Technical Solution Design
for
Get Your Car

Version: V2.0
Date: 29/10/2020
Sponsor: RMIT University
Author: Vincent Pranata, Jerald Tienzo, Youxin Zheng, and Yanfang He

Document Control

Distribution

Version	Issued	Recipient	Position
V1.0	19/10/2020	Shaahin Madani	Supervisor
V1.0	19/10/2020	Shaahin Madani	Client
V2.0	29/10/2020	Shaahin Madani	Supervisor
V2.0	29/10/2020	Shaahin Madani	Client

Amendment History

Section	Page(s)	Version	Comment
Section 2 - 3	6, 7, 9	V1.0	Completed Technical environment, overall architecture
Section 9	22	V1.1	Complete issue and risk section
Section 4	8 , 18	V1.2	Update system architected
Section 1, 4 – 5, 7	9	V1.3	Update and completed Functionalities, Non-functionalities section, database architecture
Section 6 - 8	20 , 21	V1.4	Update Implementation instruction and test result
Section 1 – 10	1 – 21	V2.0	Reviewed all sections with minor updates for submission.

.

Staff or Entities Consulted

NAME	Position / Organization
Shaahin Madani	Supervisor/ RMIT University
Shaahin Madani	Client

Related Documents

Name	Author	Description
Test documentation	Team uhhh	Test plan for unit test and test case result
Test case	Team uhhh	Document holding details of the test cases

Preface

The purpose of this document is to outline the Technical side of the Get Your Car project. It serves as guide on how to implement the system and how the system actually works. It also outlines the project's database scheme and the non-functional requirements that are implemented. The known risks and issues are also discussed in detail below.

Table of Contents

1 Introduction

This project aims to build a car share scheme website that provides a platform for users to rent cars. This project will be used by a company that owns several cars in a variety of different locations that are available to customers. The sponsor is offering this project to a company so they can provide a convenient way for their customers to use their services.

The final product will be a website that users can use to book cars for rent and return them once the booking has finished. Customers will be able to register and log themselves into the website in order to make bookings. Customers will also be able to view the details of their bookings, cars, and pricing plans. Searching for cars and their locations will also be available to users on the website. The website and product are called "GET YOUR CAR".

The technical environment we are using involve GitHub, Trello, MS Teams, Python, Flask web-frame, HTML, CSS, JavaScript, MySQL, and Google Cloud Platform (GCP).

The level of complexity: We will fulfil the customer requirement and implement all functions they need.

The benefit of the technical solution is helping the development team to develop the website. It also helps team members to understand which tools they use to cooperate with others in the team.

2 Technical Environment

For this project, the technical environments/technologies we decided to use are GitHub, Trello, MS Teams, Python, Flask web-frame, HTML, CSS, JavaScript, MySQL, and Google Cloud Platform (GCP).

GitHub is chosen for our source control technology mainly because all members already have a GitHub account and are comfortable with it. GitHub is a very popular technology that we very widely used amongst programmers. It offers strong services in relation to source code management and distribution version control. There are all aspects that will be important for us to work effectively and efficiently as a team.

Trello is our task management board due to the specification of this project that required us to use it. Trello excels at providing a virtual management board that is used for agile practices. We are specifically following the SCRUM agile methodology and using Trello allows us to organise our project into virtual boards. The boards are used to organise the tasks that are assigned to each member for the sprint. This allows for a visual representation for how the team is progressing with each task for the sprint. There are many features in Trello that allow for board customization such as assigning specific members to tasks, adding due dates, adding checklist to tasks, and adding comments/notes to a task. Board customization is Trello’s biggest strength and is a major reason why we are using it.

MS Teams is no different, it is also required for us to use MS Teams as our mean of daily communication and documentation management, due to specification of the project. MS Teams is a communication platform that allows for teams to organise their communicate and files. Team chats can be created with different rooms available to also be created in order to organise different discussion topics. Files can be uploaded to the team chat and organised in folders. There are also built-in Microsoft applications to make use of such as word and excel. While using these applications, multiple users can be on the same file at the same time, being at to work on the same document concurrently. File management and being able to work on the same document with multiple users emphasises the aspect of working effectively and efficiently as a team. These functions are important to us as a team that is why we are using MS teams as our main communication platform.

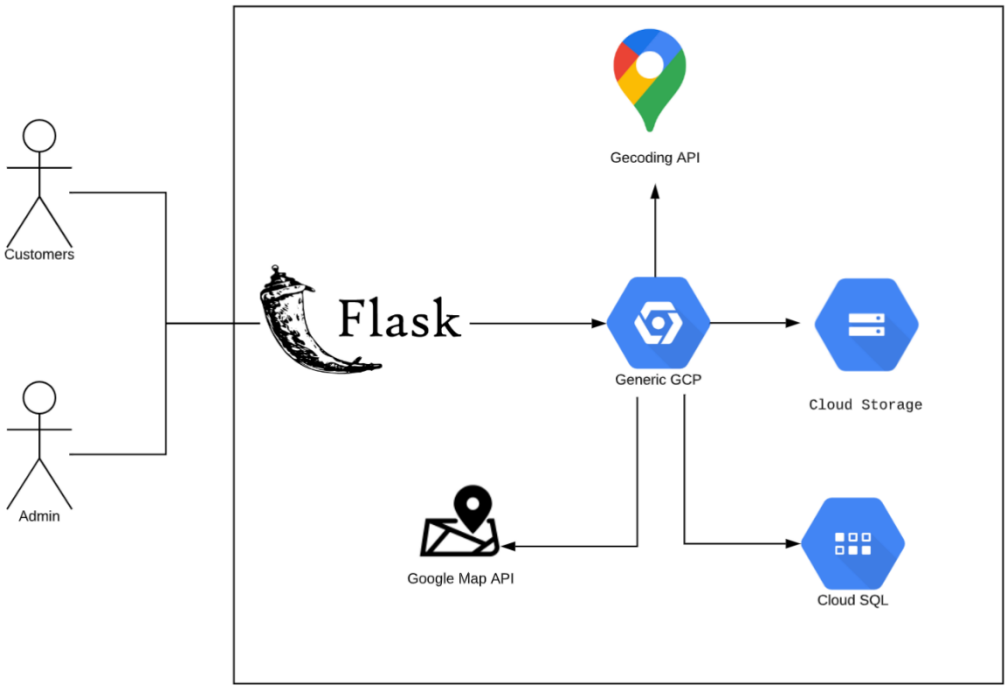
Python and Flask web-frame are chosen to be our project main language and web application frame due to its simplicity and range of library options. Python can be used to easily implement functionalities and APIs that we most likely going to be using such as encryption and multiple APIs from GCP like Cloud SQL, Cloud Storage, Google Map API, and Geocoding API. While the reason we chose Flask even though it is a relatively newer technology, because it has learned from its predecessors such as Django and Pyramid and implements extensions and other plugins useful for projects. Flask is also more suitable to be used in projects that are in the smaller scale like ours.

HTML, CSS, JavaScript are also used in order to make the pages, style, and make pop-up notifications that shows both error and successful messages. We also used some bootstrap-css that are available online that gave us easy to use layouts for our projects.

GCP is chosen as our deployment mainly because it is a free to use deployment platform for new users and have multiple useful features that are very useful for our project. Cloud SQL is a GCP function that is an online database server that uses MySQL which is used by us to store the data about customers, addresses, licenses, cars, bookings, and admins inside a database. We chose Cloud SQL because is also easy to be implemented with python when being deployed compared to the other database server options like the BigQuery. Cloud Storage is another GCP function and the only option that we have to store the dummy.sql file which consist of SQL queries to insert multiple dummy data to the Cloud SQL database. We use Cloud Storage because Cloud SQL can make a connection to the Cloud Storage to run the dummy.sql file easily from its UI. Next is the Google Map API which is used to show markers to locate each car location. While Geocoding is used to find the customer address location based on the inputted credentials to be stored so that user can easily find cars near their registered address. We chose these functions for our project since we have chosen GCP as our deployment platform and GCP provided these plugins to be used.

3 Overall Architecture

We have two types of users, customers and admins, which both interact with the program. This interaction is handled by the Flash web-frame where it will then establish a connection with the GCP project and will create more detailed connections with the GCP functionalities that are used for a project such as Cloud SQL, Cloud Storage, Google Maps API, and Geocoding API.



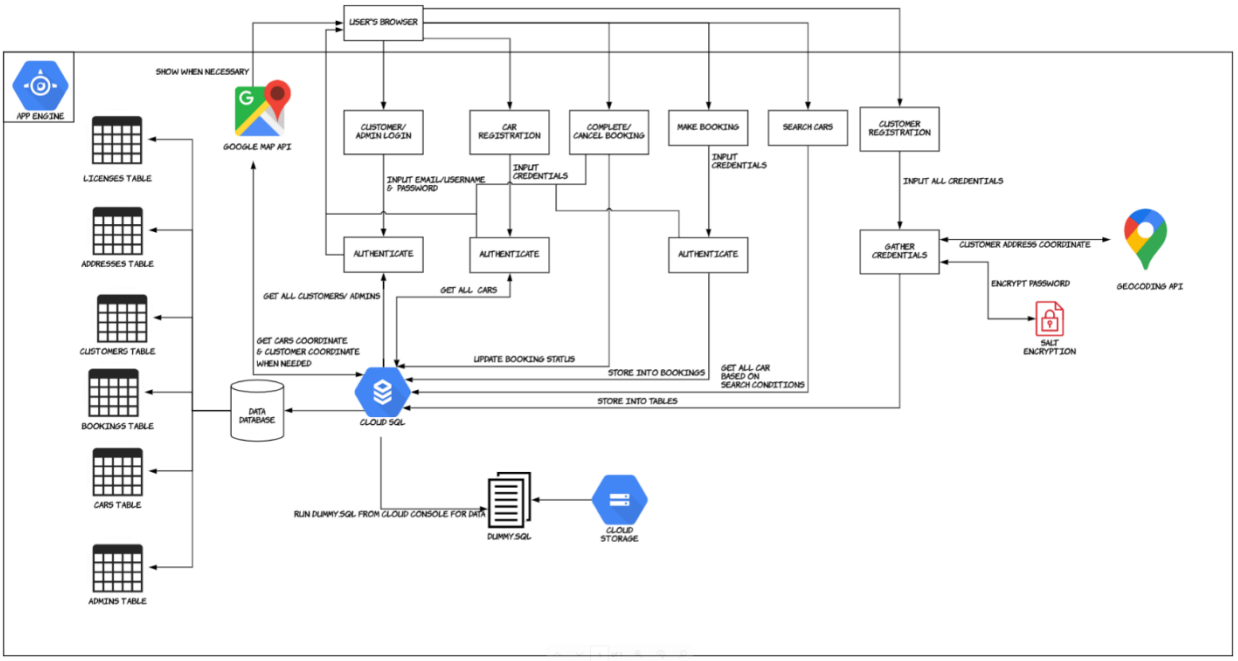
4 System Architecture

Our system is built to run inside the App Engine provided by Google Cloud Project. After it is deployed and run by a user, the user will interact with the UI and the UI will accept the customer input and send it accordingly to the backend to be processed.

The system also only has a single database called “Data” where it contains all 6 of our tables (Customers, Admins, Licenses, Addresses, Cars, and Bookings) which is in the Cloud SQL. The Cloud SQL is also able to establish a connection with the Cloud Storage to import a file called “dummy.sql” to insert dummy data into the database.

The system also establishes a connection with Geocoding and Google Maps API in order to get the coordinates of the customer addresses and show the car locations in the map.

The SALT encryption method is used by our system to encrypt the customer password to ensure confidentiality. The tables will also be accessed accordingly to what the user decided to do, for example when a user decided to login as admin, the system will take the data stored inside the Admins table, while if the user decided to login as the customer then the system will establish a connection with the Customers table. Below is the detailed architectural diagram that we made for our system.



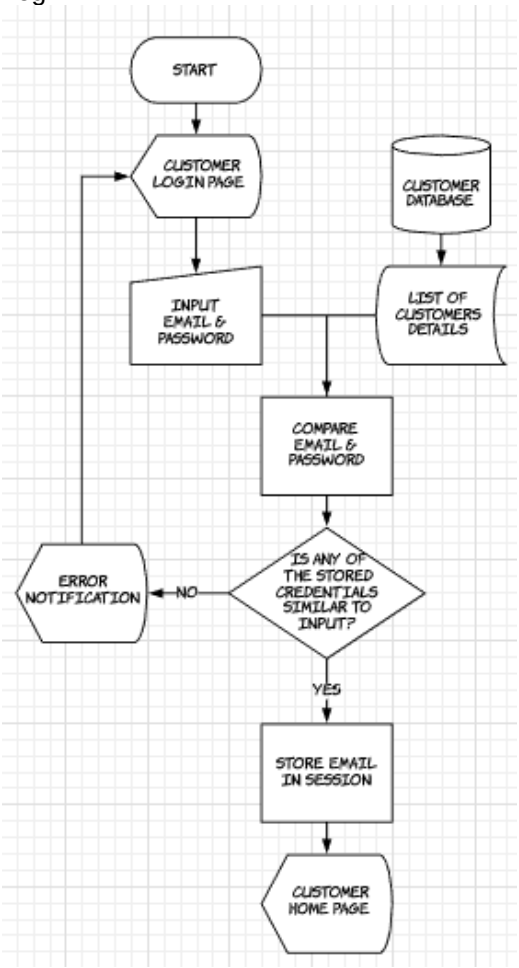
4.1 FUNCTIONALITIES/FEATURES

Feature Name	Description
Customer Registration	Customer need provide some information to prove their identity to the system such as email address, contact number and the driver license
Customer Login	Customer can use username and password to access account information such as rental history, make bookings, cancel bookings, etc
Customer Password Reset	Customer can provide correct email, date of birth and also both first name and last name to reset password
Customer Logout	Customer can click logout button to exit account.
Find Specific Cars for Customer	Customer search specific cars based on the categories provided and input the keyword into a textbox.
Choose a Plan for Customer	Customer can choose between two price plans to make their booking (Regular and Premium)
Make a Booking for Customer	Customer are able to make a booking of the available car shown after they have filled in a valid booking time
Cancel a Booking for Customer	Customer are able to cancel any ongoing booking they have
View Personal Booking History for Customer	Customer can access their bookings history by clicking the Booking History button in the navbar.
View Personal Ongoing Booking for Customer	Customer are able to access their list ongoing bookings by clicking the Ongoing Booking button in the navbar
Admin Login	Admin can use username and password to modify the customer info, register new car, view user and car list and change the information about car.
View Registered User for Admin	User can view the registered users after its login as admin
View Registered Car for Admin	User can view the registered car after its login as admin

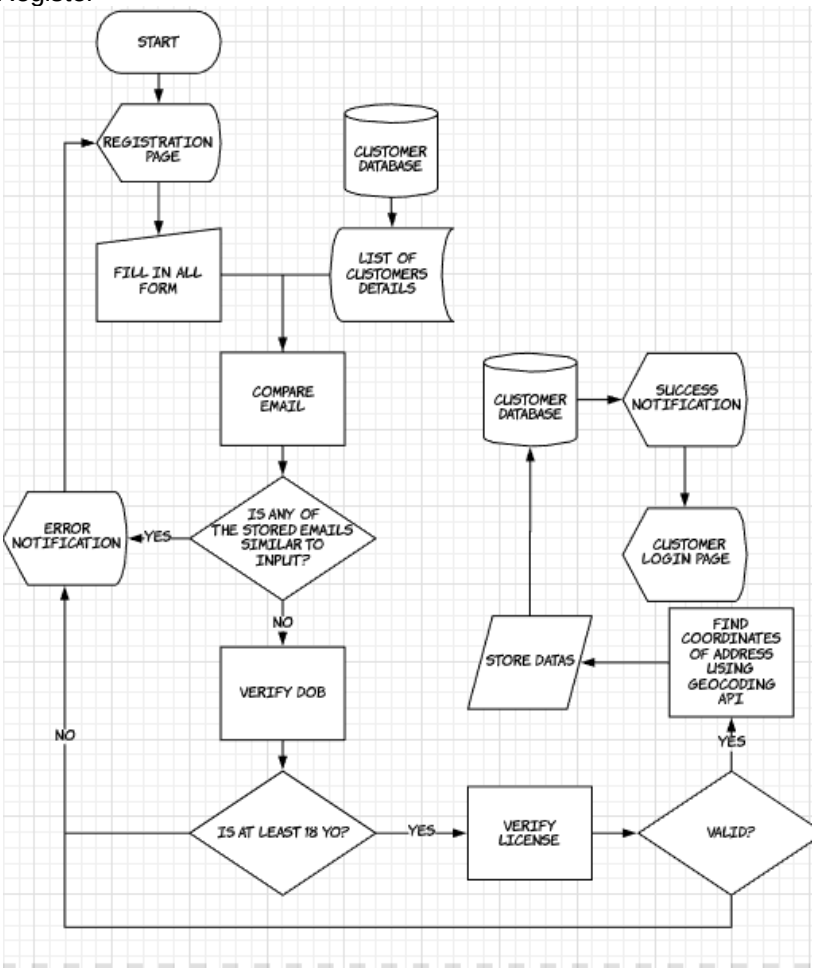
Car Registration for Admin	Admin are able to register a new car to be rented to the customer after the filled in the details and the car a valid license plate
Card Details for Payment for Customer	Customer can input correct bank car detail format to finish the payment
Cars Near Me	Once customer click a button to search cars near their registered address, it will show cars that are at least in 5 km range from the registered address.
Complete Booking for Customer	Customer are able to complete any ongoing booking they have

4.1.1 Customer Login and Registration System

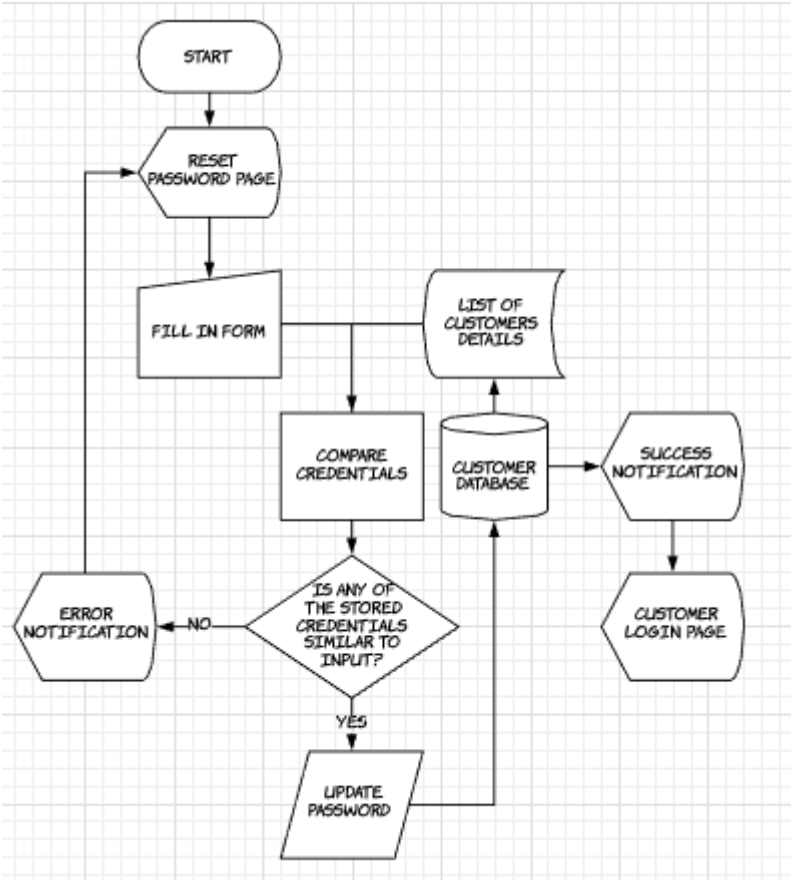
Login



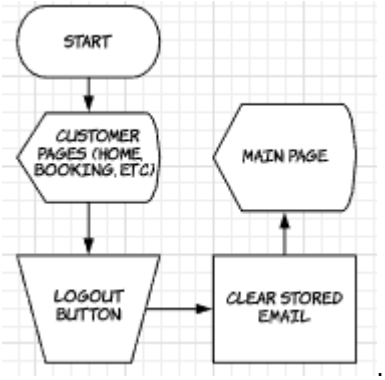
Register


START
REGISTRATION PAGE
CUSTOMER DATABASE
FILL IN ALL FORM
LIST OF CUSTOMERS DETAILS
COMPARE EMAIL
CUSTOMER DATABASE
SUCCESS NOTIFICATION
ERROR NOTIFICATION
YES
IS ANY OF THE STORED EMAILS SIMILAR TO INPUT?
CUSTOMER LOGIN PAGE
NO
STORE DATAS
FIND COORDINATES OF ADDRESS USING GEOCODING API
VERIFY DOB
NO
YES
IS AT LEAST 18 YO?
YES
VERIFY LICENSE
VALID?

Reset Password

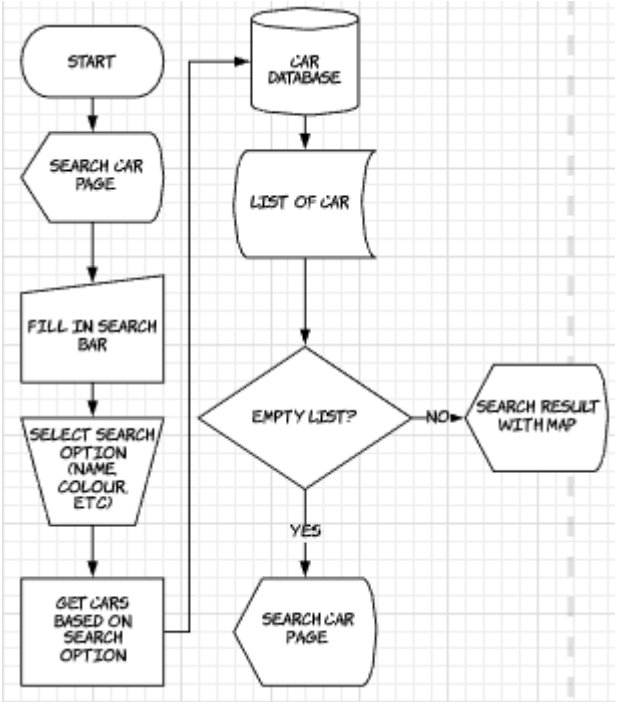


4.1.2 Customer Logout System

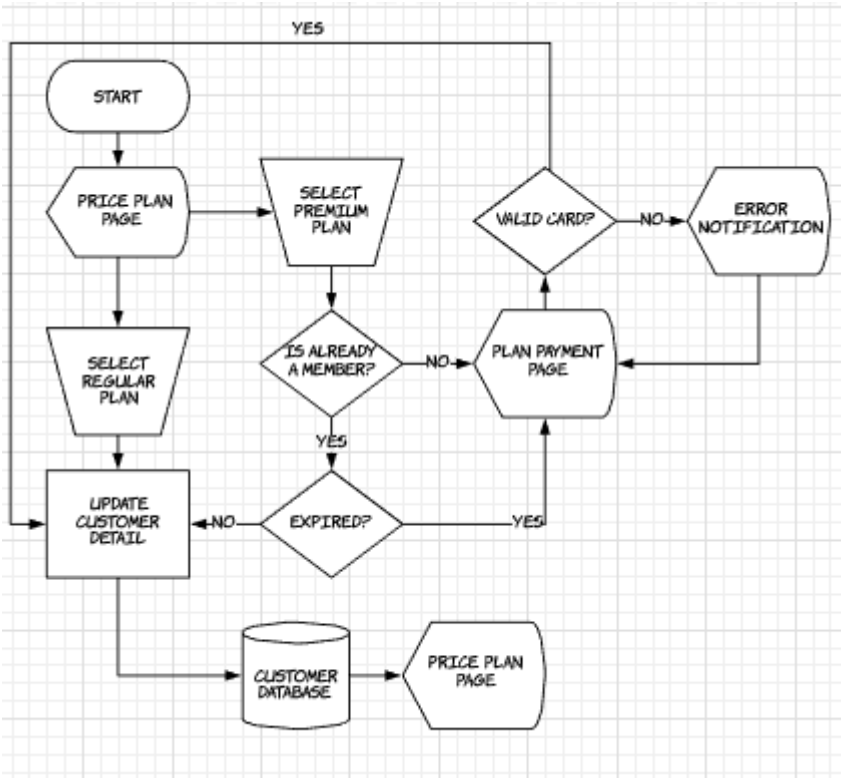


.

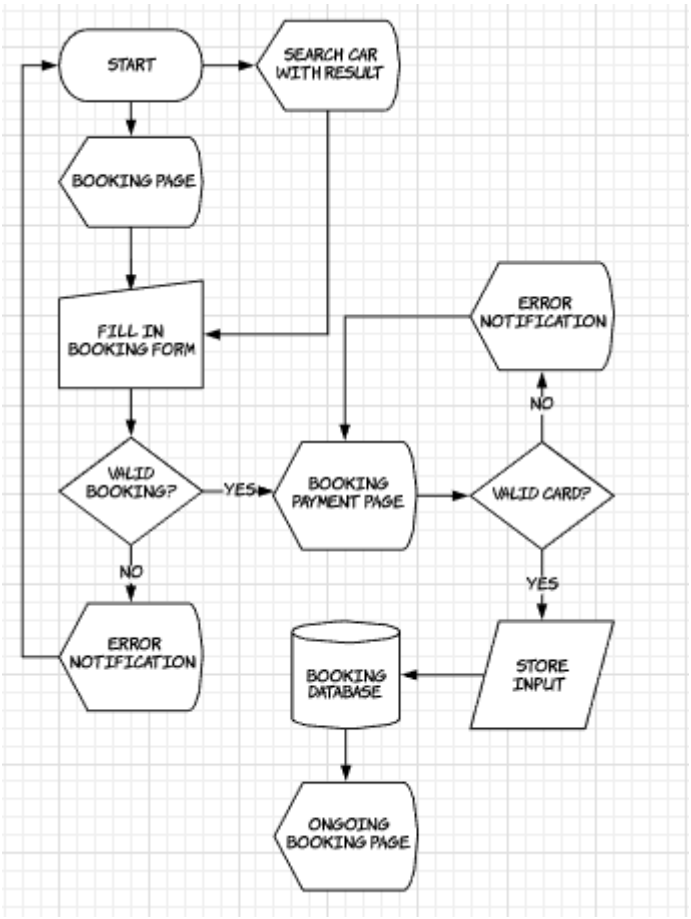
4.1.3 Find Specifics Cars



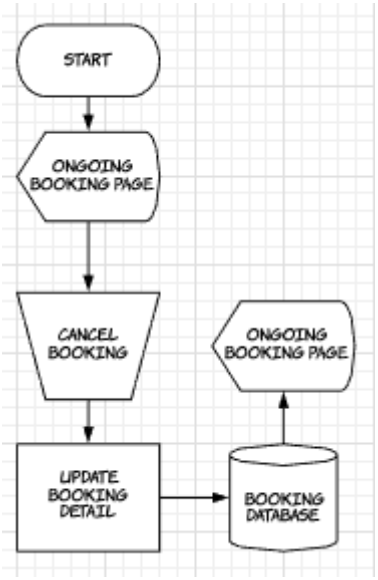
4.1.4 Choose Plan



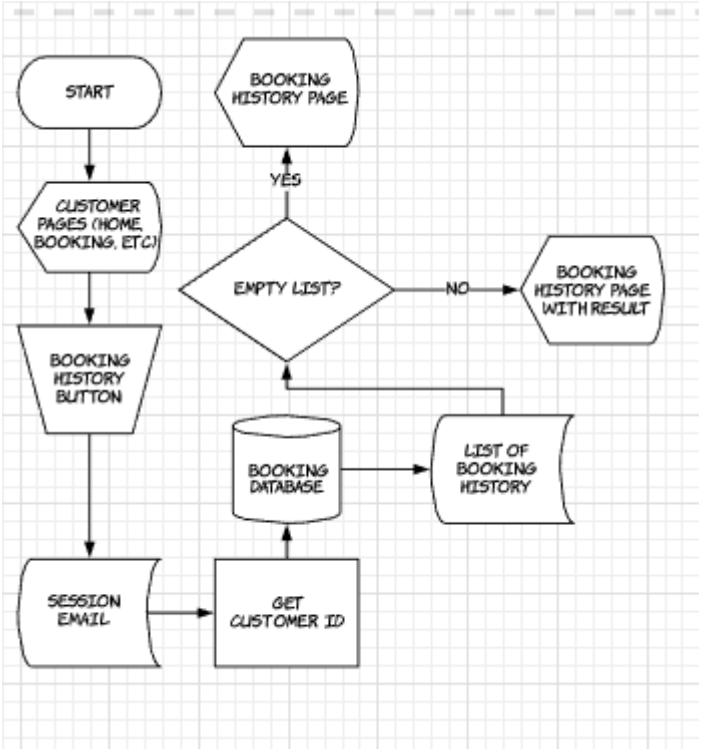
4.1.5 Make a Booking



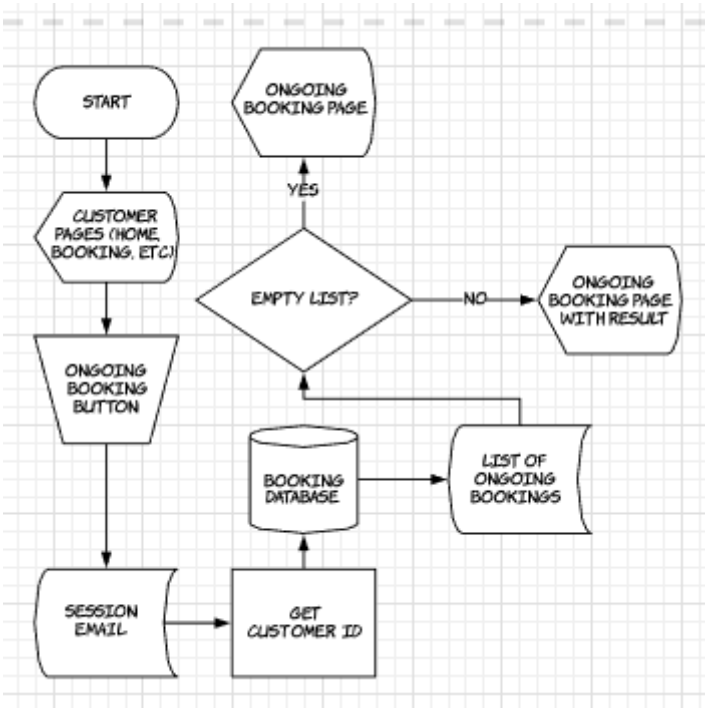
4.1.6 Cancel a Booking



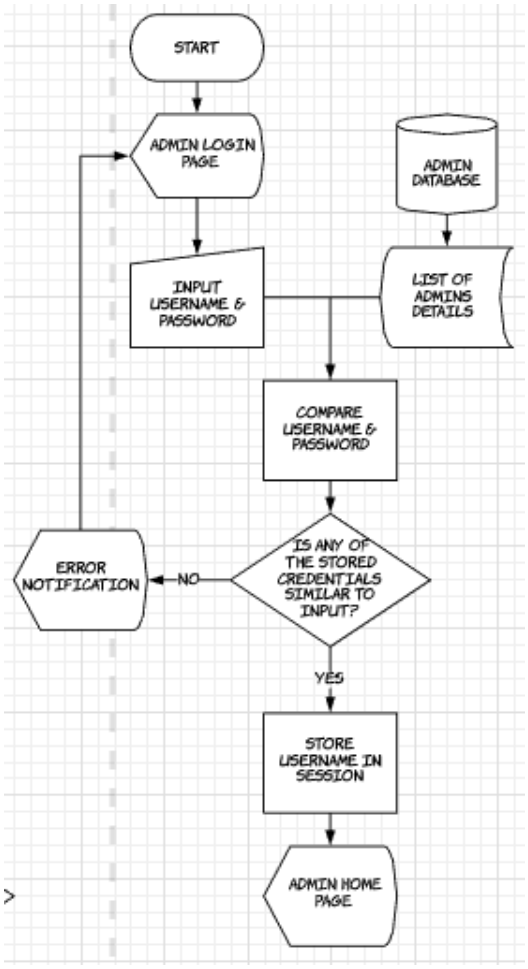
4.1.7 View Personal Booking History



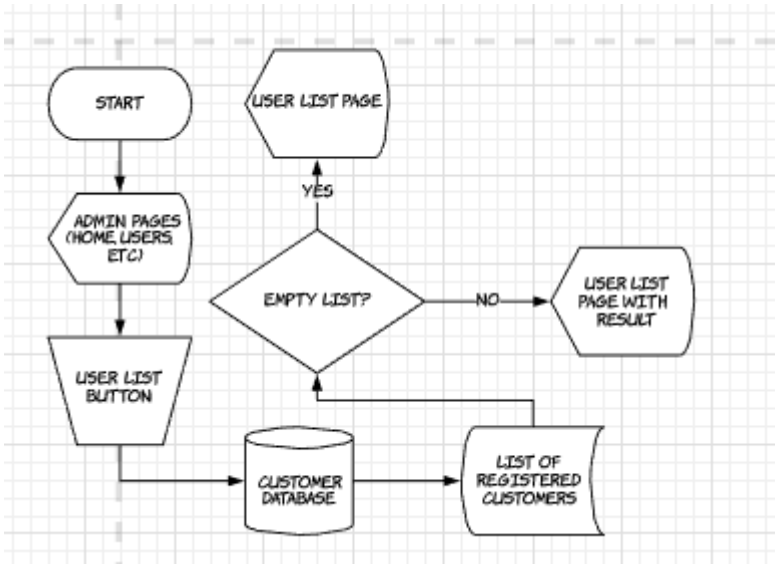
4.1.8 View Personal Ongoing Booking



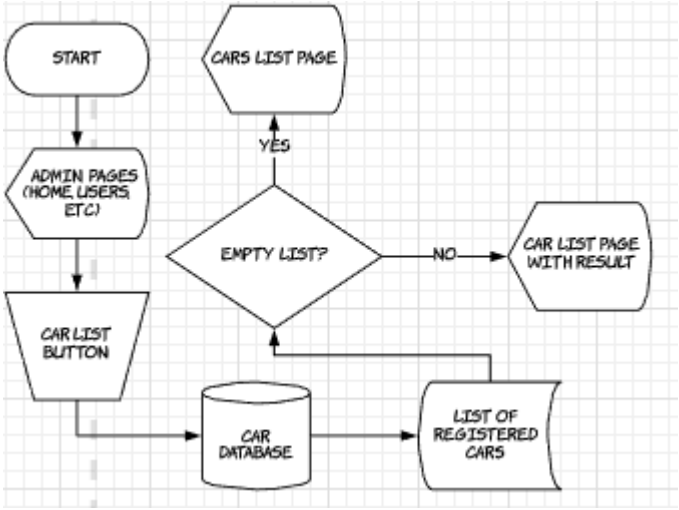
4.1.9 Admin Login System



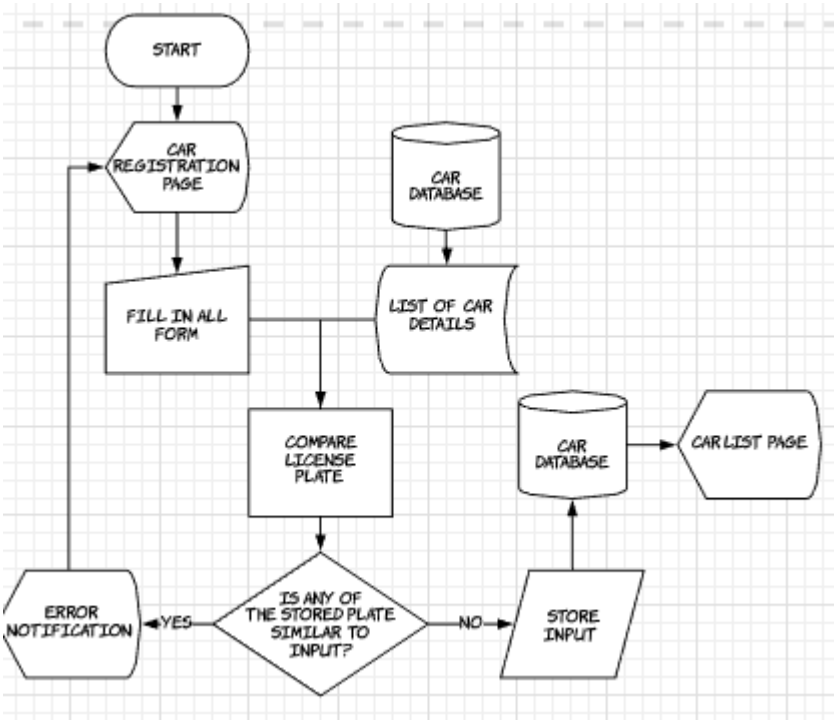
4.1.10 View Registered User



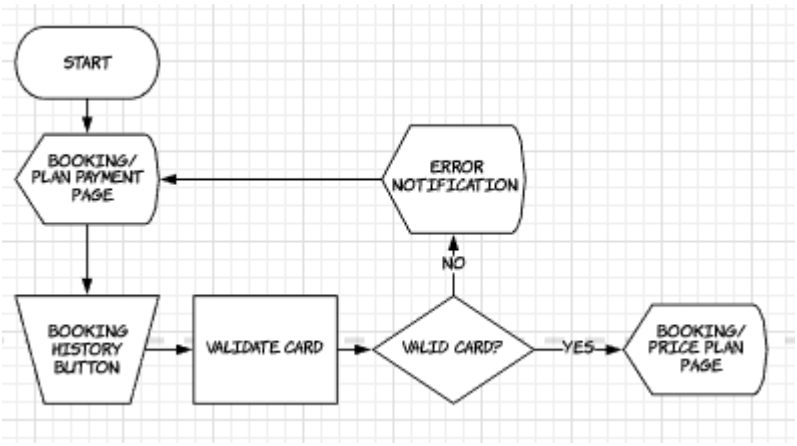
4.1.11 View Registered Car



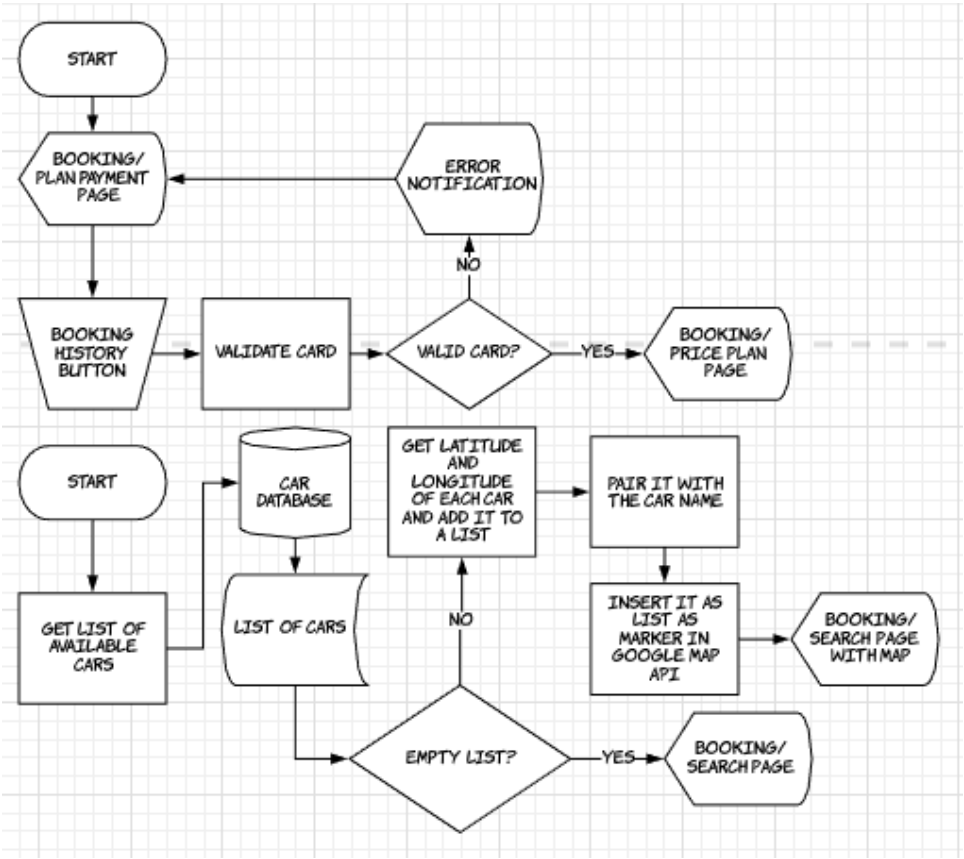
4.1.12 Car Registration



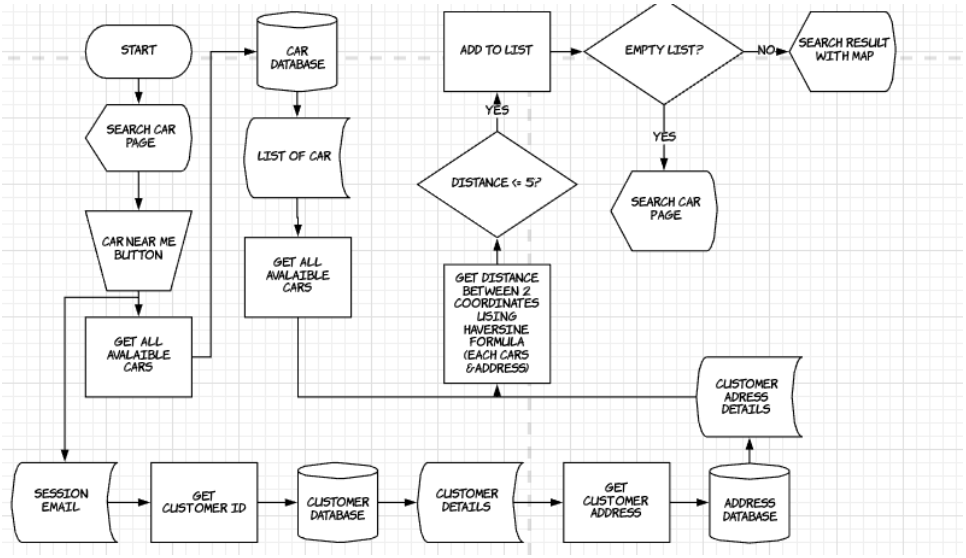
4.1.13 Card Detail for Payment



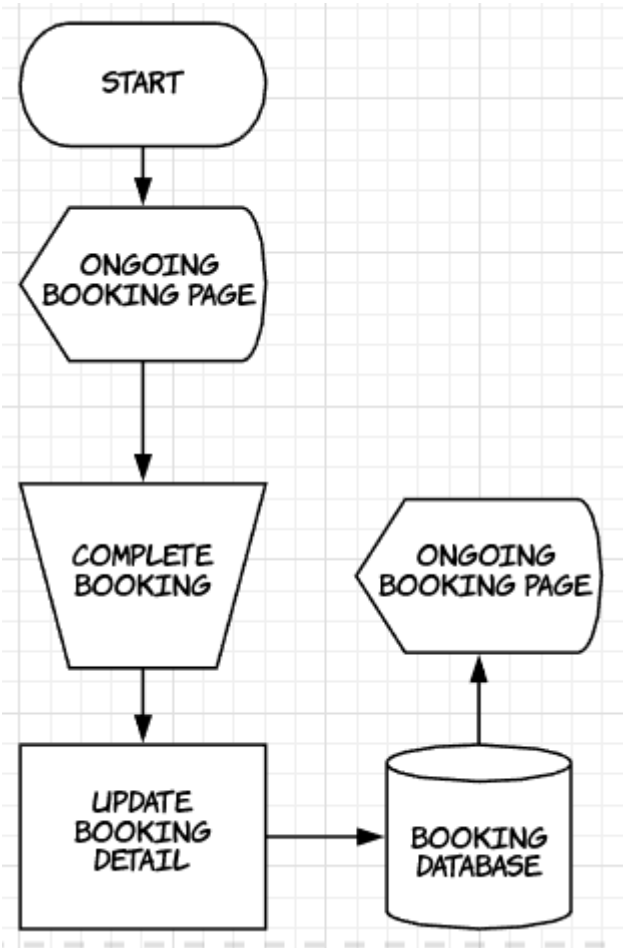
4.1.14 Show Car Locations on a Map



4.1.15 Find Cars Near Registered Address



4.1.16 Complete a Booking



5 Database Architecture

The database was built in a way that would allow for easy maintenance of the customer data. This was because we understand that were would be a lot of data that may need to be stored, so it is essential that the database is maintainable and easy to use. There is potential for scalability in the database if we wanted to expand on it further. An idea that was followed was creating the databased in third normal form but in the case of our database, had not been fully followed. Our database still allows for maintenance to be change easily to follow that normal form technique mentioned previously, and then it would allow for a lot more scalability within the database.

Below is our database schema:

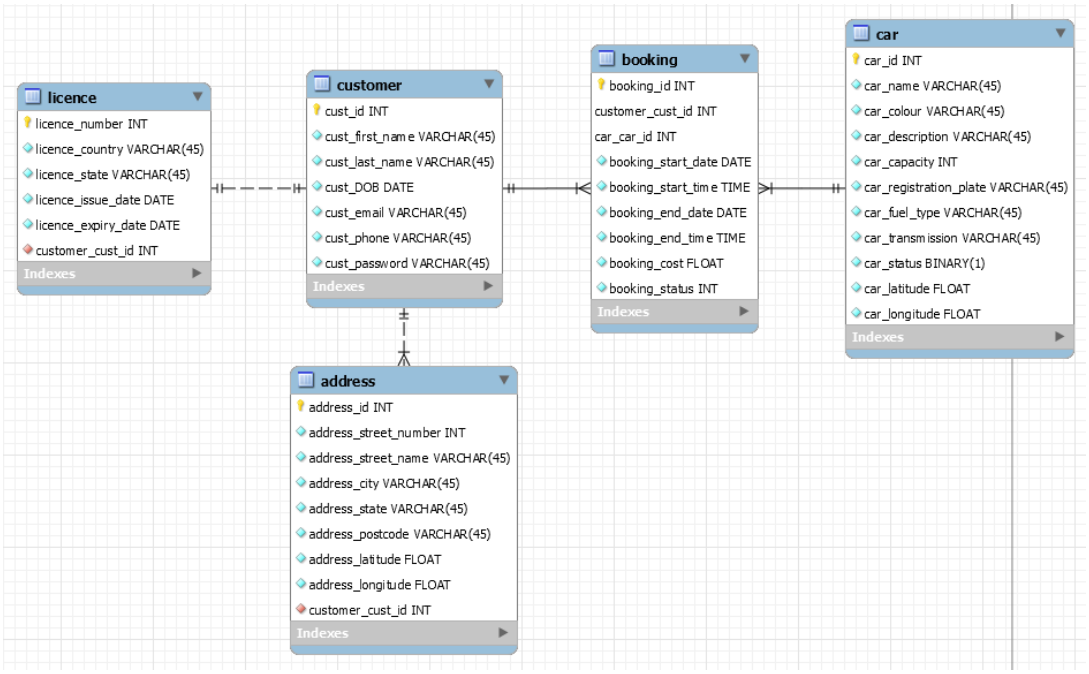
Customer(cust_ID, cust_first_name, cust_last_name, cust_DOB, cust_email, cust_phone, cust_password)

Licence(licence_number, cust_id, licence_country, licence_state, licence_issue_date, licence_expiry_date)

Car(car_ID, car_name, car_colour, car_description, car_capacity, car_registration_plate, car_fuel_type, car_trasmission, car_status, car_latitude, car_longitude)

Booking(booking_ID, cust_id, car_id, booking_start_date, booking_start_time, booking_end_date, booking_end_time, booking_cost, booking_status)

Address(address_ID, cust_id, address_street_number, address_street_name, address_city, address_state, address_postcode, address_latitude, address_longitude)



6 Implementation Instructions

I. Locally

In order to run this project locally, you will have to prepare any localhost server using apps such as XAMPP, MAMP, etc. This is required for you to have a local MySQL database. Once you set it up you can use the database by using the localhost IP address which is 127.0.0.1.5000. You will also need to also create a Project in GCP then you will have to enable both the Google Map API and Geocoding API of the project use just created. Then head to the API credentials page to create a new API key. Once the API key is created, just copy it and paste it in the code where Google Map and Geocoding are implemented. After all of that, the project should be able to run locally.

II. Deployed

In order to run this project in a deployed environment, you will have to create a new project in GCP and go to the SQL page. Then proceed to create a MySQL instance of MySQL v5.7 then go to the database page and create a new database with whatever name you want. Once you set it up you can use this database by either using its public IP address or its instance name. You will also have to enable both the Google Map API and Geocoding API of the project use just created. The head to the API credentials page to create a new API key. Once the API key is created just copy it and paste it in the code where Google Map and Geocoding are implemented. After all of that is done, you will need to create an app.yaml in your project files. Finally, just open a command prompt where the location of the app.yaml is and do the command “gcloud app deploy”. After you finish that last step, you will be able to open the deployed version of the project.

7 Non-functional specifications

- The website needs to be able to be opened on all browsers
- The website needs to be able to be opened on mobile devices
- When searching for a car, it should take less than a second to display results
- When booking a car, that car will update and not show as available in less than a second
- When cancelling a booking, the car booked will update and show as available in less than a second
- If an error occurs on page, an error message is displayed in less than 5 seconds to the user
- The website is secure and can safely hold all confidential information
- The website can be expanded and further developed in the future
- The website is robust against malicious attacks
- The website is easily maintainable
- The website follows all legal requirements and ethical considerations

8 Summary of test results

Unit Test ID	Test Name	Expected Results	Actual Results
1.1	test_hash_successful()	True	True
1.2	test_hash_fail()	False	False
1.3	test_valid_DOB()	True	True
1.4	test_invalid_DOB()	False	False
1.5	test_valid_lincense_date()	True	True
1.6	test_invalid_lincense_date()	False	False
1.7	test_validate_customer_credentials_success()	True	True
1.8	test_validate_customer_credentials_fail()	False	False
1.9	test_register_success()	True	True
1.10	test_register_fail()	False	False
1.11	test_login_success()	True	True
1.12	test_login_fail()	False	False
1.13	test_find_customer_address_coordinates()	Equal	Equal
4.1	test_validate_premium_expiry_success()	True	True
4.2	test_validate_premium_expiry_fail()	False	False
5.1	test_validate_booking_time_success()	True	True
5.2	test_validate_booking_time_fail()	False	False
12.1	test_car_registration_success()	True	True
12.2	test_car_registration_fail()	False	False
12.3	test_calculate_car_price()	Equal	Equal
13.1	test_calculate_total_booking_cost()	Equal	Equal
13.2	test_calculate_total_booking_time()	Equal	Equal
13.3	test_card_validation_success()	True	True
13.4	test_card_validation_fail()	False	False

Test Case ID	Test Description	Expected Results	Actual Results
1.1	Test the Login Functionality for Customer	Success	Success
1.2	Test the Login Functionality for Customer	Success	Success
1.3	Test the Registration Functionality for Customer	Success	Success
1.4	Test the Registration Functionality for Customer	Success	Success
1.5	Test the Registration Functionality for Customer	Success	Success
1.6	Test the Registration Functionality for Customer	Success	Success
1.7	Test the Reset Password Functionality for Customer	Success	Success
1.8	Test the Reset Password Functionality for Customer	Success	Success
1.9	Test the Reset Password Functionality for Customer	Success	Success
2.1	Test the Logout Functionality for Customer	Success	Success
3.1	Test the search car Functionality for Customer	Success	Success
3.2	Test the search car Functionality for Customer	Success	Success
3.3	Test the search car Functionality for Customer	Success	Success
4.1	Test the Switch Price Plan for Customer	Success	Success
4.2	Test the Switch Price Plan for Customer	Success	Success
4.3	Test the Switch Price Plan for Customer	Success	Success
5.1	Test the Make Booking Functionality for Customer	Success	Success
5.2	Test the Make Booking Functionality for Customer	Success	Success
6.1	Test the Cancel Booking Functionality for Customer	Success	Success
9.1	Test the Login Functionality for Admin	Success	Success
9.2	Test the Login Functionality for Admin	Success	Success
10.1	Test View user list functionality for admin	Success	Success
11.1	Test view list of car registered functionality for admin	Success	Success
11.2	Test modify car information functionality for admin	Success	Success
13.1	Test the Payment Functionality for Customer	Success	Success
13.2	Test the Payment Functionality for Customer	Success	Success
15.1	Test the Cars near to me Functionality for Customer	Success	Success
16.1	Test the Complete Booking Functionality for Customer	Success	Success

9 Known Issues & Risks

Description of Risk	Impact on project
Identity theft (information extortion): Get your car user will potentially losing their personal information and might will commit fraud by hacker.	Gey your car will be losing use customer because the unsafe environment and erode consumer confidence, diminishing get your car's rental sales
Human error or failure: Employee might inexperience or incorrect assumption.	It's easy to revelation of classified date, entry of erroneous data or accidental data deletion or modification.
Technical hardware failures(error)	Hardware can fail and cause project outage

10 Other Considerations

There were no other considerations for this project that may have affected our proposal's acceptance and delivery.