

Governors State University

## OPUS Open Portal to University Scholarship

---

All Capstone Projects

Student Capstone Projects

---

Fall 2022

### Rent-A-Car

Poojitha Konanki

Follow this and additional works at: <https://opus.govst.edu/capstones>

---

For more information about the academic degree, extended learning, and certificate programs of Governors State University, go to [http://www.govst.edu/Academics/Degree\\_Programs\\_and\\_Certifications/](http://www.govst.edu/Academics/Degree_Programs_and_Certifications/)

Visit the [Governors State Computer Science Department](#)

This Capstone Project is brought to you for free and open access by the Student Capstone Projects at OPUS Open Portal to University Scholarship. It has been accepted for inclusion in All Capstone Projects by an authorized administrator of OPUS Open Portal to University Scholarship. For more information, please contact [opus@govst.edu](mailto:opus@govst.edu).

# **RENT A CAR**

By

**Poojitha Konanki**

B-Tech,Srm University,2019

GRADUATECAPSTONE SEMINAR PROJECT

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,

With a Major in Computer Science



Governors State University  
University Park, IL 60484

2022

## **ABSTRACT**

In the current world, people are always seeking to decrease their workload by utilizing technology. They are all Travelling to get away from their worries. Using our application, we decreased the difficulty of car rental in our rent-a-car operation. The car owner posts information about the vehicle and how people can travel in it. Car owners oversee car availability. Renters can select the car they want to rent based on their preferences. The renter can see the model of the car and how much space it has. If a renter wants to book a car, they fill out a form and pay the car owner. We gave the option, renters and car owners should communicate with one another. After the trip, the renter can evaluate the vehicle.

Our team members wish to construct the application utilizing the Model, View, Controller (MVC)

framework in .Net core for the backend and HTML5, CSS3, Bootstrap, And JavaScript for the front end, with data saved in MS-SQL Server. Renter, Vehicle Owner, and Administrators use the rent-a-car system. Vehicle owners can post photographs of their vehicles along with vehicle information. Renters can reserve the car they require. The administrator is in management of the entire application.

# Table of Content

<b>1</b>	<b><i>Project Description</i></b> .....	1
1.1	Competitive Information .....	1
1.2	Relationship to Other Applications/Projects .....	1
1.3	Assumptions and Dependencies .....	1
1.4	Future Enhancements .....	2
1.5	Definitions and Acronyms.....	2
<b>2</b>	<b><i>Project Technical Description</i></b> .....	2
2.1	Application Architecture .....	3
2.2	Application Information flows .....	6
2.3	Interactions with other Applications .....	8
2.4	Capabilities .....	8
2.5	Risk Assessment and Management .....	9
<b>3</b>	<b><i>Project Requirements</i></b> .....	9
3.1	Identification of Requirements .....	9
3.2	Operations, Administration, Maintenance and Provisioning (OAM&P) .....	10
3.3	Security and Fraud Prevention .....	10
3.4	Release and Transition Plan .....	11
<b>4</b>	<b><i>Project Design Description</i></b> .....	11
<b>5</b>	<b><i>Internal/external Interface Impacts and Specification</i></b> .....	13
<b>6</b>	<b><i>Design Units Impacts</i></b> .....	25
6.1	Functional Area A/Design Unit A.....	25
6.1.1	<b><i>Functional Overview</i></b> .....	25
6.1.2	<b><i>Impacts</i></b> .....	26
6.1.3	<b><i>Requirements</i></b> .....	26
<b>7</b>	<b><i>Open Issues</i></b> .....	26
<b>8</b>	<b><i>Acknowledgements</i></b> .....	26
<b>9</b>	<b><i>References</i></b> .....	26
<b>10</b>	<b><i>Appendices</i></b> .....	27

## ***1 Project Description***

The purpose of this project is to create a platform for renters and vehicle owners to meet their requirements. We created three consoles: one for the admin, one for the vehicle owner, and one for the renter. Also available to non-registered users are the Guest Pages, which include the Homepage and Vehicle Details Page. As an administrator, you can see and control (Activate / Deactivate) vehicle owners, renters, and vehicles. And the admin can create a vehicle Category, a list of coverage location, and a list of vehicle companies and be able to view the survey and bookings. Vehicle owners may update their profile information, post a vehicle, and see a list of people that booked the vehicle and survey information. Also, having the ability to meet with the renter for a chat. Renters may use the feature to search for the vehicle. Renters can provide comments about the vehicle, as well as communicate with recruiters through their console chat. We also provide additional options on this site, such as vehicle searching by location, brand, category, and date. All users, registered and non-registered, will be able to see the home page, vehicle details page. We present the latest categories, Brands and premium vehicles and their details on the index page. The following consoles are available on the project criteria. That is Admin, Owners, and Renter consoles.

### ***1.1 Competitive Information***

Companies are offering car rental management services to renters and vehicle owners; we chose Turo.com [2] as one of the most famous competitors in the online vehicle rental management field. We can book vehicles and view the details of the vehicles. In the same manner in our project show index page provides category-wise vehicle information and top brands actively renting. We also have a list of Premium Vehicles too.

### ***1.2 Relationship to Other Applications/Projects***

This project provides a comprehensive solution that may be used in any other application [3][4]. Renters can use this project to book the vehicles in the market. Vehicle owners may present their vehicle to the open world for renters who want to search for a vehicle, and they can talk to each other about it. Admin can control the vehicle owner, renter, and vehicles.

### ***1.3 Assumptions and Dependencies***

- All users must have the basic knowledge of online website usage
- Admin know the management of vehicle owners, renters & vehicles

- For booking a vehicle renter needs to search the vehicle based on their requirements with dates and book the vehicle and pay the amount for the trip.
- All payments are clicked pay is considered as paid
- All Payments cancelled as Failed Transaction
- All bookings considered as terms and conditions provided by the vehicle owner are followed by the renter.
- All bookings considered as used on the booked dates

#### ***1.4 Future Enhancements***

This project includes all aspects of vehicle renting. We may also provide the following functionality. We can also provide applicants with GPS Tracking live locations.

- Payment gateway integration
- WhatsApp (social media) Integration for Easy Interactions
- SMS Gateway integration for Notifications.

#### ***1.5 Definitions and Acronyms***

- Administrator – Admin User of the Rent A Car Web Application
- Owner – Vehicle Owner
- Renter – Who Rent a Vehicle
- HTML5 - Hypertext Markup Language 5
- CSS3 - Cascading Style Sheet
- SaaS - Software as a service
- .NET Core – New Open-Source Software Development Platform.
- MVC - Model –View – Controller
- JSON is JavaScript Object Notation
- MS-SQL – Microsoft SQL Server
- Bootstrap – Open-Source CSS Framework for UI Development

## ***2 Project Technical Description***

This project is providing a platform for vehicle owners and renters to rent a car. This project was developed in .Net Core in MVC design pattern and, we use Entity Framework Core, LINQ to access and process Database, we used bootstrap for user interface design with the jQuery methods. We used three controllers

in this application. i.e., Home, Admin, vehicles. We use a few features of .net core for requirements of this project like we use identity for authentication purpose. With this we can do all the process involved in the login and we used ASP Role based Authorization of page access, it is used to restrict the user to access required contents. We used Dependency Injection concept with that we get context in full flow as it. We use inheritance, Polymorphism, Interface, and more object-oriented programming concepts. This project having the following pages:

Index Page, Login Page, Registration Page, Separate Dashboard for all users (Admin, Vehicle owner, Renter), Vehicles Page register and list page, booking list page, Advanced Search Page, Vehicle Details Page, Book Now Page, Pay now Page, Chat & Survey.

### ***2.1 Application Architecture***

In this web application, we used Entity Framework Core migrations feature to change the data model and update the database. This project started with Index (Home). It is the normal website view of recent Categories and Brands and Premium vehicles for any user (guest, Renters, vehicle owners, and admin). In the guest Index we show the recent premium vehicles List for easy access. Search the vehicle with the advanced filters like Category, brand, location, name of the vehicle and date availability. If they click the category, then it goes to vehicles available in this category as a list. If the renter wants to book the vehicle, they can click the book now button. Then it shows the vehicle details in the book now page if they want to confirm the vehicle then need to click the pay now button. It will redirect to the pay now page to complete the payment process. If the vehicle is booked by the renter, then the information is shown to the vehicle owner and admin in the booking list page. We have the chat option; it can be used to discuss between renters and vehicle owners.

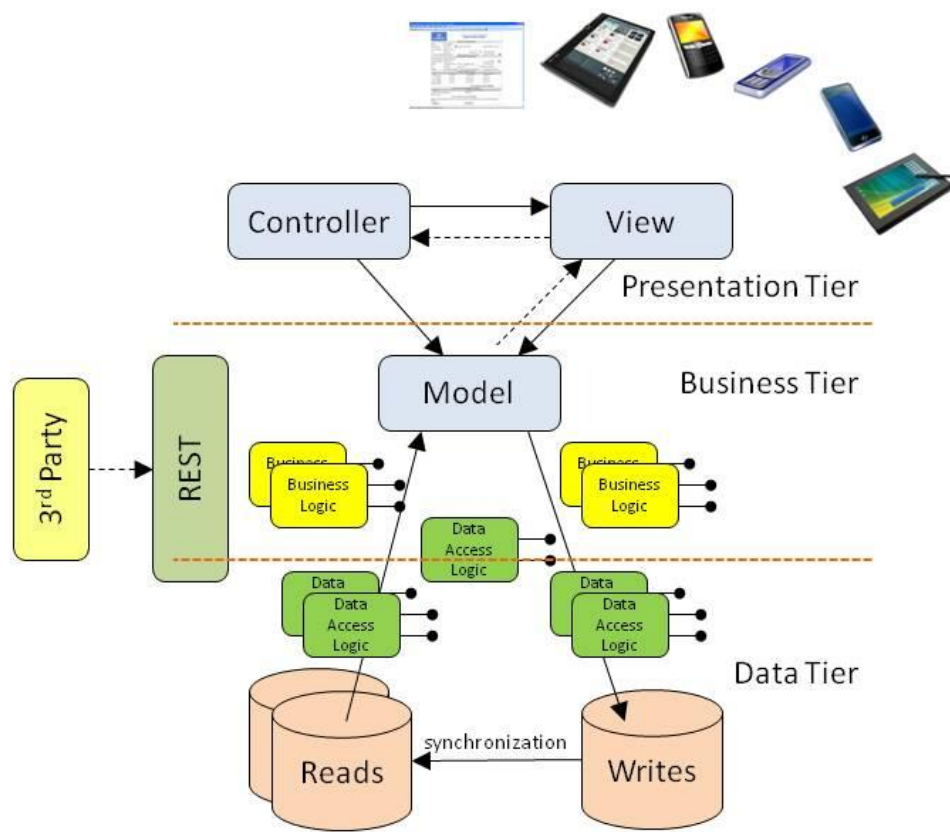


Figure 1: System Architecture [1]



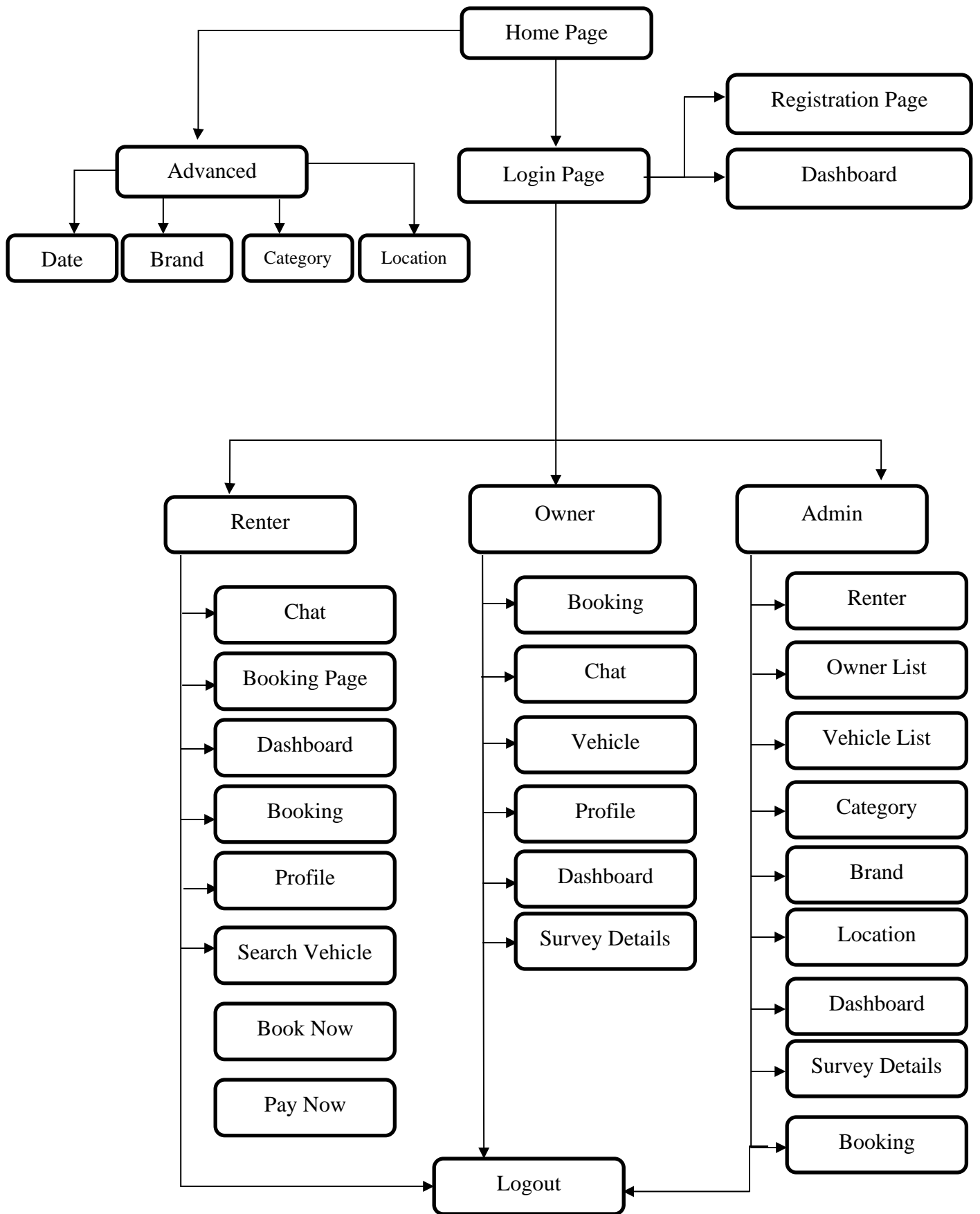


Figure 2: Application Architecture

## 2.2 Application Information flows

We used four different streams for this web application project. Four application workflow diagrams are added below for detailed understanding. In the Information workflow first, we start with a home page called an index page. In index we provide the options like recently posted premium vehicles in case the index page can be accessed by any type of users either registered or guest users. And we provide the search option, with this search option we can see all the vehicles posted in the portal with advanced search features based on location, brand, name, date availability and category. With the search result we can also see the details of vehicles. In the case of Admin console, we have the option to monitor the vehicle owner, renter, and registered vehicles and he/she can control it by activating or deactivating the items. And the admin can add, edit, and delete the Category, Brand, and Location. In case a vehicle owner can register his/her vehicle and view the bookings, discuss with the renters and feedback (survey) received from the renter. In renter portal they can view the wallet (spends) in Dashboard itself and they can book vehicles and view the bookings and payment history and can discuss with the vehicle owners.

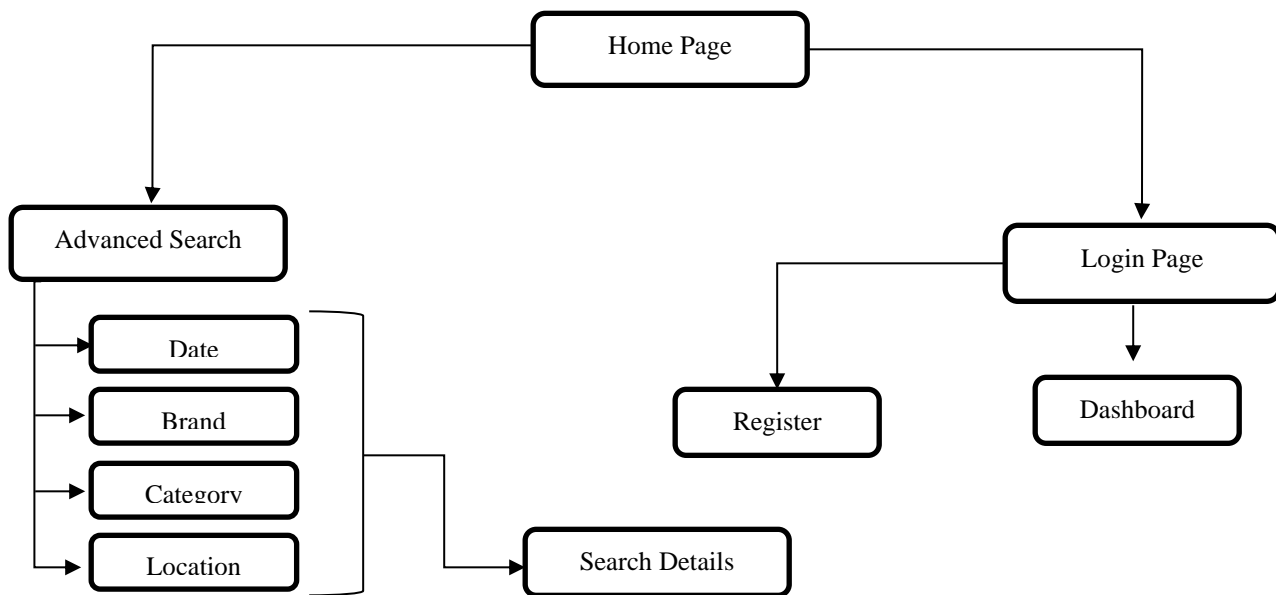


Figure 3: Guest Workflow

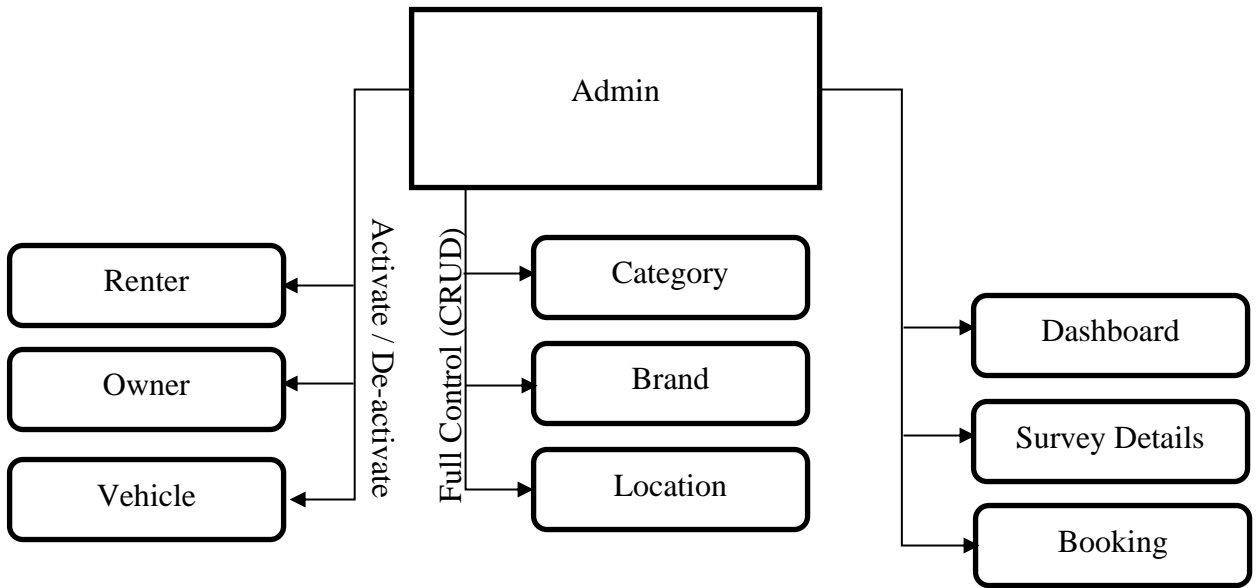


Figure 4: Admin Workflow

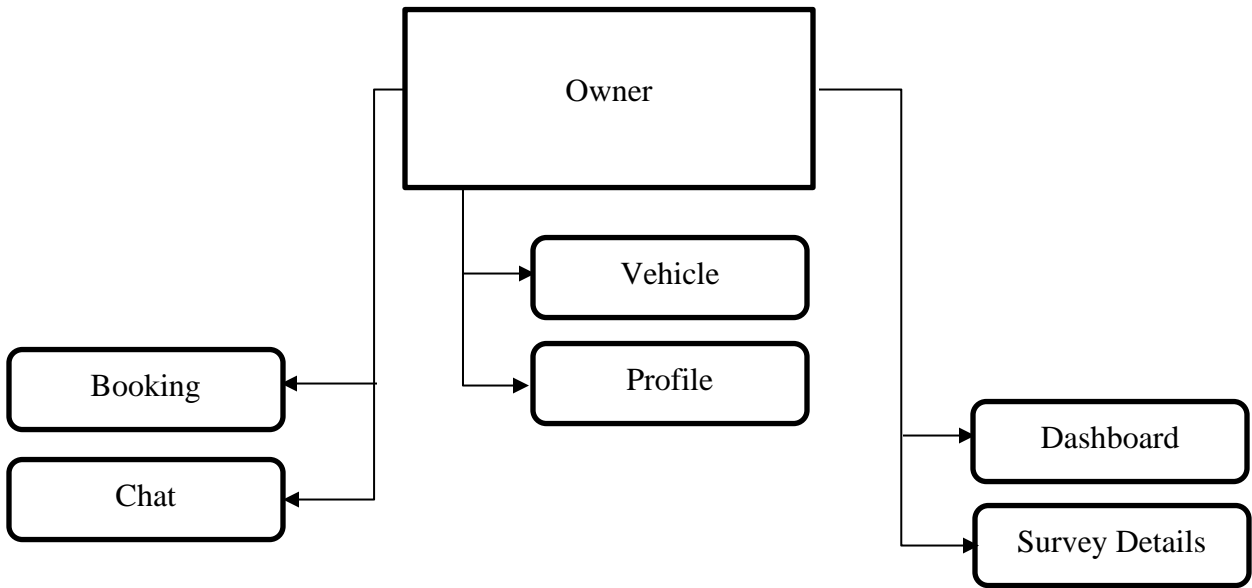


Figure 5: Owner Workflow

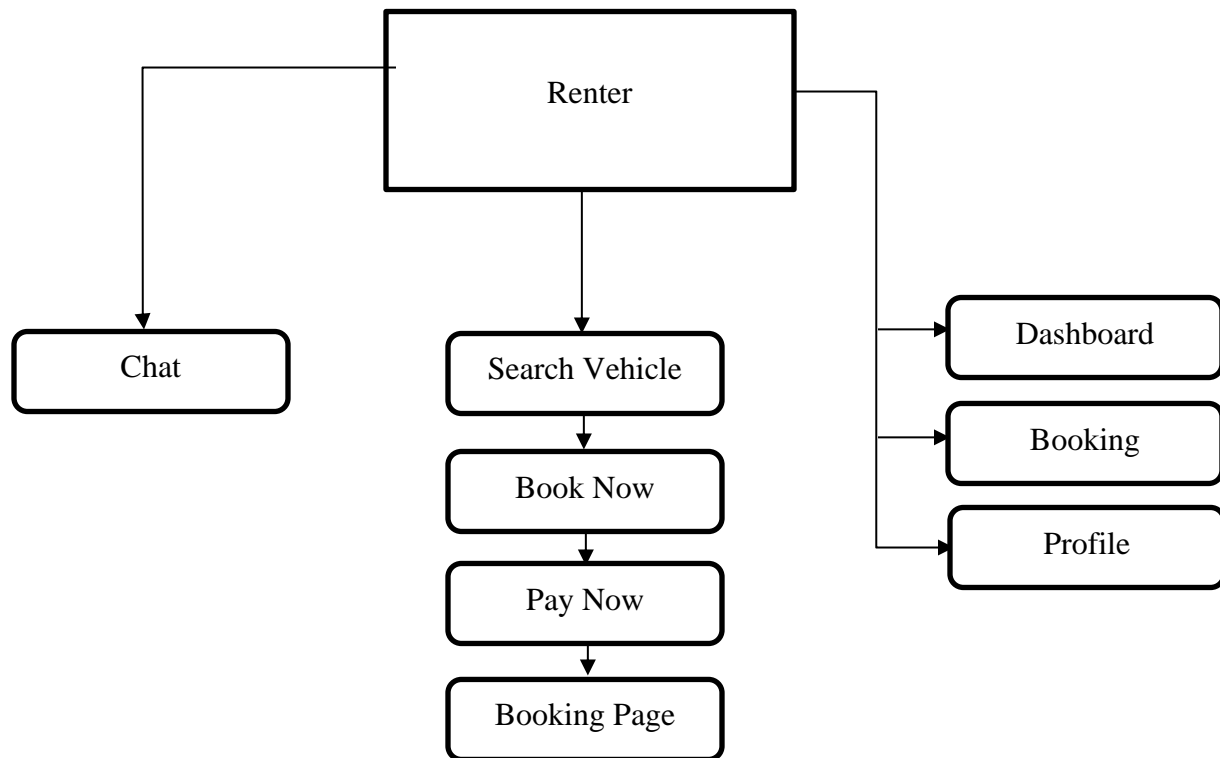


Figure 6: Renter Workflow

### 2.3 Interactions with other Applications

In this web application we are providing vehicle and booking details, in this portal we can also provide the related contents to the trip like travel kit, chitchat items, cinema tickets. If we go for that option, we need to work on the API from their respective application.

### 2.4 Capabilities

- The users should be able to access this web application from their device, whatever type of device it may be a desktop, laptop, tablet, or smartphone.
- The website is dependable and has a user-friendly interface. The user interface (UI) should enable Renter / Vehicle Owner / Administrator to easily use this platform.
- The Renter must be able to view the details of the vehicle, book the vehicle and pay for that trip.
- Renters can discuss with the vehicle owner to get the any other information need about the vehicles

- Admin can monitor and control the Vehicle Owners, renters, and vehicles and provide category, Brand, coverage location.

## 2.5 *Risk Assessment and Management*

Risk management is becoming the most challenging aspect of this web application. While we cannot predict the future with certainty, we can apply a simple and streamlined risk management process to predict the uncertainties in this web application and minimize the occurrence. Risk management helps in avoiding crises and provides stability. This improves the chance of successful project development and completion and reduces the consequences of risks. This certainly is not the end of the journey for us on the effective risk management of this web application. It is a constant learning process to be able to constantly improve our practices to increase our process efficiency and risk-free development and maintenance of the software.

## 3 *Project Requirements*

### 3.1 *Identification of Requirements*

#### <RentACAR\_2022-1 User-capability- “000101”>

**Responsive Design:** Users can be assured of getting the perfect UI / UX on any device (desktop, laptop, tablet or mobile with any resolution)

#### <RentACAR\_2022-2 User-Registration- “000102”>

**User Registration:** Renter and vehicle owners can be Register by their own, it is the essential process to do the respective action in their portal

#### <RentACAR\_2022-3 Search- “000103”>

**Advanced Search:** any user can search for the vehicle with different criteria like Location, Category, Brand of the vehicle and name of the vehicle with availability by date.

#### <RentACAR\_2022-4 Admin-capability- “000104”>

**Admin Panel:** Admin can view and activate / de-activate the owners, renters, and vehicles, and, they can add, edit, and delete category, brand, location and can view the survey details.

#### <RentACAR\_2022-5 Owner-capability- “000105”>

**Vehicle owner Portal:** vehicle owners can add their vehicles, discuss with the renters by using chat menu and they can see booking of their vehicles and survey of their vehicles. And they can update their profile.

#### <RentACAR\_2022-6 Renter-capability- “000106”>

**Renter Portal:** Renters can update their profile, discuss with the vehicle owners using chat menus and add surveys for completed the booking, search the vehicle, view the vehicle details, confirm the booking, and complete the payment using pay now page.

### ***3.2 Operations, Administration, Maintenance and Provisioning (OAM&P)***

#### **Operations**

- This application is designed to help renters search for available vehicles.
- This application is very user friendly and dependable to use.
- The function of the website for searching the vehicle should be easy and secure.
- The registration of user and login are not required to access the website. As a guest can access the home page and vehicle details page.
- The data generated from the search option should show the vehicle based on date availability.

#### **Administration**

- Only the admin can manage all the users and Default Items like category, brand.
- Only the admin can be able to create, modify and remove the Brand, category, and Location.
- The admin can activate or deactivate the renter, owner, and vehicle.
- Admin can see all the information in dashboard for easy access.

#### **Maintenance**

- Pages load faster when free of large file sizes for these complex web applications.
- This utility ensures that the regular backup is made up of the entire website and data to maintain the integrity and security.

#### **Provisioning**

- This application ensures reliable configuration for the whole application.
- The ability to monitor user progress and control the access of the application.
- Admin can have the ability to Add, Update or delete the category, location, brand easily and efficiently

### ***3.3 Security and Fraud Prevention***

The security and fraud detection process can be established through security & fraud prevention planning.

The following steps are the fraud prevention planning process are,

- Delegated responsibilities with specific role to the management (Admin)
- Applications, should be updated regularly with security batch updates.

- All user details (Renter and owner) saved through the required registration process.

### ***3.4 Release and Transition Plan***

The website development must be successfully completed and evaluated before, and it can be deployed to the live server. The website should not be launched for vehicle booking process before fixing the bugs. After booting, the data storage process will be activated, and the maintenance process will be used to maintain the continuous workflow of the web application.

## ***4 Project Design Description***

In this project, we did all the functionality in .Net Core 6.0. Which is the latest one and more efficient, faster than other platforms, open source and independent to any environment. It can work in windows or Linux or mac, with the help of .net core we use a lot of features like dependency injection, inheritance, Identity. We use a lot of features of .net core 6.0. Especially we use an identity function for users which is more useful to authenticate the user as well as we use identity Role for Authorization of the users because we use separate roles of user in the same application with different portal, so we use asp .net identity role to authorize the user. We also use the MVC design pattern in project development which is having more continent and easier to understand the code and feature enhancements.

In MVC pattern, all code sections are divided into major sections such as Model, View, and Controller. In Controller, it is used to route the user request, based on the controller action method it fetches the data from the model and uses the data in view. In this MVC pattern we have three controllers as “Home Controller, Admin Controller, Vehicle Controller” with this controller we did all the functionalities [5]. We also use Razor View for .net Core application view design. We used entity framework core for model-based operation and context access. Entity framework core is used to create a model-based database creation and update using entity framework migration feature.

In this project flow starts from the Home / Index Page which is the main first page of the application which is accessible to all either non-registered user, vehicle owner, renter, or admin. In this home page we show the recent premium vehicles. And recent category ads in the portal and recent brands added in this portal. If anyone wants to know vehicles available in that category or brand just click the show more link button. In the case of premium vehicles if the user wants to book the vehicle just click on the book now button it will redirect to the details page, in case of the current user in renter role else it will redirect to the login page. And the site index also has features like search for vehicles. After the home page, the next search page is used to search the vehicles on different criteria, like location-based search, brand-based search,

category-based search, and date availability. In search results show the vehicle when vehicle available to book else it would not show the vehicle. And we provide the chat button it will redirect to chat window if the current user becomes the renter. Then vehicle detail page, in this page detailed information is displayed like vehicle location of pickup, number of seats available, fuel type, gear type, if air-condition and navigation available or not. How far the trip, in this case we added one implementation based on the data available in the vehicle detail and renter input. In case the renter wants to book a vehicle for 10 hours but the vehicles per day amount is lower than it will provide the discount of which is the lower amount that will take consider for payment, which is applicable for day wise and month wise too. And we provide / show the terms and conditions given by the vehicle owner in this page, if the renter hires this vehicle, then he considers as accepted the terms and condition provided here too. And we provide the comments of the vehicle. And we show the location of the vehicle on a map. And if the user wants to book this vehicle, then she / he needs to register as renter then click the pay now button. Once the pay now button is clicked it will add the booking table in the database and show the payment page. In the payment page, we also show the vehicle information and invoice information and bank information to complete the payment process. In case the user wants to cancel the payment, they may click the cancel button, it will be considered as the failed payment in booking.

For admin User, after login we show the dashboard, in that we show the statistical overview like total amount latest booking, etc. And the admin has the right to create, update, and delete the coverage location, category, and brand. In the case of coverage location, we provide the latitude and longitude of the location also. And we have the renter and owner list in user's main menu. In the user list admin can activate and deactivate the user with the help of the change status button. In that same manner admin can activate / deactivate all the vehicles registered in this portal. And admin can view the survey details and booking details.

In the case of vehicle owners, after login we show the dashboard and show their statistical status. And admin have the rights to add multiple vehicles with detailed information, based on this input only this web application display the vehicle information, and also owner can update their profile also, he /she can discuss with the renter using chat option, it shows the list of renter approached to them that is show in the list if the owner want to discuss with any one just click on the respective name it will show the message details in the list with the difference in date and time and show the message like social media approach, and also we provide the survey and booking details in the respective page of their vehicles.

As a renter user, after login the application we show the dashboard window with statistical presentation. And he/ she can book the vehicle using the home button and search option discussed earlier in this chapter. And renters can discuss with the vehicle owner with the help of chat window, and they can update the



profile details. In case of booking list page having the option to provide the survey for the paid booking, in case of payment failed then it is considered as not applicable, and once the survey completed for the respective booking then it is shown as completed else it shows as available. If they click the available option, then it will redirect to survey page and then renter need to provide the feedback of the trip on that vehicle as a survey.

### 5 Internal/external Interface Impacts and Specification

#### Internal Specification:

MS SQL Server is used as a database server. It is a client-server architecture. MS SQL Server process starts with the client application sending a request and SQL Server accepts, processes, and replies to the request with processed data. As the below diagram show the three major components in MS-SQL Server Architecture:

- Protocol Layer
- Relational Engine
- Storage Engine

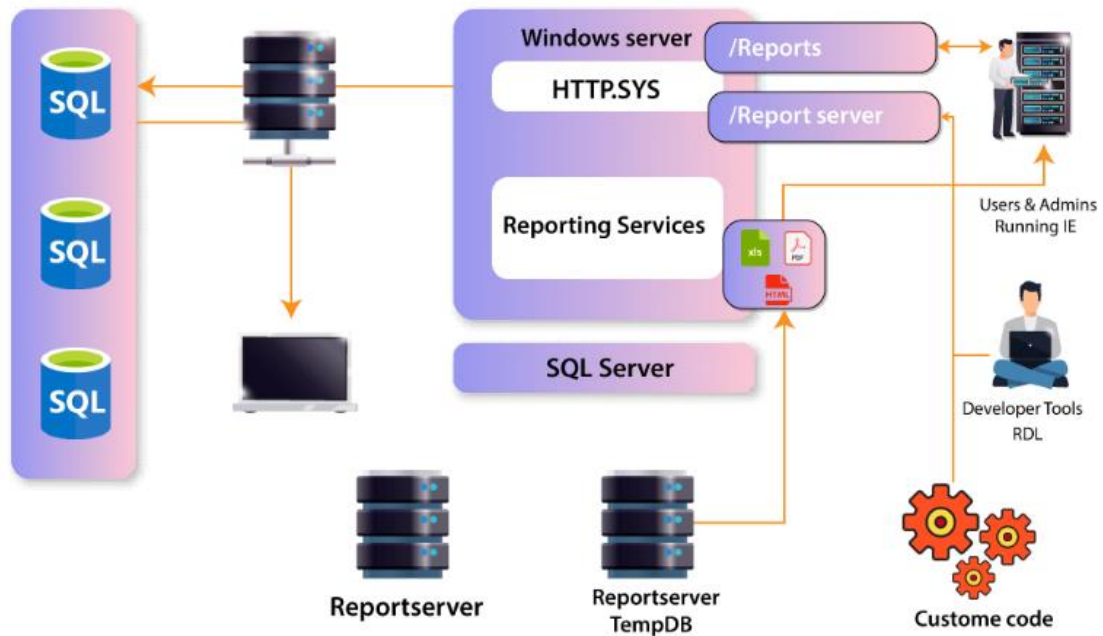


Figure 7: SQL Architecture [6]

**External Specification (Home Page)**

I Rental	Home	Category	Brands	Features	Login/Dashboard																
Search Your Car																					
Premium Categories																					
Category one		Category two		Category three																	
Brand																					
Brand one		Brand two		Brand three																	
Premium Vehicles																					
<table border="1"> <tr> <td rowspan="4">Car Image</td> <td>Hourly/ Daily/monthly price</td> </tr> <tr> <td>Air condition Status</td> </tr> <tr> <td>Seat Count</td> </tr> <tr> <td>Location</td> </tr> <tr> <td>Name</td> <td>Chat</td> </tr> <tr> <td>Brand / category</td> <td>Book Now</td> </tr> </table>		Car Image	Hourly/ Daily/monthly price	Air condition Status	Seat Count	Location	Name	Chat	Brand / category	Book Now	<table border="1"> <tr> <td rowspan="4">Car Image</td> <td>Hourly/ Daily/monthly price</td> </tr> <tr> <td>Air condition Status</td> </tr> <tr> <td>Seat Count</td> </tr> <tr> <td>Location</td> </tr> <tr> <td>Name</td> <td>Chat</td> </tr> <tr> <td>Brand / category</td> <td>Book Now</td> </tr> </table>		Car Image	Hourly/ Daily/monthly price	Air condition Status	Seat Count	Location	Name	Chat	Brand / category	Book Now
Car Image	Hourly/ Daily/monthly price																				
	Air condition Status																				
	Seat Count																				
	Location																				
Name	Chat																				
Brand / category	Book Now																				
Car Image	Hourly/ Daily/monthly price																				
	Air condition Status																				
	Seat Count																				
	Location																				
Name	Chat																				
Brand / category	Book Now																				
Get In Touch																					
				<table border="1"> <tr> <td>Name</td> <td>Mail</td> </tr> <tr> <td colspan="2">Message</td> </tr> <tr> <td colspan="2">Sent</td> </tr> </table>		Name	Mail	Message		Sent											
Name	Mail																				
Message																					
Sent																					
Contact Detail		About		Recent																	

# Table Design of Database

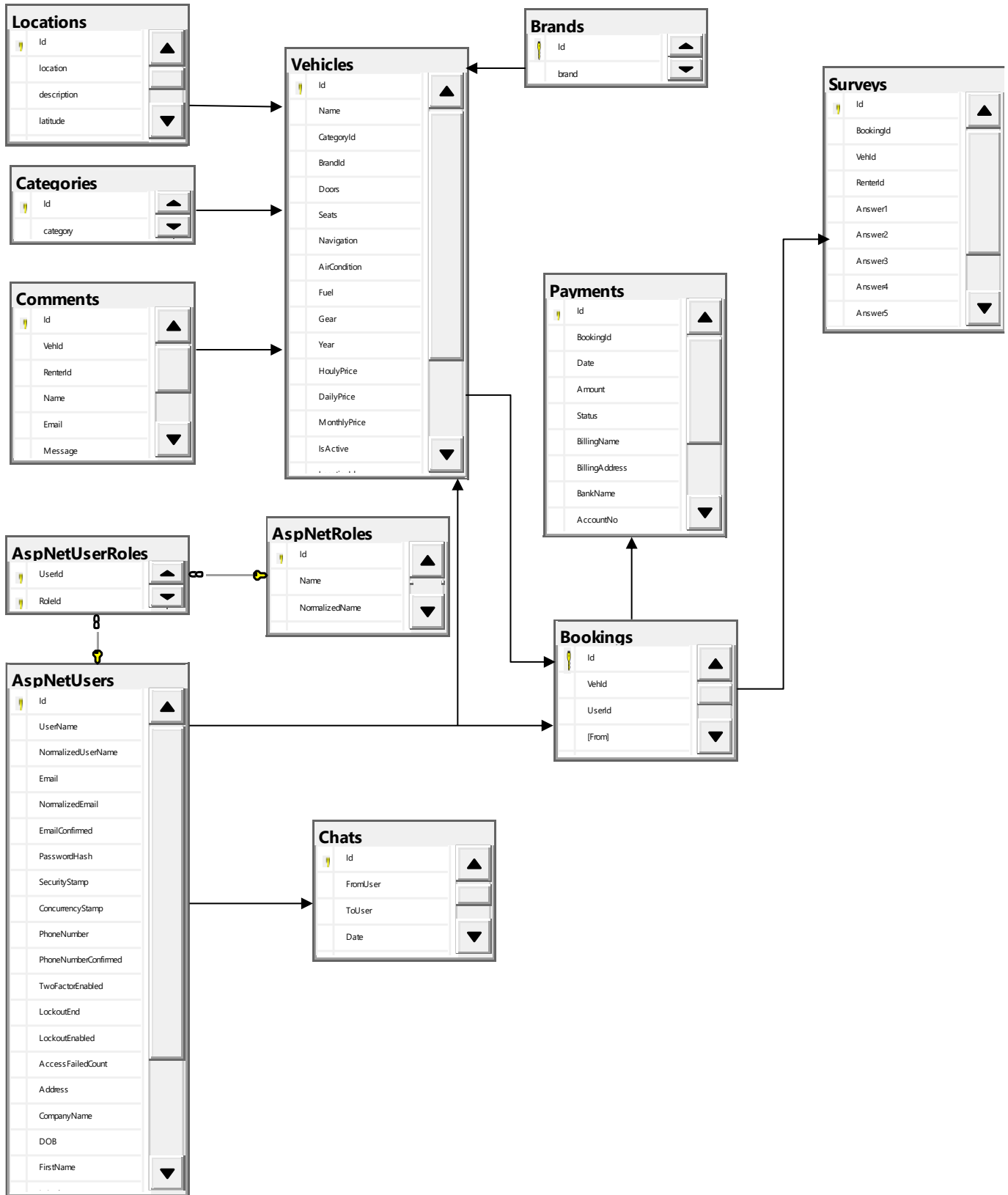


Figure 8: ER Diagram

## Site Index

Home of web application it is an index page to show the key details about the web application

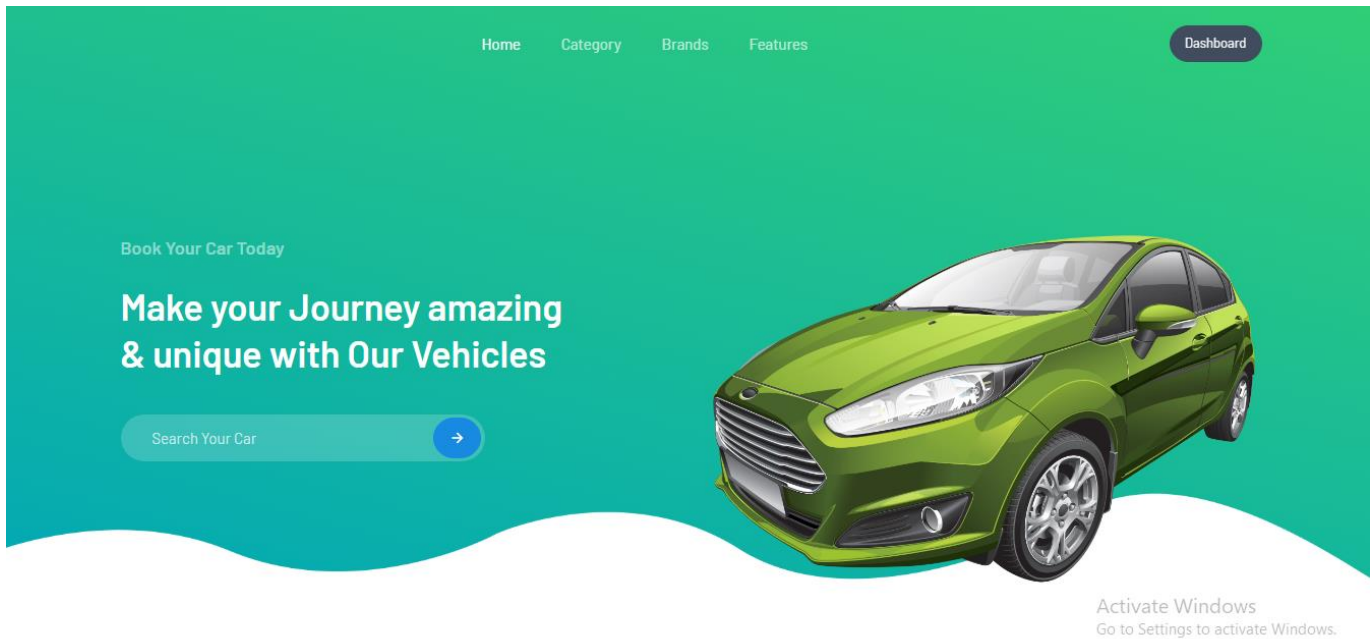


Figure 9: Home Page

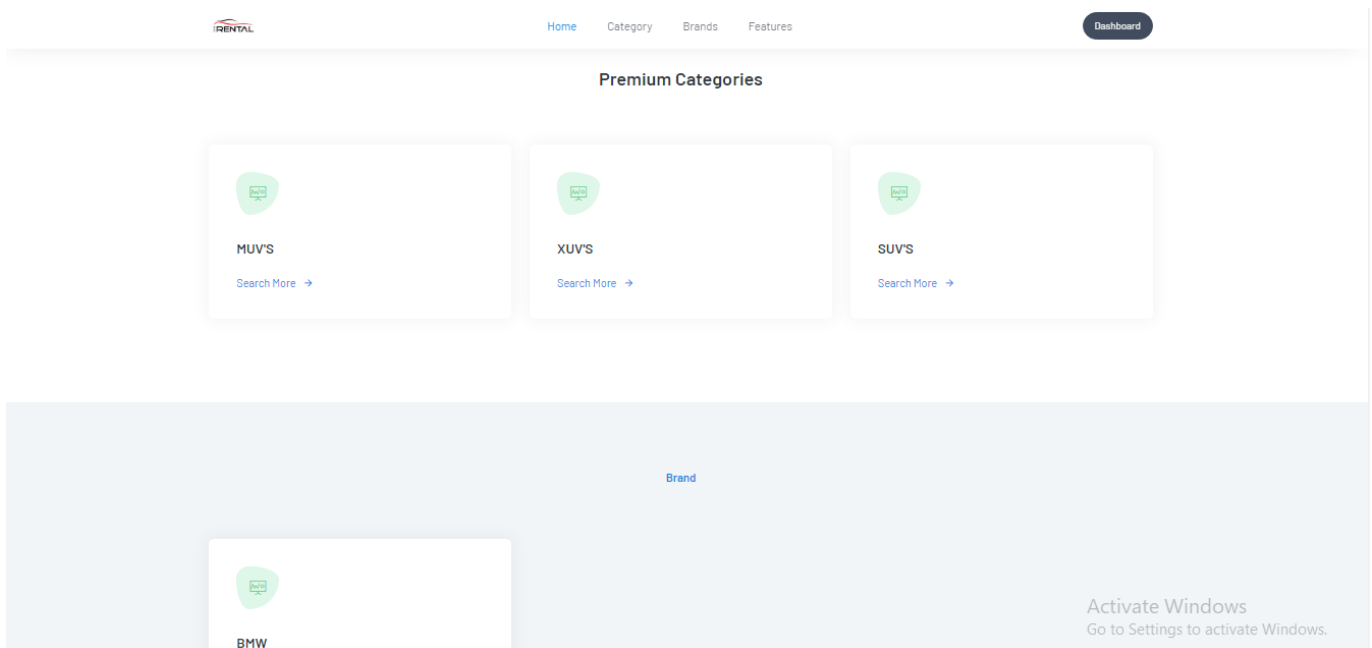


Figure 10: Categories Section

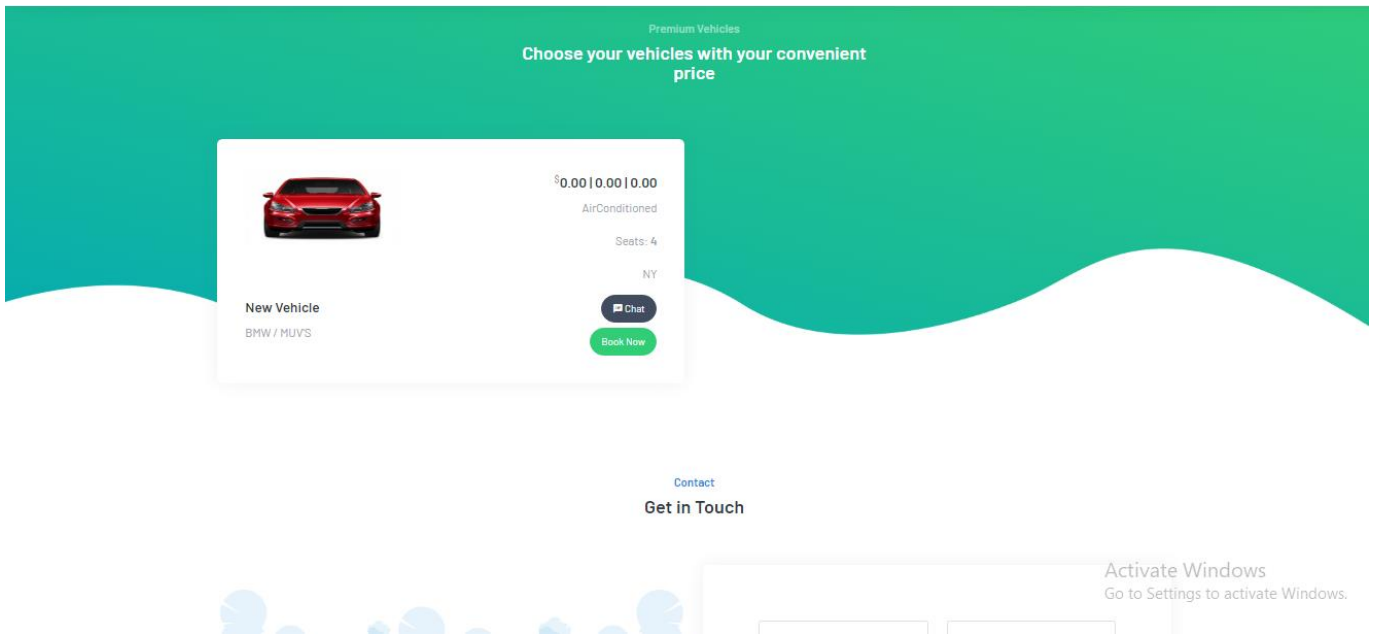


Figure 11: Highlighted Vehicle Section

## Registration Page

It is used to register renter and owner

A screenshot of a registration form on a dark blue background. The form is white and contains the following fields: "Role" (dropdown menu with "Renter" selected), "User Name" (text input with "AAAA"), "Email" (text input with "Enter your Email"), "First Name" (text input with "Enter your FirstName"), "Last Name" (text input with "Enter your LastName"), "Company Name" (text input with "Enter your CompanyName"), "DOB" (text input with "01/01/0001" and a calendar icon), and "Phone Number" (text input with "Enter your PhoneNumber"). There is a home icon in the top right corner and an "Activate Windows" notification in the bottom right corner.

Figure 12: Registration Page

## Login Page

Login page for admin, owner, and renter to go the console

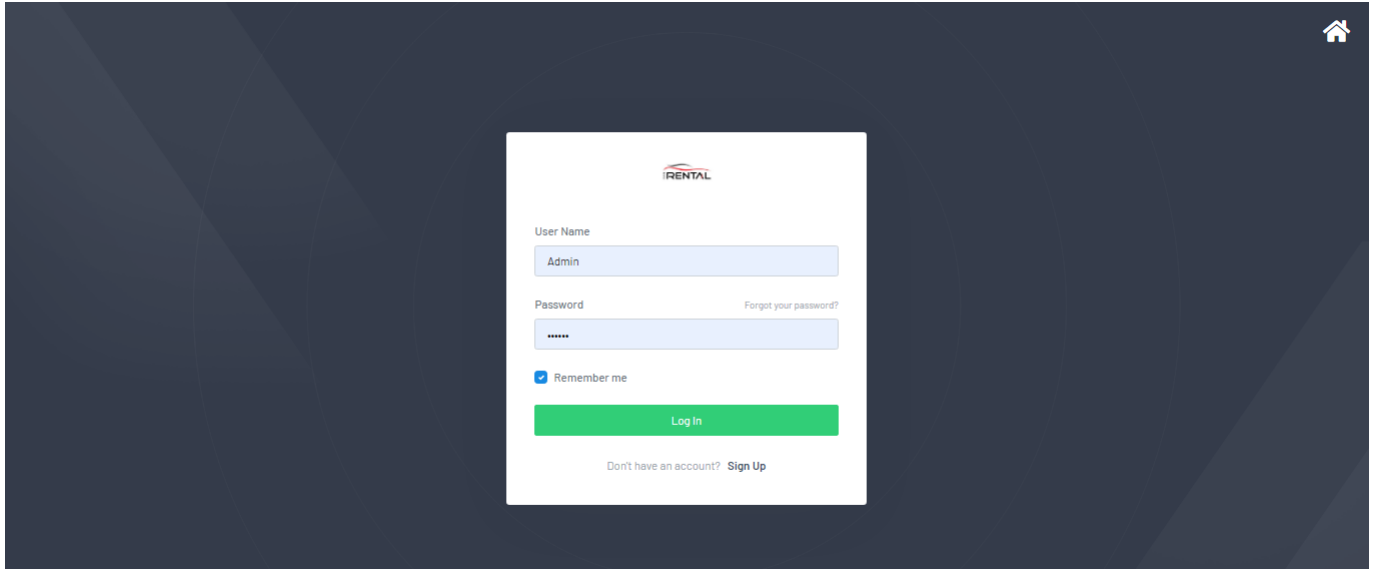


Figure 12: Login Page

## Admin Dashboard

Provide the overview of the application to admin in dashboard

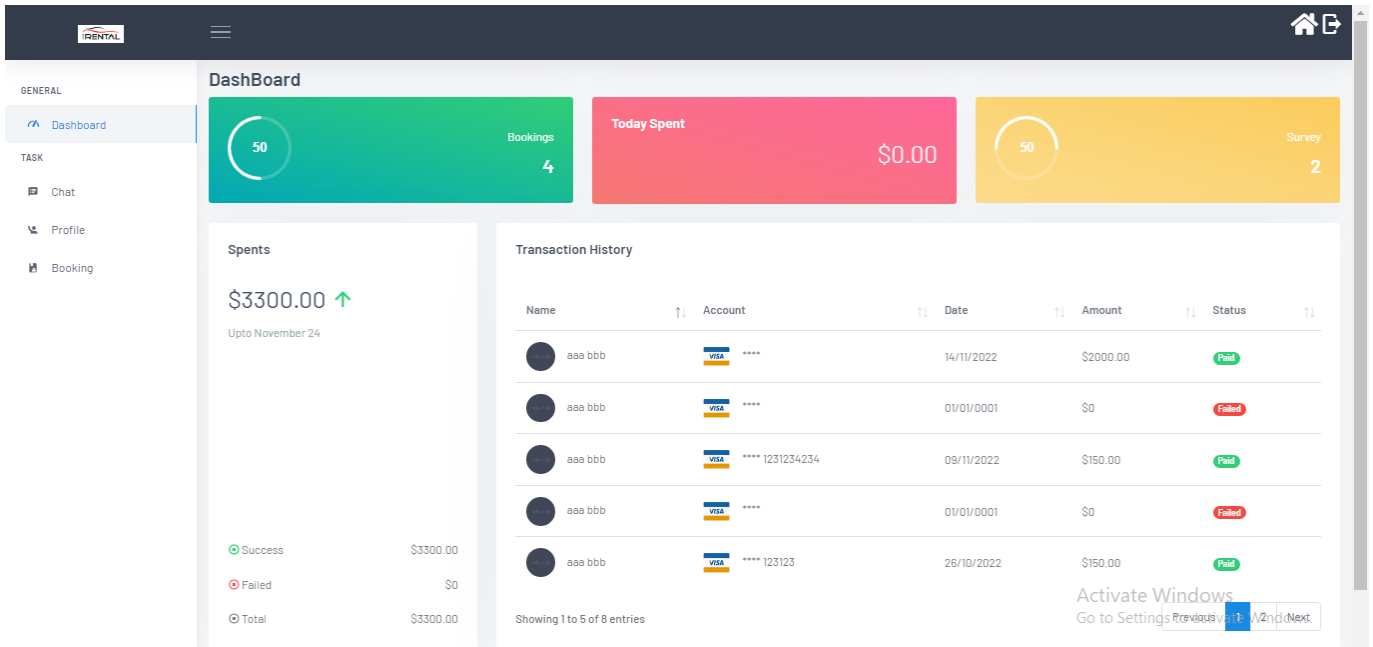


Figure 13: Admin Dashboard

## Category List Page

List of all Categories

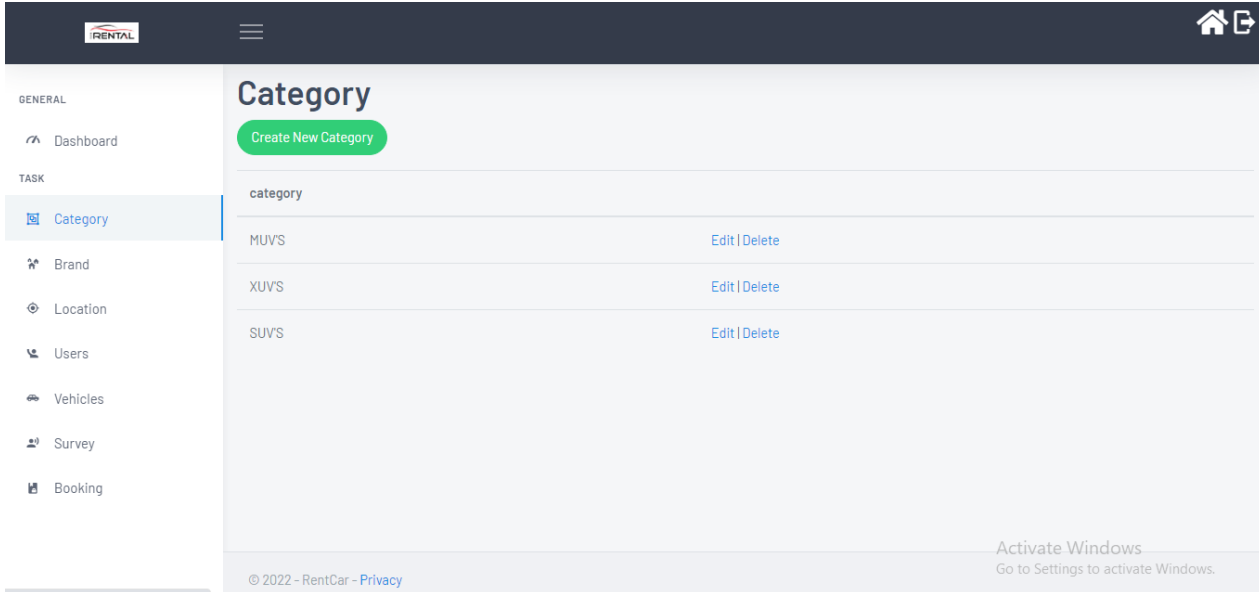


Figure 14: Category List

## Brand Creation Page

To Create a Brand

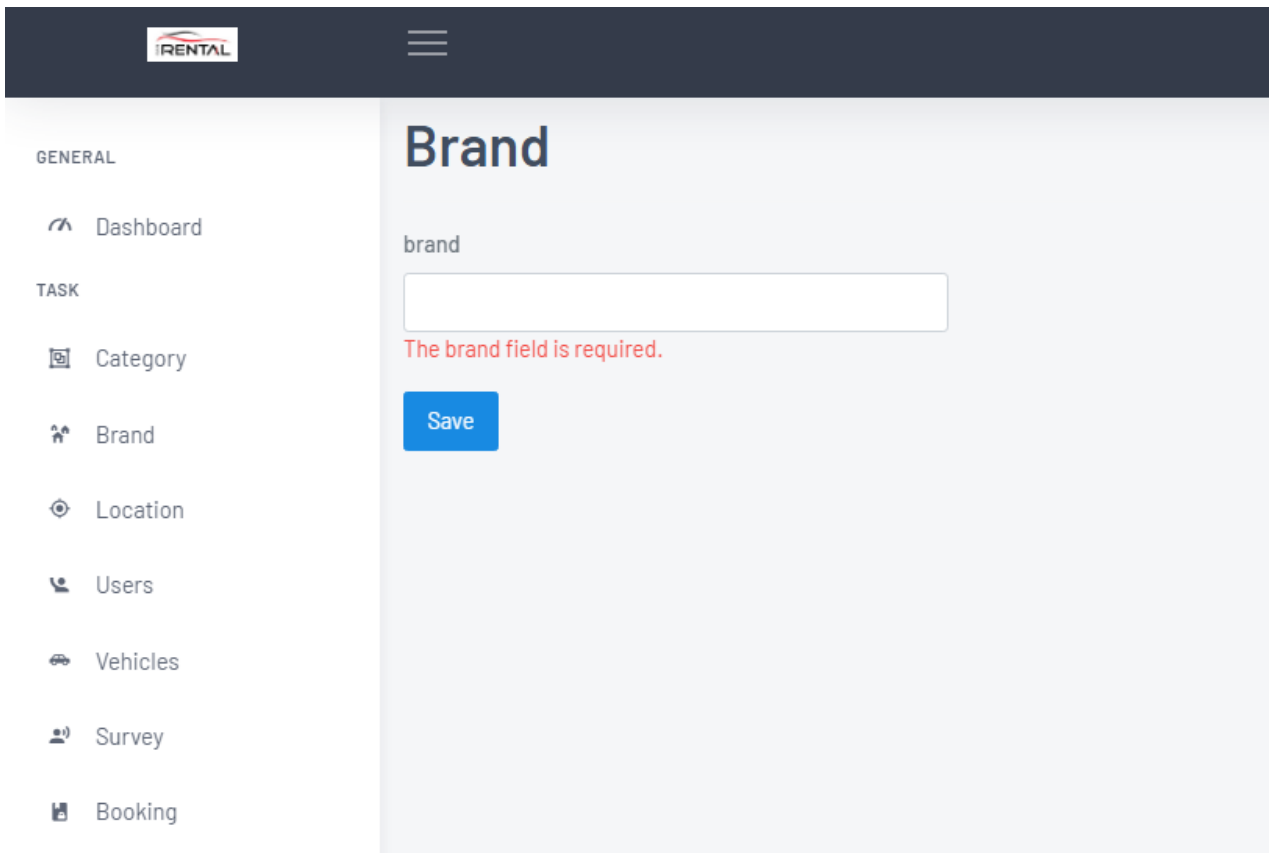


Figure 15: Brand Creation

## Location Edit

### Edit page of location

location

The location field is required.

latitude

longitude

description

The description field is required.

Save

© 2022 - RentCar - Privacy

Activate Windows  
Go to Settings to activate Windows.

Figure 16: Edit Location

## User Renter List

In user menu we have the list of owners and list of renters in two submenus

Owner Renter

First Name	Last Name	Company Name	Phone#	Address	D.O.B	Email	Status
AAA OWNER	BBB	ABC	1234	AAAAAA	01/01/0001	ow@a.c	Active Change Status

Figure 17: Registered Users List



## Vehicle Details

List of all vehicles, in that admin have a right to control the status by using change status to Activate / De-Activate the vehicle

Name	CategoryName	BrandName	LocationName	Doors / Seats	Navigation / AirCondition/ IsPremium	Fuel	Gear / Year	HourlyPrice / DailyPrice / MonthlyPrice	IsActive
ABC	MUVS	BMW	NY	4 / 4	Yes/Yes/Premium	Diesel	Automatic / 2022	10.00 / 0.00 / 0.00	IN Active <a href="#">Change Status</a>
ABC	MUVS	BMW	NY	4 / 4	Yes/Yes/Normal	Diesel	Automatic / 2022	10.00 / 50.00 / 150.00	Active <a href="#">Change Status</a>
X5	MUVS	BMW	NY	4 / 4	Yes/Yes/Premium	Petrol	Automatic / 2022	100.00 / 1000.00 / 10000.00	IN Active <a href="#">Change Status</a>
X3 New 2022	MUVS	BMW	NY	4 / 3	Yes/Yes/Premium	Diesel	Automatic / 2022	20.00 / 150.00 / 1000.00	IN Active <a href="#">Change Status</a>
New Vehicle	MUVS	BMW	NY	4 / 4	Yes/Yes/Premium	Petrol	Automatic / 2022	0.00 / 0.00 / 0.00	Active <a href="#">Change Status</a>

Figure 18: Vehicle List

## Survey Details

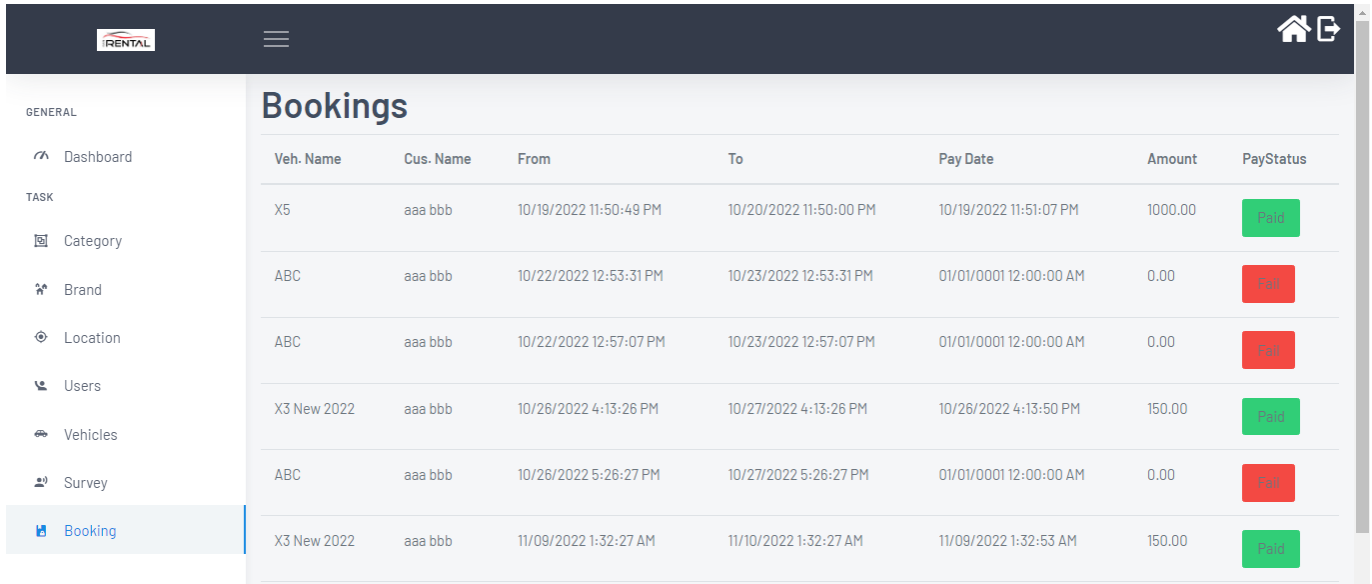
After the renter trip is completed, we provide the feedback survey section that information shows here.

rating of the vehicle?	you have your own car?	your reason?	location?	you compare cars?	Is Any change need?	EntryDate
5	Yes	travel	NY	Amount	-	10/26/2022 4:00:18 PM
5	Yes					11/14/2022 2:27:56 PM

Figure 20: Survey List

## Admin Booking Details

All booking details show to admin



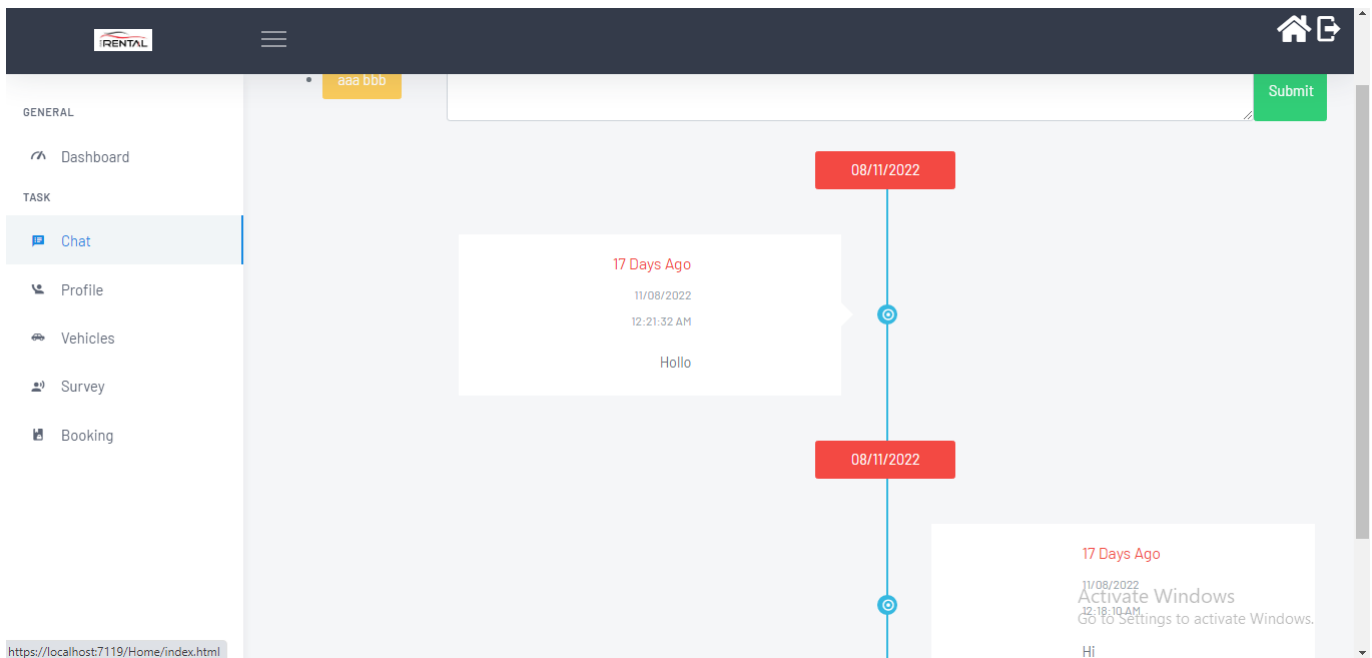
The screenshot shows the 'Bookings' section of an admin interface. On the left is a sidebar with navigation options: Dashboard, Category, Brand, Location, Users, Vehicles, Survey, and Booking (highlighted). The main area displays a table with columns: Veh. Name, Cus. Name, From, To, Pay Date, Amount, and PayStatus. The table contains six rows of booking data.

Veh. Name	Cus. Name	From	To	Pay Date	Amount	PayStatus
X5	aaa bbb	10/19/2022 11:50:49 PM	10/20/2022 11:50:00 PM	10/19/2022 11:51:07 PM	1000.00	Paid
ABC	aaa bbb	10/22/2022 12:53:31 PM	10/23/2022 12:53:31 PM	01/01/0001 12:00:00 AM	0.00	Fail
ABC	aaa bbb	10/22/2022 12:57:07 PM	10/23/2022 12:57:07 PM	01/01/0001 12:00:00 AM	0.00	Fail
X3 New 2022	aaa bbb	10/26/2022 4:13:26 PM	10/27/2022 4:13:26 PM	10/26/2022 4:13:50 PM	150.00	Paid
ABC	aaa bbb	10/26/2022 5:26:27 PM	10/27/2022 5:26:27 PM	01/01/0001 12:00:00 AM	0.00	Fail
X3 New 2022	aaa bbb	11/09/2022 1:32:27 AM	11/10/2022 1:32:27 AM	11/09/2022 1:32:53 AM	150.00	Paid

Figure 9: Booking List

## Chat

Discussion between renter and owner



The screenshot shows the 'Chat' section of the admin interface. The sidebar on the left includes: Dashboard, Profile, Vehicles, Survey, and Booking. The chat area shows a conversation with a customer named 'aaa bbb'. The chat history includes a date separator '08/11/2022', a message from the renter saying 'Hollo' on 11/08/2022 at 12:21:32 AM, and a message from the owner saying 'Hi' on 11/08/2022 at 12:18:10 AM. A 'Submit' button is visible in the top right corner of the chat area.

Figure 10: Chat History

## Survey Entry

After trip complete renter can provide feedback in survey

The screenshot shows the iRENTAL website's survey entry form. On the left is a navigation menu with 'GENERAL' (Dashboard) and 'TASK' (Chat, Profile, Booking). The main content area is titled 'Survey Form' and contains six questions:

1. Please provide the rating of the vehicle? (Dropdown menu with 'Very Good' selected)
2. Did you have your own car at that time? (Dropdown menu with 'Yes' selected)
3. What was your reason for searching a car? (Text input field with 'Your Answer')
4. Where was the car location that you were searching for? (Text input field with 'Your Answer')
5. What basis did you compare cars? (Text input field with 'Your Answer')
6. Is there anything you would change about the booking procedure? (Text input field with 'Your Answer')

A blue 'Submit' button is located at the bottom right of the form. A disclaimer states: 'We'll never share your email with anyone else.'

Figure 23: Survey Entry Form

## Search Details

Advanced search option has the filter like date, brand, category, location, and Name

The screenshot shows the iRENTAL website's search page. At the top, there is a navigation bar with 'Home', 'Category', 'Brands', 'Features', and a 'Dashboard' button. Below the navigation bar is a green banner with the text 'Make your Journey amazing & unique with Our Vehicles'. The search form includes two date pickers (11/24/2022 08:20 PM and 11/25/2022 08:20 PM), a 'Search Your Car' button, a 'BMW' dropdown menu, a 'Select Category' dropdown menu, and a 'Select Location' dropdown menu. Below the search form is a section titled 'Book your vehicles with your convenient Price' with the subtitle 'Vehicles List Based on Search Criteria'. The search results are displayed in a grid of cards. The first card shows a car image, a price range of '\$10.00 | 50.00 | 150.00', and features like 'AirConditioned' and 'Seats: 4'. The second card shows a red car image, a price range of '\$0.00 | 0.00 | 0.00', and features like 'AirConditioned', 'Seats: 4', and 'NY'. A Windows watermark is visible in the bottom right corner of the screenshot.

Figure 24: Search Page

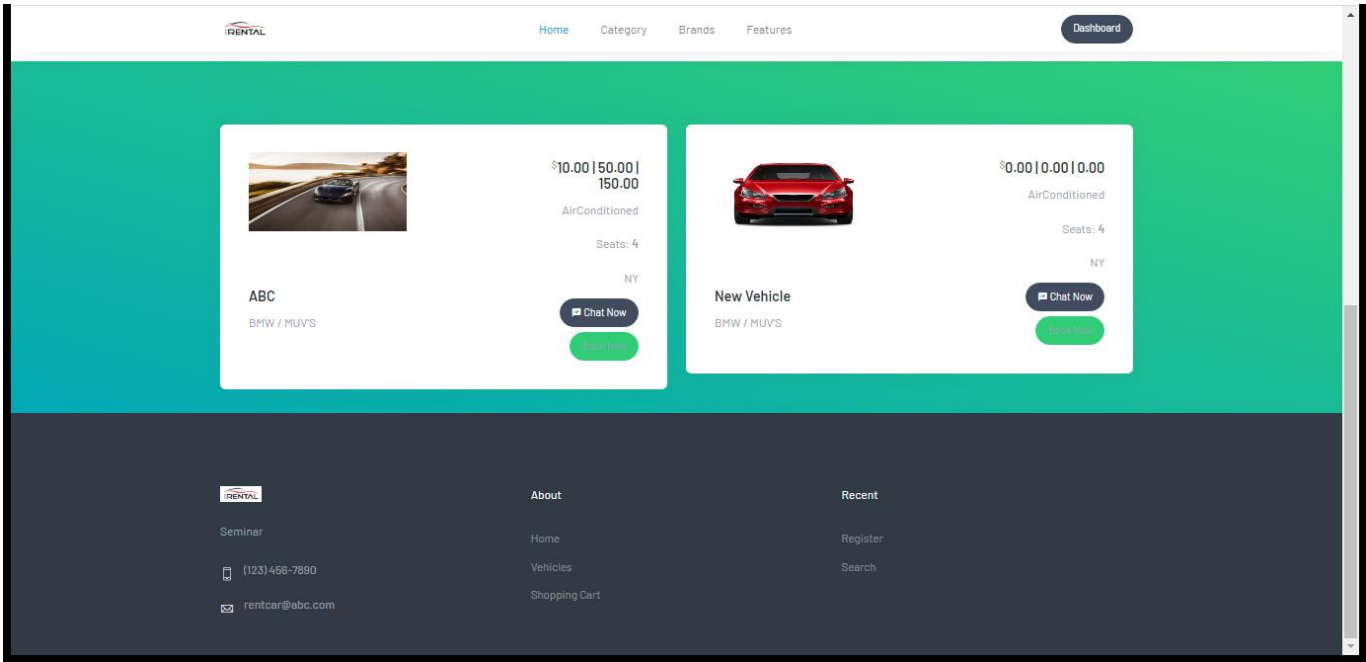


Figure 25: Search Vehicle List

**Booking Detail**

Complete Details of the vehicle selected

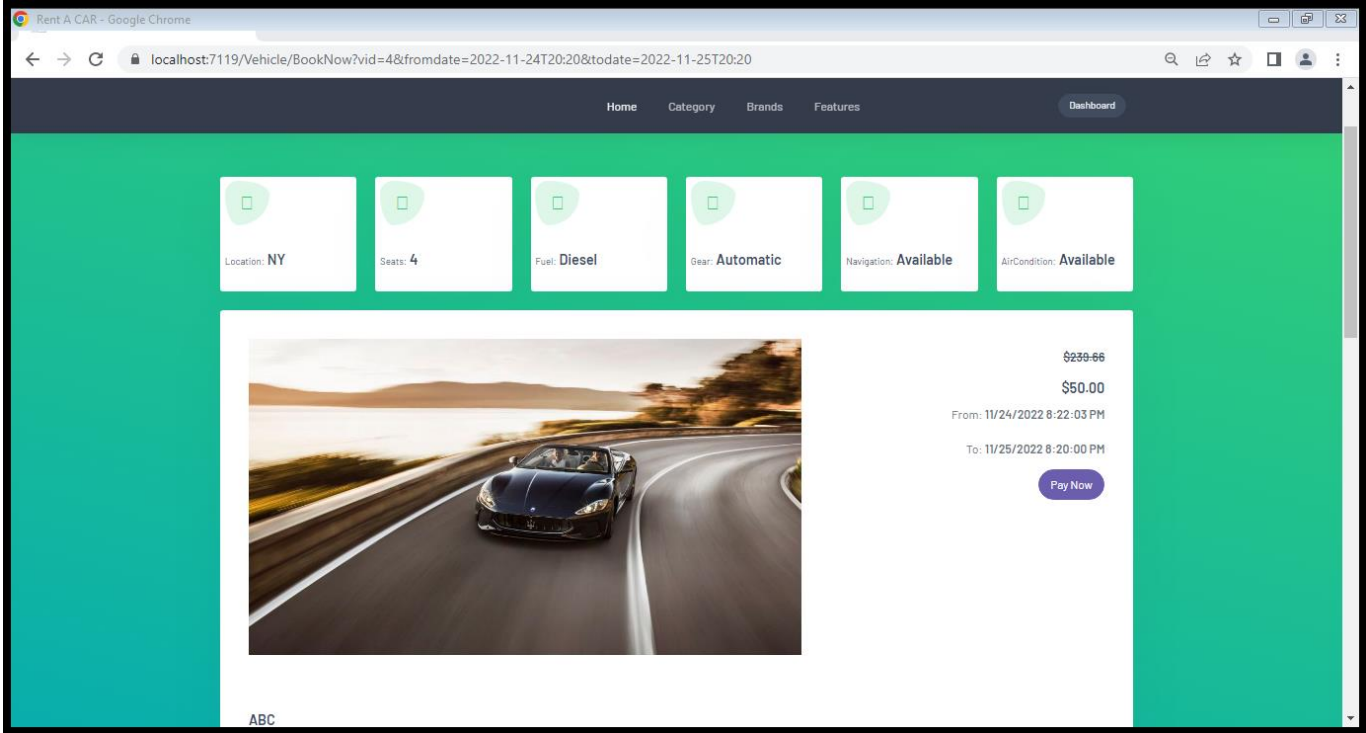


Figure 26: Booking Detail

## Payment Details

After booking conformed redirected to payment page

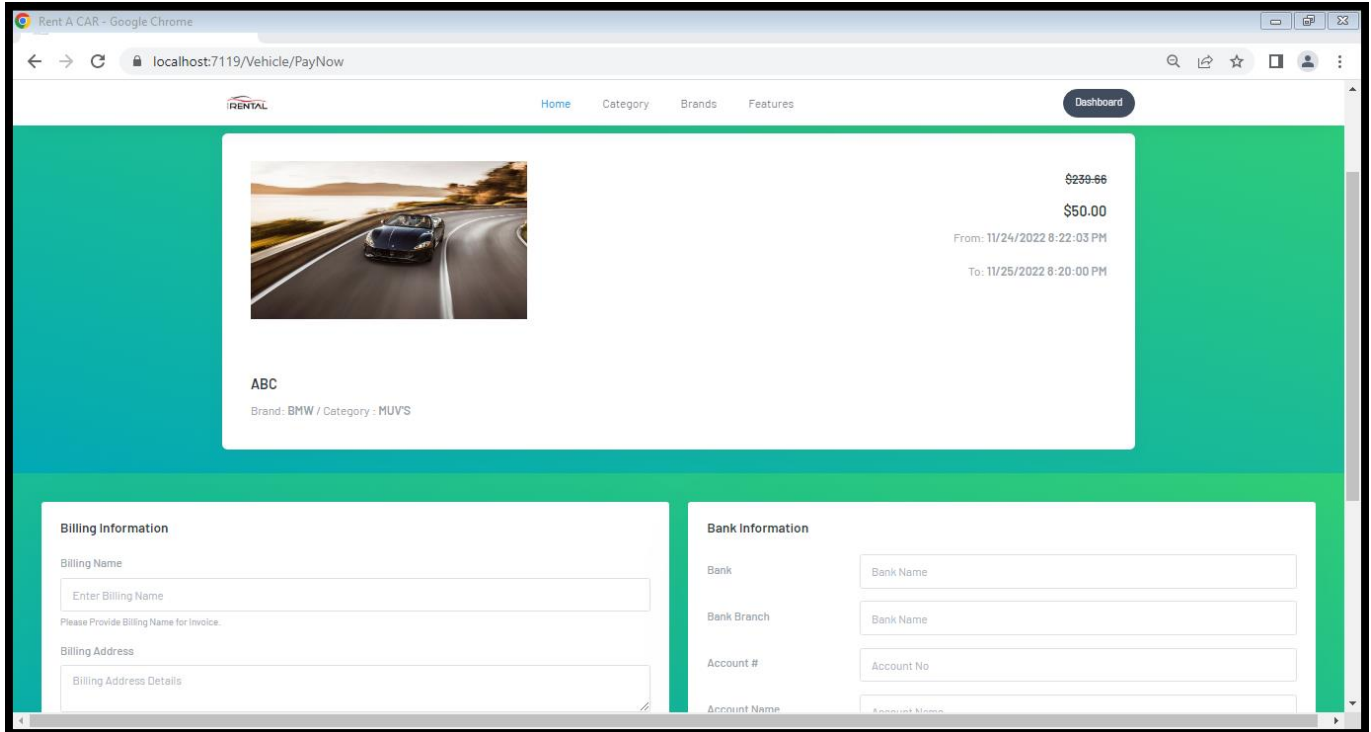


Figure 27: Payment Information

## 6 Design Units Impacts

Based on the findings of the surveys, rival service offers, literature, and best practice discoveries, it became evident that the new vehicle rental platform makes the process simpler and easier for clients. The time and money saved stand out among the various conveniences of this approach the most. The new method also always provides flexible service hours and availability, which was not possible with the prior system for renting cars.

### 6.1 Functional Area A/Design Unit A

#### 6.1.1 Functional Overview

The main purpose of functional areas is to ensure that all business activities are conducted efficiently. This is essential if the project is to achieve its aims and objectives. The database should be updated, and the customers

like vehicle info, user details and booking details will be securely stored in the database. The database will be secured with encryption, and only authentic users can access the website.

### **6.1.2 Impacts**

It helps us to verify the functionality of the application. Authentication of a user when they try to sign in into the system. They contain the goal, such as booking system, an online catalogue of the vehicle. It can also include things like approval of users and vehicles, workflows, and authorization levels.

This design unit and workflow process enables the computer working functions to adapt the system to the operational requirements and Maintenance requirements. The work process includes authentication, data update, user information update, Vehicle information create and update and all administrative process.

### **6.1.3 Requirements**

- The administrator performs the user authentication process
- login and registration not required to view this web application as a guest
- Admin should track vehicle owner, & Renter progress
- External interfaces and internal interfaces should be properly configured and dependable
- Data backup process also included for customer data protection
- The data security for the application must be used

## **7 Open Issues**

No Issues

## **8 Acknowledgements**

I would like to thank my professor Dr. Xin Chen who granted me the outstanding opportunities to prepare this excellent plan on the subject CPSC8985. This project supported us in preparing Research in the field of software engineering. We learned so many things from this project. It helped us to grow knowledge regarding research in technology and report preparation. I would like to thank my classmates and my parents who supported me to complete the project successfully.

## **9 References**

1. Image. System Architecture [https://en.wikiversity.org/wiki/Three-Tier\\_Architecture](https://en.wikiversity.org/wiki/Three-Tier_Architecture)
2. Lazov, "Profit management of Car Rental Service Companies", European Journal of Operational Research, vol. 258, no. 1, pp. 307-314, 2017
3. Falah Y H Ahmed; Ewan Bin Hazlan; Muhammad Irsyad Abdulla. "Enhancement of Mobile-Based Application for Vehicle Rental" 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE).

4. S. R. Manalu, A. Wibisurya, N. Chandra, and A. P. Oedijanto, "Development and evaluation of mobile application for room rental information with chat and push notification", 2016 International Conference on Information Management and Technology (ICIMTech), pp. 7-11, November 2016.
5. MVC Helper. <https://help.grapecity.com/componentone/PDF/MVC/c1mvchelpers.pdf>
6. Image. Server Architecture. <https://mindmajix.com/what-is-ssrs>

## 10 Appendices

```
2 references
public IActionResult Location() { return View(_db.Locations); }
0 references
public IActionResult LocationCU(int id, Location Parameter)
{
    if (id > 0 && Parameter.location==null)
    {
        var data = _db.Locations.Find(id);
        return View(data);
    }
    if (Parameter.location!= null)
    {
        if (Parameter.Id > 0)
            _db.Update(Parameter);
        else
            _db.Add(Parameter);
        var result = _db.SaveChanges();
        if (result > 0)
            return RedirectToAction(nameof(Location));
    }
    return View(new Location());
}
```

Figure 28: Location CRUD code

```

public IActionResult Chat(ChatView chat)
{
    var cuserid = this.User.FindFirstValue(ClaimTypes.NameIdentifier);
    if (string.IsNullOrEmpty(cuserid)) return RedirectToAction(nameof(SignIn));
    if (!string.IsNullOrEmpty(chat.ToUserId))
    {
        chat.Chats = _db.Chats.Where(x => x.FromUser == chat.ToUserId).ToList();
        if (chat.Chats == null) chat.Chats = new List<Chat>(); chat.Chats.AddRange(_db.Chats.Where(x => x.ToUser == chat.ToUserId).ToList());
        chat.Chats = chat.Chats.OrderByDescending(x => x.Date).ToList();
        chat.Chats.ForEach(c =>
        {
            var diff = DateTime.Now.Subtract(c.Date); var diffval = "Just Now";
            if (diff.TotalMinutes > 1)
                if (diff.TotalMinutes > 60)
                    if (diff.TotalHours > 24)
                        if (diff.TotalDays > 30)
                            if (diff.TotalDays > 365)
                                diffval = "Year Ago";
                            else
                                diffval = Convert.ToInt32((diff.TotalDays / 30)).ToString() + " Months Ago";
                        else
                            diffval = Convert.ToInt32(diff.TotalDays).ToString() + " Days Ago";
                    else
                        diffval = Convert.ToInt32(diff.TotalHours).ToString() + " Hours Ago";
                else
                    diffval = Convert.ToInt32(diff.TotalMinutes).ToString() + " Minutes Ago";
            c.Difference = diffval;
        });
        var touser = _db.Users.FirstOrDefault(x => x.Id == chat.ToUserId); chat.ToUserName = touser != null ? touser.FirstName + " " + touser.LastName;
    }
    chat.ThisUserId = cuserid;
    if (cuserid != null)
    {
        List<string> userlist = _db.Chats.Where(x => x.FromUser == cuserid).Select(x => x.ToUser).Distinct().ToList();
        if (userlist == null) userlist = new List<string>();
        userlist.AddRange(_db.Chats.Where(x => x.ToUser == cuserid).Select(x => x.FromUser).Distinct().ToList());
        if (userlist.Any())
        {
            var dbUser = _db.Users.ToList();
            var ChatUsers = dbUser.Join(userlist.Distinct(), u => u.Id, c => c, (u, c) => new MUser { Id = u.Id, FirstName = u.FirstName, LastName =
            chat.ChatUsers = ChatUsers.ToList();
        }
    }
    return View(chat);
}

```

Figure 29: Chat Controller



```
15 references
public class VehicleViewModel : Vehicle
{
    9 references
    public string CategoryName { get; set; }
    9 references
    public string BrandName { get; set; }
    8 references
    public string LocationName { get; set; }
}
10 references
public class Vehicle
{
    22 references
    public int Id { get; set; }
    20 references
    public string Name { get; set; }
    13 references
    public int CategoryId { get; set; }
    13 references
    public int BrandId { get; set; }
    11 references
    public int Doors { get; set; } = 4;
    14 references
    public int Seats { get; set; } = 4;
    11 references
    public bool Navigation { get; set; } = true;
    13 references
    public bool AirCondition { get; set; } = true;
    12 references
    public Fuel Fuel { get; set; }
    12 references
    public Gear Gear { get; set; }
    11 references
    public int Year { get; set; } = 2022;
    14 references
    public decimal HoulyPrice { get; set; }
    16 references
    public decimal DailyPrice { get; set; }
    16 references
    public decimal MonthlyPrice { get; set; }
    14 references
    public bool IsActive { get; set; } = false;
    14 references
    public int LocationId { get; set; }
    11 references
    public bool IsPremium { get; set; }
    10 references
    public string OwnerId { get; set; }
    [NotMapped]
    4 references
```

Figure 30: Vehicle Model

```

RentCar - Microsoft Visual Studio
Chat.cshtml | Dashboard.cshtml | Vehicle.cs | SiteIndex.cs | MUser.cs | SurveyList.cshtml | VehicleController.cs | HomeController.cs
1 @model Dashboard
2 @{
3     ViewData["Title"] = "Dashboard";
4     //Layout = null;
5 }
6
7 <h3>Dashboard</h3>
8 <div class="row">
9     <div class="col-xl-4">
10        <div class="card-box widget-chart-one gradient-success bx-shadow-lg">
11            <div class="float-left dir="ltr">
12                <input data-plugin="knob" data-width="80" data-height="80" data-linecap="round"
13                    data-fgColor="#ffffff" data-bgcolor="rgba(255,255,255,0.2)" value="@Model.Card1Per" data-skin="tron" data-angleOffset="180"
14                    data-readonly=true data-thickness="1" />
15            </div>
16            <div class="widget-chart-one-content text-right">
17                <p class="text-white mb-0 mt-2">Bookings</p>
18                <h3 class="text-white">@Model.Card1Value</h3>
19            </div>
20        </div> <!-- end card-box -->
21    </div> <!-- end col 1 -->
22
23
24    <div class="col-xl-4">
25        <div class="card-box gradient-danger bx-shadow-lg pb-0">
26            @if (User.IsInRole("Renter"))
27            {
28                <h4 class="header-title text-white">Today Spent</h4>
29            }
30            else
31            {
32                <h4 class="header-title text-white">Daily Sales</h4>
33                <div class="mb-3 mt-2 text-right">
34                    <p class="text-white">March 26 - April 01</p>
35                </div>
36                <h2 class="font-weight-light text-white">@$Model.Card2Value</h2>
37            </div>
38        </div> <!-- end card-box -->
39    </div> <!-- end col 2 -->
40
41
42    <div class="col-xl-4">
43        <div class="card-box widget-chart-one gradient-warning bx-shadow-lg">
44            <div class="float-left dir="ltr">
45                <input data-plugin="knob" data-width="80" data-height="80" data-linecap="round"
46                    data-fgColor="#ffffff" data-bgcolor="rgba(255,255,255,0.2)" value="@Model.Card3Per" data-skin="tron" data-angleOffset="270"
47                    data-readonly=true data-thickness="1" />
48            </div>
49        </div>
50    </div>
51 </div>
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Figure 31: Dashboard Razor View Code

```
Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
ApplicationDbContext.cs Chat.cshtml DashBoard.cshtml Vehicle.cs SiteIndex.cs
RentCar
  RentCar.Data.ApplicationDbContext
1 using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
2 using Microsoft.EntityFrameworkCore;
3 using RentCar.Models;
4
5 namespace RentCar.Data
6 {
7     27 references
8     public class ApplicationDbContext : IdentityDbContext<MUser>
9     {
10         0 references
11         public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
12             : base(options)
13         {
14             10 references
15             public DbSet<Category> Categories { get; set; }
16             10 references
17             public DbSet<Brand> Brands { get; set; }
18             10 references
19             public DbSet<Location> Locations { get; set; }
20             15 references
21             public DbSet<Vehicle> Vehicles { get; set; }
22             5 references
23             public DbSet<Booking> Bookings { get; set; }
24             3 references
25             public DbSet<Payment> Payments { get; set; }
26             2 references
27             public DbSet<Comments> Comments { get; set; }
28             3 references
29             public DbSet<SurveyDetail> Surveys { get; set; }
30             5 references
31             public DbSet<Chat> Chats { get; set; }
32         }
33     }
34 }
```

Figure 32: Db Context