2024

Lanka Nippon Biztech Institute

GoCarNow Car Rental Project

Name : Tharuka Dilshan

Index Number : UGC0122027

Batch : SE01

Dedication

This project is dedicated to my family for their endless support and understanding. Their encouragement and belief in my abilities have been my guiding light, propelling me through challenges and inspiring me to achieve excellence. I am grateful for their unwavering love and patience, which provided the foundation for this endeavor. I also dedicate this work to my mentors and lecturers whose guidance and wisdom have shaped my knowledge and skills. Their mentorship has been invaluable, shaping not just this project but also my growth as a learner and professional.

Acknowledgments

I extend my deepest gratitude to everyone who contributed to the success of this project. Special thanks to my family for their unwavering support and understanding. I am grateful to my mentors and advisors for their guidance and valuable insights. I acknowledge the assistance and collaboration of [mention any specific individuals, organizations, or teams]. I appreciate the efforts of all contributors, whose dedication and hard work made this project possible. Your support, encouragement, and belief in this endeavor have been invaluable. Thank you for being part of this journey and for making it a fulfilling experience.

Table of Contents

List of Figures

List Of Tables

List of Acronyms

DDL - Data Definition Language

ER - Entity Relationship

SQL - Structured query language

DML - Data Manipulation Language

1. Chapter 01: Introduction

1.1. Introduction

In Chapter 1, we delve into the dynamic landscape of the car rental industry, highlighting the challenges faced by traditional systems and the motivation behind our project. We aim to revolutionize the industry by fostering inclusivity, affordability, and a diverse range of vehicle options. This chapter outlines our objectives, scope, and methodology in creating an enhanced car rental system that empowers small businesses, and individual owners while offering a seamless user experience through web and mobile applications."

1.2. Introduction System

The car rental industry plays a pivotal role in the modern transportation area, the car rental industry stands as a key player, providing unparalleled flexibility and convenience to individuals and businesses. However, with this current rental system challenges, renting vehicles can be a pricey affair, and customers often find themselves stuck with a limited selection of vehicles, utilization optimization, and continuous improvement of customer experiences. In existing car rental systems, big companies rule the scene. They have set up online platforms and mobile applications that make finding, picking, and booking a rental car easy. When you hop onto these platforms, you can check out the cars they have, choose the one you want, and make a reservation. But guess what? We are shaking things up. Our system is all about making it easier for the small rental company and individual car owners to be a part of our system. Furthermore, Drivers can be part of our system. We want everyone to have a chance to get in on the action and offer their cars for rent.

1.3. Motivation

The motivation behind our project is rooted in the desire to address the drawbacks of the existing car rental system. While big companies currently dominate, we aim to revolutionize the industry by creating a more inclusive platform. We want to make renting a car affordable, diverse, and accessible to everyone, from small rental companies to individual car owners and even drivers.

1.4. Aim

The aim of the enhanced car rental system is to disrupt the traditional car rental industry by fostering inclusivity and affordability. The project empowers small businesses, individual car owners, and drivers, challenging the dominance of major companies. It seeks to provide customers with a diverse range of vehicle choices while addressing issues like high costs and limited selection. The project aims to redefine the car rental experience, making it more accessible.

1.5. Objectives of the Project

- **Enhanced Vehicle Selection**: Expand the range of available vehicles, providing customers with a broader and more personalized choice.
- **Affordability:** Tackle the issue of high costs associated with renting vehicles, making it more budget-friendly for a wider audience.
- **Diversify Options:** Break the monopoly of big companies by enabling small rental businesses, individual car owners, and even drivers to participate in our system.
- **Optimized Utilization:** Implement systems to ensure efficient utilization of vehicles, benefitting both owners and renters.
- **Administrative Control:** To empower administrators with tools for efficient management, including attraction updates and user feedback analysis.

1.6. Scope of the Project

- Creation of a web and mobile application for seamless user interaction.
- Development of a comprehensive admin panel for managing listings, users, and approvals.
- Implementation of user authentication and document verification for compliance.
- Keep users informed with timely notifications and alerts.

1.7. Methodology

1.8. Summary

In summary, Chapter 1 introduces the car rental industry's challenges and the motivation behind the project. It outlines the aim, objectives, scope, and methodology of the enhanced car rental system, highlighting the project's disruptive potential and commitment to inclusivity and affordability.

2. Chapter 02: Analyze

2.1. Introduction

In Chapter 2, we conduct a thorough review of similar systems in the car rental industry, drawing insights and references to inform the development of our GoCarNow Vehicle Rental System. We then proceed to identify key functional requirements, including user registration, availability checks, vehicle listing, booking processes, pricing structures, and administrative functionalities. Additionally, non-functional requirements such as vehicle maintenance services and industry impact are outlined to define the project's scope. Lastly, we select the Waterfall process model for its structured approach, aligning with our clear plan and detailed requirements analysis, ensuring a methodical progression toward project completion.

2.2. Review Similar System with References

When reviewing similar systems in the vehicle rental domain, it's essential to consider various aspects such as user experience, functionality, scalability, and technological stack. Systems like Car Rental System (Car Booking.com) have best practices and innovative features. we aim to incorporate successful strategies and unique functionalities into our GoCarNow vehicle rental system, ensuring competitiveness and customer satisfaction.

2.3. Identifying Functional Requirements

- **User Registration**
  - Customers can be registered with the platform.
  - Drivers can be registered with the platform.
  - Vehicle Owner can be registered with the platform.
- **Availability Check**
  - Users should be able to see the status of a vehicle (available or booked) before initiating the booking process.
- **Listing Vehicle**
  - Owners and rental businesses must authenticate themselves before listing vehicles.
  - Individual car owners can list their vehicles with detailed information (model, year, color, mileage, and features).
  - The system should support detailed descriptions and images to provide users with a complete overview.
- **Booking Vehicle**
  - Users must be authenticated before initiating the booking process.
  - Users can browse the platform, select vehicles based on preferences, and make bookings.
  - The system generates a booking request that includes user details, vehicle information, and rental duration.
- **Pricing and Billing**
  - Owners set pricing details for each vehicle, including Daily, and Monthly rates.
- **Admin Dashboard**
  - The admin can Add a Vehicle Rental Company to the system.
  - The admin can View a Vehicle Rental Company.
  - The admin can Update a Vehicle Rental Company.
  - The admin can Delete a Vehicle Rental Company.

2.4. Identifying Non-Functional Requirements

- **Vehicle Maintenance Services:** Providing detailed vehicle maintenance services, such as repairs, inspections, or routine maintenance beyond predictive maintenance for optimizing utilization.
- **Vehicle accidents and Vehicle Breakdowns:** Addressing vehicle breakdowns on the road, including roadside assistance services, falls outside the defined scope of the enhanced car rental system project.
- **Complete Industry Overhaul:** Our project does not aim to completely overturn the existing car rental industry but rather introduces enhancements to foster inclusivity.

2.5. Identifying Suitable Process Model

I chose the Waterfall method for the GoCarNow Vehicle Rental System project because I already have a clear and detailed plan. I have already identified the functional requirements of my project. This method works step by step, where I finish one thing before moving on to the next. Since has exactly what we want in each phase, it helps me stay organized.

2.6. Summary

In summary Chapter 2, we review existing car rental systems, identify functional requirements like user registration and vehicle listing, outline non-functional requirements such as maintenance services, and choose the Waterfall process model for our GoCarNow Vehicle Rental System project due to its structured approach and alignment with our detailed plan.

3. Chapter 03: Design

3.1. Introduction

In Chapter 3, we delve into the design aspects of GoCarNow, a revolutionary platform in the car rental industry. This chapter encompasses a holistic view of the system, from introducing our clients to detailing the scope, use case diagram, ER diagram, class diagram, sequence diagram, state chart diagram, activity diagram, and normalization process. Through this comprehensive exploration, we aim to provide a clear blueprint of the GoCarNow system's design and functionality.

3.2. Introduction to Clients

Welcome to the future of car rentals! In an industry dominated by big players, we're here to revolutionize the game and empower small rental companies, individual car owners, and drivers like never before.

At GoCarNow we've built a groundbreaking platform that levels the playing field, making it effortless for anyone to participate in the car rental ecosystem. Whether you're a small rental agency with a handful of vehicles or an individual looking to capitalize on your car's idle time, our system opens doors for everyone.

Gone are the days of limited choices and sky-high prices. Our platform seamlessly connects renters with a diverse range of vehicles, all while prioritizing affordability and convenience. Want to rent out your car when it's not in use? You can do that. Need a vehicle for a weekend getaway?

Join us in reshaping the car-rental landscape. Whether you're looking to generate extra income from your vehicle or find a flexible rental solution, GoCarNow is your trusted partner.

3.3. Scope Identification

- Development of a user-friendly online platform and Web application accessible to customers, small rental companies, individual car owners, and drivers.
- Integration of features that enable small businesses, individual owners, and drivers to list their vehicles for rent, expanding the pool of available cars for customers.

3.4. Use Case Diagram

A class diagram is a fundamental aspect of object-oriented modeling, providing a visual representation of the structure and relationships within a system. It embodies the blueprint of a software application by illustrating the classes, attributes, operations, and associations that define the system's objects and their interactions.

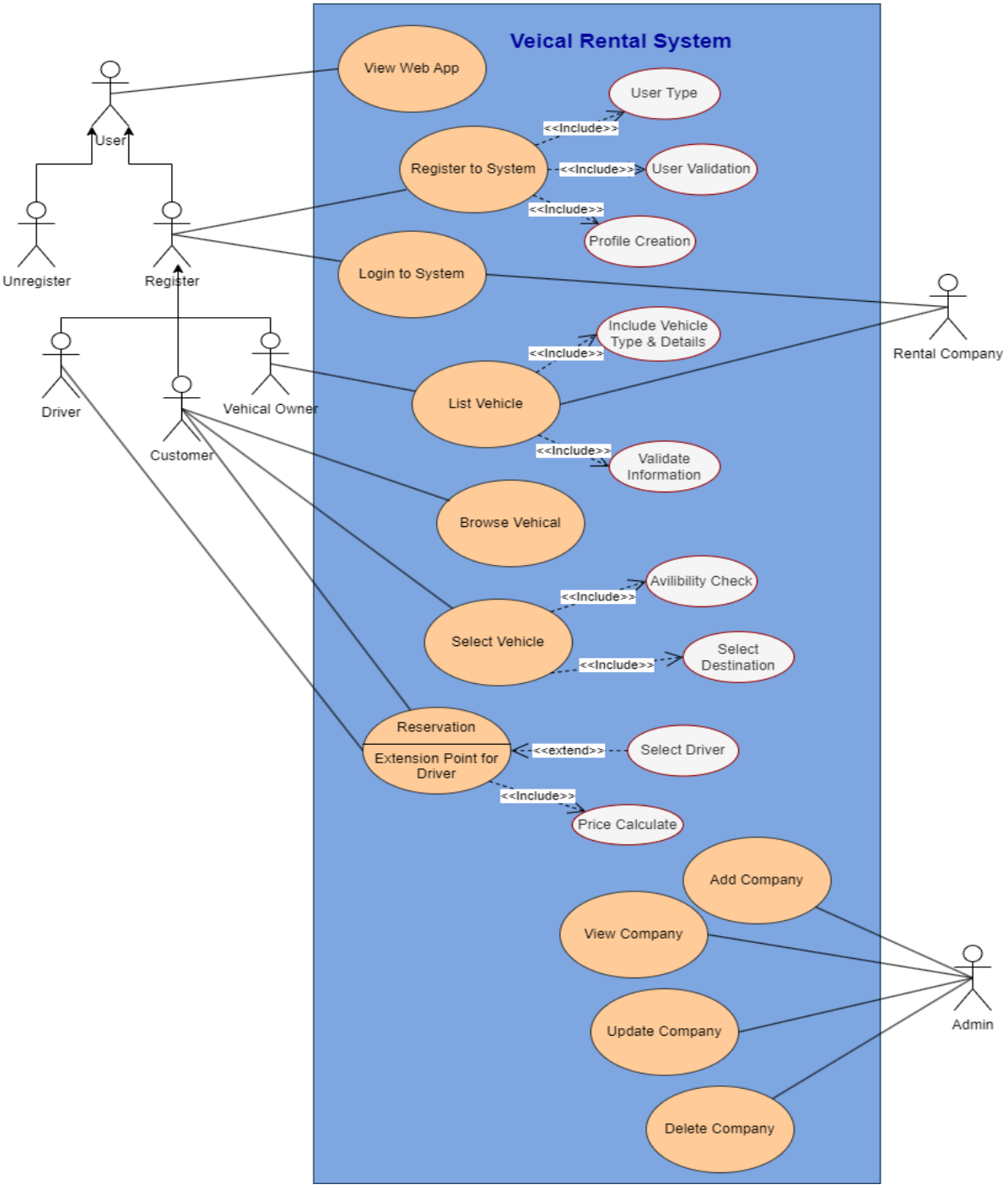


Figure 3.1: Use Case Diagram of GoCarNow

The above diagram shows the use case diagram of the system(Figure 3.1). Use case diagram is a graphical representation of the interactions between a system and its users. In the context of a vehicle rental system, the use case diagram would show how users can interact with the system to rent vehicles.

3.4.1. Identification of Roles and Cases

User:

- Unregistered User: This refers to individuals who have not yet created an account or registered with the system. Unregistered users typically have limited access to functionalities and may only be able to view basic information or browse the system.
- Registered User: Once users sign up and create an account, they become registered users. Registered users have more privileges and access to additional features compared to unregistered users. They can typically perform actions such as logging in, managing their profiles, and accessing personalized services.

Registered User (inherits from User):

- Driver: Driver may have access to features such as vehicle reservation driver profiles, only one Driver can a part one Reservation and only one Driver can part of selecting Vehicle Process.
    - **Relationship:**
        - **Driver to Reservation:** One to One Relationship
        - **Driver to Vehicle:** One to One Relationship.
- Customer: These users are registered and primarily interact with the system to rent vehicles. They can browse available vehicles, can select one Vehicle and can make one reservation for a day.
    - **Relationship:**
        - **Customer to Vehicle:** One to One Relationship.
        - **Customer to Reservation:** One to One Relationship.
- Vehicle Owner: Users who own vehicles and have registered them within the system. They may have access to features for managing their vehicle listings, availability, pricing, and other related aspects.
    - **Relationship:**
        - **Vehicle Owner to Vehicle:** Many to Many Relationship

Rental Company:

- Company who owns vehicles and have registered by Admin to the system. They may have access to features for managing their vehicle listings, availability, pricing, and other related aspects.
    - **Relationship:**

        - **Company to Admin**: Many to One Relationship
        - **Company to Vehicle:** Many to Many Relationship

Admin:

- Administrative users have special privileges and responsibilities within the system. Admin can add company, update Company and Delete Company.
    - **Relationship:**
        - **Admin to Company:** One to Many Relationship

3.4.2. Identification of Dependencies

- **Customer Login (Dependent):** The user needs to be logged in to have a reservation history and access to specific vehicles.
- **List Available Vehicles (Dependent):** The system needs to display available vehicles based on the renter's criteria (dates).
- **Vehicle Details (Dependent):** The user might want to view details of a particular vehicle before making a reservation.

3.5. ER Diagram

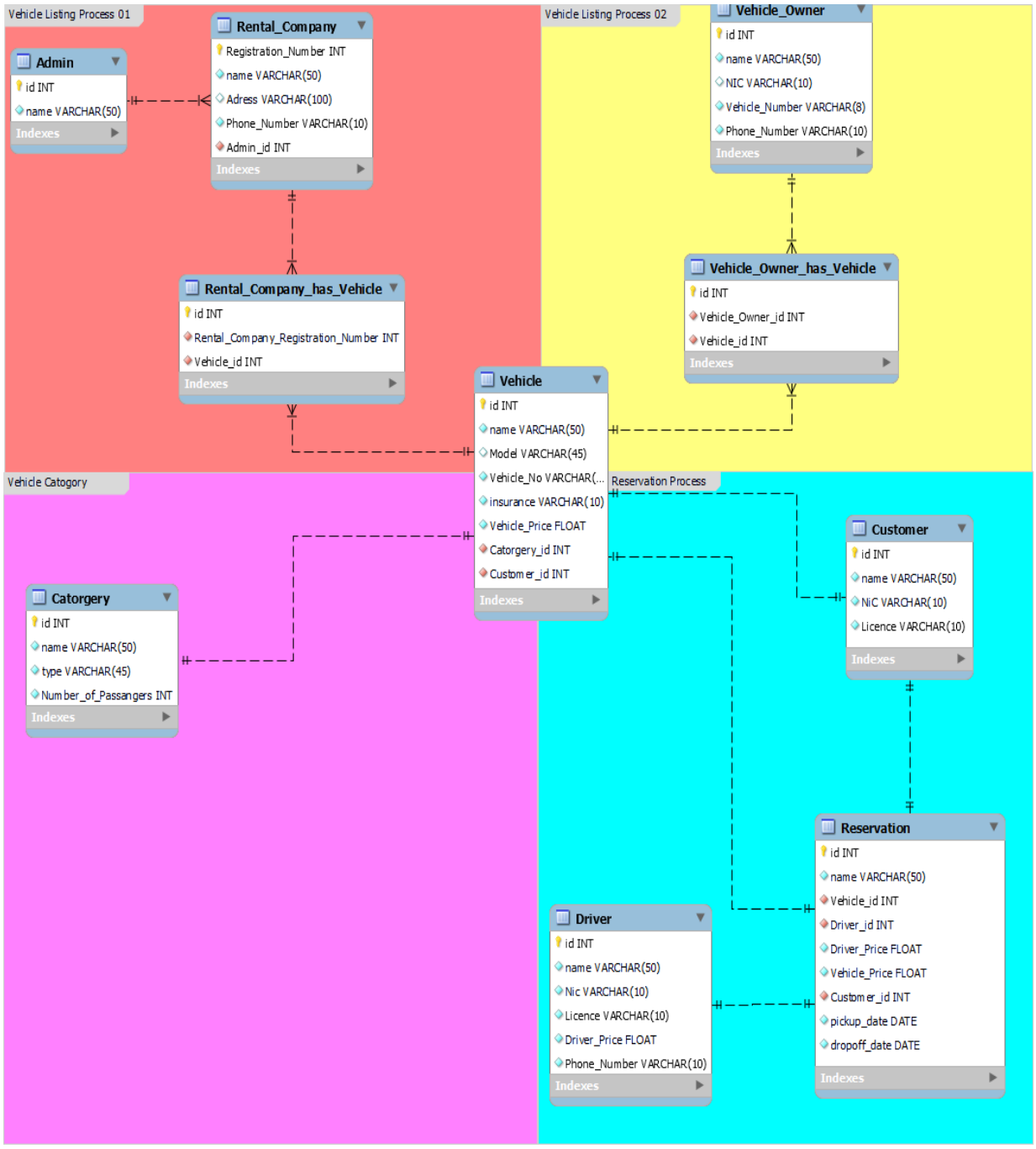


Figure 3.2: ER Diagram of GoCarNow

The above diagram shows the ER diagram of the system(Figure 3.2). An ER diagram visually depicts the entities, or key elements, within a system and the relationships that connect them. In this context, the entities represent aspects crucial to vehicle rentals, such as Customers, Vehicles, and Rental Companies. The diagram will illustrate how these entities interact, providing a clear understanding of the system's data structure.

3.5.1. Identification of Entities

- The ER diagram you sent shows the following entities:
  - Rental Company
  - Customer
  - Vehicle Owner
  - Driver
  - Vehicle
  - Vehicle Category
  - Reservation

3.5.2. Justification of Attributes

Admin

Attributes:

- id (INT): Unique identifier for each admin.
- name (VARCHAR(50)): The administrator's name.

Vehicle Owner

Attributes:

- id (INT): Unique identifier for a vehicle owner.
- name (VARCHAR(50)): The owner's name.
- Address (VARCHAR(100): The owner's address. It relevant for communication, and record-keeping.
- NIC (VARCHAR(10)): National Identity Card number is a common identifier used for verification and record-keeping.
- Phone Number (VARCHAR(10)): The owner's contact number. It allows for contacting the owner for communication purposes.

Rental Company

Attributes:

- id (INT): Unique identifier for the rental company.
- Registration Number (INT): The company's registration number represent the company's official registration number for legal purposes.
- name (VARCHAR(50)): The company's name.
- Address (VARCHAR(100): The company's address.
- Phone Number (VARCHAR(10)): The company's contact number. It allows for users to contact the company for inquiries or reservations.

Vehicle

Attributes:

- id (INT): Unique identifier for a vehicle.
- Vehicle Number (VARCHAR(8)): Vehicle's registration/license plate number it crucial for identification and verification purposes.
- name (VARCHAR(50)): This attribute provides additional details about the specific vehicle model, enhancing clarity.
- Model(VARCHAR(45)): This helps identify the vehicle's model, providing basic information to users.
- Insurance (VARCHAR(10)): Presumably indicates insurance status: 'Yes' or 'No'.
- Vehicle Price (FLOAT): The price associated with the vehicle.
- Category id (INT): This foreign key establishes a relationship with the Vehicle Category entity, enabling categorization and filtering of vehicles based on type.

Customer

Attributes:

- id (INT): Unique identifier for a customer.
- name (VARCHAR(50)): The customer's name.
- NIC (VARCHAR(10)):)): National Identity Card number is a common identifier used for verification and record-keeping.
- License (VARCHAR(10)): The customer's driver's license number is likely required to verify their eligibility to vehicles Reservation.

3.5.3. Identification of Relationships

- **Vehicle Owner - Vehicle (One-to-Many):** A vehicle owner can have multiple vehicles, but a vehicle can belong to only one owner.
- **Rental Company - Vehicle (One-to-Many):** A rental company can have multiple vehicles, but a vehicle can only belong to one rental company.
- **Vehicle - Vehicle Category (Many-to-One):** Many vehicles can belong to a single category, but a vehicle can only be part of one category.
- **Customer - Reservation (One-to-Many):** A customer can make multiple reservations, but a reservation is made by a single customer.
- **Reservation - Driver (One-to-One):** A reservation might optionally be assigned one driver

3.6. Class Diagram

A class diagram is a type of UML diagram that shows the relationships between classes and objects in a system. In the context of a vehicle rental system, a class diagram would show the different classes of things involved in the system

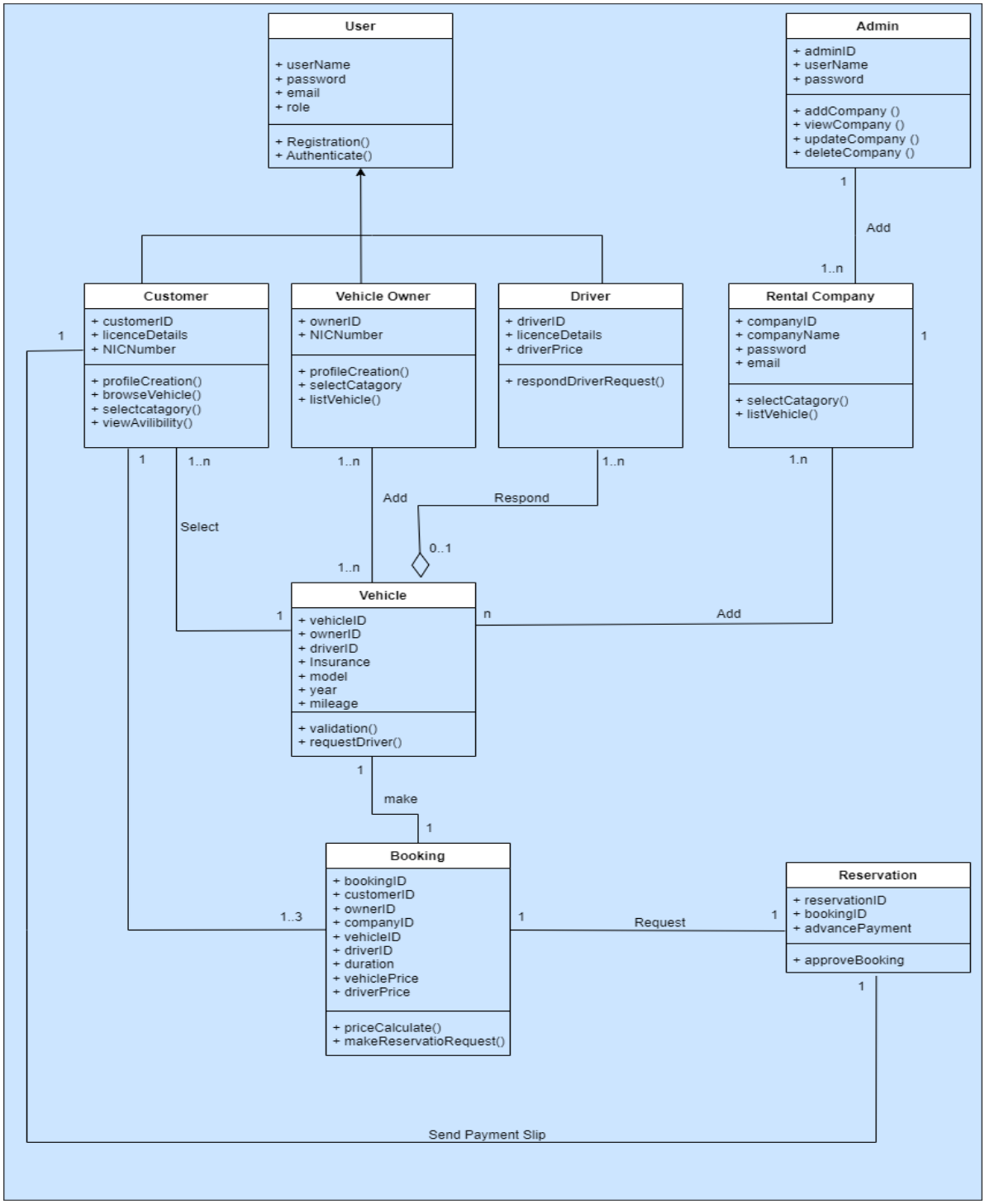


Figure 3.3: Class Diagram of GoCarNow

The above class diagram(Figure 3.3) specifically focuses on a system called GoCarNow. It shows the different classes that are involved in the system, including User, Admin, Customer, VehicleOwner, Driver, Company, Vehicle, Booking, and Reservation. The diagram also shows the relationships between these classes. For example, the diagram shows that a User can be a Customer, VehicleOwner, or Driver. It also shows that a Customer can make a Booking for a Vehicle.

3.7. Sequence Diagram

Sequence diagrams are a powerful tool used in software development to visualize the message flow between different parts of a system. They capture the interactions chronologically, showcasing the order in which messages are sent and received. This chapter delves into the world of sequence diagrams, explaining their core concepts and how they can effectively represent the dynamic behavior within a system.

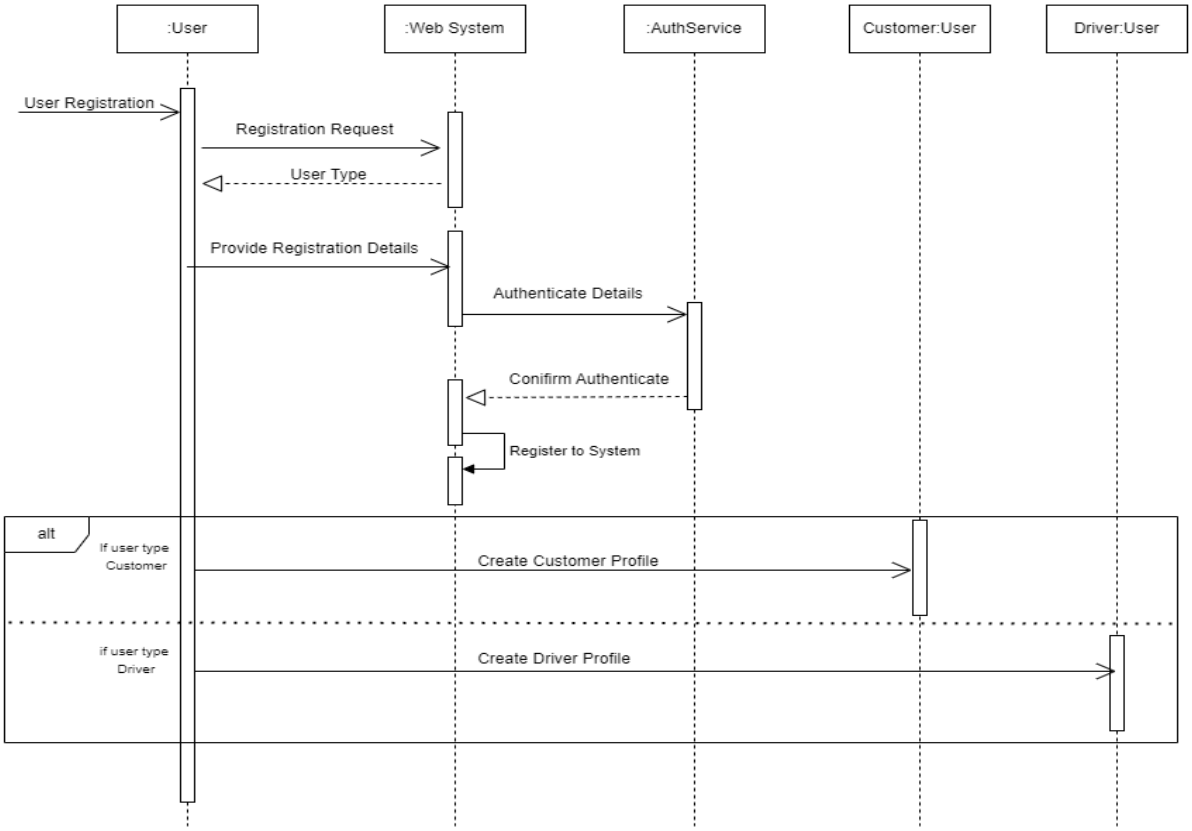


Figure 3.4: User Registration Sequence Diagram

The above sequence diagram (Figure 3.4) illustrates the message flow between a user and a web system during customer registration for a ride-hailing application. It depicts the interactions between three lifelines: the User, the Web System, and the Auth Service (Authentication Service). The User initiates the interaction by requesting registration through the Web System. The Web System captures the user-provided registration details and transmits them to the Auth Service. The Auth Service authenticates the details, and if successful, sends a confirmation signal back to the Web System. Upon receiving confirmation, the Web System registers the user in the system and notifies the User of successful registration.

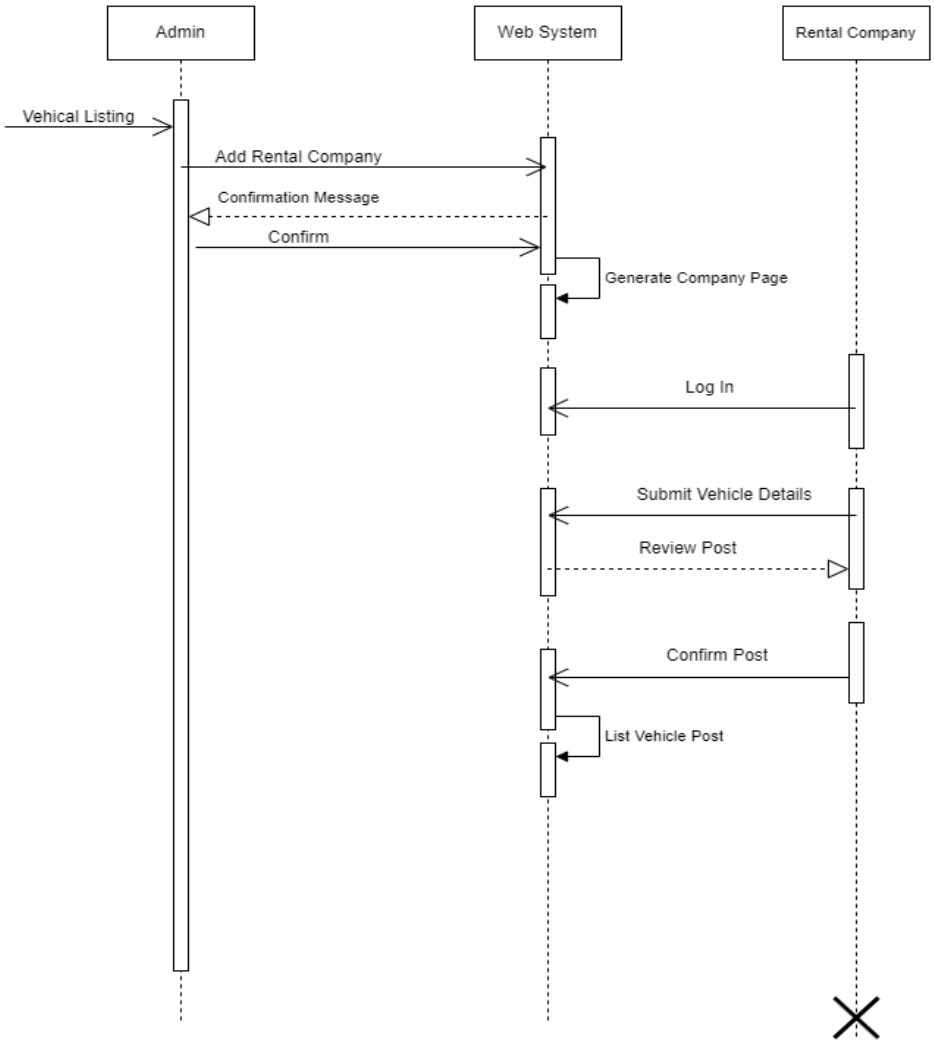


Figure 3.5: Sequence Diagram for Company List Vehicle Post

The above sequence diagram (Figure 3.5) starts with the Admin adding a rental company to the system. The system generates a confirmation message, and a company page is created. Then the company can log in and list vehicle posts.

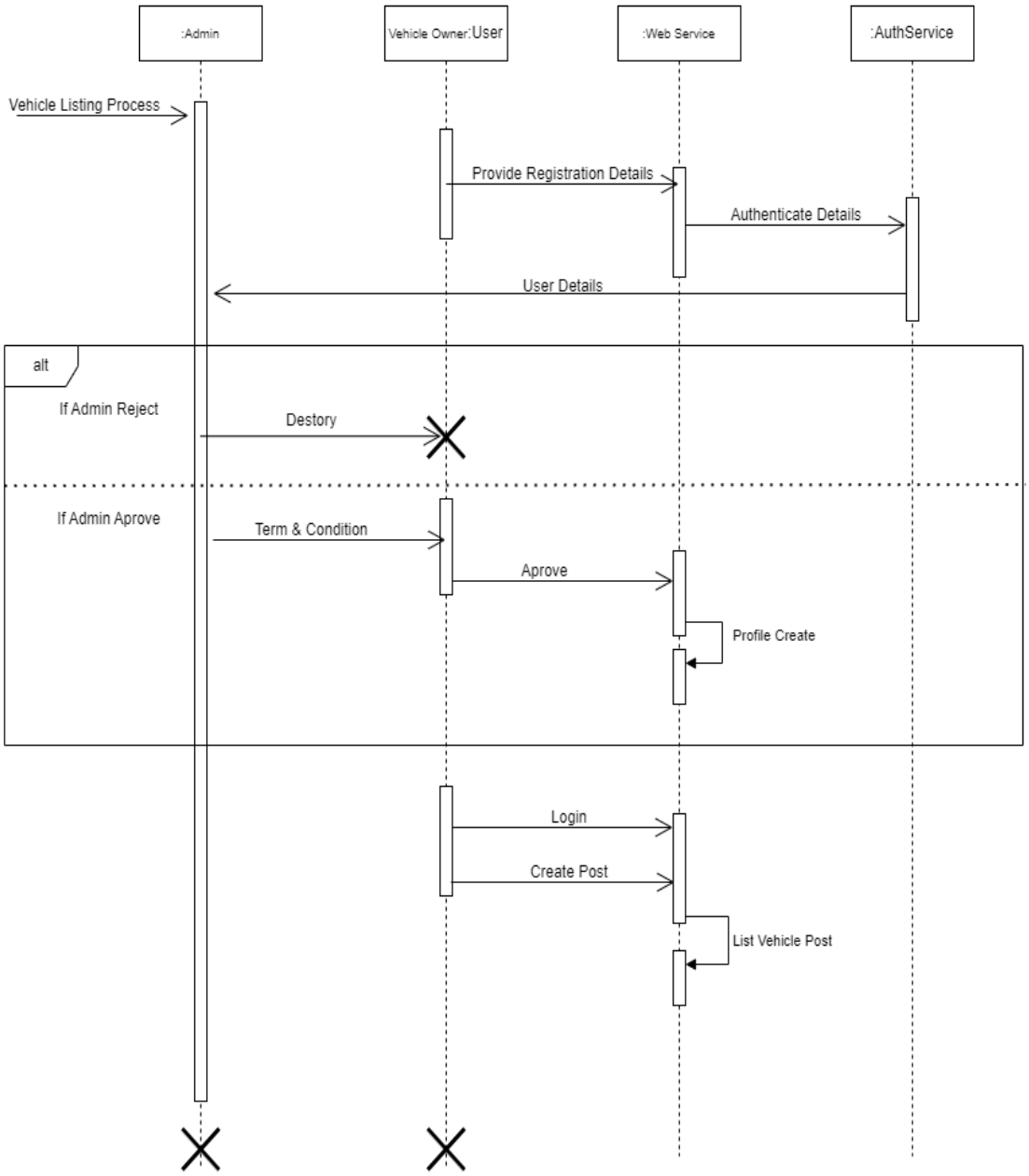


Figure 3.6: Sequence Diagram for Owner List Vehicle Post

In the above sequence diagram (Figure 3.6) vehicle owner initiates the process by providing registration details to the web service. The web service transmits this information to the Admin for approval. The Admin reviews the details and sends either an approval or rejection signal back to the web service.

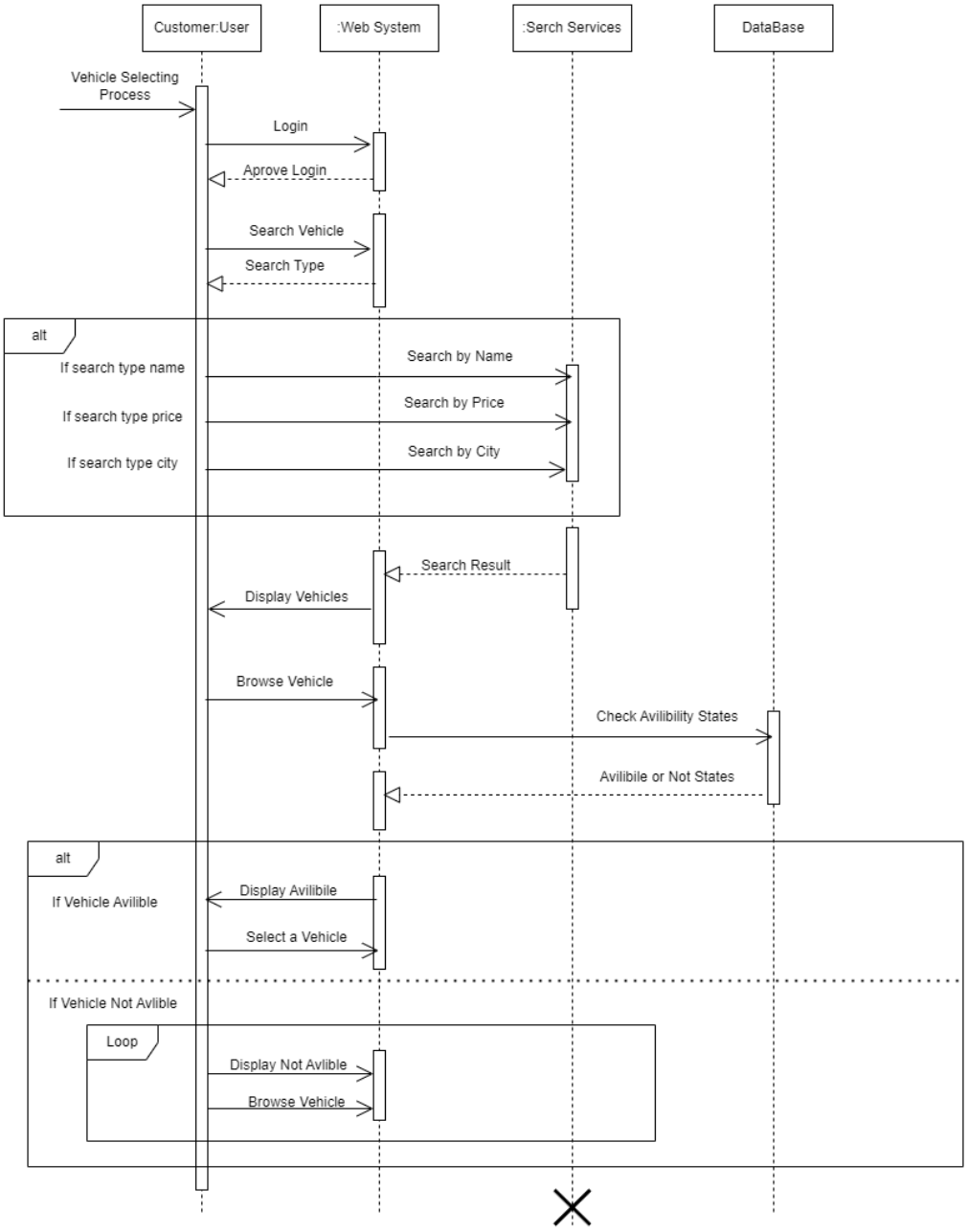


Figure 3.7: Sequence Diagram for Vehicle Selecting Process

The above sequence diagram (Figure 3.7) illustrates the message flow between a customer, a web system, and several search services involved in finding a vehicle. The customer initiates the process by selecting a search type on the web system. The web system then transmits this search type to the corresponding search service. The search service retrieves vehicle information based on the search type and transmits it back to the web system. The web system then displays the search results to the customer.

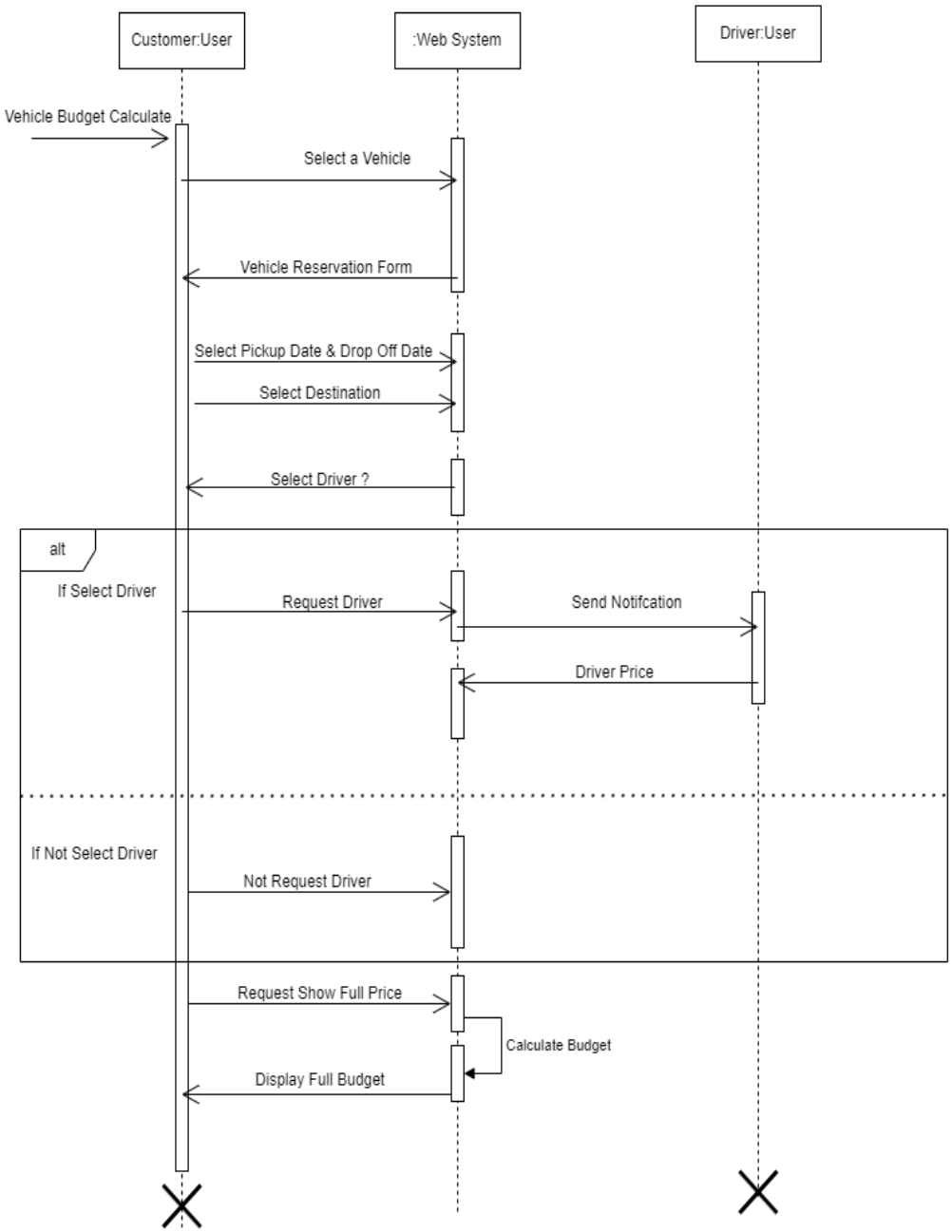


Figure 3.8: Booking Process

The above sequence diagram (Figure 3.8) depicts the message flow between a customer and a web system during the process of booking a vehicle. The diagram showcases two alternative scenarios based on whether the customer chooses to select a driver during the booking process.

3.8. State Chart Diagram

State chart diagrams are a valuable tool for visualizing the behavior of systems in response to various events. Within the context of a vehicle rental system, state chart diagrams will be a key asset in understanding the lifecycle of various elements. We'll explore how these diagrams represent the changing states of a vehicle (available, rented), a customer (registered, logged in, booking), or a reservation.

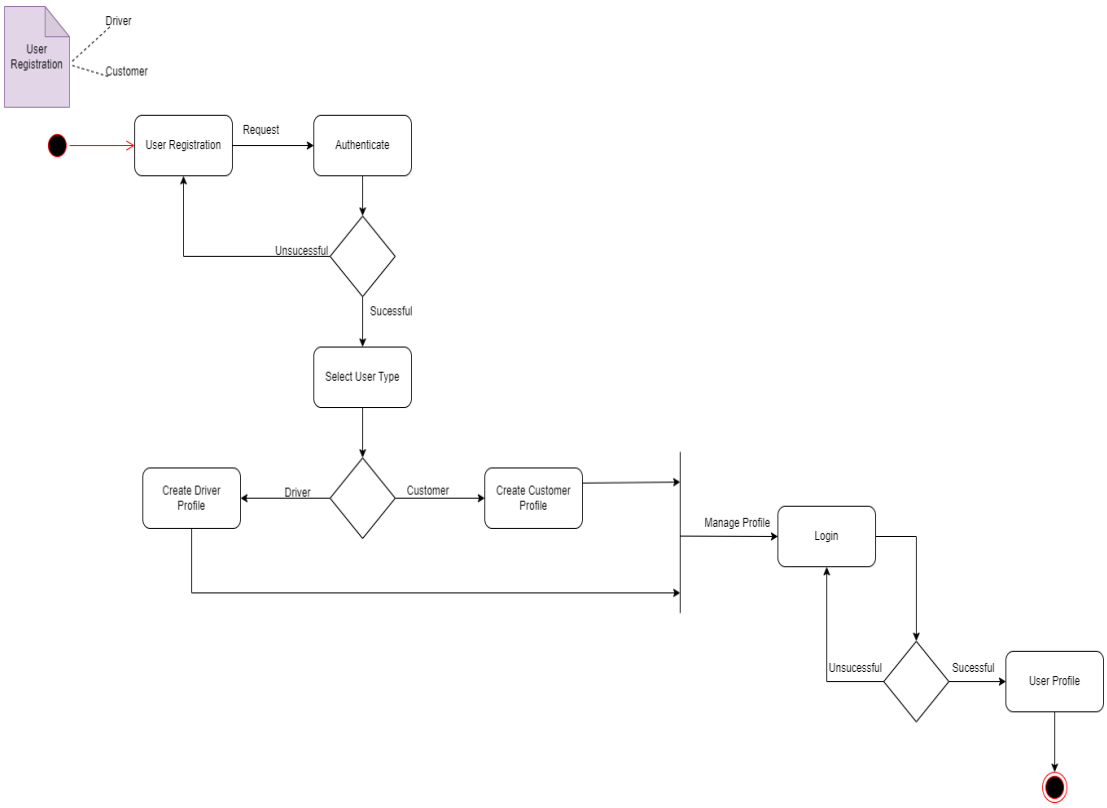


Figure 3.9: State Chart for User Registration

The above state chart diagram (Figure 3.9) depicts the various stages a user can progress through during the registration process for a ride-hailing application. It illustrates the system's behavior in response to user actions and system validations.

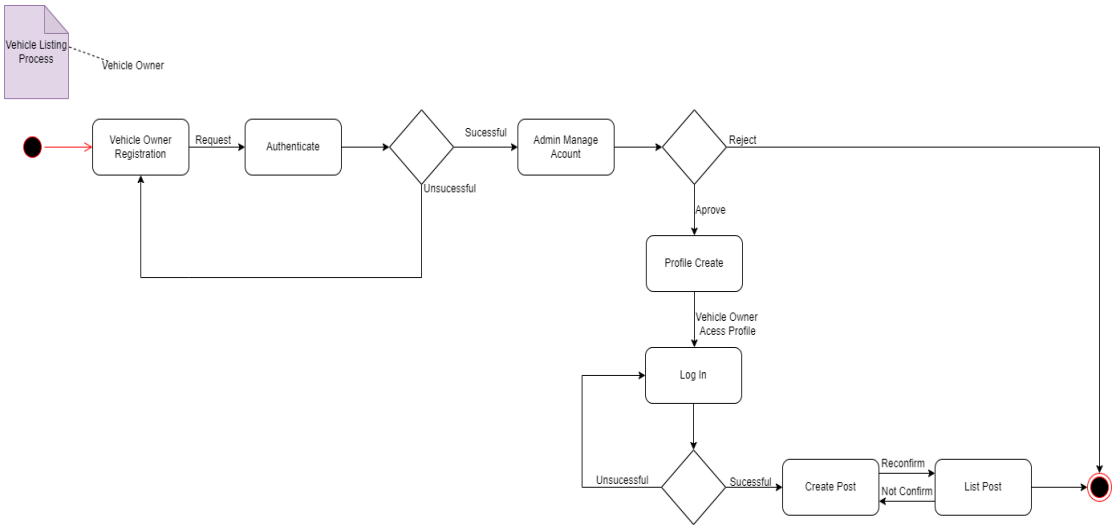


Figure 3.10: State Chart for Vehicle Listing Process

The above state chart diagram (Figure 3.10) illustrates the different states a vehicle listing can undergo on a web platform, along with the events that trigger transitions between these states. It focuses on the interaction between a vehicle owner and the web service throughout the listing process.

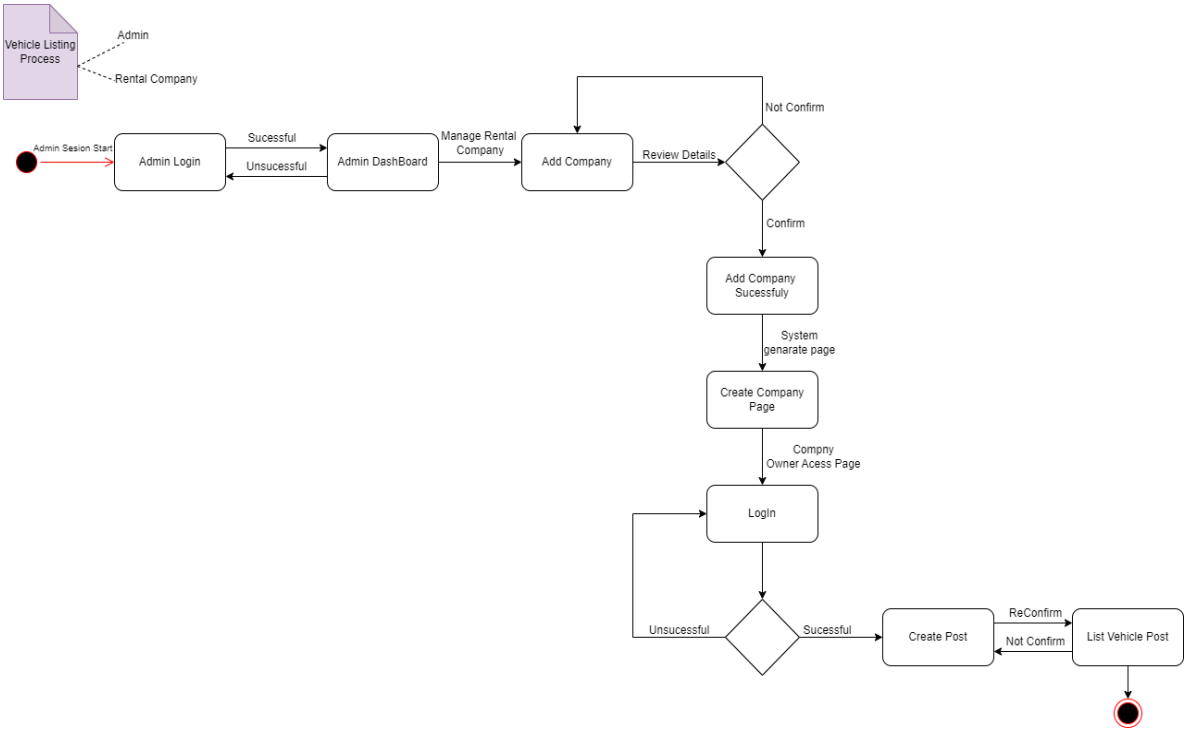


Figure 3.11: State Chart of Vehicle Listing Process 02

The above state chart diagram (Figure 3.11) highlights these possible transitions and the events that trigger them, providing a clear understanding of the vehicle listing and rental process on the rental company's system

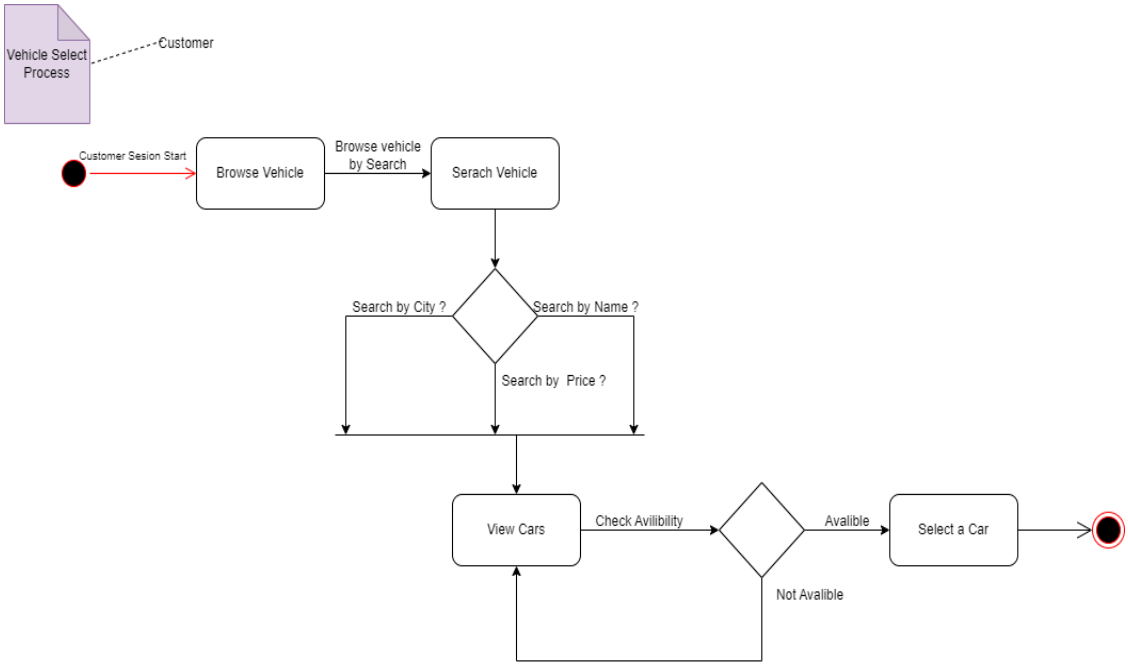


Figure 3.12: State Chart of Vehicle Select Process

The above state chart diagram (Figure 3.12) depicts the different stages a customer can progress through while selecting a vehicle for rental on a web platform. It highlights the system's behavior in response to user actions and the various availability statuses of vehicles.

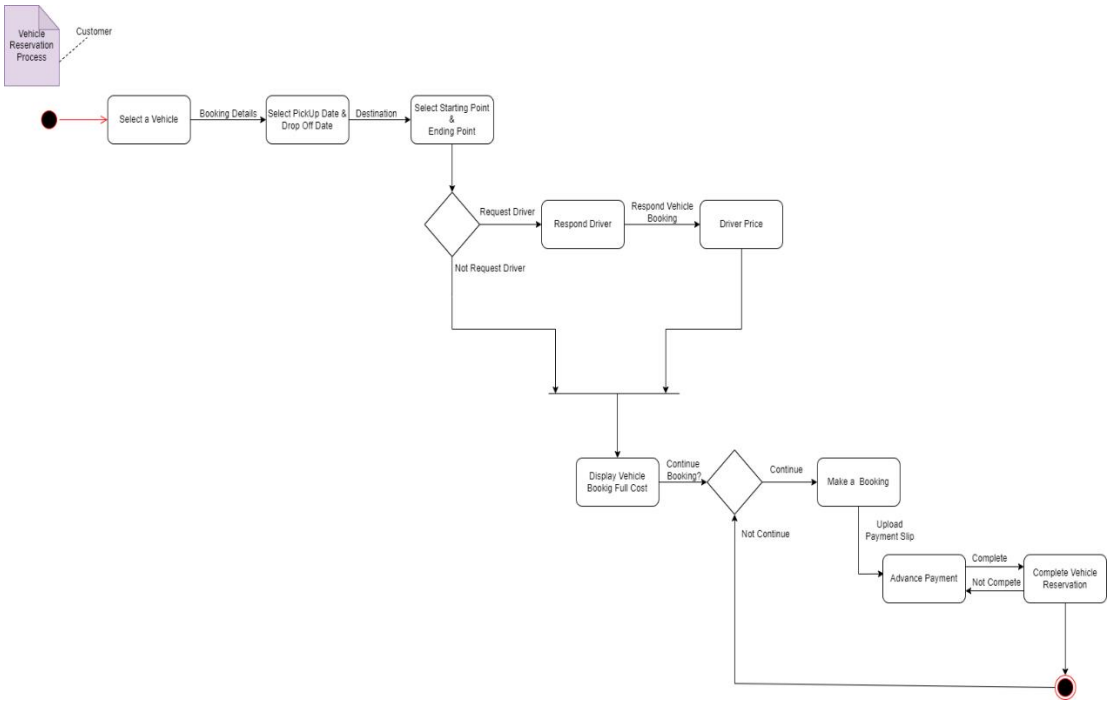


Figure 3.13: State Chart of Vehicle Reservation Process

The above state chart diagram (Figure 3.13) showcases the various stages a vehicle reservation can undergo within a rental system, along with the events that trigger transitions between these states. It illustrates the interaction between a customer and the system throughout the reservation process

3.9. Activity Diagram

Activity diagrams serve as powerful tools in software design, offering a visual representation of the sequential and parallel activities within a system. They play a crucial role in modeling the dynamic behavior of a system, showcasing the flow of actions and decisions as they occur in real time. By illustrating the sequence of activities, decision points, and the order of execution, activity diagrams provide a clear and structured view of how processes unfold within a software application. In this context, activity diagrams are invaluable in understanding, analyzing, and optimizing complex workflows, making them indispensable in the realm of software development and system design

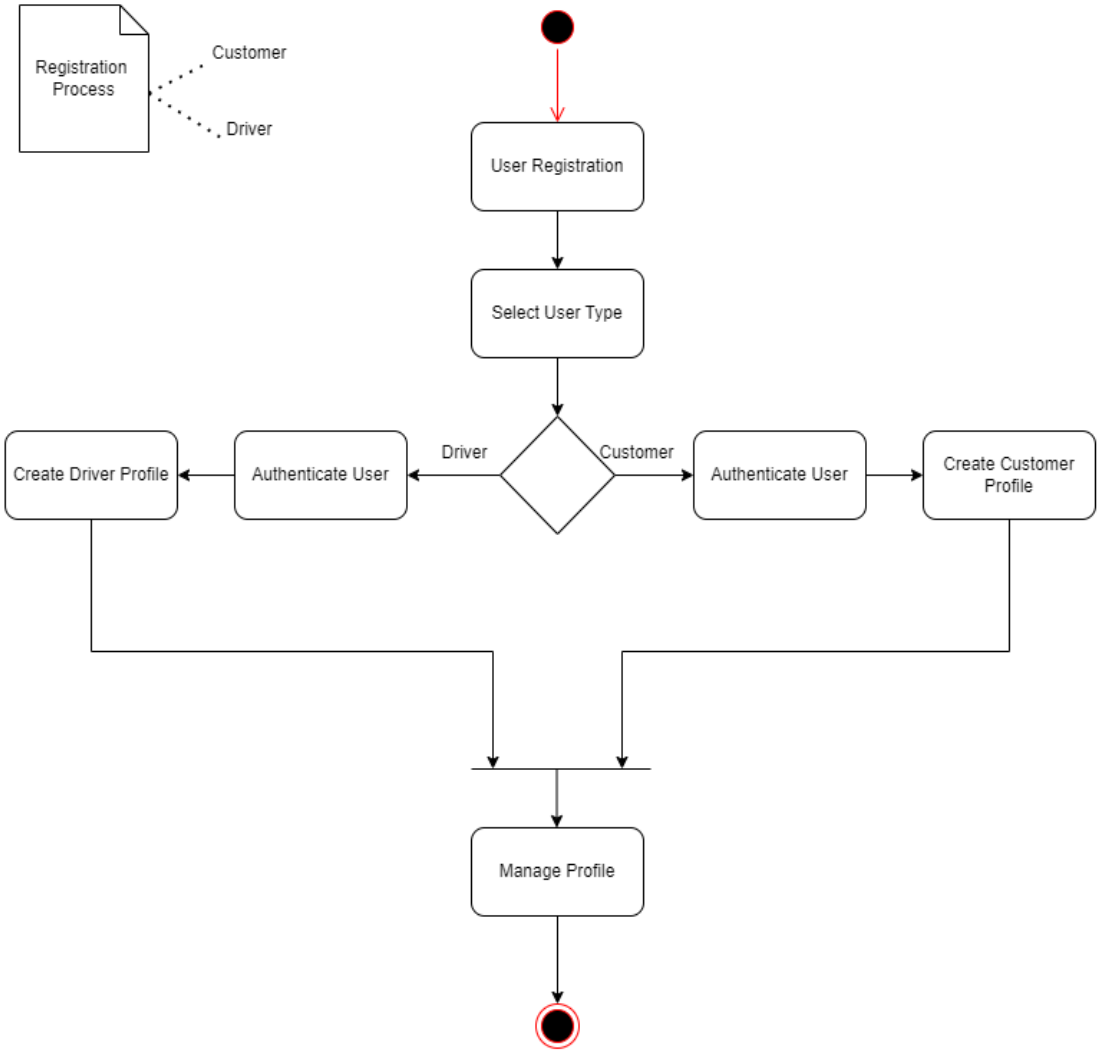


Figure 3.14: Activity Diagram for Registration

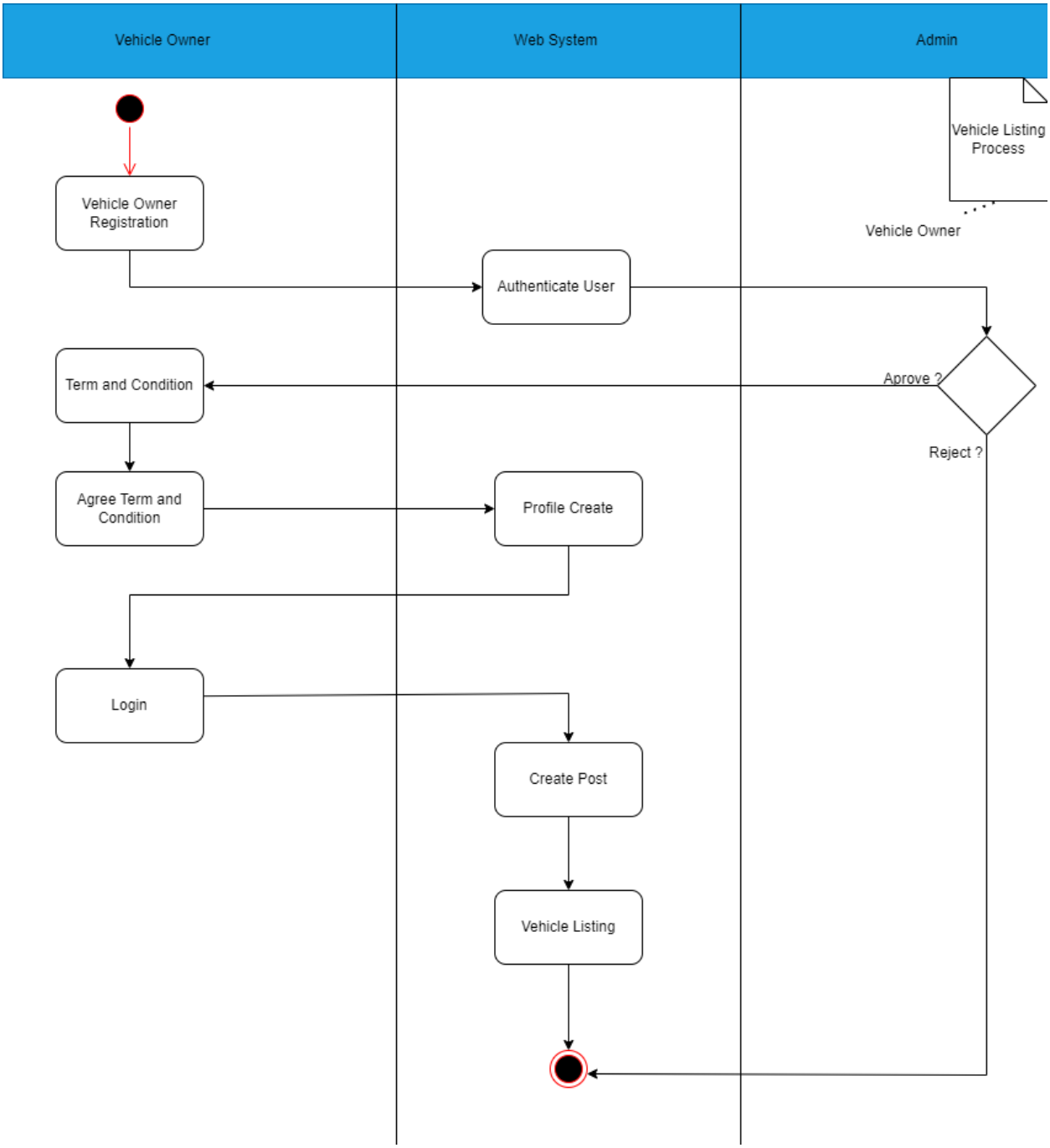


Figure 3.15: Activity Diagram for Vehicle Listing Process

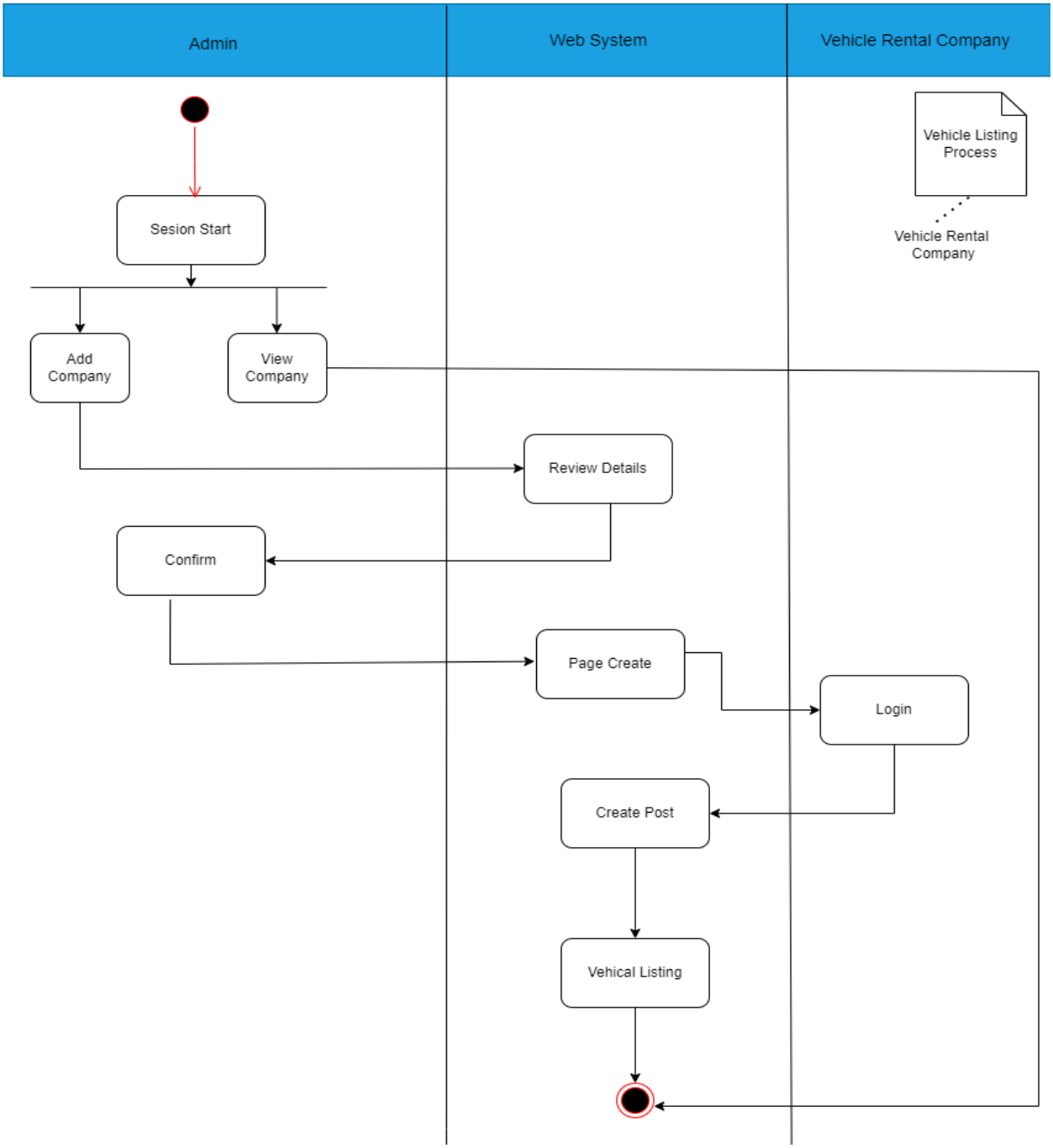


Figure 3.16: Vehicle Listing Process 02

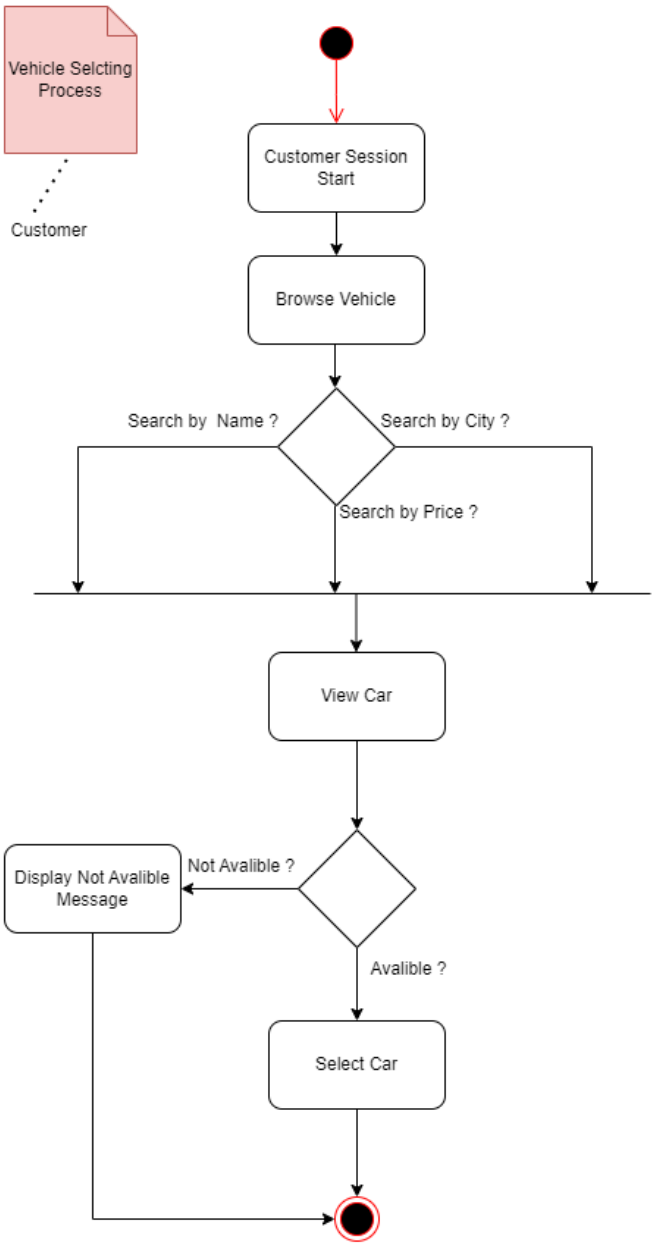


Figure 3.17: Activity Diagram for Vehicle Selecting Process

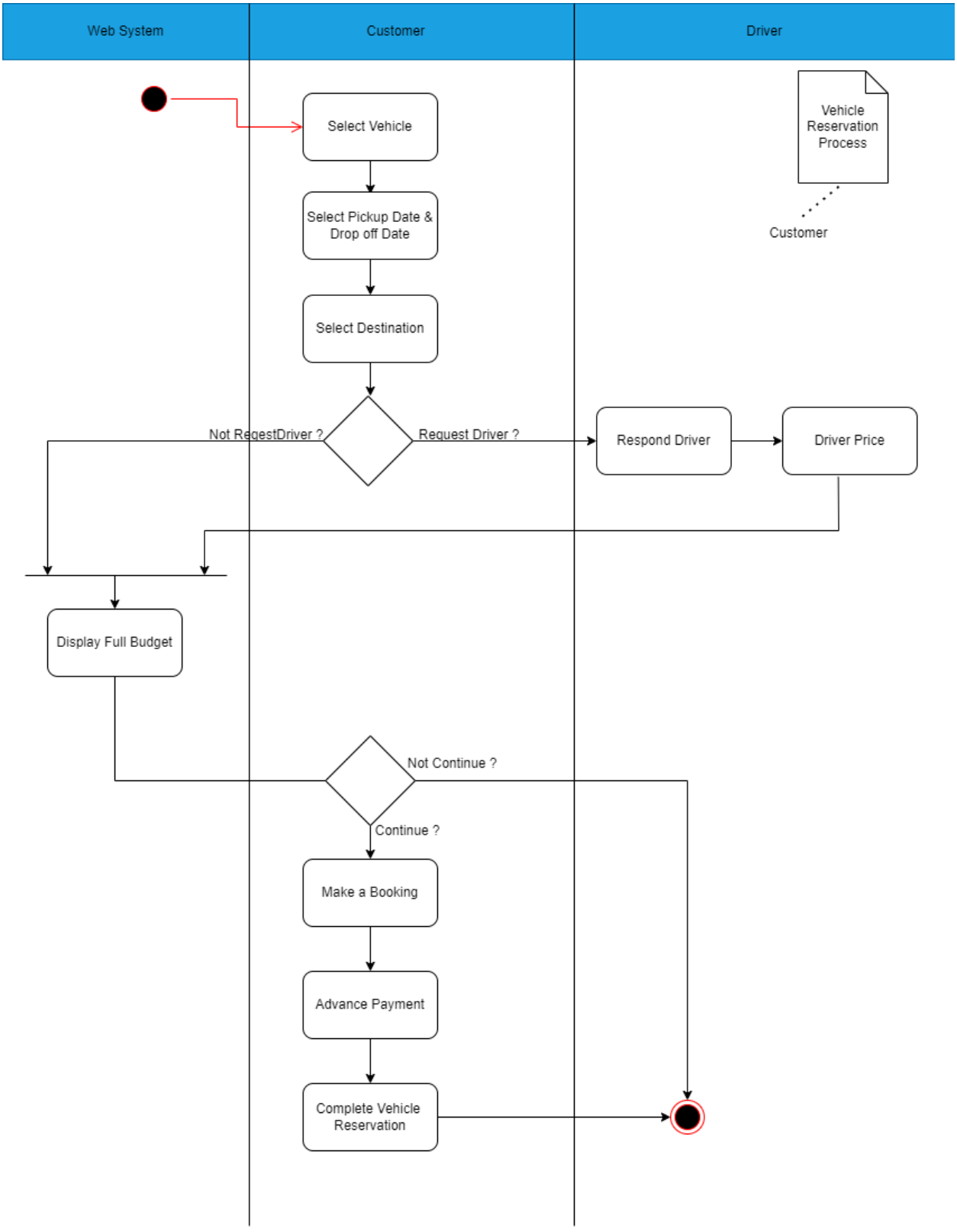


Figure 3.18: Activity Diagram for Vehicle Reservation Process

3.10. Summary

Chapter 3 of our design documentation dives deep into the architecture and functionality of GoCarNow. We begin by welcoming our clients to the future of car rentals, emphasizing our platform's mission to empower small rental companies, individual car owners, and drivers. The scope identification outlines our development goals, focusing on a user-friendly online platform and web application accessible to various stakeholders. The use case diagram visually represents how users interact with the system, followed by the ER diagram showcasing key entities and their attributes. The class diagram provides a detailed view of the system's classes and relationships. Sequence diagrams illustrate message flows in various system processes, while state chart diagrams depict dynamic states. Activity diagrams offer insights into specific processes like registration, vehicle listing, selection, and reservation. Finally, the normalization process ensures data integrity and efficiency, making Chapter 3 a comprehensive guide to understanding and designing the GoCarNow system.

4. Chapter 4: Implementation

4.1. Introduction

In Chapter 4, Implementation, we delve into the practical aspects of the GoCarNow vehicle rental system. This chapter encompasses the creation of database structures using Data Definition Language (DDL), formulation of SQL queries for data manipulation, development of comprehensive test plans for system validation, and the implementation of a user-friendly Java project to streamline rental operations. Through these efforts, we aim to bring to life a robust and efficient solution for managing vehicle rentals, catering to the needs of rental agencies, vehicle owners, drivers, and customers alike.

4.2. DDL

DDL, or Data Definition Language, is a fundamental aspect of database management systems that enables users to define and modify the structure of database objects. With DDL commands, developers can create tables to store data, define indexes for efficient querying, establish relationships between tables, and manage views for data presentation. Additionally, DDL allows for the alteration and deletion of existing database objects, providing flexibility and control in database design and maintenance.

4.2.1. Listing

```
CREATE DATABASE IF NOT EXISTS rentvehicledb;
```

Figure 4.1: DDL 01

The command (Figure 4.1)CREATE DATABASE IF NOT EXISTS rentvehicledb; is used to create a database named rentvehicledb if it doesn't already exist. It's a common SQL statement to ensure that the database is created only if it doesn't exist, preventing errors if the database already exists.

```
CREATE TABLE IF NOT EXISTS catorgery (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    type VARCHAR(20),
    Number_of_Passengers INT
);
```

Figure 4.2: DDL 02

The above DDL statement (Figure 4.2) is used to create a table named catorgery in the database. This table is designed to store categories related to vehicle rentals. It includes columns such as id (an integer primary key), name (a variable character field for category names), type (a variable character field for category types like "Luxury" or "Budget"), and Number_of_Passengers (an integer field representing the number of passengers for vehicles in this category). The IF NOT EXISTS clause ensures that the table is only created if it does not already exist in the database, preventing duplication errors.

4.3. SQL Queries

4.3.1. DML

Data Manipulation Language (DML) is a crucial aspect of database management, enabling users to interact with and modify data within tables. It encompasses a set of SQL commands designed for data manipulation operations such as inserting new records, updating existing data, deleting records, and querying data to retrieve specific information.

4.3.2. Test Plan

Project Name	GoCarNow Vehicle Rental System
Test Plan ID	DB01
Brief Introduction about the system.	The vehicle rental project is a comprehensive software solution designed to facilitate the management of vehicle rentals for various stakeholders including vehicle owners, drivers, vehicle companies, and rental agencies.
Test Objectives	1. Vehicle Availability Check 2. Filter Vehicle by Price 3. Display Rental Company Vehicle list 4. Vehicle type with count of passengers 5. Filter Driver Price Range
Features to be tested	1. Manage Company Details
Test Environment	• Devices - Laptop • Software – MYSQL Workbench
Test Approach	• Testing Levels: System testing, acceptance testing • Testing Methods: Manual testing • Testing Tools: Junit
Testing Task	• Test Planning: Define test objectives, identify test scenarios and plan test execution. • Test Design: Create test cases for each identified test scenario. • Test Execution: Execute test cases and scripts in the environment.
Test deliverables	1. Test Plan 2. Test Environment setup documentation 3. Test Summary Report 4. Test Results 5. Test Evaluation Report
Schedule	• Test Planning: [12.02.2024] - [16.02.2024] • Test Design: [18.02.2024] - [19.02.2024] • Test Development: [19.02.2024] - [23.02.2024] • Test Execution: [24.02.2024] - [24.02.2024] • Test Evaluation: [25.02.2024] - [25.02.2024]

Table 4.1: Test Plan of Queries

4.3.3. Query 01

```
SELECT vehicle.*
FROM vehicle
LEFT JOIN reservation ON vehicle.id = reservation.vehicle_id
WHERE
    (
        (reservation.pickup_date IS NULL OR reservation.pickup_date > '2024-03-10')
        OR
        (reservation.dropoff_date IS NULL OR reservation.dropoff_date < '2024-03-12')
        OR
        (reservation.pickup_date IS NULL OR reservation.dropoff_date IS NULL)
    );
```

Figure 4.3: Query 01

The above query (Figure 4.3) effectively filters out vehicles that are reserved during the period between March 10th and March 11th, 2024. Therefore, the output of the query represents a list of vehicles that are available for rent during that specific timeframe using this query customer can find available vehicles according to the customer's selected pickup date and drop off date.

Result Grid | Filter Rows: | Export: | Wrap Cell Conten

id	name	Model	Vehicle_No	insurance	Vehicle_Price	Catorgery_id
2	Boss	BMW	abc-1234	avilible	4000	2
3	GTR	Nissan	xx-2234	unavilible	5000	3

Figure 4.4: Query 01 Result

4.3.3.1. Test Case

Test Case 01	
Insert pickup date and drop off date	Tharuka Dilshan
01	Black box
Insert Reservation Date according to user need	2024.02.24
Manage Details	2.00 am

Table 4.2:Test Cases 01

Step No	Test Step	Test Case ID	Test Input	Expected result	Actual result	Test result
01	Insert Company Details	01	Pickup Date: 2024-03-10 Drop Off Date: 2024-03-12	Display Available vehicle	Display Available Vehicle	Pass

Table 4.3: Test Cases 01 Result

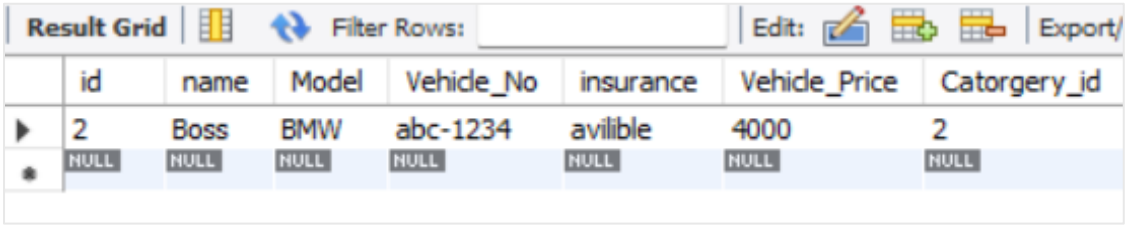
4.3.4. Query 02

```
SELECT vehicle.*
FROM vehicle
WHERE Vehicle_Price < 5000;
```

Figure 4.5: Query 02

The above query (Figure 4.5) Displaying a list of affordable vehicles to potential customers searching for rentals within a specific budget range.

Filtering data for data analysis tasks focusing on vehicles within a particular price bracket.



4.3.4.1. Test Case

Test Case 02	
Filter Vehicle by Price	Tharuka Dilshan
02	Black box
Filter Vehicle by Price entered by user	2024.02.24
Manage Details	2.05 am

Table 4.4: Test Cases 02

Step No	Test Step	Test Case ID	Test Input	Expected result	Actual result	Test result
01	Insert Vehicle Price	02	Price:5000	Display Vehicle Price less than 5000	Display Vehicle Price less than 5000	Pass

Table 4.5: Test Cases 02 Result

4.3.5. Query 03

```
SELECT vehicle.*
FROM vehicle
JOIN rental_company_has_vehicle ON vehicle.id = rental_company_has_vehicle.vehicle_id
WHERE rental_company_has_vehicle.rental_company_Registration_number = 1;
```

Figure 4.6: Query 03

The above query(Figure 4.6), Display a list of vehicles offered by a specific rental company on a website or application. Users can find all lists of vehicles offered by selected rental company

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

id	name	Model	Vehicle_No	insurance	Vehicle_Price	Catorgery_id
2	Boss	BMW	abc-1234	avilible	4000	2
3	GTR	Nissan	xx-2234	unavilible	5000	3

4.3.5.1. Test Case

Test Case 03	
Display Rental Company Vehicle	Tharuka Dilshan
03	Black box
Display All Vehicle Details According to company registration number	2024.02.24
Manage Details	2.15 am

Table 4.6: Test Case 03

Step No	Test Step	Test Case ID	Test Input	Expected result	Actual result	Test result
01	Insert rental company registration Number	03	rental_company_ Registration_number=1	Display Registration Number 1 Vehicle Details	Display Registration Number 1 Vehicle Details	Pass

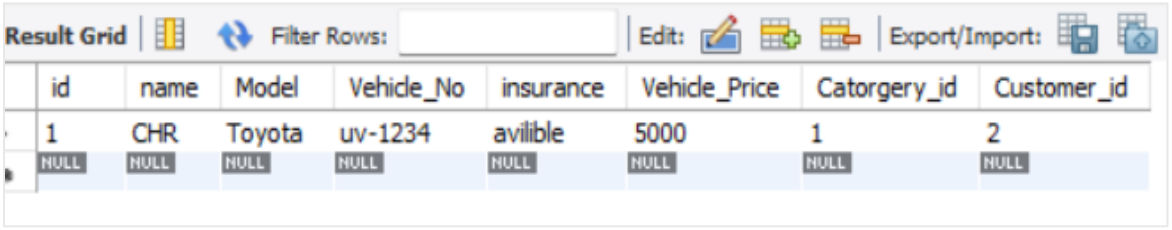
Table 4.7: Test Case 03 Result

4.3.6. Query 04

```
SELECT *
FROM vehicle
WHERE Catorgery_id IN (
    SELECT id
    FROM catorgery
    WHERE type = 'Luxery' AND Number_of_Passangers = 4
);
```

Figure 4.7: Query 04

The above query (Figure 4.7) effectively retrieves information about all vehicles from the vehicle table and then Displays a list of luxury vehicles with a 4-passenger capacity on a car rental website



id	name	Model	Vehicle_No	insurance	Vehicle_Price	Catorgery_id	Customer_id
1	CHR	Toyota	uv-1234	avilible	5000	1	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

4.3.6.1. Test Case

Test Case 04	
Vehicle type with a count of passengers	Tharuka Dilshan
04	Black box
Display Vehicle type with Number of Passengers according to user input	2024.02.24
Manage Details	2.25 am

Table 4.8: Test Case 04

Step No	Test Step	Test Case ID	Test Input	Expected result	Actual result	Test result
01	Insert Vehicle type and Number of Passengers	04	Type= Luxury Number of Passengers=4	Show Luxury 4 passengers Vehicle	Show Luxury 4 passengers Vehicle	Pass

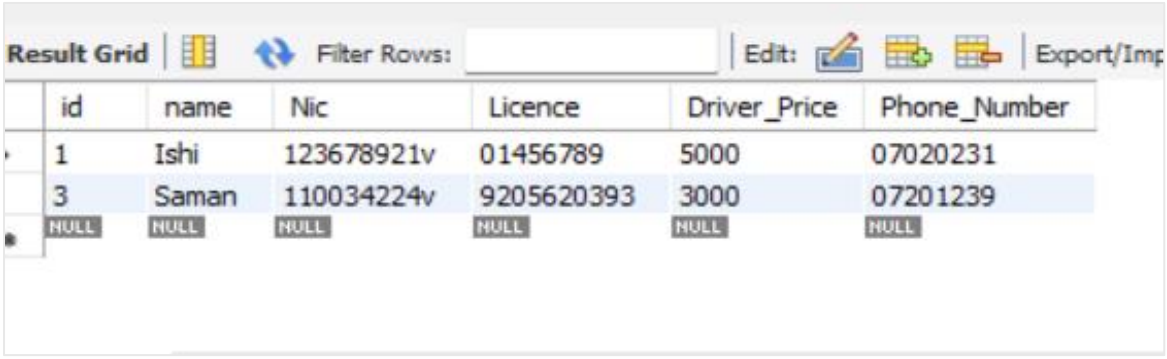
Table 4.9: Test Case 04 Result

4.3.7. Query 05

```
SELECT * FROM driver;
SELECT * FROM driver WHERE Driver_Price BETWEEN 3000 AND 10000;
```

Figure 4.8: Query 05

The above query (Figure 4.8) retrieves all information about every driver in the system. The second query focuses on drivers whose associated price (presumably a service fee or cost) falls within a specific range. This range is between 3000 and 10000Users can filter drivers by specific price range using this query.



4.3.7.1. Test Case

Test Case 05	
Filter Driver Price Range	Tharuka Dilshan
05	Black box
Filter Driver Details According to Specific Price Range	2024.02.24
Manage Details	2.15 am

Table 4.10: Test Case 06

Step No	Test Step	Test Case ID	Test Input	Expected result	Actual result	Test result
01	Insert Price Range of Driver	05	Price Range: 3000-10000	Display Driver Price Rang in 3000-10000	Display Driver Price Rang in 3000-10000	Pass

Table 4.11: Test Case 05 Result

4.4. Java Project

GoCarNow vehicle rental system is designed to make things easier for both administrators and rental company owners. Our system has an easy-to-use interface that makes adding and managing rental companies a breeze. Administrators can quickly enter important details like registration numbers, company emails, and names using simple forms and menus.

Our interface is user-friendly, making it easy for users to navigate and complete tasks efficiently. Clear prompts and visual cues help administrators register rental companies quickly, ensuring accurate information is captured and organized effectively in the system.

Additionally, vehicle owners can log in using their company email and password, granting them access to their respective accounts. This login feature enhances security and ensures that only authorized personnel can manage rental operations and access company data.

4.4.1. Login Form Interface

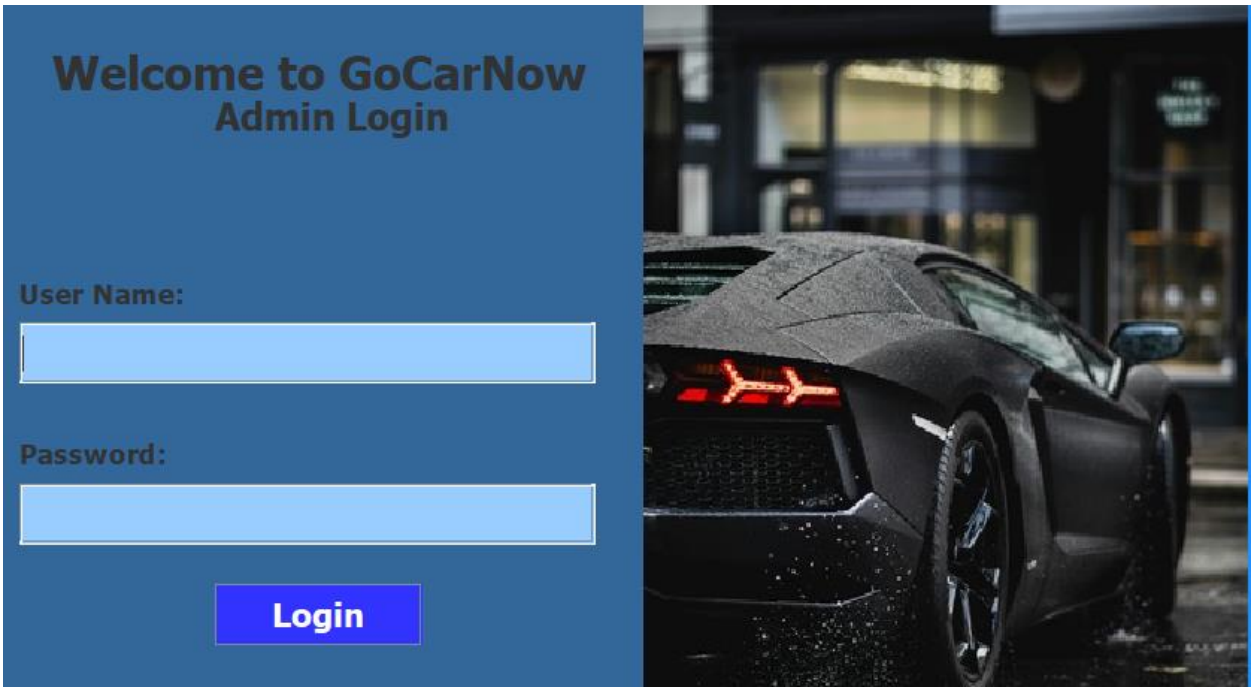


Figure 4.9: Admin Login Page

- The Admin Login Page Interface (Figure 4.9) serves as the gateway for administrators to access the administrative panel of the vehicle rental system. This interface provides a secure authentication mechanism, allowing authorized administrators to log in using their credentials. Designed with usability and security in mind, the interface presents intuitive elements such as username and password fields, along with optional features like the "Remember me" checkbox and a "Forgot password" link for password recovery. Through a combination of clear layout, thoughtful design, and robust functionality, the Admin Login Page Interface facilitates seamless access for administrators, ensuring efficient management of vehicle company registrations within the system.

4.4.2. Admin Dashboard

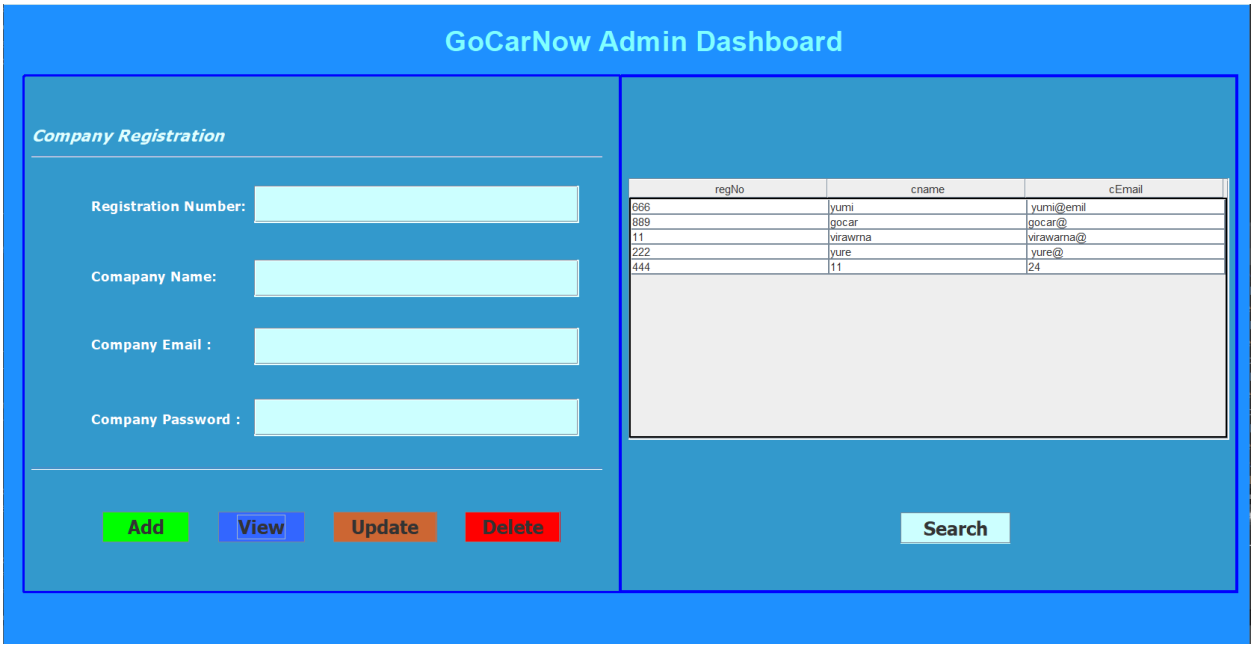


Figure 4.10: Admin Dashboard

- The Administration Interface (Figure 4.10) serves as the central hub for administrators to manage vehicle company registrations and oversee system operations within the vehicle rental platform. Designed with efficiency and usability in mind, the interface offers a suite of essential functionalities, including adding, viewing, deleting, updating, and searching for vehicle company records. Administrators can seamlessly navigate through these features, ensuring smooth management of registered vehicle companies and facilitating streamlined operations across the platform. With robust security measures and intuitive user feedback mechanisms in place, the Administration Interface empowers administrators to maintain data integrity and uphold system reliability effectively.

Functional Requirement

Add Vehicle Company:

- The system should allow administrators to add new vehicle companies to the database.
- Administrators should be able to input details such as company name, contact information, and registration documents.

View Vehicle Companies:

- Administrators should be able to view a list of all registered vehicle companies.
- The list should display relevant details such as company name, contact information, and registration status.

Delete Vehicle Company:

- Administrators should have the ability to delete vehicle company records from the system.
- Deletion should require confirmation to prevent accidental removal of records.

Update Vehicle Company Information:

- Administrators should be able to update the details of existing vehicle companies.
- They should have the capability to modify information such as contact details or registration documents.

Search for Vehicle Companies:

- Administrators should be able to search for specific vehicle companies using keywords or criteria.
- The search functionality should provide relevant results based on the search query.

Testing

Test Plan

Project Name	GoCarNow Vehicle Rental System
Test Plan ID	GCN01
Brief Introduction about the system.	The vehicle rental project is a comprehensive software solution designed to facilitate the management of vehicle rentals for various stakeholders including vehicle owners, drivers, vehicle companies, and rental agencies.
Test Objectives	1. Insert Company Data 2. Update/Edit Company Data 3. Search Company Data 4. Delete Company Data 5. View Company Data
Features to be tested	1. Manage Company Details
Test Environment	• Devices - Laptop • Software – Eclipse IDE • Database – WAMP server database
Test Approach	• Testing Levels: System testing, acceptance testing • Testing Methods: Manual testing • Testing Tools: Junit
Testing Task	• Test Planning: Define test objectives, identify test scenarios and plan test execution. • Test Design: Create test cases for each identified test scenario. • Test Execution: Execute test cases and scripts in the environment.
Test deliverables	1. Test Plan 2. Test Environment setup documentation 3. Test Summary Report 4. Test Results 5. Test Evaluation Report
Schedule	• Test Planning: [12.02.2024] - [16.02.2024] • Test Design: [18.02.2024] - [19.02.2024] • Test Development: [19.02.2024] - [23.02.2024] • Test Execution: [24.02.2024] - [24.02.2024] • Test Evaluation: [25.02.2024] - [25.02.2024]

Table 4.12: Test Plan of Java

Test Cases

Insert Company Details.

Test Case 01	
Insert Manager Details	Tharuka Dilshan
01	Black box
Insert Company Details to the Database	2024.02.24
Manage Details	2.00 am

Table 4.13: Insert Company Details Test Case

Step No	Test Step	Test Case ID	Test Input	Expected result	Actual result	Test result
01	Insert Company Details	01	Registration Number: 001 Company Name: Riyasewna Email: Riyasewana@gmail.com Password: 12345678	Show Insert Data Successfully Message	Data Insert Successfully	Pass

Table 4.14:: Insert Company Details Test Case Result

2. Update Company Details.

Test Case 02	
Update Manager Details	Tharuka Dilshan
02	Black box
Update Company Details to the Database	2024.02.24
Manage Details	2.05 am

Table 4.15: Update Company Details Test Case

Step No	Test Step	Test Case ID	Test Input	Expected result	Actual result	Test result
01	Update Company Details	02	CompanyName: Riyasewana Car Email: riyasewanacar@gmail,com	Show Insert Data Edited Successfully Message	Data Edited Successfully	Pass

Table 4.16: Update Company Details Test Case Result

3. View Company Details

Test Case 03	
Update Manager Details	Tharuka Dilshan
03	Black box
View Company Details from the Database	2024.02.24
Manage Details	2.15 am

Table 4.17: View Company Details Test Case

Step No	Test Step	Test Case ID	Test Input	Expected result	Actual result	Test result
01	View Company Details	03	RegNo:101	Show All Colum	Display All Colum	Pass

Table 4.18: View Company Details Test Result

4. Delete Company Details

Test Case 04	
Delete Company Details	Tharuka Dilshan
03	Black box
Delete Company Details from the Database	2024.02.24
Manage Details	2.25 am

Table 4.19: Delete Company Details Test Case

Step No	Test Step	Test Case ID	Test Input	Expected result	Actual result	Test result
01	Delete Company Details	04	RegNo:1	Show Data Deleted Successfully Message	Data Deleted Successfully	Pass

Table 4.20: Delete Company Details Test Case Result

5. Search Company Details

Test Case 05	
Update Manager Details	Tharuka Dilshan
05	Black box
Search Company Details from the Database	2024.02.24
Manage Details	2.15 am

Table 4.21:Search Company Test Case

Step No	Test Step	Test Case ID	Test Input	Expected result	Actual result	Test result
01	View Company Details	05	RegNo:2	Show Company Details where RegNo equal to 2	RegNo: 2 Company Name: RolankaCar Email:rolanka@gmail.com	Pass

Table 4.22: Search Company Test Case

6. Annexure

1 . company Class.java

```
import java.sql.PreparedStatement;

public class company {

    protected int regNo;
    protected String cname;
    protected String cEmail;
    protected String cPassword;
    protected Mysqlconnection dbc;



    //parameterlized constructor
    public company(int regNo,String cname,String cEmail,String cPassword) {
        this.regNo=regNo;
        this.cname=cname;
        this.cEmail=cEmail;
        this.cPassword=cPassword;


    }
    //constructor
    public company() {
        dbc=new Mysqlconnection();

    }



    //insert data
    public Boolean insert(int regNo,String cname,String cEmail,String cPassword) {
        try {
            String mysql="INSERT INTO companyregister(regNo,cname,cEmail,cPassword)VALUES(?,?,?,?)";
            PreparedStatement statement=dbc.connectDb().prepareStatement(mysql);
```

```
//insert data
public Boolean insert(int regNo,String cname,String cEmail,String cPassword) {
    try {
        String mysql="INSERT INTO companyregister(regNo,cname,cEmail,cPassword)VALUES(?,?,?,?)";
        PreparedStatement statement=dbc.connectDb().prepareStatement(mysql);

        statement.setInt(1,regNo);
        statement.setString(2,cname);
        statement.setString(3,cEmail);
        statement.setString(4,cPassword);

        if(regNo==0||cname==null||cEmail==null||cPassword==null||cPassword.isEmpty()) {
            return false;

        }else {
            int rowsInserted=statement.executeUpdate();
            if(rowsInserted>0) {
                return true;
            }else {
                return false;
            }
        }


    }catch(SQLException e) {
        return false;
    }

}
```

```
    //view method
   public  DefaultTableModel view(){
       DefaultTableModel model = new DefaultTableModel();


       try {
           String mysql="SELECT regNo, cname, cEmail FROM companyregister";
           PreparedStatement statement=dbc.connectDb().prepareStatement(mysql);

           ResultSet resultSet = statement.executeQuery();

           //colums names
           int columCount = resultSet.getMetaData().getColumnCount();
           for(int i = 1;i <= columCount;i++) {
               model.addColumn(resultSet.getMetaData().getColumnName(i));
           }


           //raw data
           while(resultSet.next()) {
               Object[] rowData = new Object [columCount];
               for(int i=0; i < columCount;i++) {
                   rowData[i] = resultSet.getObject(i+1);
               }
               model.addRow(rowData);
           }


       } catch (SQLException e) {
           // TODO Auto-generated catch block
           e.printStackTrace();
       }









       return model;
}
```

```java
// Edit method
public boolean edit(int regNo, String newName, String newEmail) throws SQLException {
    // SQL statement for update operation
    String mysql = "UPDATE companyregister SET cname = ?, cEmail = ? WHERE regNo = ?";

    try (
        // Establish connection and create PreparedStatement
            PreparedStatement statement=dbc.connectDb().prepareStatement(mysql);
    ) {
        // Set the parameters for the PreparedStatement
        statement.setString(1, newName);
        statement.setString(2, newEmail);
        statement.setInt(3, regNo);

        // Execute the update operation
        int rowsUpdated = statement.executeUpdate();

        // Check if any rows were affected
        if (rowsUpdated > 0) {
            return true; // Update successful
        } else {
            return false; // No rows were updated (perhaps no matching record found)
        }
    }
}
```

```java
// Delete method
public boolean delete(int regNo) throws SQLException {
    // SQL statement for delete operation
    String mysql = "DELETE FROM companyregister WHERE regNo = ?";

    try (
        // Establish connection and create PreparedStatement
            PreparedStatement statement=dbc.connectDb().prepareStatement(mysql);

    ) {
        // Set the parameter for the PreparedStatement
        statement.setInt(1, regNo);

        // Execute the delete operation
        int rowsDeleted = statement.executeUpdate();

        // Check if any rows were affected
        if (rowsDeleted > 0) {
            return true; // Deletion successful
        } else {
            return false; // No rows were deleted (perhaps no matching record found)
        }
    }
}
```

```
    public ResultSet searchByRegNo(int regNo) {
        try {
            String mysql = "SELECT * FROM companyregister WHERE regNo = ?";
            PreparedStatement statement = dbc.connectDb().prepareStatement(mysql);
            statement.setInt(1, regNo);
            return statement.executeQuery();
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }



}
```

3.Database Connection

```
package vehiclerental;

import java.sql.Connection;

public class Mysqlconnection {
    private String username;
    private String pwd;
    private String dbname;
    private Connection conn;

    public Mysqlconnection() {
        this.dbname = "javadb";
        this.username = "root";
        this.pwd = "";
        connectDb();
     }



    public Connection connectDb() {
        // TODO Auto-generated method stub
        try {
            conn=DriverManager.getConnection("jdbc:mysql://localhost:3309/"+dbname,username,pwd);
            return conn;
        }catch(SQLException e) {
            System.out.print(e);
            return conn;
        }
    }

}
```

4.Login Class

```java
package vehiclerental;

import java.sql.Connection;

public class LoginService {
      private static final String JDBC_URL = "jdbc:mysql://localhost:3309/javadb";
        private static final String JDBC_USERNAME = "root";
        private static final String JDBC_PASSWORD = "";

        public static boolean validateAdmin(String username, String password) {
            boolean isValid = false;
            Connection conn = null;
            PreparedStatement stmt = null;
            ResultSet rs = null;

            try {
                conn = DriverManager.getConnection(JDBC_URL, JDBC_USERNAME, JDBC_PASSWORD);
                String query = "SELECT * FROM admin WHERE username = ? AND password = ?";
                stmt = conn.prepareStatement(query);
                stmt.setString(1, username);
                stmt.setString(2, password);
                rs = stmt.executeQuery();
                isValid = rs.next();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return isValid;
        }

}
```

5.Login Interface

```
import java.awt.EventQueue;

public class Login extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField nameField;
    private JPasswordField passwordField;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Login frame = new Login();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public Login() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 703, 461);
        contentPane = new JPanel();
        contentPane.setBackground(new Color(30, 144,255));
```

```
JButton btnNewButton = new JButton("Login");
btnNewButton.setForeground(new Color(255, 255, 255));
btnNewButton.setBounds(115, 315, 111, 33);
panel.add(btnNewButton);
btnNewButton.setBackground(new Color(51, 51, 255));
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {



        String username = nameField.getText();
        String password = new String(passwordField.getPassword());

        if (LoginService.validateAdmin(username, password)) {
            JOptionPane.showMessageDialog(null, "Login Successful!");
            AdminDashboard obj=new AdminDashboard();
            obj.setVisible(true);
        } else {
            JOptionPane.showMessageDialog(null, "Invalid Login. Try Again!");
        }
    }
});
btnNewButton.setFont(new Font("Tahoma", Font.BOLD, 18));

nameField = new JTextField();
nameField.setBounds(10, 175, 309, 33);
panel.add(nameField);
nameField.setForeground(Color.DARK_GRAY);
nameField.setBackground(new Color(153, 204, 255));
nameField.setFont(new Font("Tahoma", Font.PLAIN, 15));
nameField.setColumns(10);

passwordField = new JPasswordField();
```

```
        btnNewButton.setFont(new Font("Tahoma", Font.BOLD, 18));

        nameField = new JTextField();
        nameField.setBounds(10, 175, 309, 33);
        panel.add(nameField);
        nameField.setForeground(Color.DARK_GRAY);
        nameField.setBackground(new Color(153, 204, 255));
        nameField.setFont(new Font("Tahoma", Font.PLAIN, 15));
        nameField.setColumns(10);

        passwordField = new JPasswordField();
        passwordField.setBounds(10, 261, 309, 33);
        panel.add(passwordField);
        passwordField.setForeground(Color.DARK_GRAY);
        passwordField.setBackground(new Color(153, 204, 255));

        JLabel lblNewLabel_1_1 = new JLabel("Password:");
        lblNewLabel_1_1.setBounds(10, 239, 164, 13);
        panel.add(lblNewLabel_1_1);
        lblNewLabel_1_1.setFont(new Font("Tahoma", Font.BOLD, 15));

        JLabel lblNewLabel_1 = new JLabel("User Name:");
        lblNewLabel_1.setBounds(10, 153, 164, 13);
        panel.add(lblNewLabel_1);
        lblNewLabel_1.setFont(new Font("Tahoma", Font.BOLD, 15));
    }
}
```

6.Admin Dashboard

```java
package vehiclerental;

import java.awt.EventQueue;

public class AdminDashboard extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField regField;
    private JTextField nameField;
    private JTextField emailField;
    private JPasswordField passwordField;
    private JTable table;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    AdminDashboard frame = new AdminDashboard();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
```

```
public AdminDashboard() {
    setBackground(new Color(0, 204, 255));
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 1348, 787);
    contentPane = new JPanel();
    contentPane.setBackground(new Color(30, 144, 255));
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

    setContentPane(contentPane);
    contentPane.setLayout(null);

    JLabel lblNewLabel = new JLabel("GoCarNow Admin Dashboard");
    lblNewLabel.setForeground(new Color(128, 255, 255));
    lblNewLabel.setFont(new Font("SansSerif", Font.BOLD, 30));
    lblNewLabel.setBounds(472, 23, 443, 30);
    contentPane.add(lblNewLabel);

    JPanel registrationPanel = new JPanel();
    registrationPanel.setBorder(new LineBorder(new Color(0, 0, 255), 3, true));
    registrationPanel.setBackground(new Color(51, 153, 204));
    registrationPanel.setBounds(20, 75, 641, 555);
    contentPane.add(registrationPanel);
    registrationPanel.setLayout(null);

    JLabel lblNewLabel_2 = new JLabel("Comapany Name: ");
    lblNewLabel_2.setForeground(new Color(255, 255, 255));
    lblNewLabel_2.setBounds(74, 198, 164, 35);
    lblNewLabel_2.setFont(new Font("Tahoma", Font.BOLD, 15));
    registrationPanel.add(lblNewLabel_2);

    regField = new JTextField();
    regField.setForeground(Color.GRAY);
    regField.setFont(new Font("Tahoma", Font.PLAIN, 18));
```

```
public Login() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 703, 461);
    contentPane = new JPanel();
    contentPane.setBackground(new Color(30, 144,255));
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

    setContentPane(contentPane);
    contentPane.setLayout(null);

    JPanel panel = new JPanel();
    panel.setBackground(new Color(51, 102, 153));
    panel.setBounds(10, 32, 669, 371);
    contentPane.add(panel);
    panel.setLayout(null);

    JLabel lblNewLabel = new JLabel("Admin Login");
    lblNewLabel.setBounds(115, 52, 126, 25);
    panel.add(lblNewLabel);
    lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 20));

    JLabel lblNewLabel_2 = new JLabel("Welcome to GoCarNow");
    lblNewLabel_2.setBounds(28, 28, 318, 27);
    panel.add(lblNewLabel_2);
    lblNewLabel_2.setFont(new Font("Tahoma", Font.BOLD, 25));

    JLabel lblNewLabel_3 = new JLabel("");
    lblNewLabel_3.setBounds(345, 0, 343, 371);
    panel.add(lblNewLabel_3);
    lblNewLabel_3.setIcon(new ImageIcon("C:\\Users\\Tharuka\\Downloads\\car.jpg"));

    JButton btnNewButton = new JButton("Login");
    btnNewButton.setForeground(new Color(255, 255, 255));
```

```
        panel.add(btnNewButton);
        btnNewButton.setBackground(new Color(51, 51, 255));
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {



                String username = nameField.getText();
                String password = new String(passwordField.getPassword());

                if (LoginService.validateAdmin(username, password)) {
                    JOptionPane.showMessageDialog(null, "Login Successful!");
                    AdminDashboard obj=new AdminDashboard();
                    obj.setVisible(true);
                } else {
                    JOptionPane.showMessageDialog(null, "Invalid Login. Try Again!");
                }
            }
        });
        btnNewButton.setFont(new Font("Tahoma", Font.BOLD, 18));

        nameField = new JTextField();
        nameField.setBounds(10, 175, 309, 33);
        panel.add(nameField);
        nameField.setForeground(Color.DARK_GRAY);
        nameField.setBackground(new Color(153, 204, 255));
        nameField.setFont(new Font("Tahoma", Font.PLAIN, 15));
        nameField.setColumns(10);

        passwordField = new JPasswordField();
        passwordField.setBounds(10, 261, 309, 33);
        panel.add(passwordField);
        passwordField.setForeground(Color.DARK_GRAY);
```

```
        nameField = new JTextField();
        nameField.setBounds(10, 175, 309, 33);
        panel.add(nameField);
        nameField.setForeground(Color.DARK_GRAY);
        nameField.setBackground(new Color(153, 204, 255));
        nameField.setFont(new Font("Tahoma", Font.PLAIN, 15));
        nameField.setColumns(10);

        passwordField = new JPasswordField();
        passwordField.setBounds(10, 261, 309, 33);
        panel.add(passwordField);
        passwordField.setForeground(Color.DARK_GRAY);
        passwordField.setBackground(new Color(153, 204, 255));

        JLabel lblNewLabel_1_1 = new JLabel("Password:");
        lblNewLabel_1_1.setBounds(10, 239, 164, 13);
        panel.add(lblNewLabel_1_1);
        lblNewLabel_1_1.setFont(new Font("Tahoma", Font.BOLD, 15));

        JLabel lblNewLabel_1 = new JLabel("User Name:");
        lblNewLabel_1.setBounds(10, 153, 164, 13);
        panel.add(lblNewLabel_1);
        lblNewLabel_1.setFont(new Font("Tahoma", Font.BOLD, 15));
    }
}
```

4.4.3. References

https://www.geeksforgeeks.org/introduction-to-java-swing/

https://www.tutorialspoint.com/human_computer_interface/object_oriented_programming.htm

https://www.javatpoint.com/java-swing

4.5. Summary

Chapter 4, Implementation, covers the foundational steps required to build and execute the GoCarNow vehicle rental system. It begins with DDL, where database structures are defined and modified to store essential data. SQL queries are then crafted to manipulate and retrieve data effectively, ensuring smooth interaction with the database. The chapter also outlines a detailed test plan, highlighting key objectives and testing methodologies to validate the system's functionality thoroughly. Finally, the Java project implementation adds a user-friendly interface for administrators, rental company owners, and other stakeholders to interact seamlessly with the system, enhancing overall efficiency and usability.