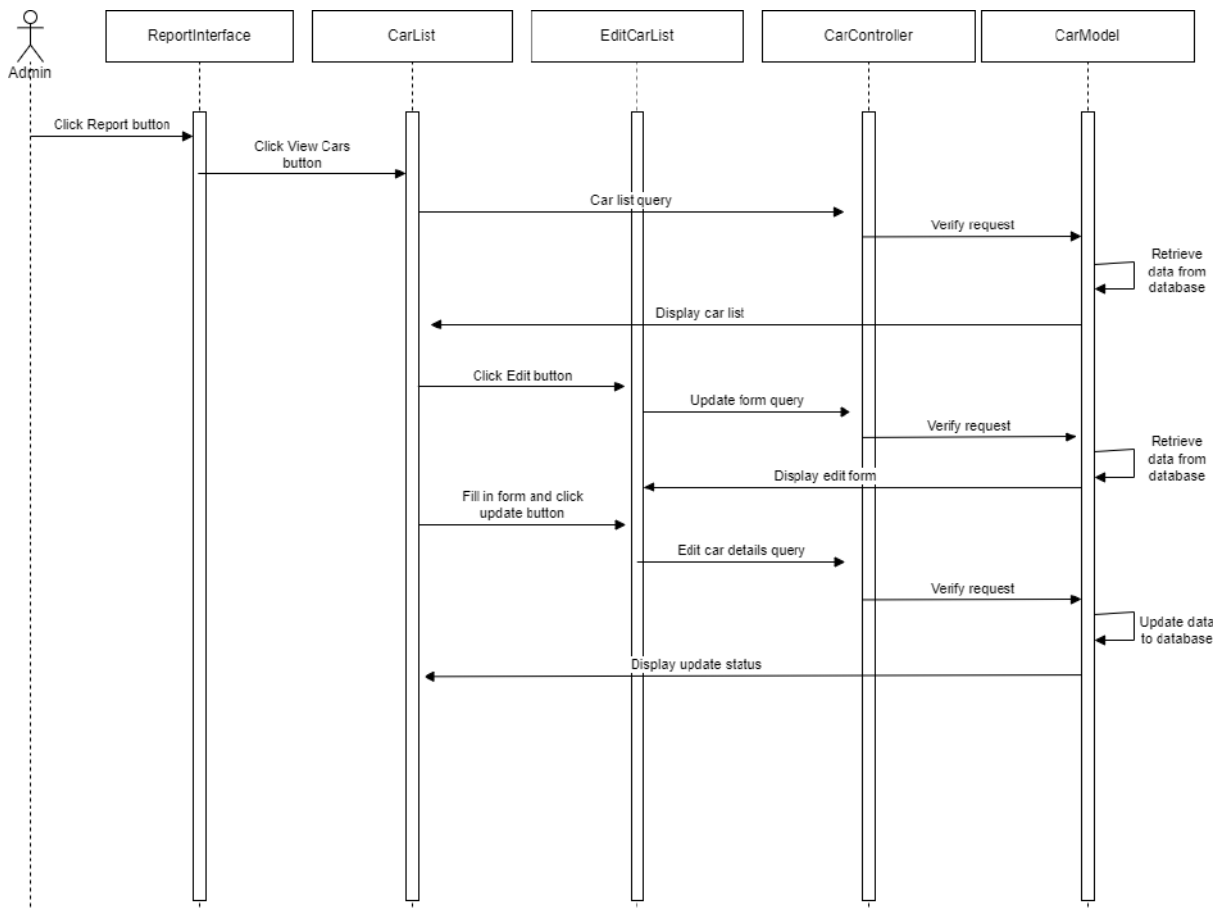


Figure 2.2.4.3 Manage Users Basic Flow (Edit Car Rental List)

The Edit Car Rental List Basic Flow is for Admin to view and modify the list of car rentals registered into the system. It consists of three interfaces, one controller and one model. The interfaces are ReportInterface, CarRentalList and EditCarRentalList. The controller is the

CarRentalController and the model is the CarRentalModel. This sequence diagram shows the basic flow for the Edit CarRental List function in the Manage Users module in *2.1.2.4 Manage Users Use Case Description*.

The Admin will first click on the Report button which will redirect them to the ReportInterface interface. The Admin then clicks on the View CarRental List button and they will be redirected to the CarRentalList interface. The CarRentalController will then verify the request and retrieves data from the database in the CarRentalModel. The system will then display the car rental list in the CarRentalList interface.

Then, the Admin clicks on the Update button in the CarRentalList interface. The CarRentalController will then verify the request and retrieves details of the particular car rental from the database in the CarRentalModel. The system will then display details of the car rental in the EditCarRentalList interface. The Admin then edits the existing data of the car rental and clicks the Update button. The CarRentalController will then verify the data and updates the data to the database. The system will then display the update status.

**2.2.4.4 Delete Car Basic Flow**

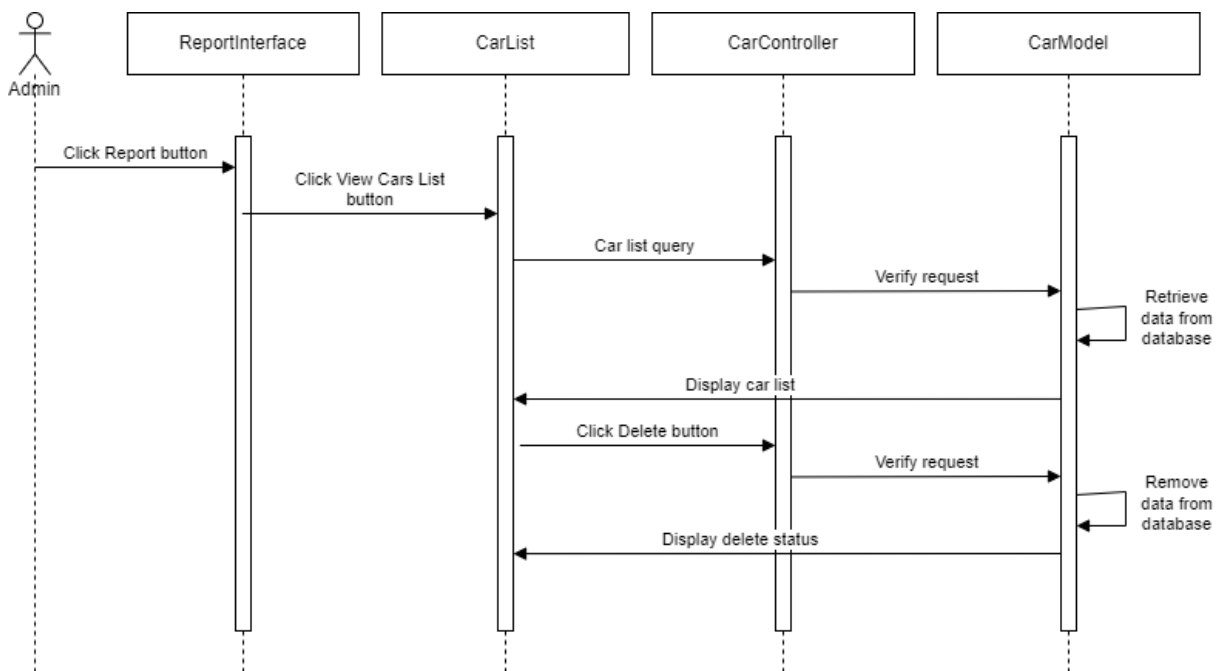


Figure 2.2.4.4 Manage Users Basic Flow (Delete Car Rental)

The Delete Car Rental Basic Flow is for Admin to delete any data from the list of car rentals. It consists of two interfaces which are ReportInterface and CarRentalList, one controller which is the CarRentalController and one model which is CarRentalModel. This sequence diagram

shows the basic flow for the Delete Car Rental function in the Manage Users module in 2.1.2.4 *Manage Users Use Case Description*.

The Admin will first click on the Report button which will redirect them to the ReportInterface interface. The Admin then clicks on the View Car Rental List button and they will be redirected to the CarRentalList interface. The CarRentalController will then verify the request and retrieves data from the database in the CarRentalModel. The system will then display the car rental list in the CarRentalList interface.

Then, the Admin clicks on the Delete button in the CarRentalList interface. The CarRentalController will then verify the delete request and removes data from the database in CarRentalModel. The system will then display the delete status.

**2.2.4.5 Manage Car Basic Flow**

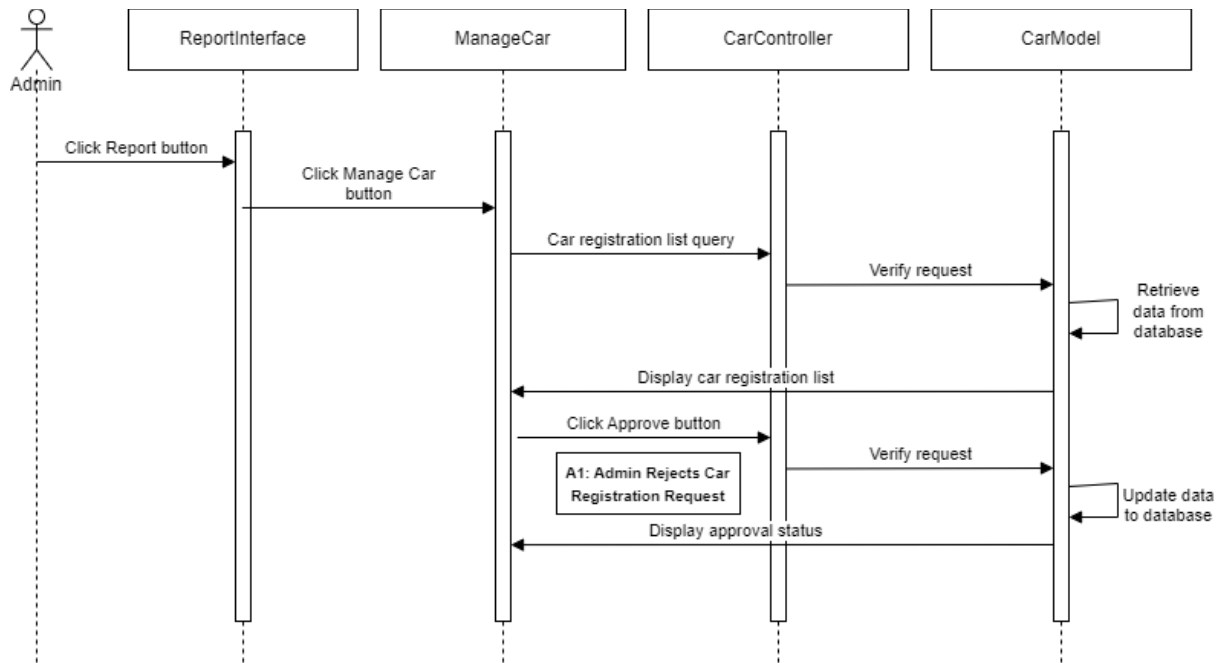


Figure 2.2.4.5 Manage Users Basic Flow (Manage Car)

2.2.4.6 Reject Car Alternative Flow

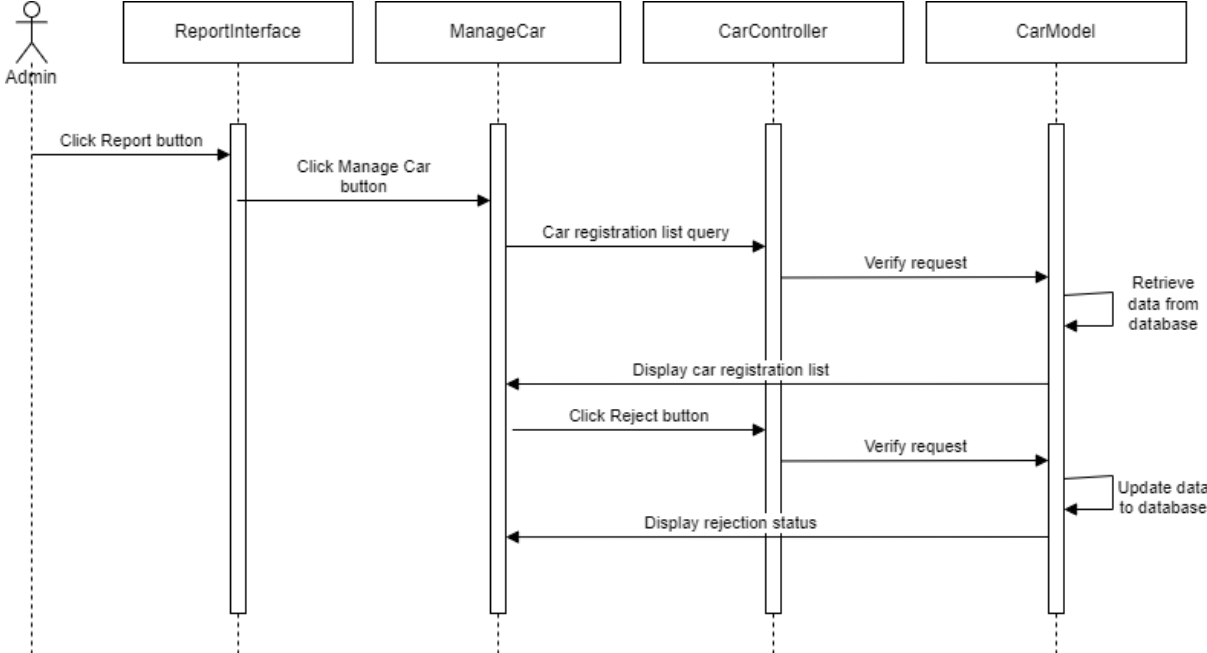


Figure 2.2.4.6 Manage Users Alternative Flow (Reject Car)

## 2.2.5 Manage Report

### 2.2.5.1 View Report Basic Flow

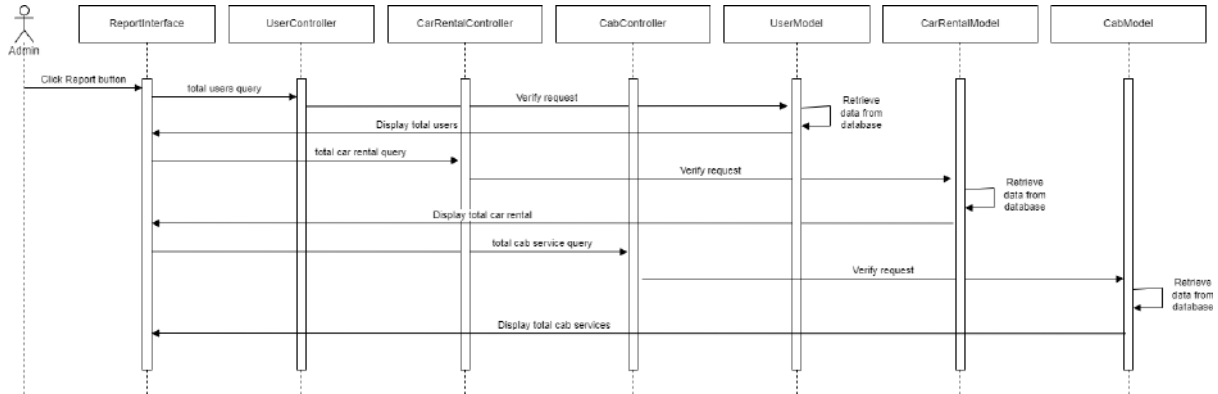


Figure 2.2.5.1 Manage Report Basic Flow (View Report)

The View Report Basic Flow is for Admin to view reports of the system such as the total number of users in the system, the total number of car rentals registered into the system and the total number of cab services registered into the system. It consists of one interface which is the ReportInterface, three controllers and three models. The controllers are UserController, CarRentalController and CabController. The models are UserModel, CarRentalModel and CabModel. This sequence diagram shows the basic flow for the View Report in the Manage Report module in *2.1.2.5 Manage Report Use Case Description*.

The Admin will first click on the Report button which will redirect them to the ReportInterface interface. The UserController will retrieve total number of users from the database in the UserModel and displays the total users in the ReportInterface. The CarRentalController will retrieve the total number of car rentals from the database in CarRentalModel and displays the total number of car rentals in the ReportInterface. The CabController will retrieve the total number of cab service from the database in the CabModel and displays the total number of cab service in the ReportInterface.

## CHAPTER 3

### 3.1 INTERFACE DESIGN

#### 3.1.1 Register and Login

The Register interface is for new users who want to use the system. The users will be required to enter their credentials such as Name, ID, Password and Confirm Password to register into the system. The figure below shows the Register interface.

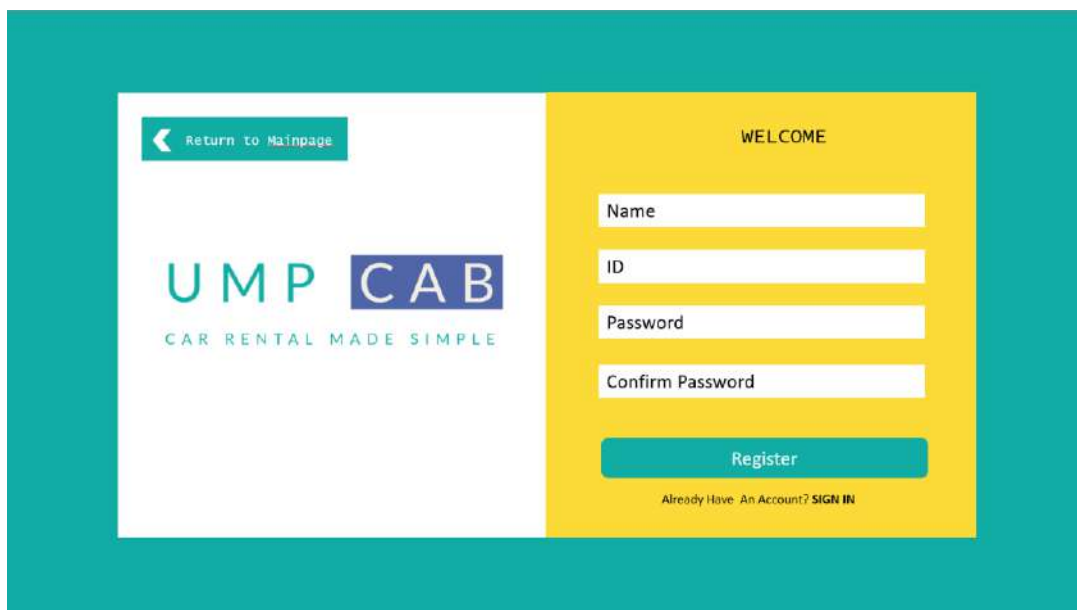
The image shows a web interface for registering a new user. On the left, there is a white panel with a teal header containing a back arrow and the text "Return to Mainpage". Below this is the logo for "UMP CAB" with the tagline "CAR RENTAL MADE SIMPLE". On the right, there is a yellow panel with the heading "WELCOME". It contains four input fields labeled "Name", "ID", "Password", and "Confirm Password". Below these fields is a teal "Register" button. At the bottom of the yellow panel, there is a link that says "Already Have An Account? SIGN IN".

Figure 3.1.1.1 Register Interface

The Login interface is for existing system to login into the system to access it. Users are required to enter credentials such as their ID and password. The figure below shows the Login interface.



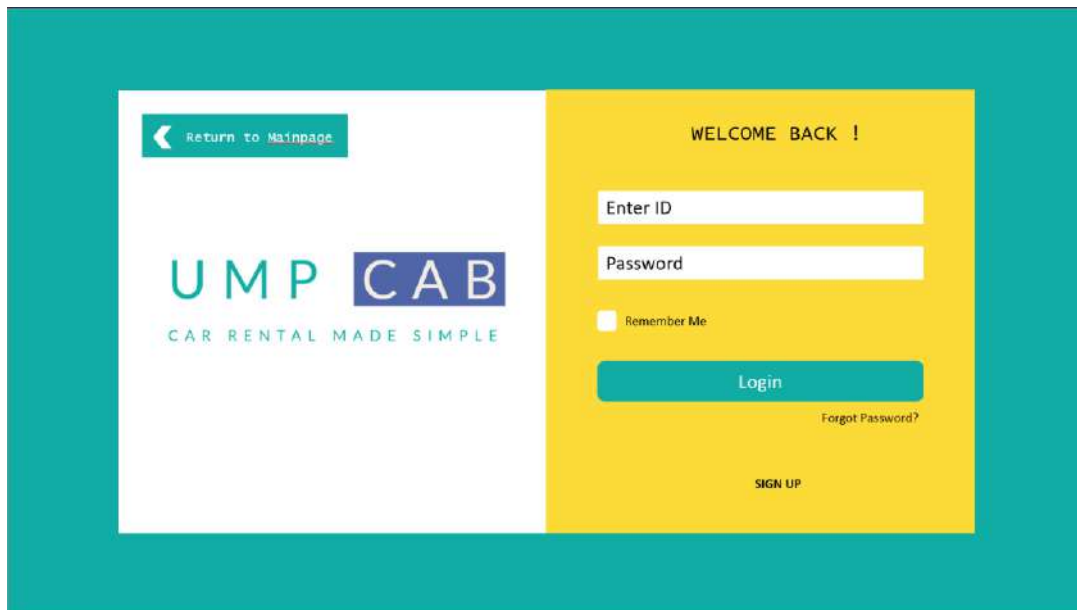


Figure 3.1.1.2 Login Interface

In any case the users forget their password for the system, they can always access the Forgot Password interface through the Login interface. Users will be prompted to enter their email address in order to reset their password. The figure below shows the Forgot Password interface.

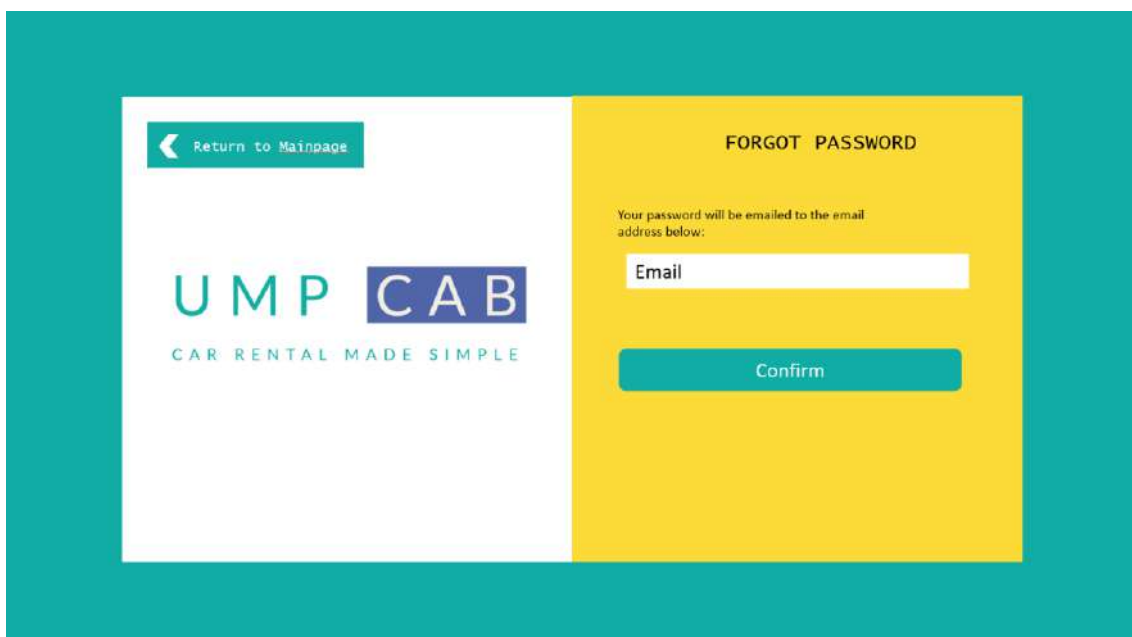


Figure 3.1.1.3 Forgot Password Interface

### 3.1.2 Mainpage

The Mainpage is the first page that users will see when they access the system. The Mainpage consists of buttons to allows users to Register Car, Browse for Car Rental services, Register Cab and Browse for Cab Services. The figure below shows the Mainpage of the system.

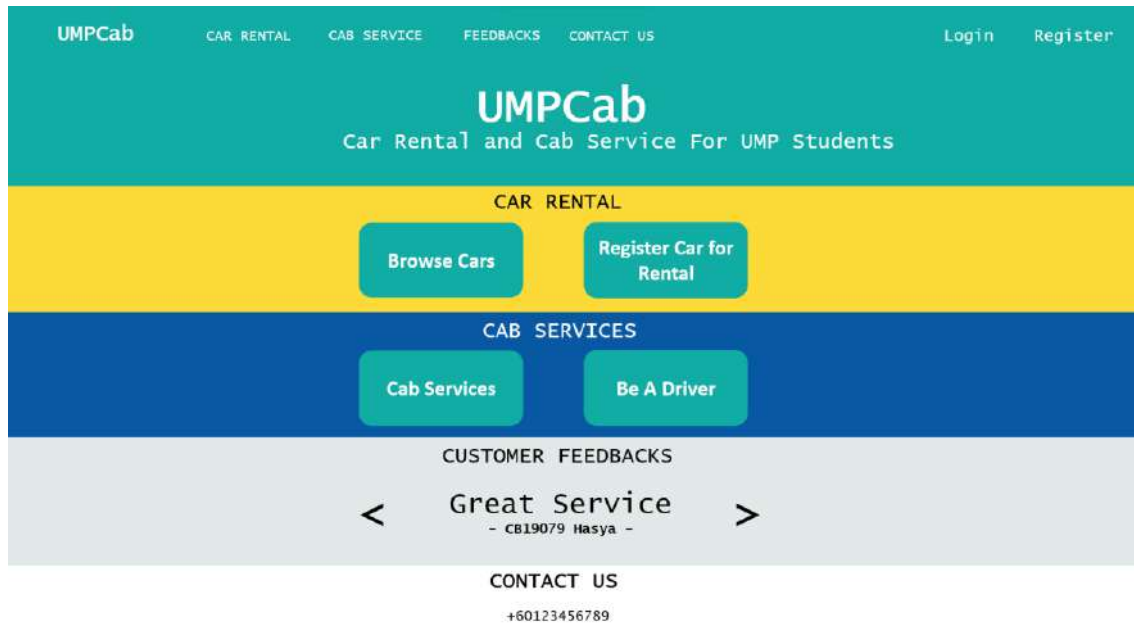


Figure 3.1.2.1 Mainpage Interface

### 3.1.3 Dashboard

After login and registration, all users will be redirected to the Dashboard interface as shown in the figure below. In the Dashboard interface, users can view their earnings (if they have registered cars or cab in the system), their total number of bookings completed (if they have registered cars or cab in the system) and their on-going bookings (if they have registered cars or cab in the system).

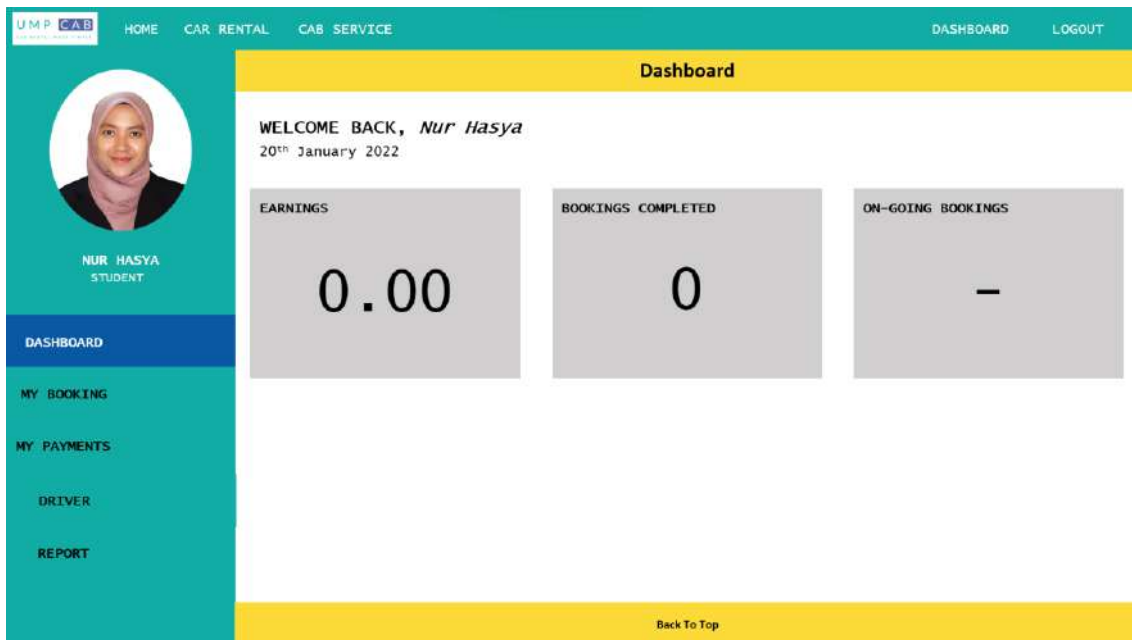


Figure 3.1.3.1 Dashboard Interface

### 3.1.4 My Booking

For the My Booking interface, users can view a list of their completed and ongoing bookings for both car rental and cabs. The figure below shows the My Booking interface.

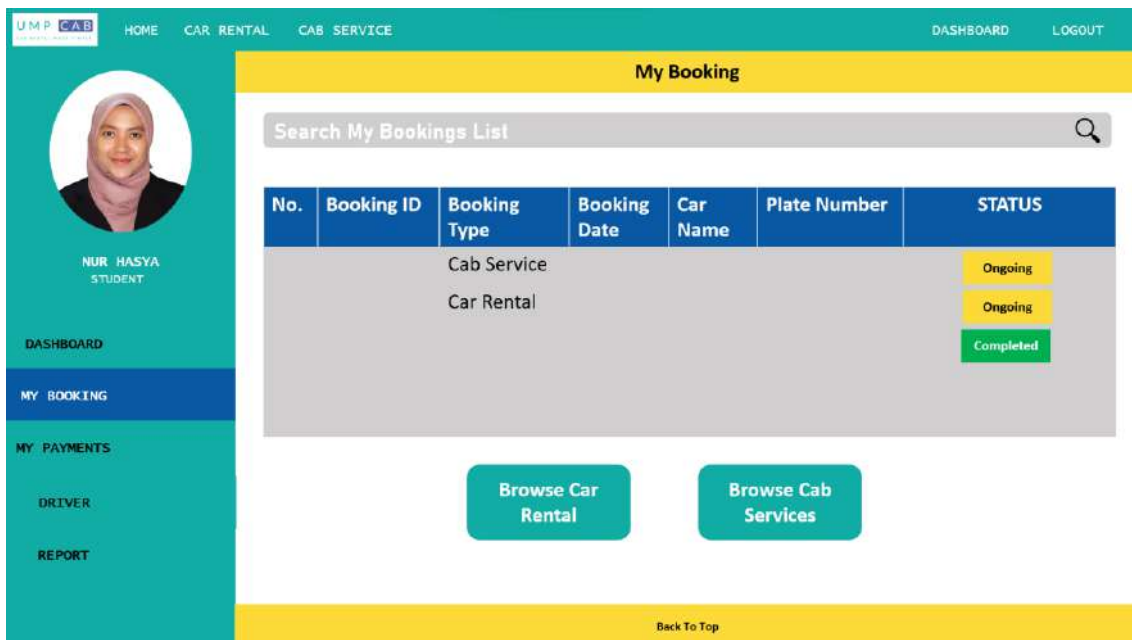


Figure 3.1.4.1 My Booking Interface

### 3.1.5 Driver Dashboard

The Driver Dashboard are for users who have cars or cabs registered into the system. In this interface, both Renters and Drivers can view the car and cab they have registered into the system. The figure below shows the Driver Dashboard interface.

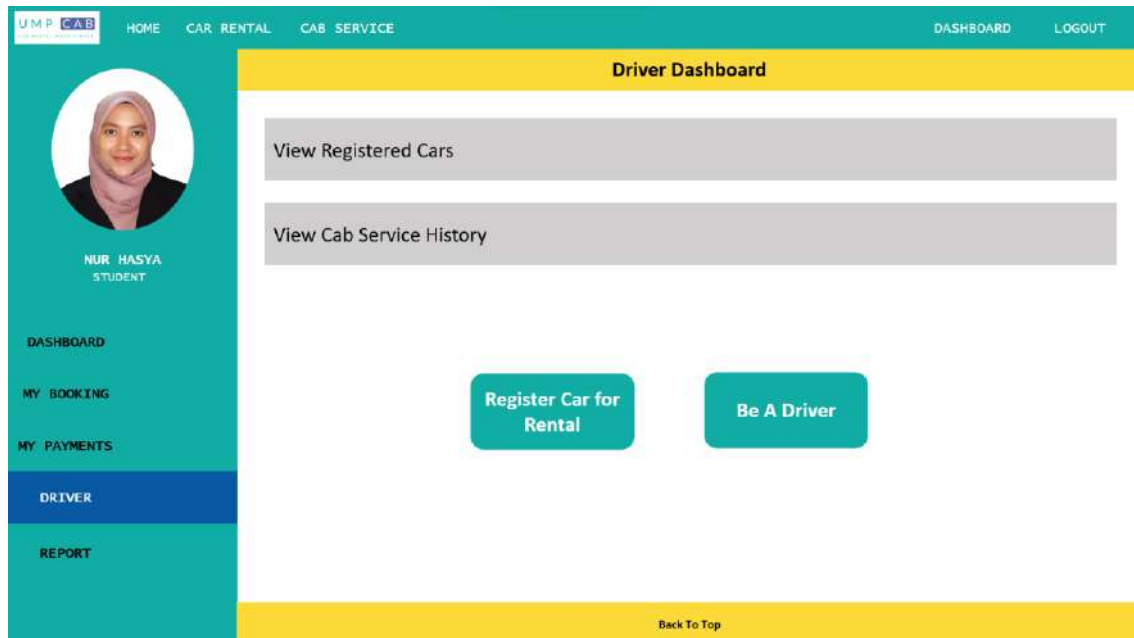


Figure 3.1.5.1 Driver Dashboard Interface

When the Driver/Renter clicks on the View Registered Cars, they will be redirected to the Registered Cars interface. The Registered Cars interface is as shown in the figure below.

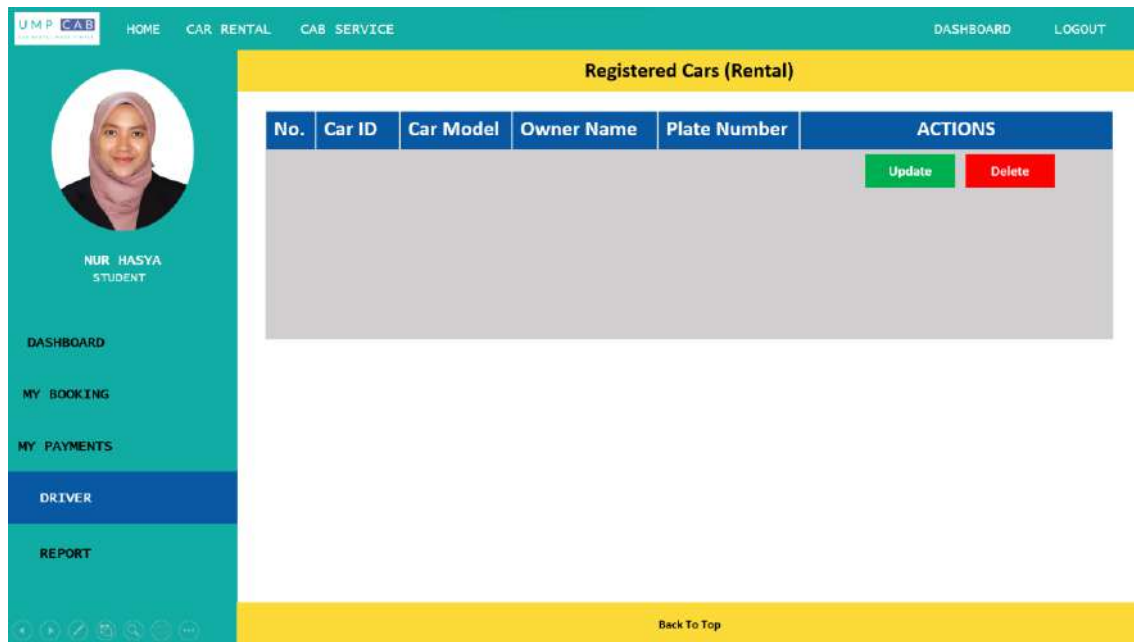


Figure 3.1.5.2 Registered Cars Interface

When the Driver/Renter clicks on the View Cab Service History, they will be redirected to the Cab Service Registered interface. The Cab Service Registered interface is as shown in the figure below.

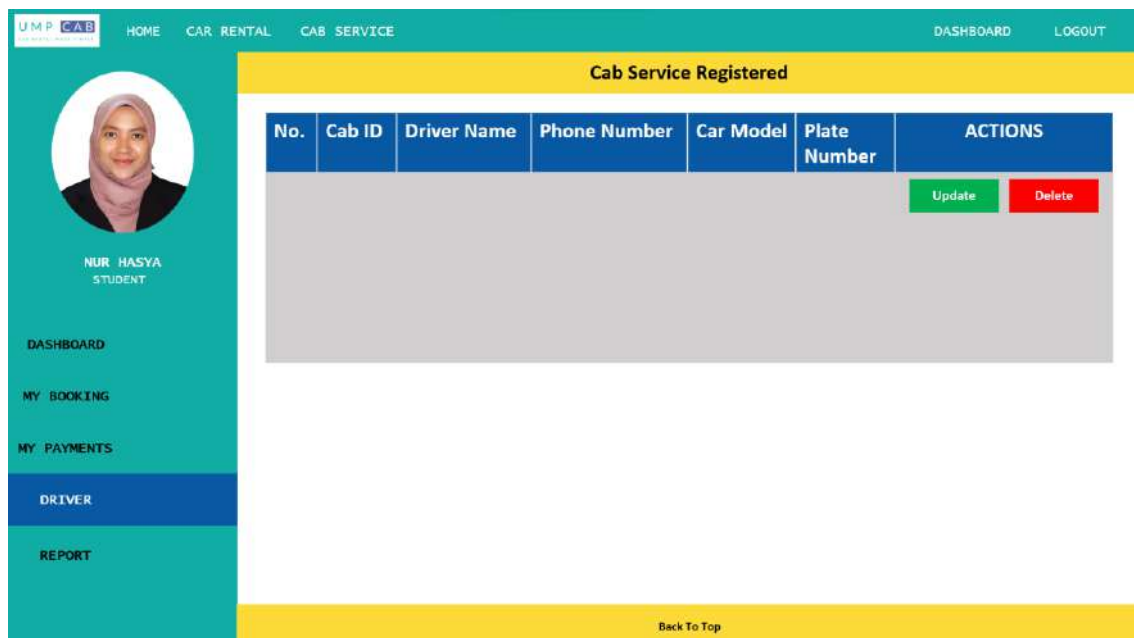


Figure 3.1.5.3 Cab Service Registered Interface

### 3.1.6 Report

For the Report interface, only Admin can access it to view the report about the system and also manage the users, car rentals and cabs registered into the system. The figure below shows the Report interface.

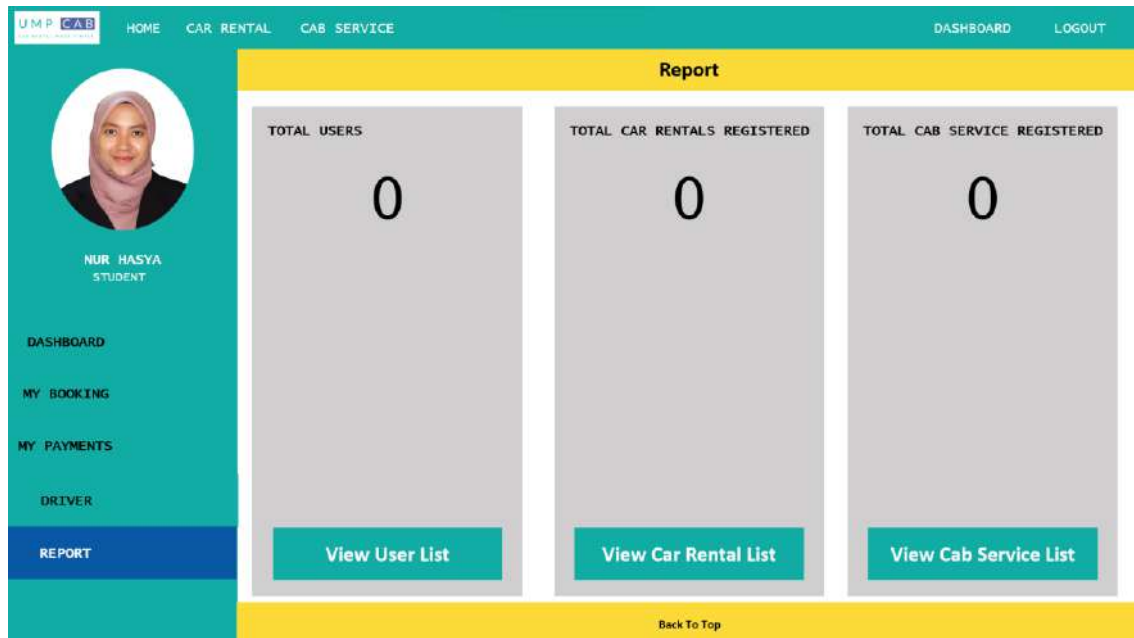


Figure 3.1.6.1 Report Interface

When the Admin clicks on the View User List, they will be redirected to the User List interface. The User List interface is as shown in the figure below.

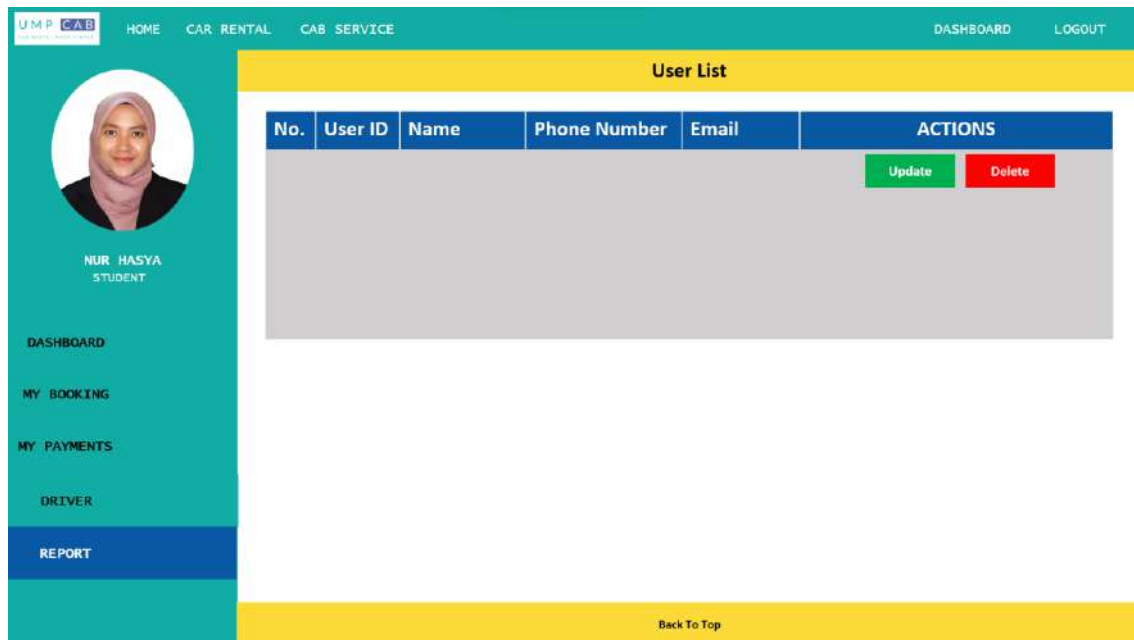


Figure 3.1.6.2 User List Interface

When the Admin clicks on the View Car Rental List, they will be redirected to the Car Rental List interface. The Car Rental List interface is as shown in the figure below.

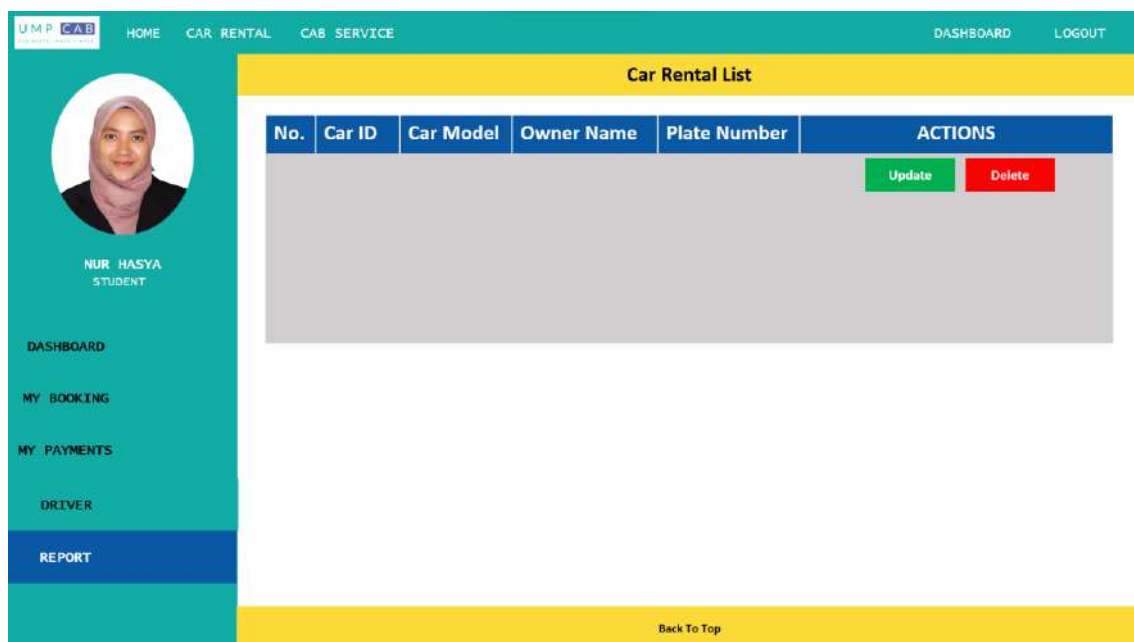


Figure 3.1.6.3 Car Rental List Interface

When the Admin clicks on the View Cab Service List, they will be redirected to the Cab Service List interface. The Cab Service List interface is as shown in the figure below.

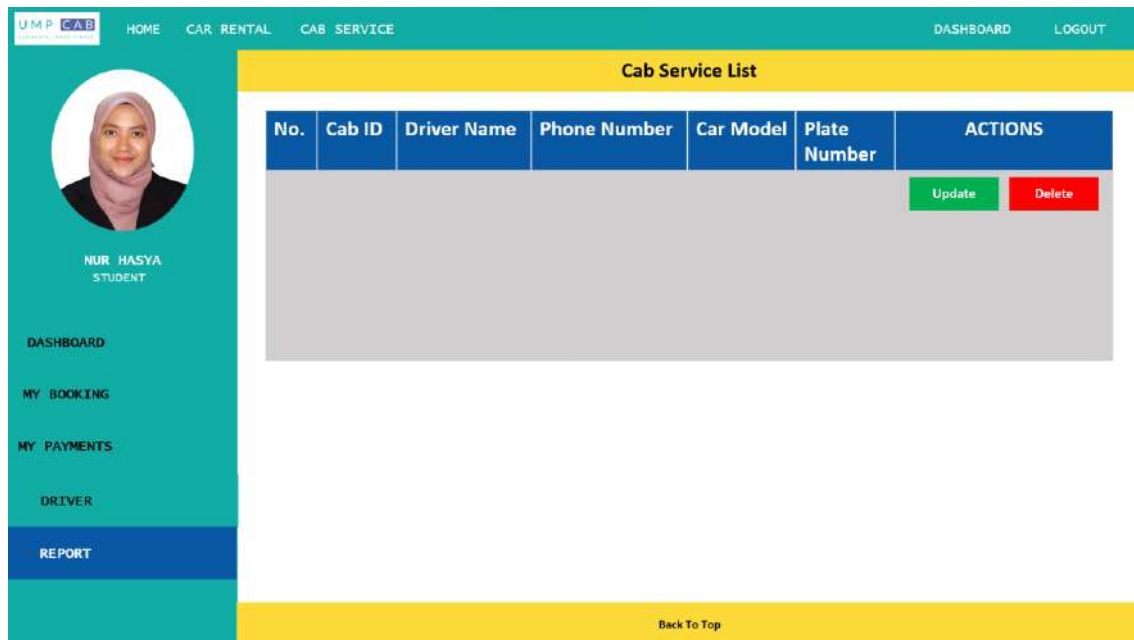


Figure 3.1.6.4 Cab Service List Interface

### 3.1.7 Car Registration

For users who want to register their car for rental, they will be redirected to the Register Car for Rental interface. The users will be required to provide their credentials for the car rental registration. The Register Car for Rental interface is as shown in the figure below.

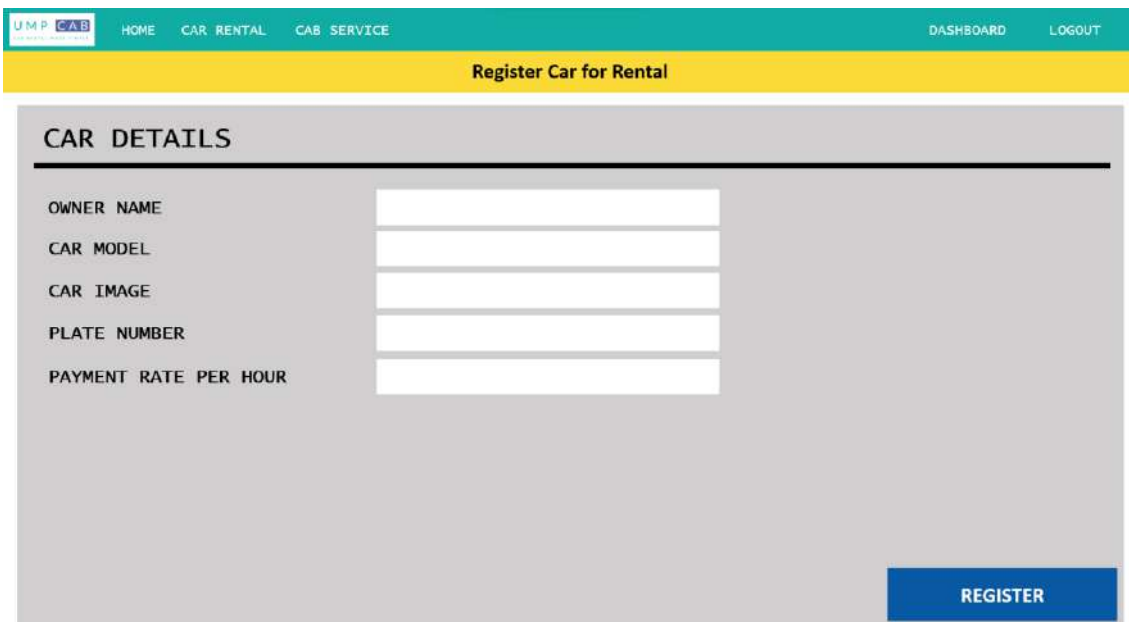


Figure 3.1.7.1 Register Car for Rental Interface



### 3.1.8 Car Rental Booking

For the car rental booking, users will first browse through a list of available cars for rental in the Car Rental Homepage interface. In this interface, users can also search for car rental services based on the search details they enter. The Car Rental Homepage interface is as shown in the figure below.

The screenshot displays the 'Car Rental List' interface. At the top, there is a navigation bar with 'HOME', 'CAR RENTAL', 'CAB SERVICE', 'DASHBOARD', and 'LOGOUT'. Below this is a search form with two columns: 'PICK-UP' and 'DROP-OFF'. The 'PICK-UP' column contains 'Pick-Up Location', 'Pick-Up Date', and 'Pick-Up Time' fields. The 'DROP-OFF' column contains 'Drop-Off Location', 'Drop-Off Date', and 'Drop-Off Time' fields. A 'Search' button is located below the form. Below the search form is a section titled 'All Cars' with a 'Filter By' button. This section displays four car models in a grid: PERODUA MYVI (red), PERODUA AXIA (white), PERODUA BEZZA (silver), and PERODUA AXIA (white). Each car card includes a 'View Details' button.

Figure 3.1.8.1 Car Rental Booking Interface

When the user clicks on the Search button, the system will display a list of available cars for rental based on their search results. Users can also filter the search results according to their own preferences. The figure below shows the Search Result interface.

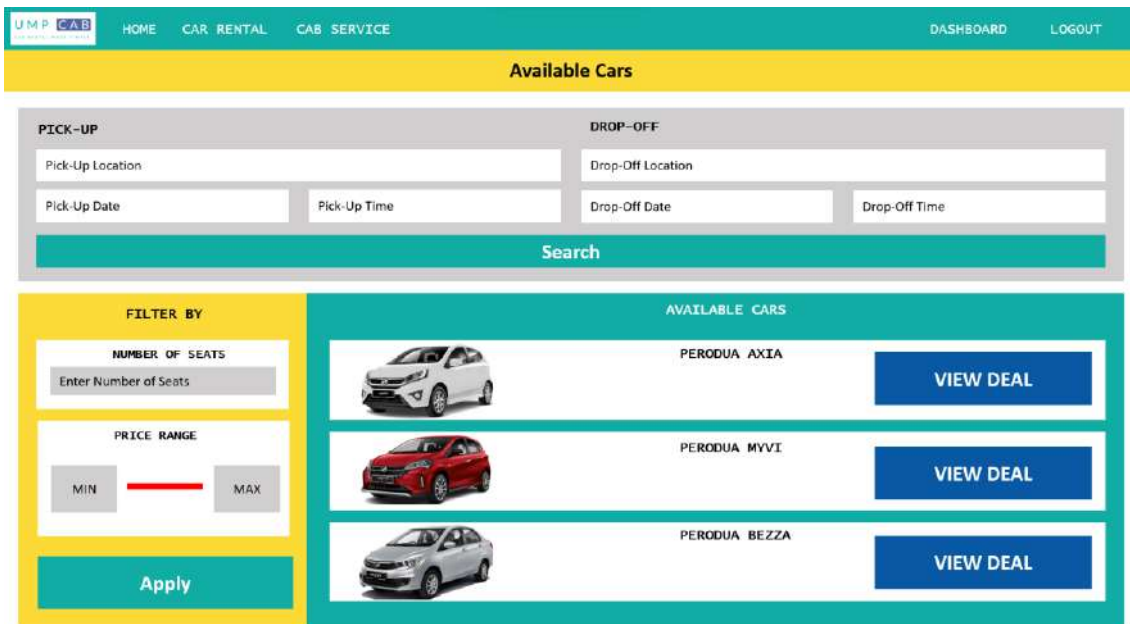


Figure 3.1.8.2 Search Result Interface

Users will then click on the View Deal button, and they will be redirected to the Car Details interface as shown below.

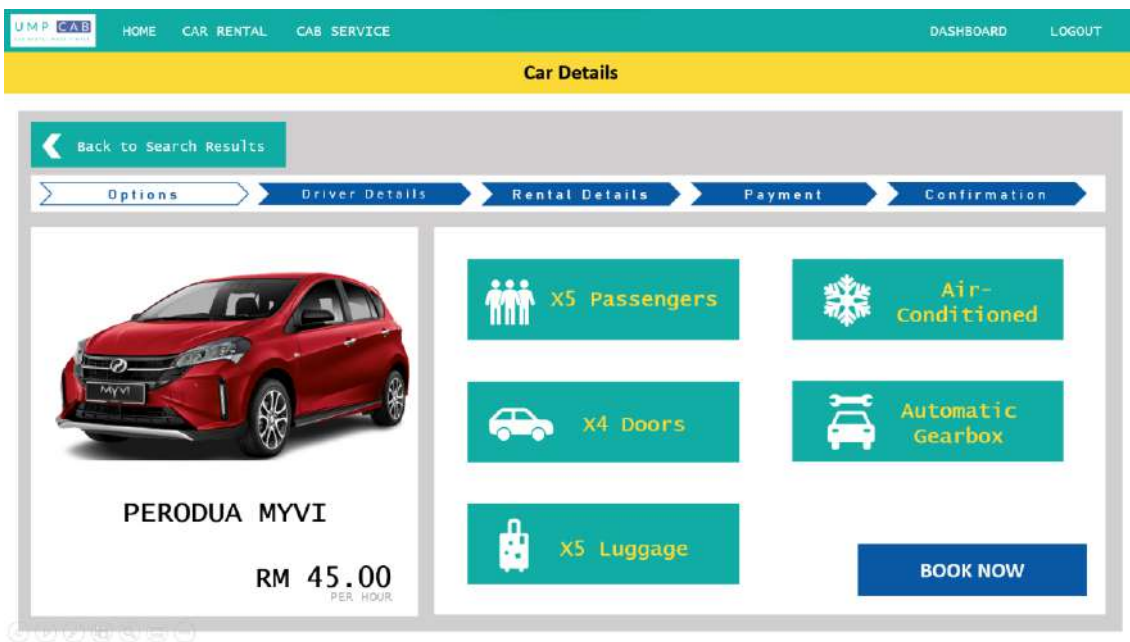


Figure 3.1.8.3 Car Details Interface

Users can then proceed to provide their details for the car rental such as Driver Details, Rental Details. The figures below show the interfaces for the respective car rental details.

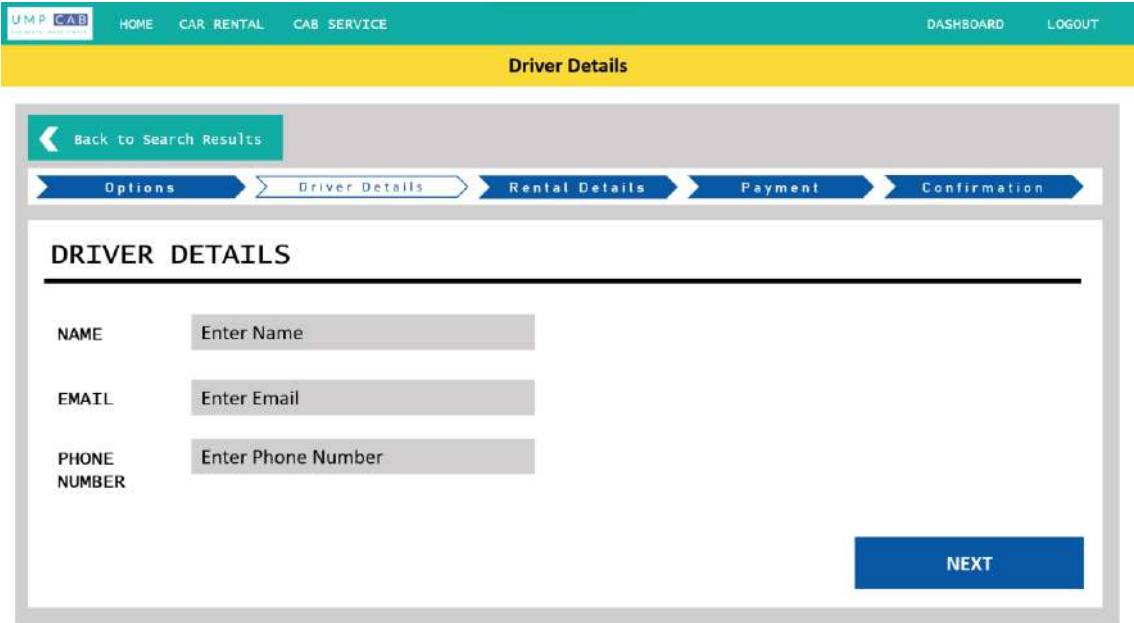


Figure 3.1.8.4 Driver Details Interface

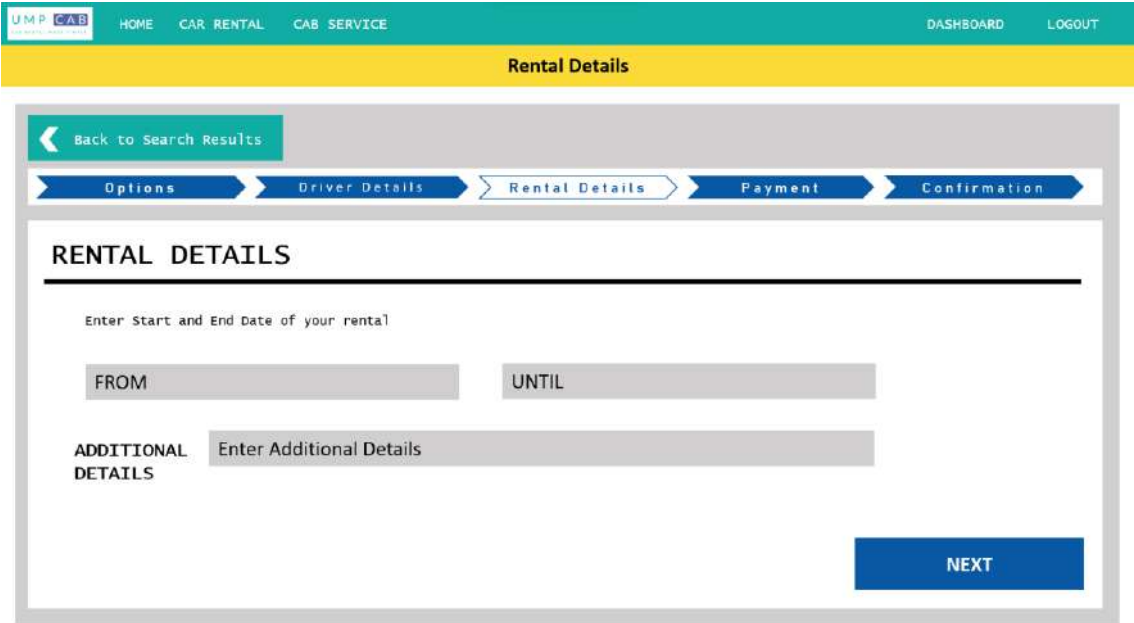


Figure 3.1.8.5 Rental Details Interface


UMP CAB HOME CAR RENTAL CAB SERVICE DASHBOARD LOGOUT

Confirm Rental

Back to Search Results

Options Driver Details Rental Details Payment Confirmation

### CONFIRM YOUR RENTAL DETAILS



**PERODUA MYVI**

- ✓ 5 Passenger Seats
- ✓ 4 Doors
- ✓ 5 Luggage Capacity
- ✓ Air-Conditioned
- ✓ Automatic Gearbox

**Driver Details:** *chase*

Name: AUL HASYA BENTJ MOHD NORJIN  
 Email: nurhasyanhdordin@gmail.com  
 Phone Number: 011-14909117

**Rental Details:**

From 1 January 2022 Until 3 January 2022 *chase*

**Payment Details:**

Payment Method: online banking (bank iclan) *chase*  
 Amount: RM 60.00

**CONFIRM BOOKING**

Figure 3.1.8.6 Confirm Rental Interface

### 3.1.9 Cab Registration

For users who want to register their car up for cab service, they will be redirected to the Register Car for Cab Service interface where they are required to provide appropriate credentials. The Register Car for Cab Service interface is as shown in the figure below.

UMP CAB HOME CAR RENTAL CAB SERVICE DASHBOARD LOGOUT

Register Car For Cab Service

### CAB DETAILS

DRIVER NAME

CAR MODEL

CAR IMAGE

PLATE NUMBER

PAYMENT RATE PER HOUR

**REGISTER**

Figure 3.1.9.1 Register Car for Cab Service Interface

### 3.1.10 Cab Booking

For cab booking, users will first browse through available cab services on a map in the Cab Service Homepage interface. In this interface, users can also search the map based on the location they want by entering their location details in the Search corner. The Cab Service Homepage interface is shown in the figure below.

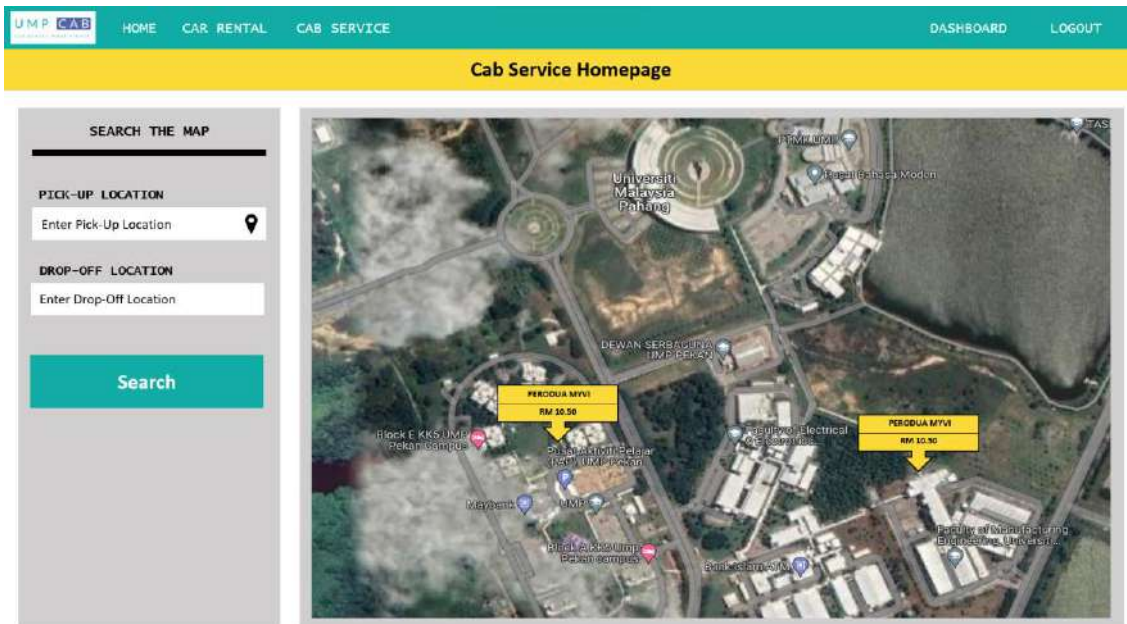


Figure 3.1.10.1 Cab Service Homepage Interface

To proceed booking a cab, users need to click on any available cab services on the map. They will then be redirected to the Cab Details interface as shown in the figure below.

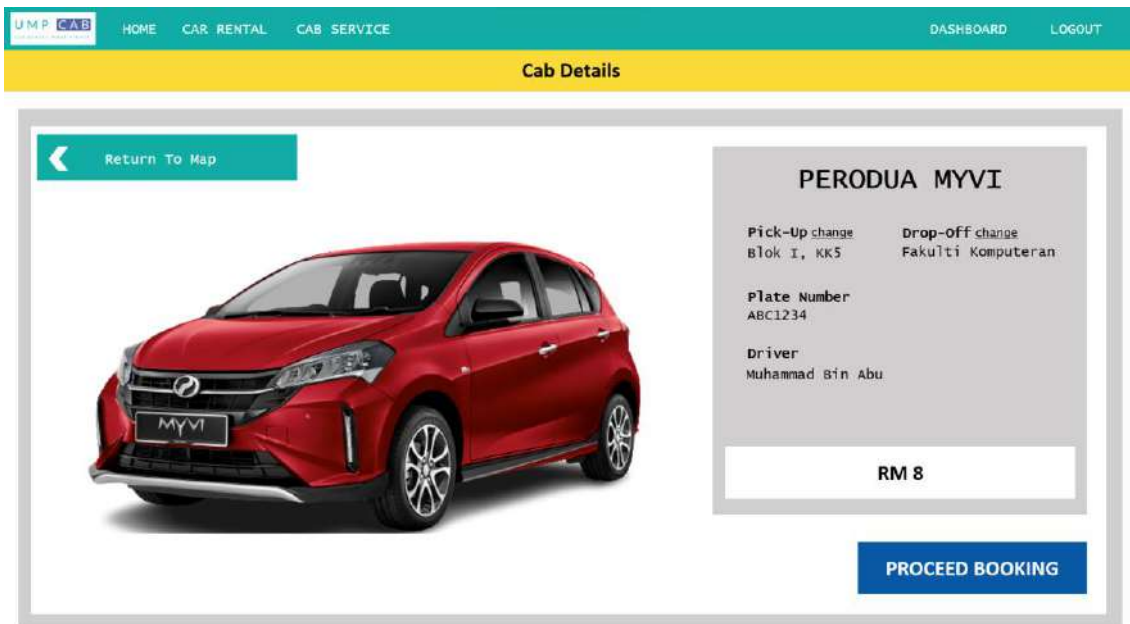


Figure 3.1.10.2 Cab Details Interface

### 3.1.11 Car Rental Review

Users are required to rate and review the car rental after they have completed the rental period. For that, they will be redirected to the Car Rental Review interface as shown in the figure below.

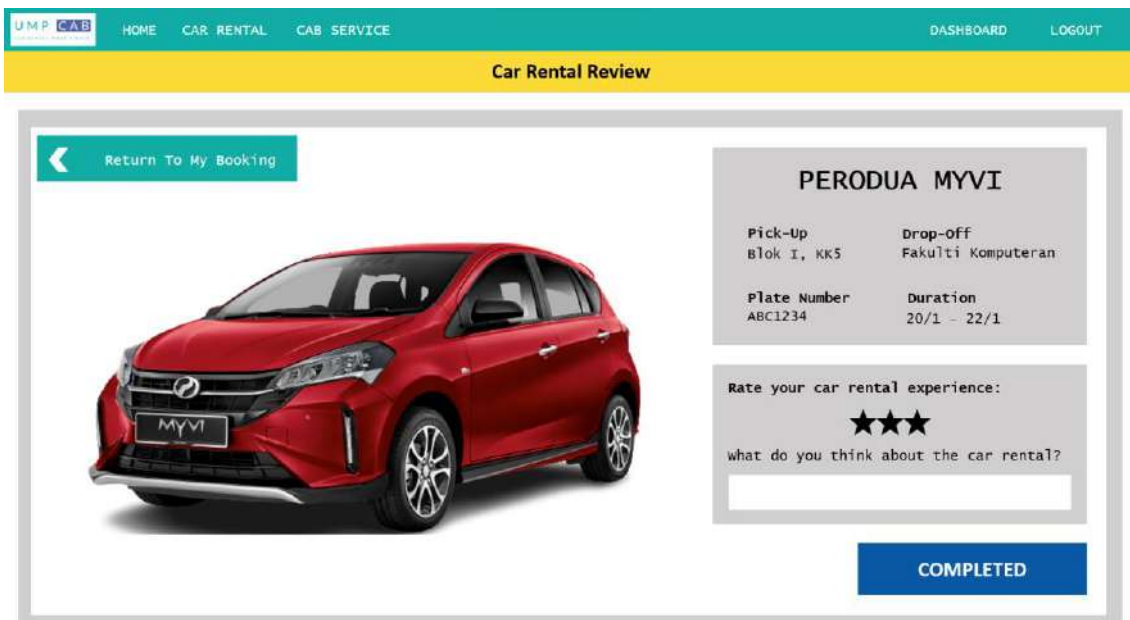


Figure 3.1.11.1 Car Rental Review Interface

### 3.1.12 Cab Review

Users are required to rate and review the cab service after they have completed the cab ride. For that, they will be redirected to the Cab Review interface as shown in the figure below.

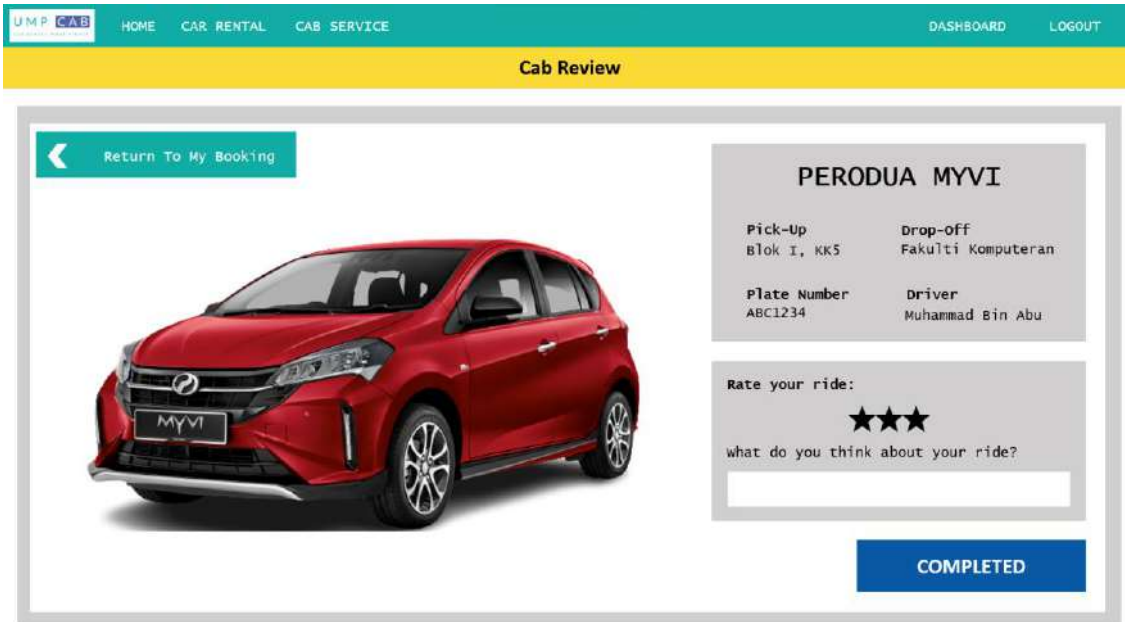


Figure 3.1.12.1 Cab Review Interface



## 3.2 HARDWARE AND SOFTWARE SPECIFICATION

### 3.2.1 Hardware Specifications

Table 3.2.1 Hardware Specifications

Description	Tools
Processor	Intel Pentium or later
Memory	At least 2GB, preferably higher
Hard Disk	At least 10GB
Screen Resolution	1280x1024 or larger

### 3.2.2 Software Specifications

Table 3.2.2 Software Specifications

Description	Tools
Operating System	Windows 10 and above
Supported Browser	Chrome, Mozilla
Web Server	XAMPP Server
Database Server	MySQL
Programming Language	PHP, HTML, CSS, Bootstrap
Framework	Laravel
IDE	Visual Studio Code



## **APPENDIX B**

### **SOFTWARE DESIGN DOCUMENT (SDD)**

2022

# SOFTWARE DESIGN DESCRIPTION (SDD)

[WEB-BASED CAR RENTAL AND CAB  
SERVICE SYSTEM FOR UMP STUDENTS]



**DOCUMENT APPROVAL**

	Name	Date
<p><b>Authenticated by:</b></p> <p>_____</p> <p>NUR HASYA BINTI MOHD NORDIN</p>	<p>NUR HASYA BINTI MOHD NORDIN</p>	
<p><b>Approved by:</b></p> <p>_____</p> <p>Client</p>		

Software : \_\_\_\_\_

Archiving Place : \_\_\_\_\_

## TABLE OF CONTENT

CONTENT	PAGE
DOCUMENT APPROVAL .....	ii
TABLE OF CONTENT.....	iii
LIST OF FIGURES .....	iv
LIST OF TABLES .....	v
LIST OF APPENDICES .....	vi
1.1 PROJECT DESCRIPTION.....	1
1.2 SYSTEM IDENTIFICATION .....	2
1.3 ARCHITECTURE / BLUE PRINT .....	3
1.4 ARCHITECTURE / BLUEPRINT DESCRIPTION .....	4
1.4.1 Manage Car Rental .....	4
1.4.2 Manage Cab Service .....	6
1.4.3 Manage Car Review .....	8
1.4.4 Manage Users .....	10
1.4.5 Manage Report .....	11
2.1 DETAILED DESCRIPTION .....	13
2.1.1 Manage Car Rental .....	13
2.1.2 Manage Cab Service .....	32
2.1.3 Manage Car Review .....	50
2.1.4 Manage Users .....	66
2.1.5 Manage Report .....	78
2.2 ERD.....	84
2.3 DATA DICTIONARY.....	85

**LIST OF FIGURES**

Figure 1.3.1 MVC Package Diagram.....	3
Figure 1.4.1 Manage Car Rental Package Diagram.....	4
Figure 1.4.2 Manage Cab Service Package Diagram .....	6
Figure 1.4.3 Manage Car Review Package Diagram.....	8
Figure 1.4.4 Manage Users Package Diagram.....	10
Figure 1.4.5 Manage Report Package Diagram .....	12
Figure 2.1.1 Manage Car Rental Detailed Design .....	13
Figure 2.1.2 Manage Cab Service Detailed Design.....	32
Figure 2.1.3 Manage Car Review Detailed Design .....	50
Figure 2.1.4 Manage Users Detailed Description.....	66
Figure 2.1.5 Manage Report Detailed Description .....	78

## LIST OF TABLES

Table 1.1 Modules Description and Actors Involved .....	1
Table 1.4.1.1 Manage Car Rental Class Description (ManageCarRental) .....	5
Table 1.4.1.2 Manage Car Rental Class Description (App) .....	6
Table 1.4.2.1 Manage Cab Service Class Description (ManageCabService).....	7
Table 1.4.2.2 Manage Cab Service Class Description (App) .....	7
Table 1.4.3.1 Manage Car Review Class Description (ManageCarReview).....	8
Table 1.4.3.2 Manage Car Review Class Description (App).....	9
Table 1.4.4.1 Manage User Class Description (ManageUser).....	10
Table 1.4.4.2 Manage User Class Description (App) .....	11
Table 1.4.5.1 Manage Report Class Description (ManageReport).....	12
Table 1.4.5.2 Manage User Class Description (App) .....	12
Table 2.3.1 Admin Table Data Dictionary.....	<b>Error! Bookmark not defined.</b>
Table 2.3.2 User Table Data Dictionary .....	85
Table 2.3.3 Booking Table Data Dictionary .....	<b>Error! Bookmark not defined.</b>
Table 2.3.4 Payment Table Data Dictionary.....	<b>Error! Bookmark not defined.</b>
Table 2.3.5 CarRental Table Data Dictionary .....	85
Table 2.3.6 CabService Table Data Dictionary .....	86
Table 2.3.7 Car Table Data Dictionary .....	88
Table 2.3.8 Report Table Data Dictionary.....	<b>Error! Bookmark not defined.</b>

**LIST OF APPENDICES**

## CHAPTER 1

### 1.1 PROJECT DESCRIPTION

This system consists of five modules which Manage Car Rental, Manage Cab Service, Manage Car Review, Manage Users and Manage Report. The table below shows the description and actors involved for each module.

**Table 1.1 Modules Description and Actors Involved**

<b>Modules</b>	<b>Description</b>	<b>Actors Involved</b>
Manage Car Rental	<p>This module allows UMP students (Renter) who own cars to register their car up for rental in the system.</p> <p>This module also allows Rentee to update and delete their registered car.</p> <p>This module also allows UMP students (Rentee) to book for car rental offered in the system.</p>	Rentee and Renter
Manage Cab Service	<p>This module allows UMP students (Driver) to register their car for cab services in the system.</p> <p>This module also allows Driver to update and delete their registered cars.</p> <p>This module also allows UMP students (Passenger) to book for cab services offered in the system.</p>	Driver and Passenger
Manage Car Review	<p>This module allows UMP students (Rentee) to provide reviews and feedbacks for the car rental services they have booked. This module also allows UMP students (Passenger) to provide reviews and</p>	Rentee and Passenger



	feedbacks for the cab services they have ride and completed.	
Manage Users	<p>This module allows the Administrator of the system to monitor the users of the system and also the cars registered into the system.</p> <p>This module also allows the Administrator to view, modify and delete information from the system.</p> <p>This module also allows the Administrator to view a list of car registration request and approve or reject their requests.</p>	Admin
Manage Report	This module allows the Administrator of the system to view the report of the system such as the total number of users in the system and the total number of cars registered into the system.	Admin

## 1.2 SYSTEM IDENTIFICATION

System Title: Web-Based Car Rental and Cab Service System for UMP Students

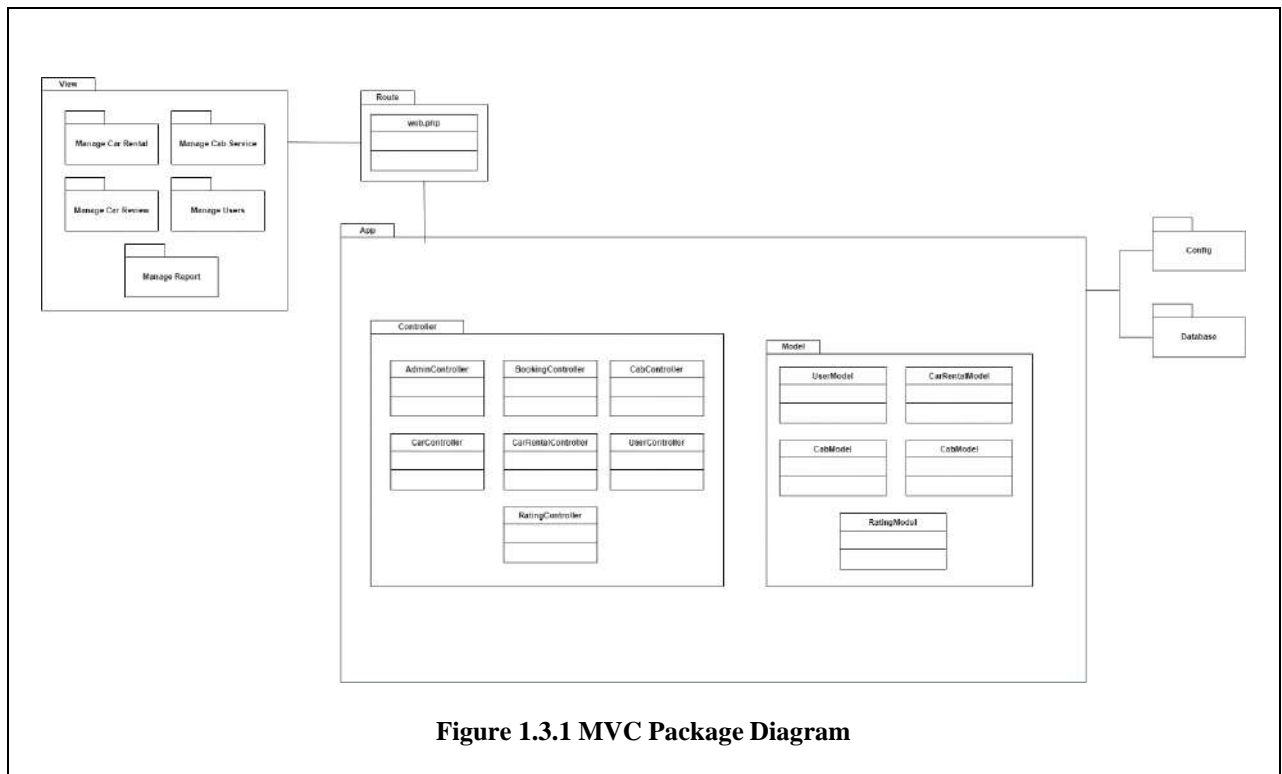
System Abbreviation: UMPCab

Year: 2022

Version: Version 1

System Identification Number: **SDD-UMPCab-2022-V1**

### 1.3 ARCHITECTURE / BLUE PRINT



**Figure 1.3.1 MVC Package Diagram**

The above diagram shows an MVC Package Diagram for the system. Model-View-Controller, or MVC for short, is an architectural pattern that divides an application layer into three logical components: model, view, and controller. Each component is designed to handle various parts of application development.

All of the data-related logic that the user works with is represented by the Model. This could be the data being transmitted between the View and Controller components, or it could be any other business logic-related data. For this system, there are five Models which are CarModel, CarRentalModel, CabModel, RatingModel, and UserModel.

All of the application's UI logic is handled by the View component.. For this system, the Views are separated for each module.

The Controller serves as a link between the Model and View components, processing all business logic and incoming requests, manipulating data using the Model, and interacting with the Views to generate the final output. Controllers are known to act as a 'bridge' between the View and the Model. This system consists of seven Controllers which are AdminController, BookingController, CabController, CarController, CarRentalController, UserController, and RatingController.

## 1.4 ARCHITECTURE / BLUEPRINT DESCRIPTION

### 1.4.1 Manage Car Rental

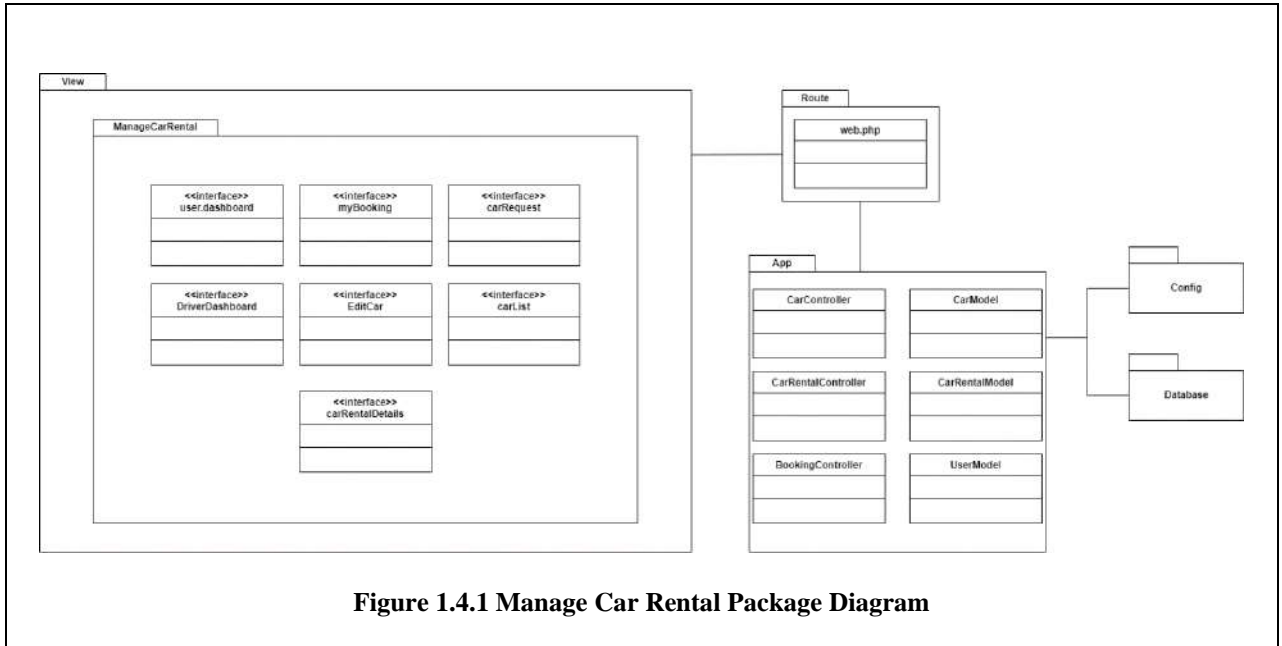


Figure 1.4.1 Manage Car Rental Package Diagram

**ManageCarRental****Table 1.4.1.1 Manage Car Rental Class Description (ManageCarRental)**

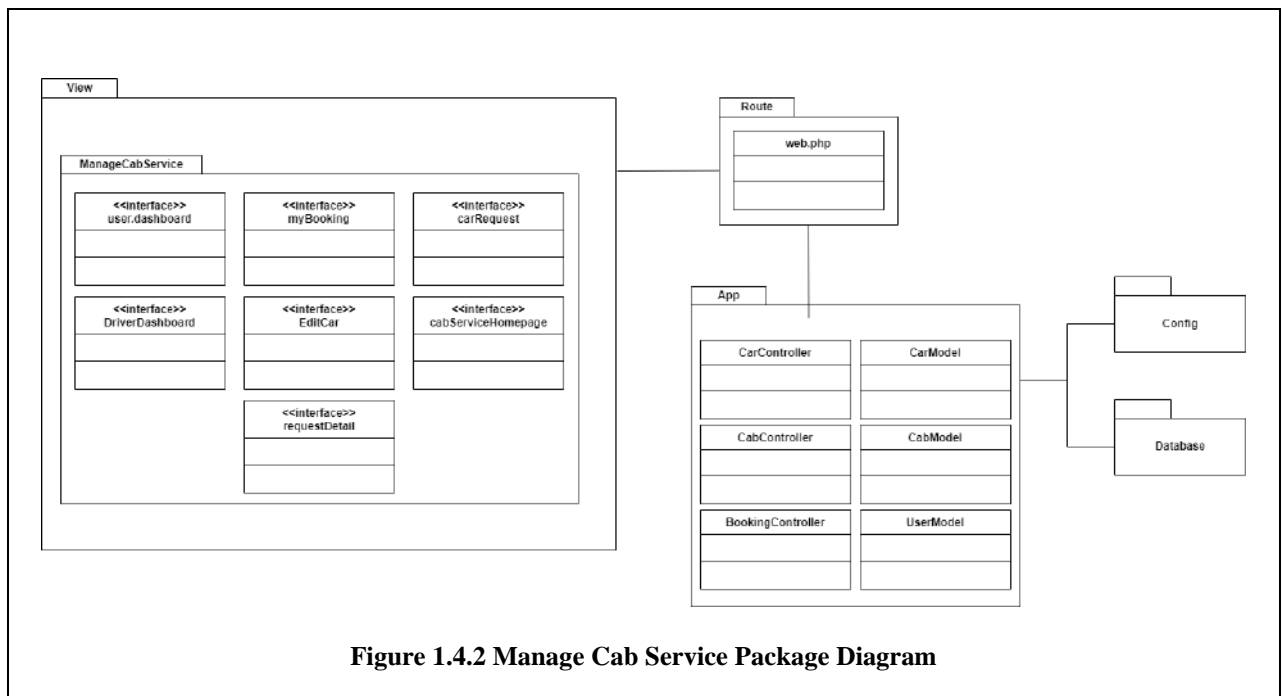
<b>Class Name</b>	<b>Description</b>
user.dashboard.blade.php	An interface that allows the Users to choose multiple options such as Book Car for rental, Book a carpool, Register car and View My Bookings.
myBooking.blade.php	An interface that allows Users to view their ongoing bookings as well as the history of their past bookings.
carRequest.blade.php	An interface that allows Users who have cars registered into the system to view car rental and cab ride requests in the system.
DriverDashboard.blade.php	An interface that allows Users to register their own car into the system and make them available for either car rental or carpool service.
EditCar.blade.php	An interface that allows User to edit and update the details of their registered car.
carList.blade.php	An interface that allows User to view a list of available car rental services in the system.
carRentalDetails.blade.php	An interface that allows User to provide their credentials to rent any car.

**App**

**Table 1.4.1.2 Manage Car Rental Class Description (App)**

Class Name	Description
CarController.php	This controller is used to retrieve, and display car related data.
CarRentalController.php	This controller is used to retrieve, and display car rentals related data.
BookingController.php	This controller is used to retrieve, and display car booking related data.
Car.php	This model is used to store and retrieve data regarding car in the database.
CarRental.php	This model is used to store and retrieve data regarding car rental in the database.
User.php	This model is used to store and retrieve data regarding user in the database.

**1.4.2 Manage Cab Service**



**Figure 1.4.2 Manage Cab Service Package Diagram**

**ManageCabService**

Table 1.4.2.1 Manage Cab Service Class Description (ManageCabService)

Class Name	Description
user.dashboard.blade.php	An interface that allows the Users to choose multiple options such as Book Car for rental, Book a carpool, Register car and View My Bookings.
myBooking.blade.php	An interface that allows Users to view their ongoing bookings as well as the history of their past bookings.
carRequest.blade.php	An interface that allows Users who have cars registered into the system to view car rental and cab ride requests in the system.
DriverDashboard.blade.php	An interface that allows Users to register their own car into the system and make them available for either car rental or carpool service.
EditCar.blade.php	An interface that allows User to edit and update the details of their registered car.
cabServiceHomepage.blade.php	An interface that allows User to view a map and choose their destination.
requestDetail.blade.php	An interface that allows Driver to view details of the cab ride request they have accepted.

## App

Table 1.4.2.2 Manage Cab Service Class Description (App)

Class Name	Description
CabController.php	This controller is used to retrieve and display cab services related data.
CarController.php	This controller is used to retrieve, and display car related data.
BookingController.php	This controller is used to retrieve, and display car booking related data.
Car.php	This model is used to store and retrieve data regarding car in the database.

Cab.php	This model is used to store and retrieve data regarding cab in the database.
User.php	This model is used to store and retrieve data regarding user in the database.

### 1.4.3 Manage Car Review

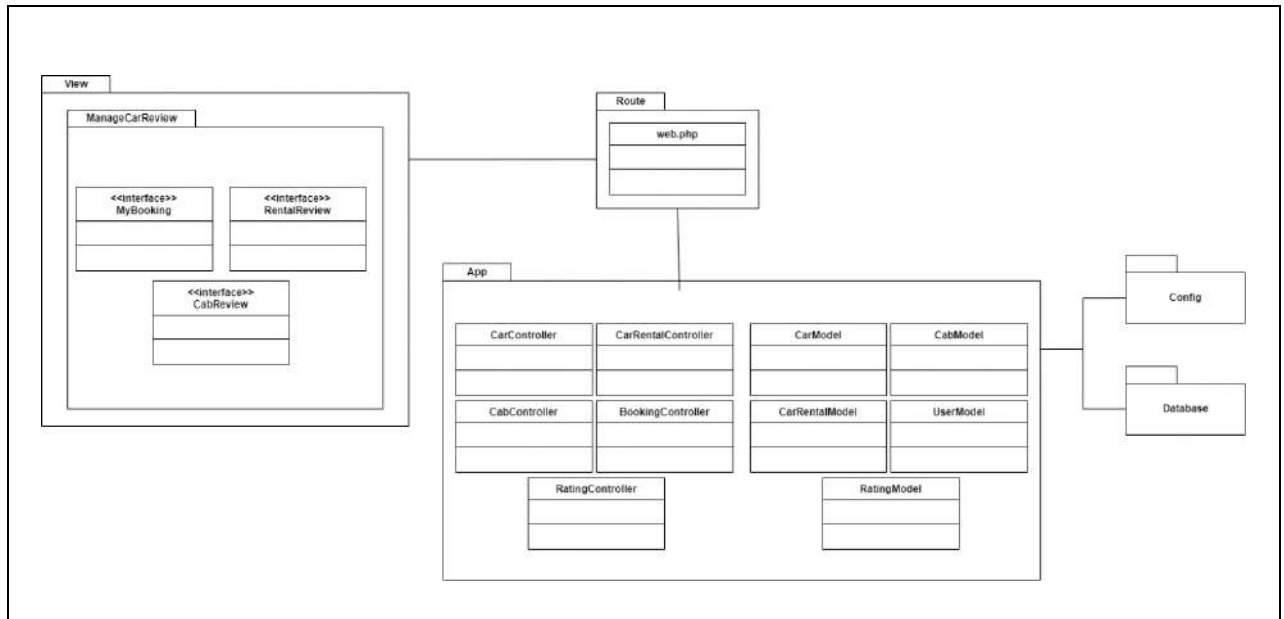


Figure 1.4.3 Manage Car Review Package Diagram

#### ManageCarReview

Table 1.4.3.1 Manage Car Review Class Description (ManageCarReview)

Class Name	Description
myBookings.blade.php	An interface that allows User to view a list of all their bookings for both car rental and cab services.
RentalReview.blade.php	An interface that allows User to provide reviews and rating to the cars that they have completed the rental period.
CabReview.blade.php	An interface that allows User to provide reviews and rating to the cabs that they have ride.

#### App

Table 1.4.3.2 Manage Car Review Class Description (App)

Class Name	Description
CarRentalController.php	This controller is used to retrieve, and display car rentals related data.
CabController.php	This controller is used to retrieve and display cab services related data.
CarController.php	This controller is used to retrieve, and display car related data.
BookingController.php	This controller is used to retrieve, and display car booking related data.
RatingController.php	This controller is used to retrieve, and display car rating related data.
CarRental.php	This model is used to store and retrieve data regarding car rental in the database.
Cab.php	This model is used to store and retrieve data regarding car rental in the database.
Car.php	This model is used to store and retrieve data regarding car in the database.
Rating.php	This model is used to store and retrieve data regarding rating in the database.
User.php	This model is used to store and retrieve data regarding user in the database.



### 1.4.4 Manage Users

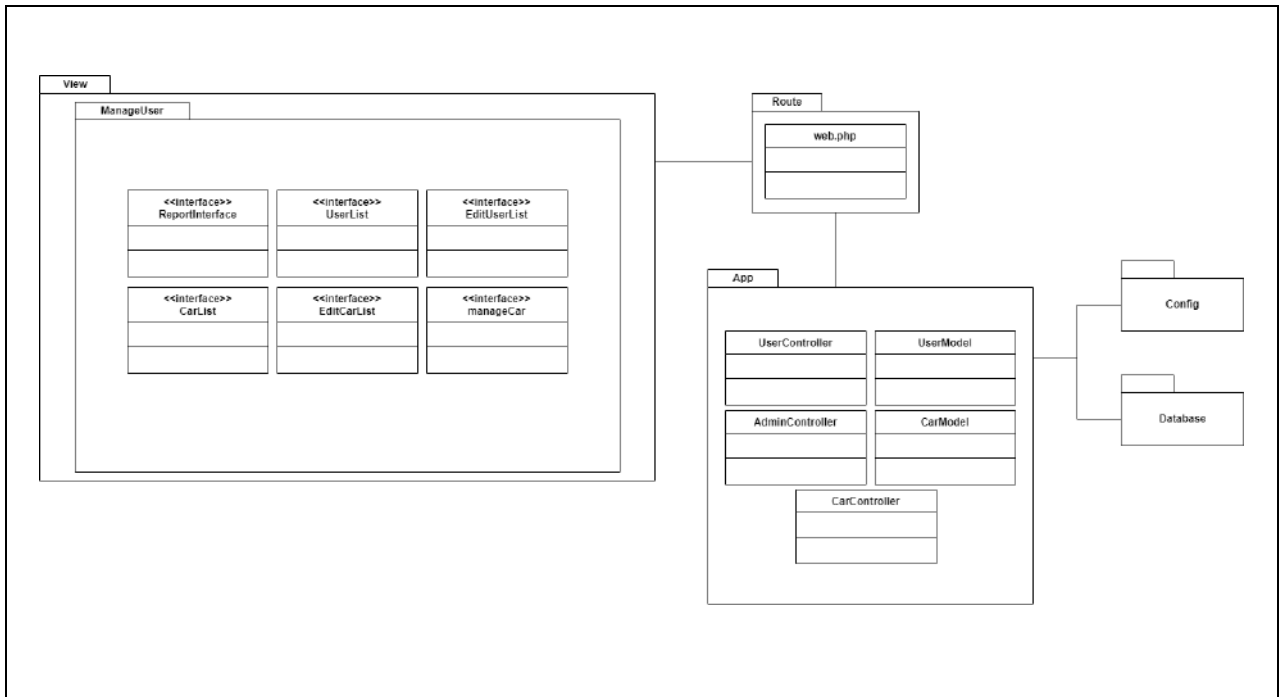


Figure 1.4.4 Manage Users Package Diagram

#### ManageUser

Table 1.4.4.1 Manage User Class Description (ManageUser)

Class Name	Description
ReportInterface.blade.php	An interface that allows Admin to view the reports of the system such as the total number of users in the system and the total cars into the system.
UserList.blade.php	An interface that allows Admin to view a list of users registered into the system and manage them by modifying and deleting any user information.
EditUserList.blade.php	An interface that allows Admin to update the existing user details by providing appropriate information.
CarList.blade.php	An interface that allows Admin to view a list of cars registered into the system and manage them by modifying and deleting any car information.
EditCarList.blade.php	An interface that allows Admin to update the existing car details by providing appropriate information.
manageCar.php	An interface that allows Admin to accept or reject car registration request.

**App**

**Table 1.4.4.2 Manage User Class Description (App)**

Class Name	Description
UserController.php	This controller is used to retrieve and display user related data.
User.php	This model is used to store and retrieve data regarding users in the database.
CarRentalController.php	This controller is used to retrieve, and display car rentals related data.
CarRental.php	This model is used to store and retrieve data regarding car rental in the database.
CabController.php	This controller is used to retrieve and display cab services related data.
Cab.php	This model is used to store and retrieve data regarding car rental in the database.

**1.4.5 Manage Report**

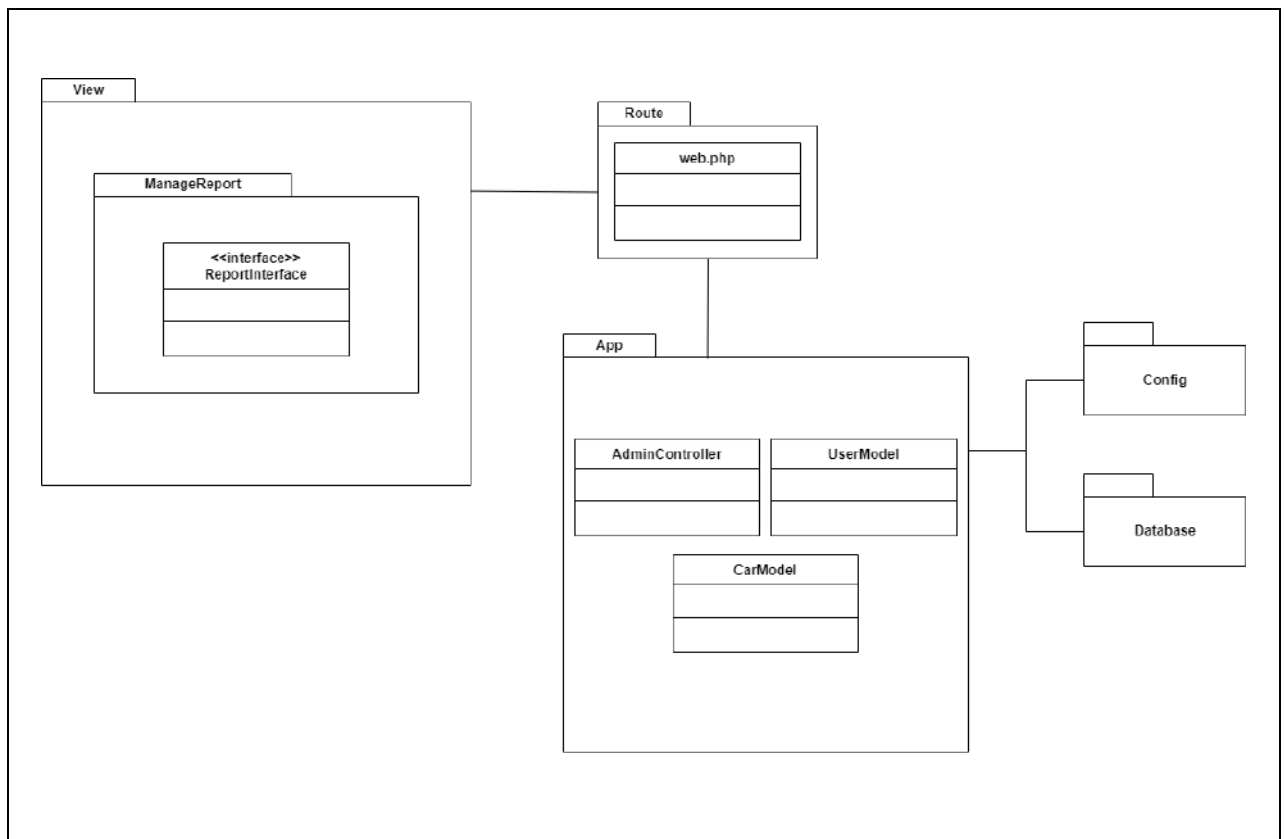


Figure 1.4.5 Manage Report Package Diagram

**ManageReport**

Table 1.4.5.1 Manage Report Class Description (ManageReport)

Class Name	Description
ReportInterface.blade.php	An interface that allows Admin to view the reports of the system such as the total number of users in the system and the total cars into the system.

**App**

Table 1.4.5.2 Manage User Class Description (App)

Class Name	Description
AdminController.php	This controller is used to retrieve and display Admin related data.
User.php	This model is used to store and retrieve data regarding users in the database.
Car.php	This model is used to store and retrieve data regarding car in the database.

# CHAPTER 2

## 2.1 DETAILED DESCRIPTION

### 2.1.1 Manage Car Rental

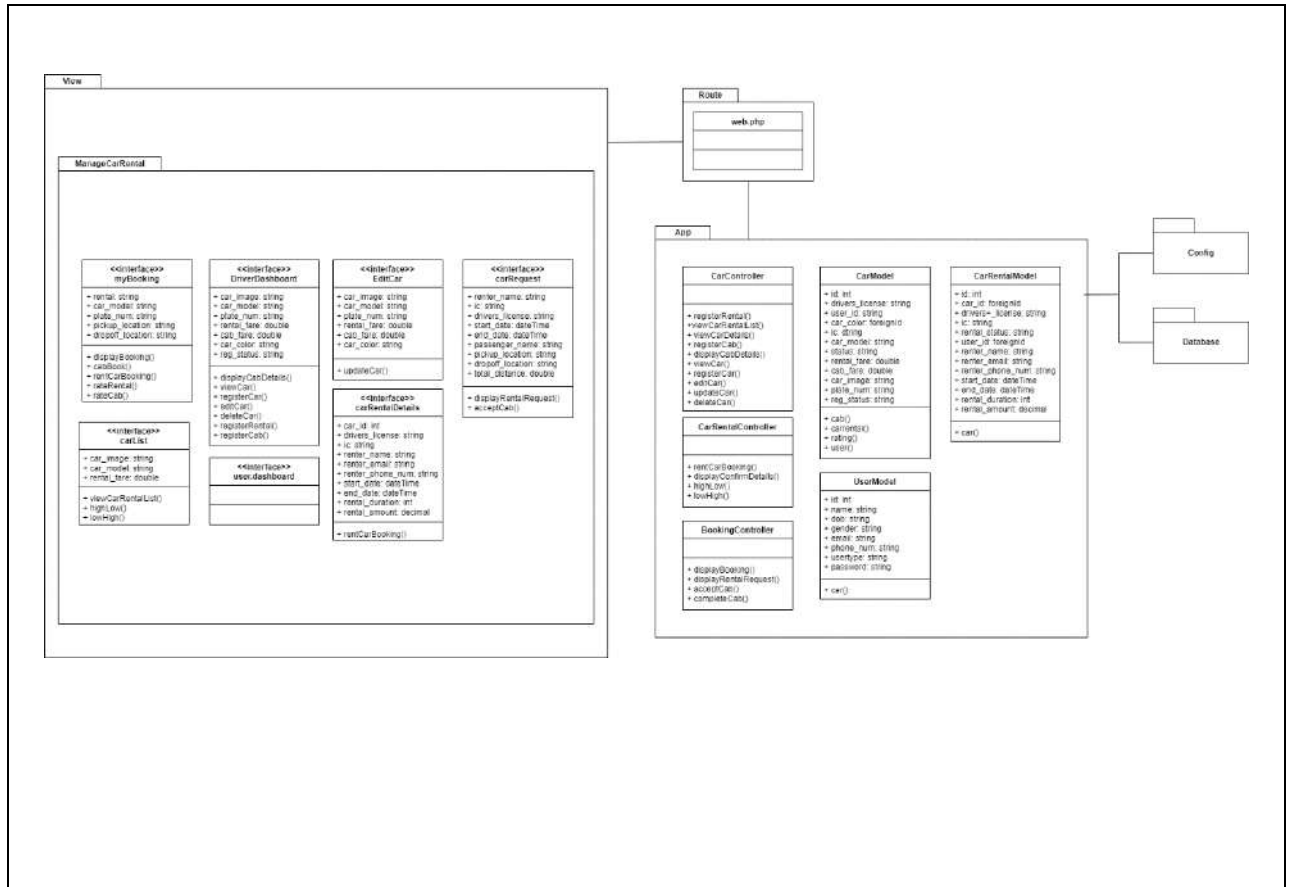


Figure 2.1.1 Manage Car Rental Detailed Design

#### user.dashboard.blade.php

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows the Users to choose multiple options such as Book Car for rental, Book a carpool, Register car and View My Bookings.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	Book Car for rental	Button
	Book a carpool	Button
	Register car	Button

	View My Bookings	Button
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	-	-
<b>Algorithm</b>	<p>START</p> <p>System display “Book Car for rental” button</p> <p>System display “Book a carpool” button</p> <p>System display “Register car” button</p> <p>System display “View My Bookings” button</p> <p>IF User click “Book Car for rental” button</p> <p>    Redirect to carList page</p> <p>ELSE IF User click “Book a carpool” button</p> <p>    Redirect to cabServiceHomepage page</p> <p>ELSE IF User click “Register car” button</p> <p>    Redirect to DriverDashboard page</p> <p>ELSE</p> <p>    Redirect to myBooking page</p> <p>ENDIF</p> <p>END</p>	

**myBooking.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Users to view their ongoing bookings as well as the history of their past bookings.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	rental	string
	car_model	string

	plate_num	string
	pickup_location	string
	dropoff_location	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	displayBooking()	This method is to display all the list of bookings for the current logged in user.
	cabBook()	This method is to allow users to request for a cab ride.
	rentCarBooking()	This method is to save user's car rental booking data.
	rateRental()	This method is to save car rental rating.
	rateCab()	This method is to save cab ride rating.
<b>Algorithm</b>	<b>displayBooking()</b>	
	<p>START</p> <p>RETRIEVE rental, car_model, plate_num, pickup_location, dropoff_location</p> <p>DISPLAY in myBooking page</p> <p>END</p>	
	<b>cabBook()</b>	
	<p>START</p> <p>ADD pickup_location, dropoff_location, total_distance</p> <p>STORE pickup_location, dropoff_location, total_distance</p> <p>RETURN myBooking page</p> <p>END</p>	
	<b>rentCarBooking()</b>	
	<p>START</p>	

	<p>ADD pickup_location, dropoff_location, total_distance</p> <p>STORE pickup_location, dropoff_location, total_distance</p> <p>RETURN myBooking page</p> <p>END</p> <p><b>rateRental()</b></p> <p>START</p> <p>ADD car_id, rating, feedback</p> <p>STORE car_id, rating, feedback</p> <p>RETURN myBooking page</p> <p>END</p> <p><b>rateCab()</b></p> <p>START</p> <p>ADD car_id, rating, feedback</p> <p>STORE car_id, rating, feedback</p> <p>RETURN myBooking page</p> <p>END</p>
--	--

**carRequest.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Users to view their ongoing bookings as well as the history of their past bookings.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	renter_name	string
	ic	string
	drivers_license	string

	start_date	dateTime
	end_date	dateTime
	passenger_name	string
	pickup_location	string
	dropoff_location	string
	total_distance	double
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	displayRentalRequest()	This method is to display a list of available car rental or cab ride request.
	acceptCab()	This method is to allow driver to accept cab ride request.
<b>Algorithm</b>	<p><b>displayRentalRequest()</b></p> <p>START</p> <p>RETRIEVE renter_name, ic, drivers_license, start_date, end_date, passenger_name, pickup_location, dropoff_location, total_distance</p> <p>DISPLAY in carRequest page</p> <p>END</p> <p><b>acceptCab()</b></p> <p>START</p> <p>RETRIEVE renter_name, ic, drivers_license, start_date, end_date, passenger_name, pickup_location, dropoff_location, total_distance</p> <p>DISPLAY in carRequest page</p> <p>User clicks “Accept Request” button</p> <p>Redirect to requestDetails page</p> <p>END</p>	



<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Users to register their own car into the system and make them available for either car rental or carpool service.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_image	string
	car_model	string
	plate_num	string
	rental_fare	double
	cab_fare	double
	car_color	string
	reg_status	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	displayCabDetails()	This method is to display details of the chosen cab.
	viewCar()	This method is to display details of registered car.
	registerCar()	This method is to register new car into the system.
	editCar()	This method is to edit and update registered car details
	deleteCar()	This method is to delete registered car.
	registerRental()	This method is to register car for rental.
	registerCab()	This method is to register car for carpool.
<b>Algorithm</b>	<b>viewCar()</b>  START  RETRIEVE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status  DISPLAY in DriverDashboard page	

END

**registerCar()**

START

ADD car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status

STORE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status

DISPLAY DriverDashboard page

END

**editCar()**

START

RETRIEVE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status

DISPLAY in EditCar page

END

**deleteCar()**

START

READ car\_id

DELETE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status from database

RETURN DriverDashboard page

END

**registerRental()**

START

	<p>User click “Register for rental” button</p> <p>UPDATE rental</p> <p>END</p> <p><b>registerCab()</b></p> <p>START</p> <p>User click “Register for carpool” button</p> <p>UPDATE cab</p> <p>END</p>
--	--

**EditCar.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows User to edit and update the details of their registered car.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_image	string
	car_model	string
	plate_num	string
	rental_fare	double
	cab_fare	double
	car_color	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	updateCar()	This method is to update the details of registered car into the system.
<b>Algorithm</b>	<p><b>updateCar()</b></p> <p>START</p> <p>READ car_id</p>	

	<p>UPDATE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status into the database</p> <p>RETURN DriverDashboard page</p> <p>END</p>
--	---

**carList.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows User to view a list of available car rental services in the system.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_image	string
	car_model	string
	rental_fare	double
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	viewCarRentalList()	This method is to display a list of available car rental services.
	highLow()	This method is to sort the car rental list in terms of rental_fare from highest to lowest.
	lowHigh()	This method is to sort the car rental list in terms of rental_fare from lowest to highest.
<b>Algorithm</b>	<p><b>viewCarRentalList()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE car_image, car_model, rental_fare</p> <p style="padding-left: 40px;">DISPLAY car_image, car_model, rental_fare</p> <p>END</p> <p><b>highLow()</b></p> <p>START</p>	

	<p>RETRIEVE car_image, car_model, rental_fare</p> <p>DISPLAY car_image, car_model, rental_fare with rental_fare from highest to lowest</p> <p>END</p> <p><b>lowHigh()</b></p> <p>START</p> <p>RETRIEVE car_image, car_model, rental_fare</p> <p>DISPLAY car_image, car_model, rental_fare with rental_fare from lowest to highest</p> <p>END</p>
--	--

**carRentalDetails.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows User to provide their credentials to rent any car.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_id	string
	drivers_license	string
	ic	string
	renter_name	string
	renter_email	string
	renter_phone_num	string
	start_date	dateTime
	end_date	dateTime
	rental_duration	int
	rental_amount	decimal
	<b>Methods</b>	<b>Method Name</b>

	rentCarBooking()	This method is to save user's car rental booking data.
<b>Algorithm</b>	<b>rentCarBooking()</b> START ADD pickup_location, dropoff_location, total_distance STORE pickup_location, dropoff_location, total_distance RETURN myBooking page END	

### CarController.php

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve, and display car related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	registerRental()	This method is to register car for rental.
	viewCarRentalList()	This method is to display a list of available car rental services.
	viewCarDetails()	This method is to display details of the chosen car rental.
	registerCab()	This method is to register car for carpool.
	displayCabDetails()	This method is to display details of the chosen cab.
	viewCar()	This method is to display details of registered car.
	registerCar()	This method is to register new car into the system.
	editCar()	This method is to edit and update registered car details

	updateCar()	This method is to update the details of registered car into the system.
	deleteCar()	This method is to delete registered car.
<b>Algorithm</b>	<p><b>viewCarRentalList()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE car_image, car_model, rental_fare</p> <p style="padding-left: 40px;">DISPLAY car_image, car_model, rental_fare</p> <p>END</p> <p><b>viewCarDetails()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE car_id</p> <p style="padding-left: 40px;">DISPLAY car_id</p> <p>END</p> <p><b>viewCar()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status</p> <p style="padding-left: 40px;">DISPLAY in DriverDashboard page</p> <p>END</p> <p><b>registerCar()</b></p> <p>START</p> <p style="padding-left: 40px;">ADD car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status</p> <p style="padding-left: 40px;">STORE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status</p> <p style="padding-left: 40px;">DISPLAY DriverDashboard page</p>	

END

**editCar()**

START

RETRIEVE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status

DISPLAY in EditCar page

END

**updateCar()**

START

READ car\_id

UPDATE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status into the database

RETURN DriverDashboard page

END

**deleteCar()**

START

READ car\_id

DELETE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status from database

RETURN DriverDashboard page

END

**registerRental()**

START



	<p>User click “Register for rental” button</p> <p>UPDATE rental</p> <p>END</p> <p><b>registerCab()</b></p> <p>START</p> <p>User click “Register for carpool” button</p> <p>UPDATE cab</p> <p>END</p>
--	--

### CarRentalController.php

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve, and display car rentals related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	rentCarBooking()	This method is to save user’s car rental booking data.
	highLow()	This method is to sort the car rental list in terms of rental_fare from highest to lowest.
	lowHigh()	This method is to sort the car rental list in terms of rental_fare from lowest to highest.
<b>Algorithm</b>	<p><b>rentCarBooking()</b></p> <p>START</p> <p>ADD pickup_location, dropoff_location, total_distance</p> <p>STORE pickup_location, dropoff_location, total_distance</p> <p>RETURN myBooking page</p>	

	<p>END</p> <p><b>highLow()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE car_image, car_model, rental_fare</p> <p style="padding-left: 40px;">DISPLAY car_image, car_model, rental_fare with rental_fare from highest to lowest</p> <p>END</p> <p><b>lowHigh()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE car_image, car_model, rental_fare</p> <p style="padding-left: 40px;">DISPLAY car_image, car_model, rental_fare with rental_fare from lowest to highest</p> <p>END</p>
--	--

### BookingController.php

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve, and display car booking related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	displayBooking()	This method is to display all the list of bookings for the current logged in user.
	displayRentalRequest()	This method is to display a list of available car rental or cab ride request.
	acceptCab()	This method is to allow driver to accept cab ride request.

	completeCab()	This method is to allow Drivers to complete their cab ride request
<b>Algorithm</b>	<p><b>displayBooking()</b></p> <p>START</p> <p>    RETRIEVE rental, car_model, plate_num, pickup_location, dropoff_location</p> <p>    DISPLAY in myBooking page</p> <p>END</p> <p><b>displayRentalRequest()</b></p> <p>START</p> <p>    RETRIEVE renter_name, ic, drivers_license, start_date, end_date, passenger_name, pickup_location, dropoff_location, total_distance</p> <p>    DISPLAY in carRequest page</p> <p>END</p> <p><b>acceptCab()</b></p> <p>START</p> <p>    RETRIEVE renter_name, ic, drivers_license, start_date, end_date, passenger_name, pickup_location, dropoff_location, total_distance</p> <p>    DISPLAY in carRequest page</p> <p>    User clicks “Accept Request” button</p> <p>    Redirect to requestDetails page</p> <p>END</p> <p><b>completeCab()</b></p> <p>START</p> <p>    User click the “Ride completed” button</p>	

	UPDATE status into database  DISPLAY carRequest page  END
--	---

### Car.php

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding car in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_id	int
	drivers_license	string
	user_id	int
	car_color	string
	ic	string
	car_model	string
	status	string
	rental_fare	double
	cab_fare	double
	car_image	string
	plate_num	string
	reg_status	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	cab()	The Car hasMany Cab
	carrental()	The Car hasMany CarRentals
	rating()	The Car hasMany Rating
	user()	The Car belongsTo User

<b>Algorithm</b>	-
------------------	---

**CarRental.php**

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding car rental in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_rental_id	int
	car_id	int
	drivers_license	string
	ic	string
	rental_status	string
	user_id	string
	renter_name	string
	renter_email	string
	renter_phone_num	string
	start_date	dateTime
	end_date	dateTime
	rental_duration	int
	rental_amount	decimal
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	car()	The CarRental belongsTo Car
<b>Algorithm</b>	-	

**User.php**

<b>Class Type</b>	Model Class
-------------------	-------------

<b>Responsibility</b>	This model is used to store and retrieve data regarding user in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	user_id	int
	name	string
	dob	string
	gender	string
	email	string
	phone_num	string
	usertype	string
	password	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	car()	The User hasMany Car
<b>Algorithm</b>	-	

### 2.1.2 Manage Cab Service

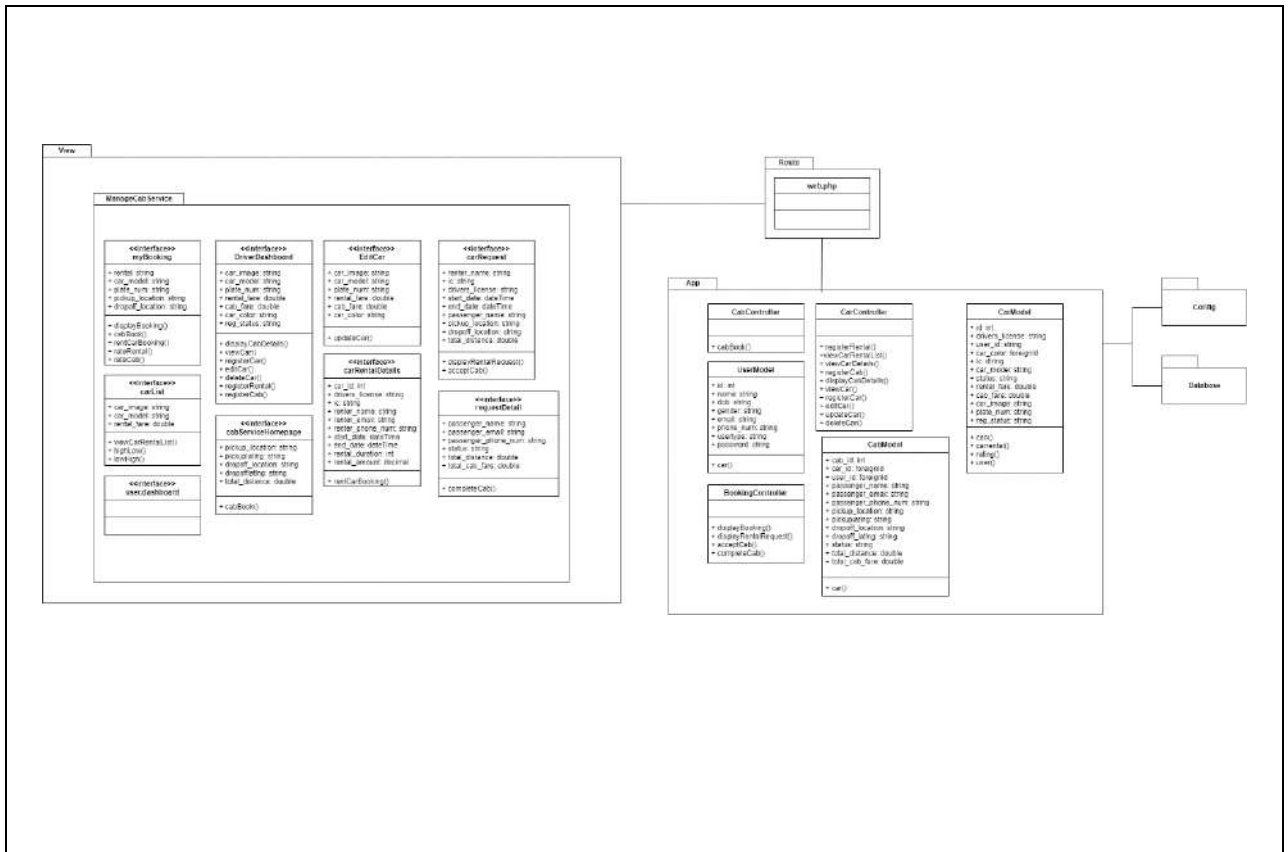


Figure 2.1.2 Manage Cab Service Detailed Design

user.dashboard.blade.php

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows the Users to choose multiple options such as Book Car for rental, Book a carpool, Register car and View My Bookings.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	Book Car for rental	Button
	Book a carpool	Button
	Register car	Button
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	-	-
<b>Algorithm</b>	START	

	<p>System display “Book Car for rental” button</p> <p>System display “Book a carpool” button</p> <p>System display “Register car” button</p> <p>System display “View My Bookings” button</p> <p>IF User click “Book Car for rental” button</p> <p style="padding-left: 40px;">Redirect to carList page</p> <p>ELSE IF User click “Book a carpool” button</p> <p style="padding-left: 40px;">Redirect to cabServiceHomepage page</p> <p>ELSE IF User click “Register car” button</p> <p style="padding-left: 40px;">Redirect to DriverDashboard page</p> <p>ELSE</p> <p style="padding-left: 40px;">Redirect to myBooking page</p> <p>ENDIF</p> <p>END</p>
--	--

**myBooking.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Users to view their ongoing bookings as well as the history of their past bookings.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	rental	string
	car_model	string
	plate_num	string
	pickup_location	string
	dropoff_location	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>



	displayBooking()	This method is to display all the list of bookings for the current logged in user.
	cabBook()	This method is to allow users to request for a cab ride.
	rentCarBooking()	This method is to save user's car rental booking data.
	rateRental()	This method is to save car rental rating.
	rateCab()	This method is to save cab ride rating.
<b>Algorithm</b>	<p><b>displayBooking()</b></p> <p>START</p> <p>    RETRIEVE rental, car_model, plate_num, pickup_location, dropoff_location</p> <p>    DISPLAY in myBooking page</p> <p>END</p> <p><b>cabBook()</b></p> <p>START</p> <p>    ADD pickup_location, dropoff_location, total_distance</p> <p>    STORE pickup_location, dropoff_location, total_distance</p> <p>    RETURN myBooking page</p> <p>END</p> <p><b>rentCarBooking()</b></p> <p>START</p> <p>    ADD pickup_location, dropoff_location, total_distance</p> <p>    STORE pickup_location, dropoff_location, total_distance</p> <p>    RETURN myBooking page</p> <p>END</p>	

	<p><b>rateRental()</b></p> <p>START</p> <p>    ADD car_id, rating, feedback</p> <p>    STORE car_id, rating, feedback</p> <p>    RETURN myBooking page</p> <p>END</p> <p><b>rateCab()</b></p> <p>START</p> <p>    ADD car_id, rating, feedback</p> <p>    STORE car_id, rating, feedback</p> <p>    RETURN myBooking page</p> <p>END</p>
--	--

**carRequest.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Users to view their ongoing bookings as well as the history of their past bookings.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	renter_name	string
	ic	string
	drivers_license	string
	start_date	dateTime
	end_date	dateTime
	passenger_name	string

	pickup_location	string
	dropoff_location	string
	total_distance	double
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	displayRentalRequest()	This method is to display a list of available car rental or cab ride request.
	acceptCab()	This method is to allow driver to accept cab ride request.
<b>Algorithm</b>	<p><b>displayRentalRequest()</b></p> <p>START</p> <p>RETRIEVE renter_name, ic, drivers_license, start_date, end_date, passenger_name, pickup_location, dropoff_location, total_distance</p> <p>DISPLAY in carRequest page</p> <p>END</p> <p><b>acceptCab()</b></p> <p>START</p> <p>RETRIEVE renter_name, ic, drivers_license, start_date, end_date, passenger_name, pickup_location, dropoff_location, total_distance</p> <p>DISPLAY in carRequest page</p> <p>User clicks “Accept Request” button</p> <p>Redirect to requestDetails page</p> <p>END</p>	

**DriverDashboard.blade.php**

<b>Class Type</b>	Boundary Class
<b>Responsibility</b>	An interface that allows Users to register their own car into the system and make them available for either car rental or carpool service.

Attributes	Attributes Name	Attributes Type
	car_image	string
	car_model	string
	plate_num	string
	rental_fare	double
	cab_fare	double
	car_color	string
	reg_status	string
Methods	Method Name	Description
	displayCabDetails()	This method is to display details of the chosen cab.
	viewCar()	This method is to display details of registered car.
	registerCar()	This method is to register new car into the system.
	editCar()	This method is to edit and update registered car details
	deleteCar()	This method is to delete registered car.
	registerRental()	This method is to register car for rental.
	registerCab()	This method is to register car for carpool.
Algorithm	<p><b>viewCar()</b></p> <p>START</p> <p>    RETRIEVE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status</p> <p>    DISPLAY in DriverDashboard page</p> <p>END</p> <p><b>registerCar()</b></p>	

START

ADD car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status

STORE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status

DISPLAY DriverDashboard page

END

**editCar()**

START

RETRIEVE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status

DISPLAY in EditCar page

END

**deleteCar()**

START

READ car\_id

DELETE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status from database

RETURN DriverDashboard page

END

**registerRental()**

START

User click "Register for rental" button

UPDATE rental

END

	<p><b>registerCab()</b></p> <p>START</p> <p style="padding-left: 40px;">User click “Register for carpool” button</p> <p style="padding-left: 40px;">UPDATE cab</p> <p>END</p>
--	---

**EditCar.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows User to edit and update the details of their registered car.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_image	string
	car_model	string
	plate_num	string
	rental_fare	double
	cab_fare	double
	car_color	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	updateCar()	This method is to update the details of registered car into the system.
<b>Algorithm</b>	<p><b>updateCar()</b></p> <p>START</p> <p style="padding-left: 40px;">READ car_id</p> <p style="padding-left: 40px;">UPDATE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status into the database</p> <p style="padding-left: 40px;">RETURN DriverDashboard page</p>	

	END
--	-----

**cabServiceHomepage.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows User to view a map and choose their destination.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	pickup_location	string
	pickuplatlng	string
	dropoff_location	string
	dropofflatlng	string
	total_distance	double
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	cabBook()	This method is to allow users to request for a cab ride.
<b>Algorithm</b>	<b>cabBook()</b> START ADD pickup_location, dropoff_location, total_distance STORE pickup_location, dropoff_location, total_distance RETURN myBooking page END	

**requestDetail.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Driver to view details of the cab ride request they have accepted.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	passenger_name	string

	passenger_email	string
	passenger_phone_num	string
	status	string
	total_distance	double
	total_cab_fare	double
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	completeCab()	This method is to allow Drivers to complete their cab ride request
<b>Algorithm</b>	<b>completeCab()</b>  START  User click the “Ride completed” button  UPDATE status into database  DISPLAY carRequest page  END	

### CabController.php

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve and display cab services related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	cabBook()	This method is to allow users to request for a cab ride.
<b>Algorithm</b>	<b>cabBook()</b>  START  ADD pickup_location, dropoff_location, total_distance  STORE pickup_location, dropoff_location, total_distance	



	RETURN myBooking page END
--	------------------------------

**CarController.php**

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve, and display car related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	registerRental()	This method is to register car for rental.
	viewCarRentalList()	This method is to display a list of available car rental services.
	viewCarDetails()	This method is to display details of the chosen car rental.
	registerCab()	This method is to register car for carpool.
	displayCabDetails()	This method is to display details of the chosen cab.
	viewCar()	This method is to display details of registered car.
	registerCar()	This method is to register new car into the system.
	editCar()	This method is to edit and update registered car details
	updateCar()	This method is to update the details of registered car into the system.
	deleteCar()	This method is to delete registered car.
<b>Algorithm</b>	<b>viewCarRentalList()</b> START RETRIEVE car_image, car_model, rental_fare	

	<pre>    DISPLAY car_image, car_model, rental_fare END  <b>viewCarDetails()</b> START     RETRIEVE car_id     DISPLAY car_id END  <b>viewCar()</b> START     RETRIEVE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status     DISPLAY in DriverDashboard page END  <b>registerCar()</b> START     ADD car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status     STORE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status     DISPLAY DriverDashboard page END  <b>editCar()</b> START     RETRIEVE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status</pre>
--	---

	<p>DISPLAY in EditCar page</p> <p>END</p> <p><b>updateCar()</b></p> <p>START</p> <p>    READ car_id</p> <p>    UPDATE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status into the database</p> <p>    RETURN DriverDashboard page</p> <p>END</p> <p><b>deleteCar()</b></p> <p>START</p> <p>    READ car_id</p> <p>    DELETE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status from database</p> <p>    RETURN DriverDashboard page</p> <p>END</p> <p><b>registerRental()</b></p> <p>START</p> <p>    User click “Register for rental” button</p> <p>    UPDATE rental</p> <p>END</p> <p><b>registerCab()</b></p> <p>START</p>
--	---

	User click “Register for carpool” button  UPDATE cab  END
--	---

### BookingController.php

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve, and display car booking related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	displayBooking()	This method is to display all the list of bookings for the current logged in user.
	displayRentalRequest()	This method is to display a list of available car rental or cab ride request.
	acceptCab()	This method is to allow driver to accept cab ride request.
	completeCab()	This method is to allow Drivers to complete their cab ride request
<b>Algorithm</b>	<b>displayBooking()</b>  START  RETRIEVE rental, car_model, plate_num, pickup_location, dropoff_location  DISPLAY in myBooking page  END  <b>displayRentalRequest()</b>  START	

	<p>RETRIEVE renter_name, ic, drivers_license, start_date, end_date, passenger_name, pickup_location, dropoff_location, total_distance</p> <p>DISPLAY in carRequest page</p> <p>END</p> <p><b>acceptCab()</b></p> <p>START</p> <p>RETRIEVE renter_name, ic, drivers_license, start_date, end_date, passenger_name, pickup_location, dropoff_location, total_distance</p> <p>DISPLAY in carRequest page</p> <p>User clicks “Accept Request” button</p> <p>Redirect to requestDetails page</p> <p>END</p> <p><b>completeCab()</b></p> <p>START</p> <p>User click the “Ride completed” button</p> <p>UPDATE status into database</p> <p>DISPLAY carRequest page</p> <p>END</p>
--	--

**Car.php**

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding car in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_id	int

	drivers_license	string
	user_id	int
	car_color	string
	ic	string
	car_model	string
	status	string
	rental_fare	double
	cab_fare	double
	car_image	string
	plate_num	string
	reg_status	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	cab()	The Car hasMany Cab
	carrental()	The Car hasMany CarRentals
	rating()	The Car hasMany Rating
	user()	The Car belongsTo User
<b>Algorithm</b>	-	

**Cab.php**

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding cab in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	cab_id	int
	car_id	int
	user_id	int

	passenger_name	string
	passenger_name	string
	passenger_phone_num	string
	pickup_location	string
	pickuplatlng	string
	dropoff_location	string
	dropofflatlng	string
	status	string
	total_distance	double
	total_cab_fare	double
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	car()	The Cab belongsTo Cab
<b>Algorithm</b>	-	

**User.php**

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding user in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	user_id	int
	name	string
	dob	string
	gender	string
	email	string
	phone_num	string
	usertype	string

	password	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	car()	The User hasMany Car
<b>Algorithm</b>	-	



### 2.1.3 Manage Car Review

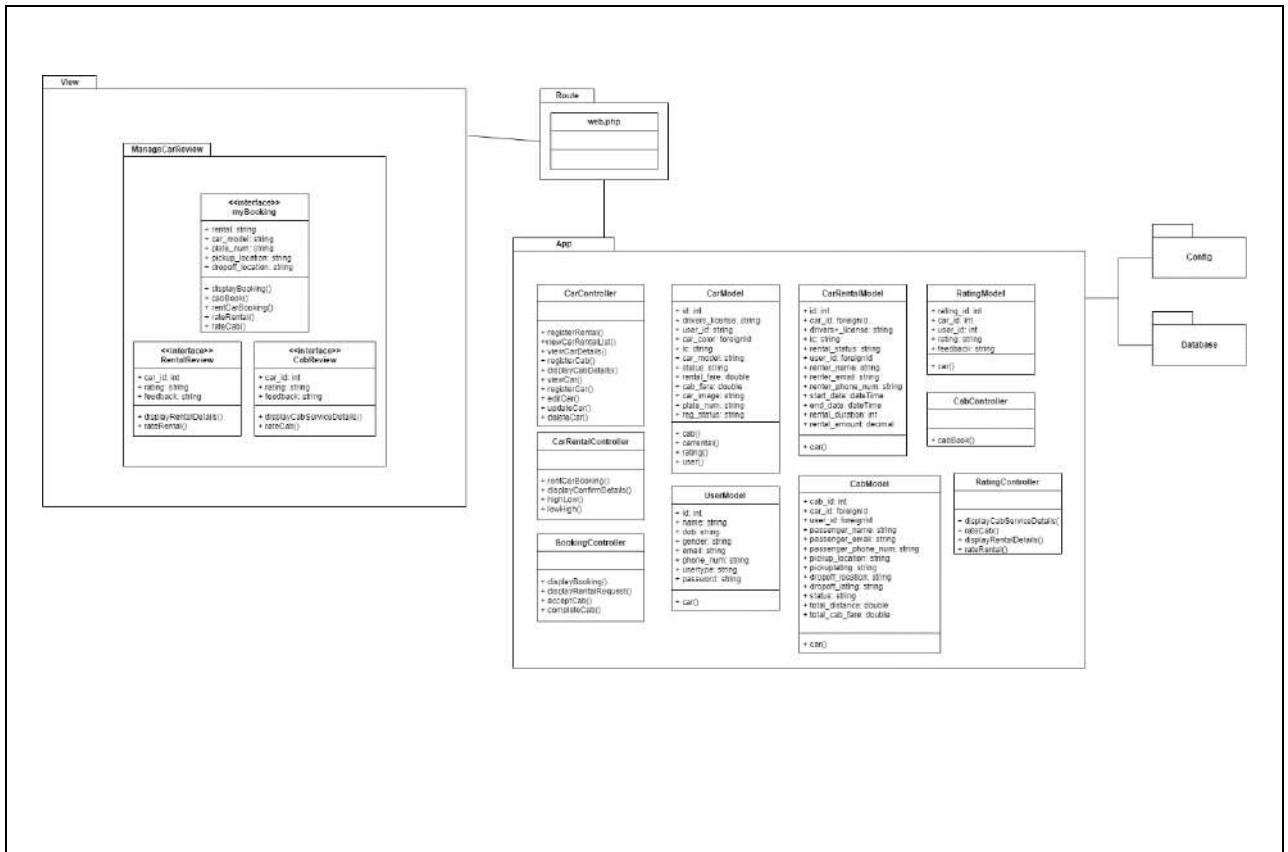


Figure 2.1.3 Manage Car Review Detailed Design

#### myBooking.blade.php

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Users to view their ongoing bookings as well as the history of their past bookings.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	rental	string
	car_model	string
	plate_num	string
	pickup_location	string
	dropoff_location	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	displayBooking()	This method is to display all the list of bookings for the current logged in user.

	cabBook()	This method is to allow users to request for a cab ride.
	rentCarBooking()	This method is to save user's car rental booking data.
	rateRental()	This method is to save car rental rating.
	rateCab()	This method is to save cab ride rating.
<b>Algorithm</b>	<p><b>displayBooking()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE rental, car_model, plate_num, pickup_location, dropoff_location</p> <p style="padding-left: 40px;">DISPLAY in myBooking page</p> <p>END</p> <p><b>cabBook()</b></p> <p>START</p> <p style="padding-left: 40px;">ADD pickup_location, dropoff_location, total_distance</p> <p style="padding-left: 40px;">STORE pickup_location, dropoff_location, total_distance</p> <p style="padding-left: 40px;">RETURN myBooking page</p> <p>END</p> <p><b>rentCarBooking()</b></p> <p>START</p> <p style="padding-left: 40px;">ADD pickup_location, dropoff_location, total_distance</p> <p style="padding-left: 40px;">STORE pickup_location, dropoff_location, total_distance</p> <p style="padding-left: 40px;">RETURN myBooking page</p> <p>END</p>	

	<p><b>rateRental()</b></p> <p>START</p> <p>    ADD car_id, rating, feedback</p> <p>    STORE car_id, rating, feedback</p> <p>    RETURN myBooking page</p> <p>END</p> <p><b>rateCab()</b></p> <p>START</p> <p>    ADD car_id, rating, feedback</p> <p>    STORE car_id, rating, feedback</p> <p>    RETURN myBooking page</p> <p>END</p>
--	--

### RentalReview.blade.php

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows User to provide reviews and rating to the cars that they have completed the rental period.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_id	int
	rating	string
	feedback	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	displayRentalDetails()	This method is to display the car rental details before rating
	rateRental()	This method is to save car rental rating.
<b>Algorithm</b>	<b>displayRentalDetails()</b>	

	<p>START</p> <p style="padding-left: 40px;">RETRIEVE car_id, car_model, plate_num, pickup_location, dropoff_location</p> <p style="padding-left: 40px;">DISPLAY car_id, car_model, plate_num, pickup_location, dropoff_location</p> <p>END</p> <p><b>rateRental()</b></p> <p>START</p> <p style="padding-left: 40px;">ADD car_id, rating, feedback</p> <p style="padding-left: 40px;">STORE car_id, rating, feedback</p> <p style="padding-left: 40px;">RETURN myBooking page</p> <p>END</p>
--	--

**CabReview.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows User to provide reviews and rating to the cabs that they have ride.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_id	int
	rating	string
	feedback	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	displayRentalDetails()	This method is to display the car rental details before rating
	rateRental()	This method is to save car rental rating.
<b>Algorithm</b>	<p><b>displayRentalDetails()</b></p> <p>START</p>	

	<p style="margin-left: 40px;">RETRIEVE car_id, car_model, plate_num, pickup_location, dropoff_location</p> <p style="margin-left: 40px;">DISPLAY car_id, car_model, plate_num, pickup_location, dropoff_location</p> <p>END</p> <p><b>rateRental()</b></p> <p>START</p> <p style="margin-left: 40px;">ADD car_id, rating, feedback</p> <p style="margin-left: 40px;">STORE car_id, rating, feedback</p> <p style="margin-left: 40px;">RETURN myBooking page</p> <p>END</p>
--	--

**CarRentalController.php**

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve, and display car rentals related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	rentCarBooking()	This method is to save user's car rental booking data.
	highLow()	This method is to sort the car rental list in terms of rental_fare from highest to lowest.
	lowHigh()	This method is to sort the car rental list in terms of rental_fare from lowest to highest.
<b>Algorithm</b>	<p><b>rentCarBooking()</b></p> <p>START</p> <p style="margin-left: 40px;">ADD pickup_location, dropoff_location, total_distance</p>	

	<p>STORE pickup_location, dropoff_location, total_distance</p> <p>RETURN myBooking page</p> <p>END</p> <p><b>highLow()</b></p> <p>START</p> <p>RETRIEVE car_image, car_model, rental_fare</p> <p>DISPLAY car_image, car_model, rental_fare with rental_fare from highest to lowest</p> <p>END</p> <p><b>lowHigh()</b></p> <p>START</p> <p>RETRIEVE car_image, car_model, rental_fare</p> <p>DISPLAY car_image, car_model, rental_fare with rental_fare from lowest to highest</p> <p>END</p>
--	--

**CabController.php**

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve and display cab services related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	cabBook()	This method is to allow users to request for a cab ride.
<b>Algorithm</b>	cabBook()	

	<p>START</p> <p>ADD pickup_location, dropoff_location, total_distance</p> <p>STORE pickup_location, dropoff_location, total_distance</p> <p>RETURN myBooking page</p> <p>END</p>
--	--

### CarController.php

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve, and display car related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	registerRental()	This method is to register car for rental.
	viewCarRentalList()	This method is to display a list of available car rental services.
	viewCarDetails()	This method is to display details of the chosen car rental.
	registerCab()	This method is to register car for carpool.
	displayCabDetails()	This method is to display details of the chosen cab.
	viewCar()	This method is to display details of registered car.
	registerCar()	This method is to register new car into the system.
	editCar()	This method is to edit and update registered car details
	updateCar()	This method is to update the details of registered car into the system.
deleteCar()	This method is to delete registered car.	

Algorithm	
	<p><b>viewCarRentalList()</b></p> <p>START</p> <p>    RETRIEVE car_image, car_model, rental_fare</p> <p>    DISPLAY car_image, car_model, rental_fare</p> <p>END</p>
	<p><b>viewCarDetails()</b></p> <p>START</p> <p>    RETRIEVE car_id</p> <p>    DISPLAY car_id</p> <p>END</p>
	<p><b>viewCar()</b></p> <p>START</p> <p>    RETRIEVE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status</p> <p>    DISPLAY in DriverDashboard page</p> <p>END</p>
	<p><b>registerCar()</b></p> <p>START</p> <p>    ADD car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status</p> <p>    STORE car_image, car_model, plate_num, rental_fare, cab_fare, car_color, reg_status</p> <p>    DISPLAY DriverDashboard page</p> <p>END</p>



**editCar()**

START

RETRIEVE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status

DISPLAY in EditCar page

END

**updateCar()**

START

READ car\_id

UPDATE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status into the database

RETURN DriverDashboard page

END

**deleteCar()**

START

READ car\_id

DELETE car\_image, car\_model, plate\_num, rental\_fare, cab\_fare,  
car\_color, reg\_status from database

RETURN DriverDashboard page

END

**registerRental()**

START

User click "Register for rental" button

UPDATE rental

	<p>END</p> <p><b>registerCab()</b></p> <p>START</p> <p style="padding-left: 40px;">User click “Register for carpool” button</p> <p style="padding-left: 40px;">UPDATE cab</p> <p>END</p>
--	--

### BookingController.php

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve, and display car booking related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	displayBooking()	This method is to display all the list of bookings for the current logged in user.
	displayRentalRequest()	This method is to display a list of available car rental or cab ride request.
	acceptCab()	This method is to allow driver to accept cab ride request.
	completeCab()	This method is to allow Drivers to complete their cab ride request
<b>Algorithm</b>	<p><b>displayBooking()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE rental, car_model, plate_num, pickup_location, dropoff_location</p> <p style="padding-left: 40px;">DISPLAY in myBooking page</p> <p>END</p>	

	<p><b>displayRentalRequest()</b></p> <p>START</p> <p>RETRIEVE renter_name, ic, drivers_license, start_date, end_date, passenger_name, pickup_location, dropoff_location, total_distance</p> <p>DISPLAY in carRequest page</p> <p>END</p> <p><b>acceptCab()</b></p> <p>START</p> <p>RETRIEVE renter_name, ic, drivers_license, start_date, end_date, passenger_name, pickup_location, dropoff_location, total_distance</p> <p>DISPLAY in carRequest page</p> <p>User clicks “Accept Request” button</p> <p>Redirect to requestDetails page</p> <p>END</p> <p><b>completeCab()</b></p> <p>START</p> <p>User click the “Ride completed” button</p> <p>UPDATE status into database</p> <p>DISPLAY carRequest page</p> <p>END</p>
--	--

**RatingController.php**

<b>Class Type</b>	Controller Class
<b>Responsibility</b>	This controller is used to retrieve, and display car rating related data.

Attributes	Attributes Name	Attributes Type
	-	-
Methods	Method Name	Description
	displayCabServiceDetails()	This method is to display cab ride details before rating
	rateCab()	This method is to save cab ride rating.
	displayRentalDetails()	This method is to display the car rental details before rating
	rateRental()	This method is to save car rental rating.
Algorithm	<p><b>displayCabServiceDetails()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE car_id, car_model, plate_num, pickup_location, dropoff_location</p> <p style="padding-left: 40px;">DISPLAY car_id, car_model, plate_num, pickup_location, dropoff_location</p> <p>END</p> <p><b>rateCab()</b></p> <p>START</p> <p style="padding-left: 40px;">ADD car_id, rating, feedback</p> <p style="padding-left: 40px;">STORE car_id, rating, feedback</p> <p style="padding-left: 40px;">RETURN myBooking page</p> <p>END</p> <p><b>displayRentalDetails()</b></p> <p>START</p>	

	<p>RETRIEVE car_id, car_model, plate_num, pickup_location, dropoff_location</p> <p>DISPLAY car_id, car_model, plate_num, pickup_location, dropoff_location</p> <p>END</p> <p><b>rateRental()</b></p> <p>START</p> <p>ADD car_id, rating, feedback</p> <p>STORE car_id, rating, feedback</p> <p>RETURN myBooking page</p> <p>END</p>
--	---

**CarRental.php**

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding car rental in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_rental_id	int
	car_id	int
	drivers_license	string
	ic	string
	rental_status	string
	user_id	string
	renter_name	string
	renter_email	string
	renter_phone_num	string

	start_date	dateTime
	end_date	dateTime
	rental_duration	int
	rental_amount	decimal
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	car()	The CarRental belongsTo Car
<b>Algorithm</b>	-	

### Cab.php

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding cab in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	cab_id	int
	car_id	int
	user_id	int
	passenger_name	string
	passenger_name	string
	passenger_phone_num	string
	pickup_location	string
	pickuplatlng	string
	dropoff_location	string
	dropofflatlng	string
	status	string
	total_distance	double
	total_cab_fare	double

Methods	Method Name	Description
	car()	The Cab belongsTo Cab
Algorithm	-	

### Car.php

Class Type	Model Class	
Responsibility	This model is used to store and retrieve data regarding car in the database.	
Attributes	Attributes Name	Attributes Type
	car_id	int
	drivers_license	string
	user_id	int
	car_color	string
	ic	string
	car_model	string
	status	string
	rental_fare	double
	cab_fare	double
	car_image	string
	plate_num	string
	reg_status	string
Methods	Method Name	Description
	cab()	The Car hasMany Cab
	carrental()	The Car hasMany CarRentals
	rating()	The Car hasMany Rating
	user()	The Car belongsTo User

<b>Algorithm</b>	-
------------------	---

**Rating.php**

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding rating in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	rating_id	int
	car_id	int
	user_id	int
	rating	string
	feedback	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	car()	The Rating belongsTo Car
<b>Algorithm</b>	-	

**User.php**

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding user in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	user_id	int
	name	string
	dob	string
	gender	string
	email	string
	phone_num	string



	usertype	string
	password	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	car()	The User hasMany Car
<b>Algorithm</b>	-	

### 2.1.4 Manage Users

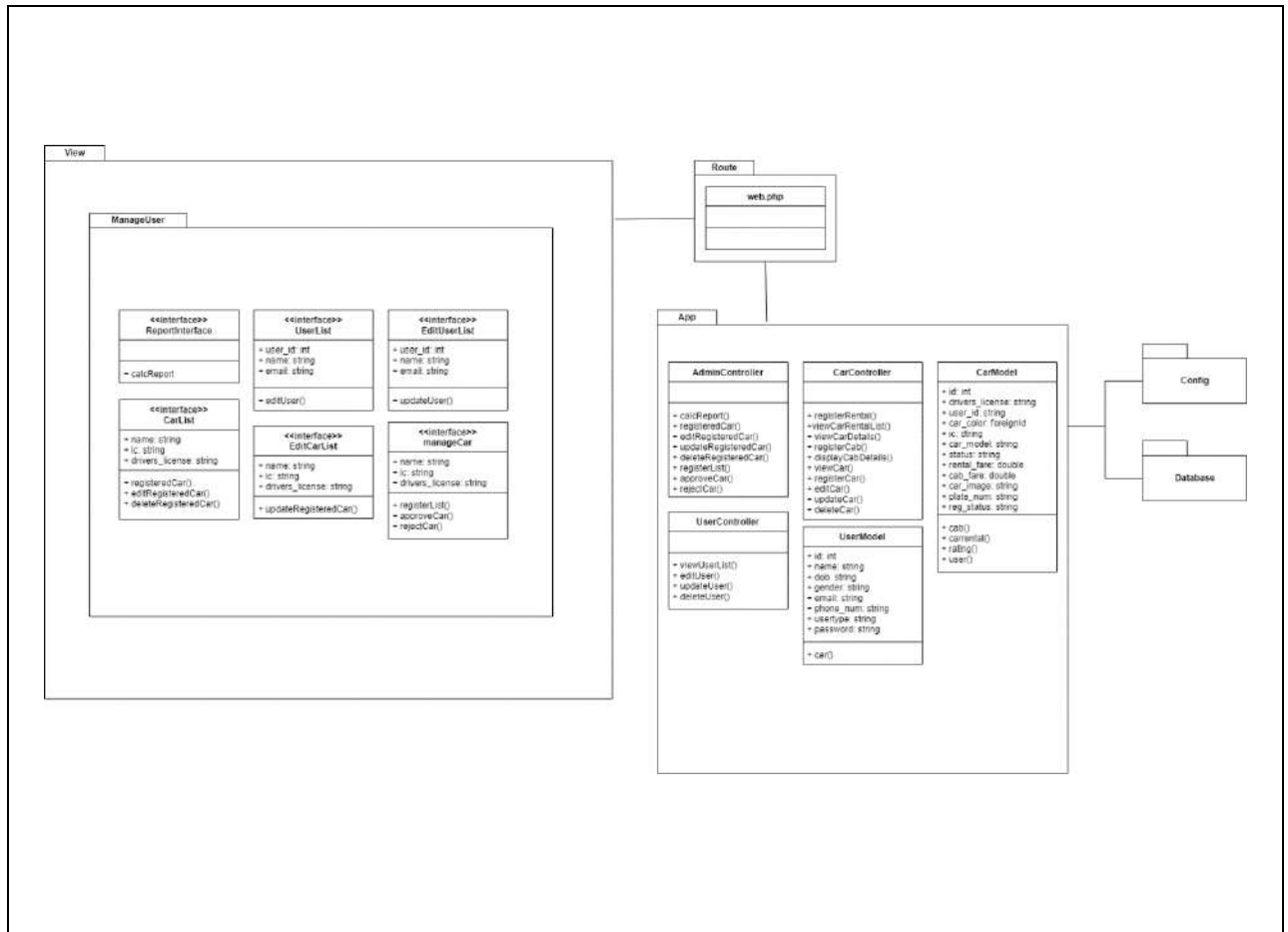


Figure 2.1.4 Manage Users Detailed Description

### ReportInterface.blade.php

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Admin to view the reports of the system such as the total number of users in the system and the total cars into the system.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>

	View User List	button
	View Cars	button
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	calcReport()	This method is to calculate the total number of users and cars registered in the system.
<b>Algorithm</b>	<b>calcReport()</b> START System displays “View User List” button System displays “View Cars” button  IF User click “View User List” button Redirect to UserList page ELSE Redirect to CarList page ENDIF END	

**UserList.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Admin to view a list of users registered into the system and manage them by modifying and deleting any user information.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	user_id	int
	name	string
	email	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>

	editUser()	This method is to edit existing users in the system.
<b>Algorithm</b>	<b>editUser()</b>  START  RETRIEVE user_id, name, email  DISPLAY user_id, name, email  END	

**EditUserList.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Admin to update the existing user details by providing appropriate information.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	user_id	int
	name	string
	email	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	updateUser()	This method is to update existing users in the system.
<b>Algorithm</b>	<b>updateUser()</b>  START  EDIT user_id, name, email  STORE user_id, name, email  END	

**CarList.blade.php**

<b>Class Type</b>	Boundary Class
-------------------	----------------

<b>Responsibility</b>	An interface that allows Admin to view a list of cars registered into the system and manage them by modifying and deleting any car information.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	name	string
	ic	string
	drivers_license	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	registeredCar()	This method is to display a list of registered cars inside the system.
	editRegisteredCar()	This method is to edit the existing details of the registered car.
	deleteRegisteredCar()	This method is to delete the registered cars.
<b>Algorithm</b>	<p><b>registeredCar()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE name, ic, drivers_license</p> <p style="padding-left: 40px;">DISPLAY name, ic, drivers_license</p> <p>END</p> <p><b>editRegisteredCar()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE name, ic, drivers_license</p> <p style="padding-left: 40px;">DISPLAY in EditCar page</p> <p>END</p> <p><b>deleteRegisteredCar()</b></p> <p>START</p> <p style="padding-left: 40px;">READ car_id</p>	

	DELETE name, ic, drivers_license from database  RETURN CarList page  END
--	--

**EditCarlist.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Admin to update the existing car details by providing appropriate information.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	name	string
	ic	string
	drivers_license	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	updateRegisteredCar()	This method is to update the registered car details into the database
<b>Algorithm</b>	<b>updateRegisteredCar()</b>  START  READ car_id  UPDATE name, ic, drivers_license into the database  RETURN CarList page  END	

**manageCar.blade.php**

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Admin to accept or reject car registration request.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>

	name	string
	ic	string
	drivers_license	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	registerList()	This method is to display a list of car registration request list.
	approveCar()	This method is for the Admin to approve car registration request.
	rejectCar()	This method is for the Admin to reject car registration request.
<b>Algorithm</b>	<b>registerList()</b>	
	START RETRIEVE name, ic, drivers_license DISPLAY name, ic, drivers_license END	
	<b>approveCar()</b>	
	START RETRIEVE name, ic, drivers_license DISPLAY name, ic, drivers_license Admin clicks “Approve” button STORE reg_status into database END	
	<b>rejectCar()</b>	
	START RETRIEVE name, ic, drivers_license	

	<p>DISPLAY name, ic, drivers_license</p> <p>Admin clicks “Reject” button</p> <p>STORE reg_status into database</p> <p>END</p>
--	---

**UserController.php**

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve and display user related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	viewUserList()	This method is to display a list of users registered inside the system.
	editUser()	This method is to edit existing users in the system.
	updateUser()	This method is to update existing users in the system.
	deleteUser()	This method is to delete users that are registered into the system.
<b>Algorithm</b>	<p><b>viewUserlist()</b></p> <p>START</p> <p style="padding-left: 40px;">RETRIEVE id, name, dob, gender, email, phone_num, usertype, password</p> <p style="padding-left: 40px;">DISPLAY id, name, dob, gender, email, phone_num, usertype, password</p> <p>END</p> <p><b>editUser()</b></p>	

	<pre> START     RETRIEVE user_id, name, email     DISPLAY user_id, name, email END  <b>updateUser()</b> START     EDIT user_id, name, email     STORE user_id, name, email END  <b>deleteUser()</b> START     READ user_id     DELETE id, name, dob, gender, email, phone_num, usertype, password from database     RETURN UserList page END </pre>
--	---

### CarRentalController.php

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve, and display car rentals related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	rentCarBooking()	This method is to save user's car rental booking data.



	highLow()	This method is to sort the car rental list in terms of rental_fare from highest to lowest.
	lowHigh()	This method is to sort the car rental list in terms of rental_fare from lowest to highest.
<b>Algorithm</b>	<p><b>rentCarBooking()</b></p> <p>START</p> <p>    ADD pickup_location, dropoff_location, total_distance</p> <p>    STORE pickup_location, dropoff_location, total_distance</p> <p>    RETURN myBooking page</p> <p>END</p> <p><b>highLow()</b></p> <p>START</p> <p>    RETRIEVE car_image, car_model, rental_fare</p> <p>    DISPLAY car_image, car_model, rental_fare with rental_fare from highest to lowest</p> <p>END</p> <p><b>lowHigh()</b></p> <p>START</p> <p>    RETRIEVE car_image, car_model, rental_fare</p> <p>    DISPLAY car_image, car_model, rental_fare with rental_fare from lowest to highest</p> <p>END</p>	

**CabController.php**

<b>Class Type</b>	Controller Class
<b>Responsibility</b>	This controller is used to retrieve and display cab services related data.

Attributes	Attributes Name	Attributes Type
	-	-
Methods	Method Name	Description
	cabBook()	This method is to allow users to request for a cab ride.
Algorithm	<b>cabBook()</b> START ADD pickup_location, dropoff_location, total_distance STORE pickup_location, dropoff_location, total_distance RETURN myBooking page END	

### User.php

Class Type	Model Class	
Responsibility	This model is used to store and retrieve data regarding user in the database.	
Attributes	Attributes Name	Attributes Type
	user_id	int
	name	string
	dob	string
	gender	string
	email	string
	phone_num	string
	usertype	string
	password	string
Methods	Method Name	Description
	car()	The User hasMany Car

<b>Algorithm</b>	-
------------------	---

**CarRental.php**

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding car rental in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_rental_id	int
	car_id	int
	drivers_license	string
	ic	string
	rental_status	string
	user_id	string
	renter_name	string
	renter_email	string
	renter_phone_num	string
	start_date	dateTime
	end_date	dateTime
	rental_duration	int
	rental_amount	decimal
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	car()	The CarRental belongsTo Car
<b>Algorithm</b>	-	

**Cab.php**

<b>Class Type</b>	Model Class
-------------------	-------------

<b>Responsibility</b>	This model is used to store and retrieve data regarding cab in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	cab_id	int
	car_id	int
	user_id	int
	passenger_name	string
	passenger_name	string
	passenger_phone_num	string
	pickup_location	string
	pickuplatlng	string
	dropoff_location	string
	dropofflatlng	string
	status	string
	total_distance	double
	total_cab_fare	double
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	car()	The Cab belongsTo Cab
<b>Algorithm</b>	-	

### 2.1.5 Manage Report

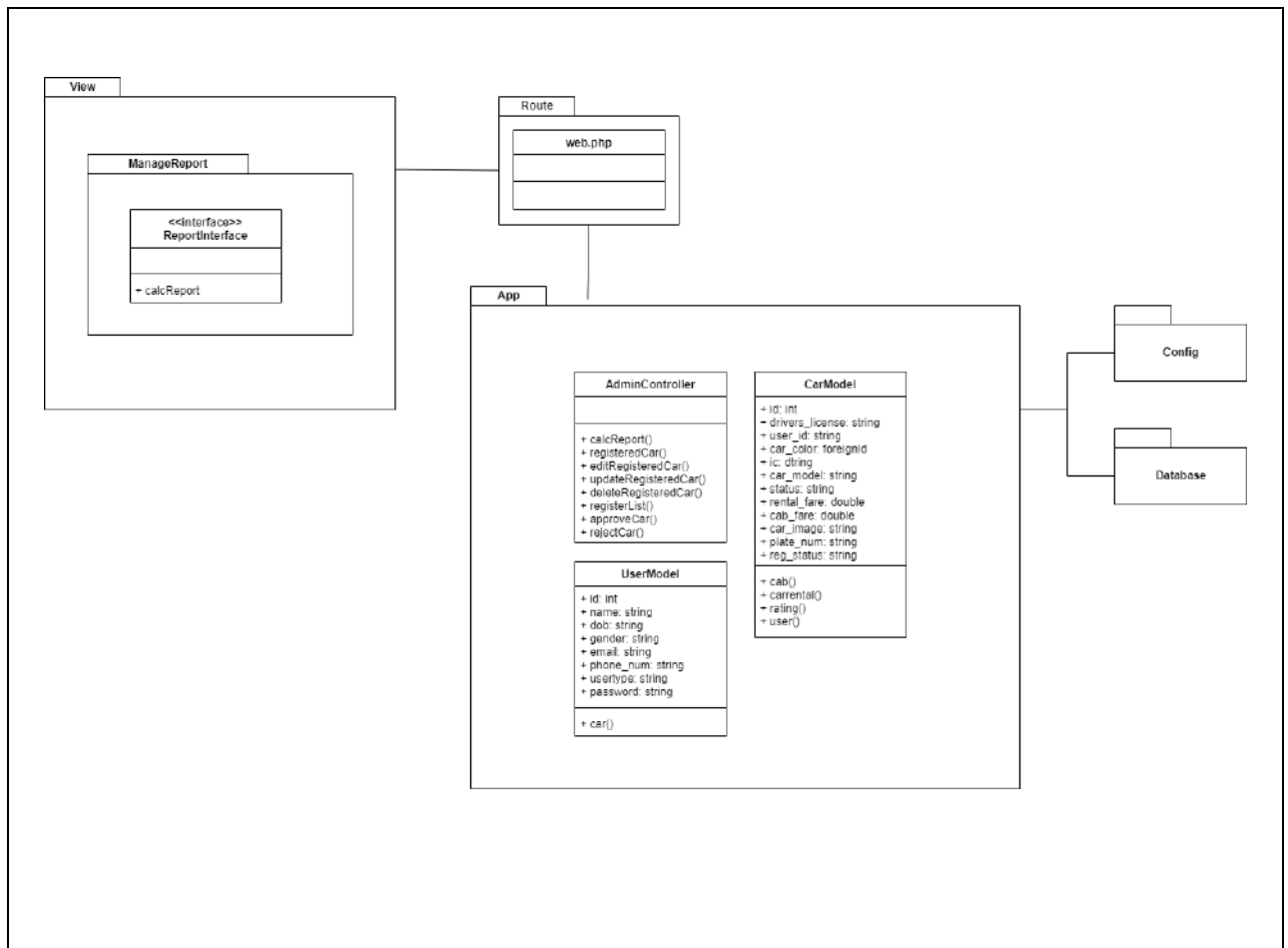


Figure 2.1.5 Manage Report Detailed Description

#### ReportInterface.blade.php

<b>Class Type</b>	Boundary Class	
<b>Responsibility</b>	An interface that allows Admin to view the reports of the system such as the total number of users in the system and the total cars into the system.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	View User List	button
	View Cars	button
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	calcReport()	This method is to calculate the total number of users and cars registered in the system.
<b>Algorithm</b>	<b>calcReport()</b>  START	

	<p>System displays “View User List” button</p> <p>System displays “View Cars” button</p> <p>IF User click “View User List” button</p> <p style="padding-left: 40px;">Redirect to UserList page</p> <p>ELSE</p> <p style="padding-left: 40px;">Redirect to CarList page</p> <p>ENDIF</p> <p>END</p>
--	---

### AdminController.php

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is used to retrieve and display Admin related data.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	-	-
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	calcReport()	This method is to calculate the total number of users and cars registered in the system.
	registeredCar()	This method is to display a list of registered cars inside the system.
	editRegisteredCar()	This method is to edit the existing details of the registered car.
	updateRegisteredCar()	This method is to update the registered car details into the database
	deleteRegisteredCar()	This method is to delete the registered cars.
	registerList()	This method is to display a list of car registration request list.

	approveCar()	This method is for the Admin to approve car registration request.
	rejectCar()	This method is for the Admin to reject car registration request.
<b>Algorithm</b>	<p><b>calcReport()</b></p> <p>START</p> <p>    System displays “View User List” button</p> <p>    System displays “View Cars” button</p> <p>    IF User click “View User List” button</p> <p>        Redirect to UserList page</p> <p>    ELSE</p> <p>        Redirect to CarList page</p> <p>    ENDIF</p> <p>END</p> <p><b>registeredCar()</b></p> <p>START</p> <p>    RETRIEVE name, ic, drivers_license</p> <p>    DISPLAY name, ic, drivers_license</p> <p>END</p> <p><b>editRegisteredCar()</b></p> <p>START</p> <p>    RETRIEVE name, ic, drivers_license</p> <p>    DISPLAY in EditCar page</p> <p>END</p>	

**updateRegisteredCar()**

START

READ car\_id

UPDATE name, ic, drivers\_license into the database

RETURN CarList page

END

**deleteRegisteredCar()**

START

READ car\_id

DELETE name, ic, drivers\_license from database

RETURN CarList page

END

**registerList()**

START

RETRIEVE name, ic, drivers\_license

DISPLAY name, ic, drivers\_license

END

**approveCar()**

START

RETRIEVE name, ic, drivers\_license

DISPLAY name, ic, drivers\_license

Admin clicks "Approve" button



	<p>STORE reg_status into database</p> <p>END</p> <p><b>rejectCar()</b></p> <p>START</p> <p>RETRIEVE name, ic, drivers_license</p> <p>DISPLAY name, ic, drivers_license</p> <p>Admin clicks “Reject” button</p> <p>STORE reg_status into database</p> <p>END</p>
--	---

### User.php

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding user in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	user_id	int
	name	string
	dob	string
	gender	string
	email	string
	phone_num	string
	usertype	string
	password	string
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	car()	The User hasMany Car

<b>Algorithm</b>	-
------------------	---

**Car.php**

<b>Class Type</b>	Model Class	
<b>Responsibility</b>	This model is used to store and retrieve data regarding car in the database.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	car_id	int
	drivers_license	string
	user_id	int
	car_color	string
	ic	string
	car_model	string
	status	string
	rental_fare	double
	cab_fare	double
	car_image	string
	plate_num	string
	reg_status	string
	<b>Methods</b>	<b>Method Name</b>
cab()		The Car hasMany Cab
carrental()		The Car hasMany CarRentals
rating()		The Car hasMany Rating
user()		The Car belongsTo User
<b>Algorithm</b>	-	

2.2 ERD



Figure 2.2.1 Entity Relationship Diagram (ERD)

## 2.3 DATA DICTIONARY

### User Table

Table 2.3.1 User Table Data Dictionary

Data Name	Description	Data Type	Constraint
id	ID for user	INT	PK
name	Users name	STRING	
dob	Users date of birth	STRING	
gender	Users gender	STRING	
email	Users email	STRING	
phone_num	Users phone number	STRING	
usertype	Type of user either 0 or 1 where:  0 => Regular user  1 => Administrator	STRING	
password	Users password	STRING	

### CarRental Table

Table 2.3.2 CarRental Table Data Dictionary

Data Name	Description	Data Type	Constraint
id	ID for car rental	INT	PK
car_id	ID for car	INT	FK
user_id	ID for user	INT	FK
drivers_license	Renter's drivers license	STRING	
ic	Renter's ic	STRING	
renter_name	Renter's name	STRING	

renter_email	Renter's email	STRING	
renter_phone_num	The renter's phone number	STRING	
rental_status	The status of the car rental either <b>"Ongoing"</b> or <b>"Completed"</b>	STRING	
rental_duration	The time duration between the start date and end date of the car rental	INT	
start_date	The start date for the car rental booking	DATETIME	
end_date	The end date for the car rental booking	DATETIME	
rental_amount	The total rental fare	DECIMAL	
created_at	The timestamps where the car rental booking is made	TIMESTAMPS	

## Cab Table

Table 2.3.3 Cab Table Data Dictionary

Data Name	Description	Data Type	Constraint
id	ID for cab	INT	PK
car_id	ID for car	INT	FK
user_id	ID for user	INT	FK
passenger_name	Passenger's name	STRING	
passenger_email	Passenger's email	STRING	
passenger_phone_num	Passenger's phone number	STRING	

pickup_location	Pick-up location for cab ride	STRING	
pickupalatlng	Latitude and longitude for pick-up location	STRING	
dropoff_location	Drop-off location for cab ride	STRING	
dropofflatlng	Latitude and longitude for drop-off location	STRING	
status	<p>Cab booking status which are <b>“Waiting”</b>, <b>“Accepted”</b> and <b>“Completed”</b>.</p> <p><b>Waiting</b> =&gt; When the passenger has made a cab ride request and is waiting for a driver to accept their request</p> <p><b>Accepted</b> =&gt; When a driver has accepted a passenger’s cab ride request</p> <p><b>Completed</b> =&gt; When the driver has completed the cab ride request</p>	STRING	
total_distance	The total distance between the pick-up and drop-off location in KM	DOUBLE	
total_cab_fare	The total fare for the cab ride	DOUBLE	

created_at	The timestamps when the cab booking is made.	TIMESTAMPS	
------------	--	------------	--

## Car Table

Table 2.3.4 Car Table Data Dictionary

Data Name	Description	Data Type	Constraint
id	ID for car	INT	PK
user_id	ID for user	INT	FK
drivers_license	User's (car owner) drivers license	STRING	
ic	User's (car owner) ic	STRING	
car_color	Car's color	STRING	
car_model	Car's model	STRING	
cab_fare	Car's fare per KM (if registered for carpool)	DOUBLE	
rental_fare	Car's fare per hour (if registered for rental)	DOUBLE	
rental	Car's rental registration status which can either be <i>NULL</i> or " <b>Car Rental</b> "	STRING	
cab	Car's carpool registration status which can either be <i>NULL</i> or " <b>Cab Service</b> "	STRING	
car_image	Car's image	STRING	
plate_num	Car's plate number	STRING	

reg_status	Car's registration status which are "Waiting for Approval", "Approved" and "Rejected"	STRING	
status	Car's status if they are booked for rental or not booked. The status are:  "Booked" => If the car is either booked for rental or carpool  "Vacant" => If the car is not booked for rental nor carpool	STRING	
created_at	The timestamps when the car is registered	TIMESTAMPS	

## Rating Table

Table 2.3.5 Rating Table Data Dictionary

Data Name	Description	Data Type	Constraint
id	ID for rating	INT	PK
car_id	ID for car	INT	FK
user_id	ID for user	INT	FK
rating	Star rating given to cars	STRING	
feedback	Feedbacks given to cars	LONGTEXT	
created_at	The timestamp when the rating and feedback is given	TIMESTAMPS	





## **APPENDIX C**

### **USER ACCEPTANCE TESTING (UAT)**

## **TABLE OF CONTENTS**

<b>1.0</b>	<b><i>TESTING REPORT</i></b>	<b>3</b>
1.1	Manage Car Rental Module	4
1.2	Manage Cab Service Module	5
1.3	Manage Car Review	7
1.4	Manage Users	8
1.5	Manage Report	9
<b>2.0</b>	<b><i>SYSTEM TESTING APPROVAL</i></b>	<b>10</b>

## **1.0 TESTING REPORT**

The purpose of this section is to outline the User Acceptance Testing (UAT) procedures for the application. Approval of this testing that reviewers are sure that following the executions of the test plan, the resulting system will be regarded thoroughly tested and appropriate for implementation

Kalaivani A/P Ramani, a UMP Student was chosen to go through the system. This form records any faults or difficulties discovered.

### 1.1 Manage Car Rental Module

No.	Module	Function	Input	Expected Output	Status		Comments
					PASS	FAILED	
1	Manage Car Rental	Allow Rentee to register car into the system [Register Car]	Car registration details and Register button	Please wait for your car to be Approved before making any car rental or cab service			-
		Allow Rentee to register car for rental [Register Car for Rental]	Register for rental button	Your car is registered for rental			-
		Allow Rentee to view car rental history [View Rental History]	History button	Car Rental History List			-
		Allow Rentee to edit registered car [Edit Car]	Car details and Update button	Car Details Successfully Updated			-

		Allow Rentee to delete registered car [Delete Car]	Delete button	Car Successfully Deleted			-
		Allow Renter to browse and book for car rental services [Rent Cars]	Renter details and Rental details	You have successfully booked this car!			-

### 1.2 Manage Cab Service Module

No.	Module	Function	Input	Expected Output	Status		Comments
					PASS	FAILED	
1	Manage Cab Service	Allow Driver to register car into the system [Register Car]	Car registration details and Register button	Please wait for your car to be Approved before making any car rental or cab service			-
		Allows Driver to register car for cab service [Register Car for Cab Service]	Register for carpool button	Your car is registered for cab service			-

		Allow Drivers to accept cab requests [Accept Cab Request]	Accept Request button	Do not forget to mark this ride as completed as soon as you have dropped off passenger at their destination.			-
		Allow Drivers to edit registered car [Edit Car]	Car details and Update button	Car Details Successfully Updated			-
		Allow Drivers to delete registered car [Delete Car]	Delete button	Car Successfully Deleted			-
		Allow Driver to view cab request history [View Cab History]	History button	Cab Request History List			PASS
		Allow Passenger to book cab services [Book Cab]	Destination details	Please wait for a driver to accept your request			

### 1.3 Manage Car Review

No.	Module	Function	Input	Expected Output	Status		Comments
					Yes	No	
1	Manage Car Review	Allow Rentee to leave rating and review on cars they have rent [Review Car Rental]	Car Review and Rating	Thank you for rating this car!			-
		Allow Passenger to leave rating and reviews on cabs they have ride [Review Cab Services]	Cab Review and Rating	Thank you for rating this car!			-



#### 1.4 Manage Users



No.	Module	Function	Input	Expected Output	Status		Comments
					PASS	FAILED	
1	Manage Users	Allow Admin to edit existing details of users registered into the system [Edit User List]	User details	User Details Updated Successfully			-
		Allow Admin to delete existing users from the system [Delete User]	Delete button	User Successfully Deleted			-
		Allow Admin to edit details of existing cars registered into the system [Edit Car]	Car details	Car Details Updated Successfully			-
		Allow Admin to delete existing cars from the system [Delete Car]	Delete button	Car Successfully Deleted			-
		Allow Admin to approve car	Approve button	Car Registration Approved!			-

		registration request [Manage Car]					
		Allow Admin to reject car registration [Reject Car]	Reject button	Car Registration Rejected			-

### 1.5 Manage Report

No.	Module	Function	Input	Expected Output	Status		Comments
					PASS	FAILED	
1	Manage Report	Allow Admin to view report of the system [View Report]	Report button	Total number of users, Total number of cars registered			There was an error when trying to display the total number of users and the total number of cars registered


## 2.0 SYSTEM TESTING APPROVAL

GENERAL INFORMATION		
<b>PROJECT NAME</b>	WEB-BASED CAR RENTAL AND CAB SERVICE SYSTEM FOR UMP STUDENTS	
<b>TIME TAKEN</b>	9 months	
<b>CLIENT</b>	KALAIVANI A/P RAMANI	
<b>APPLICATION NAME</b>	UMPCAB	
	<b>NAME</b>	<b>DATE</b>
Verified by: 	NUR HASYA BINTI MOHD NORDIN	23/12/2022
Developer		
Approved by: 	KALAIVANI A/P RAMANI	23/12/2022
Client		



## APPENDIX D

### USABILITY TESTING (GOOGLE FORM QUESTIONNAIRE)





UMPCAB  
CAR RENTAL MADE SIMPLE

### Web-Based Car Rental and Cab Service system for UMP Students (UMPCab)

Hello, I am Nur Hasya, a final year software engineering student from Faculty of Computing in Universiti Malaysia Pahang. Here is my final year project entitled the Web-Based Car Rental and Cab Service System for UMP Students (UMPCab).

The Car Rental and Cab Service System for UMP Students (UMPCab) is a web-based system developed using Laravel framework. The main purpose for the development of this project is to ease UMP Students in looking for available cars for rental around college. Students can also look for available cab services that are offered around the campus. The stakeholders involved in this system are UMP students and the administrator of the system.

Your response is highly appreciated. Have a good day!

 [nurhasyamohdnordin@gmail.com](mailto:nurhasyamohdnordin@gmail.com) (not shared)   
[Switch accounts](#)

\*Required

What is your gender? \*

Male  
 Female

What year are you in? \*

- First Year
- Second Year
- Third Year
- Final Year

Next

Clear form



CAR RENTAL MADE SIMPLE

## Web-Based Car Rental and Cab Service system for UMP Students (UMPCab)

 nurhasyamohdnordin@gmail.com (not shared)  
[Switch accounts](#)



\*Required

### SYSTEM SURVEY QUESTION

Based on your experience using UMPCab, please answer these set of questionnaires. Thank you for your time.

1 - I think that I would like to use this system \*

1      2      3      4      5  
Strongly Disagree                  Strongly Agree

2 - I found the system unnecessarily complex \*

1      2      3      4      5  
Strongly Disagree                  Strongly Agree

3 - I thought the system was easy to use \*

	1	2	3	4	5	
Strongly Diagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

4 - I think that I would need the support of a technical person to be able to use this system \*

	1	2	3	4	5	
Strongly Diagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

5 - I found the various functions in the system were well integrated \*

	1	2	3	4	5	
Strongly Diagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

6 - I thought there was too much inconsistency in this system \*

	1	2	3	4	5	
Strongly Diagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree



7 - I would imagine that most people would learn to use this system very quickly \*

	1	2	3	4	5	
Strongly Diagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

8 - I found the system very inconvenient to use

	1	2	3	4	5	
Strongly Diagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

9 - I felt very confident using the system \*

	1	2	3	4	5	
Strongly Diagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

10 - I needed to learn a lot of things before I could get going with this system \*

	1	2	3	4	5	
Strongly Diagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Back

Next

Clear form

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#).

Google Forms

**APPENDIX E**  
**SUS SCORE CALCULATION**

## **TABLE OF CONTENTS**

<b>Step 1</b>	<b>3</b>
<b>Step 2</b>	<b>4</b>
<b>Step 3</b>	<b>5</b>
<b>Step 4</b>	<b>5</b>

### Step 1

Convert user ratings from the 10 questions into points.

- For odd numbered questions (Q1, Q3, Q5, Q7, Q9)
  - o  $\text{Points} = \text{Rating} - 1$
- For even numbered questions (Q2, Q4, Q6, Q8, Q10)
  - o  $\text{Points} = 5 - \text{Rating}$
- The table below shows the ratings given by 20 respondents before the points are calculated:

Respondents	Questions										Total Points
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	
1	5	1	5	1	5	1	5	1	5	1	
2	5	5	5	5	5	5	5	5	5	5	
3	5	1	5	1	5	1	5	2	5	1	
4	5	2	4	2	4	2	5	5	5	2	
5	4	3	3	3	3	3	3	4	3	3	
6	5	3	5	2	4	3	4	1	4	1	
7	5	2	4	2	5	1	5	2	5	2	
8	5	1	5	1	5	1	5	1	5	1	
9	4	3	4	2	5	1	4	2	4	3	
10	4	1	4	2	4	1	4	1	4	3	
11	5	1	5	1	5	1	5	1	5	1	
12	4	3	4	3	4	2	5	2	4	3	
13	5	2	4	2	5	1	4	2	4	2	
14	4	2	4	1	5	1	4	1	4	2	
15	5	1	5	1	5	1	5	1	5	1	
16	5	3	4	2	5	1	4	1	5	1	
17	5	2	4	2	5	1	4	1	4	2	
18	5	1	3	1	5	1	5	1	5	3	
19	4	2	4	2	5	1	4	1	4	3	
20	5	2	5	1	4	1	4	1	5	1	

## Step 2

Total up the points for each respondents:

- The table below shows the ratings after the points are calculated:

Respondents	Questions										Total Points
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	
1	4	4	4	4	4	4	4	4	4	4	40
2	4	0	4	0	4	0	4	0	4	0	20
3	4	2	4	4	4	4	4	3	4	4	37
4	4	3	3	3	3	3	4	0	4	3	30
5	3	2	2	2	2	2	2	1	2	2	20
6	4	2	4	3	3	2	3	4	3	4	32
7	4	3	3	3	4	4	4	3	4	3	35
8	4	4	4	4	4	4	4	4	4	4	40
9	3	2	3	3	4	4	3	3	3	2	30
10	3	4	3	3	3	4	3	4	3	2	32
11	4	4	4	4	4	4	4	4	4	4	40
12	3	2	3	2	3	3	4	3	3	2	28
13	4	3	3	3	4	4	3	3	3	3	33
14	3	3	3	4	4	4	3	4	3	3	34
15	4	4	4	4	4	4	4	4	4	4	40
16	4	2	3	3	4	4	3	4	4	4	35
17	4	3	3	3	4	4	3	4	3	3	34
18	4	4	2	4	4	4	4	4	4	2	36
19	3	3	3	3	4	4	3	4	3	2	32
20	4	3	4	4	3	4	3	4	4	4	37

### Step 3

Multiply the respondent's total points with 2.5 to get an individual user's score

Respondents	Questions										Total Points	User's score
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10		
1	4	4	4	4	4	4	4	4	4	4	40	100
2	4	0	4	0	4	0	4	0	4	0	20	50
3	4	2	4	4	4	4	4	3	4	4	37	92.5
4	4	3	3	3	3	3	4	0	4	3	30	75
5	3	2	2	2	2	2	2	1	2	2	20	50
6	4	2	4	3	3	2	3	4	3	4	32	80
7	4	3	3	3	4	4	4	3	4	3	35	87.5
8	4	4	4	4	4	4	4	4	4	4	40	100
9	3	2	3	3	4	4	3	3	3	2	30	75
10	3	4	3	3	3	4	3	4	3	2	32	80
11	4	4	4	4	4	4	4	4	4	4	40	100
12	3	2	3	2	3	3	4	3	3	2	28	70
13	4	3	3	3	4	4	3	3	3	3	33	82.5
14	3	3	3	4	4	4	3	4	3	3	34	85
15	4	4	4	4	4	4	4	4	4	4	40	100
16	4	2	3	3	4	4	3	4	4	4	35	87.5
17	4	3	3	3	4	4	3	4	3	3	34	85
18	4	4	2	4	4	4	4	4	4	2	36	90
19	3	3	3	3	4	4	3	4	3	2	32	80
20	4	3	4	4	3	4	3	4	4	4	37	92.5

### Step 4

Calculate the SUS score using the formula below:

$$SUS\ Score = \frac{Total\ User\ Scores}{(Total\ Number\ of\ Users)}$$

$$SUS\ Score = \frac{1662.5}{20} = 83.13$$