Software Requirements Specification

for

Car Rental Management System

Prepared by Group #1

Dept. of Computer Science & Engineering

University of Rajshahi

14 April 2023.5

Group #1 members:

- **Ahnaf Shahrear Khan** **19103.576101**
- **Toufiqul Islam** **1910876107**
- **Tareq Munawer Siddiqui** **1911176114**
- **Md Nayem Molla** **1910476108**
- **Md. Forhan Shahriar Fahim** **1910476120**
- **Ashiq Uddin Pranto** **1911176108**
- **Md Mamun Miah** **1910476108**
- **Sk. Solaiman Abdullah** **1910976149**

- **Moshiur Rahman** **191047613.51**
- **Mst. Mhamuda Khatun** **1912076104**
- **Adrita Alam** **191227613.53.5**
- **Md. Hasibul Alam** **18103.576148**
- **Selim Reja** **1810676112**

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of the project is to make a Car Rental Management System which will automate the processes involved in managing rental cars including car listing, car inspections, reservation, booking management, rent agreements, rent collection and maintenance requests.

This project is a prototype for the Car Rental Management System which is intended to benefit car owners and car renters. The project has been implemented under the guidance of a university professor.

1.2 Document Abbreviations

This document uses the following conventions.

SRS	System Requirement Specification
CRMS	Car Rental Management System
DB	Database
DDB	Distributed Database
ER	Entity Relationship

1.3 Project Scope

The scope of the CRMS is to ease the car renting system and to create a convenient and easy-to-use interface for both car owners and car renters. The system is based on a relational database with car & car owners' information provided by the car owners and car renters' information provided by the car renter. The whole system is managed by the administrators.

1.4 References

Websites Links:

1. IEEE Recommended Practice for Software Requirements Specification (IEEE-STD-83.50- 1998). Available at https://personal.utdallas.edu/~chung/RE/IEEE83.50-1993.5.pdf
2. A study from the Geeks for geeks website. https://www.geeksforgeeks.org/software-requirement-specification-srs-format/
1. A further study from JavaTpoint https://www.javatpoint.com/software-requirement-specifications
2. Sample SRS study from https://gephi.org/users/gephi_srs_document.pdf

1.5 Overview

The CRMS is a software application that enables car owners and renters to connect and manage cars for rental, payments, and maintenance requests. This system will be designed to be intuitive and user-friendly for both car owners and renters. It will include features such as car listing, rental agreements, rental payments, and maintenance requests. The system will operate on a web-based platform and will be accessible through a web browser.

2. Overall Description

2.1 Product Perspective

The CRMS provides the following information:

- **Car Information**

  Types of car, availability, rent information(pricing), car information(car papers, location, no. of seats, photos or videos, size etc).

- **Carowners' information**

  Personal information including the car owner's name, profession, contact info i.e. cell phone number, mail etc., photo and national ID card. It also includes ratings and feedback from previous car renters.

- **Car renters’ information**

Personal information including the car renters' name, profession, contact info i.e. cell phone number, mail etc., photo, driving license and national ID card. It also includes ratings and feedback from previous car owners.

- **Drivers' information**

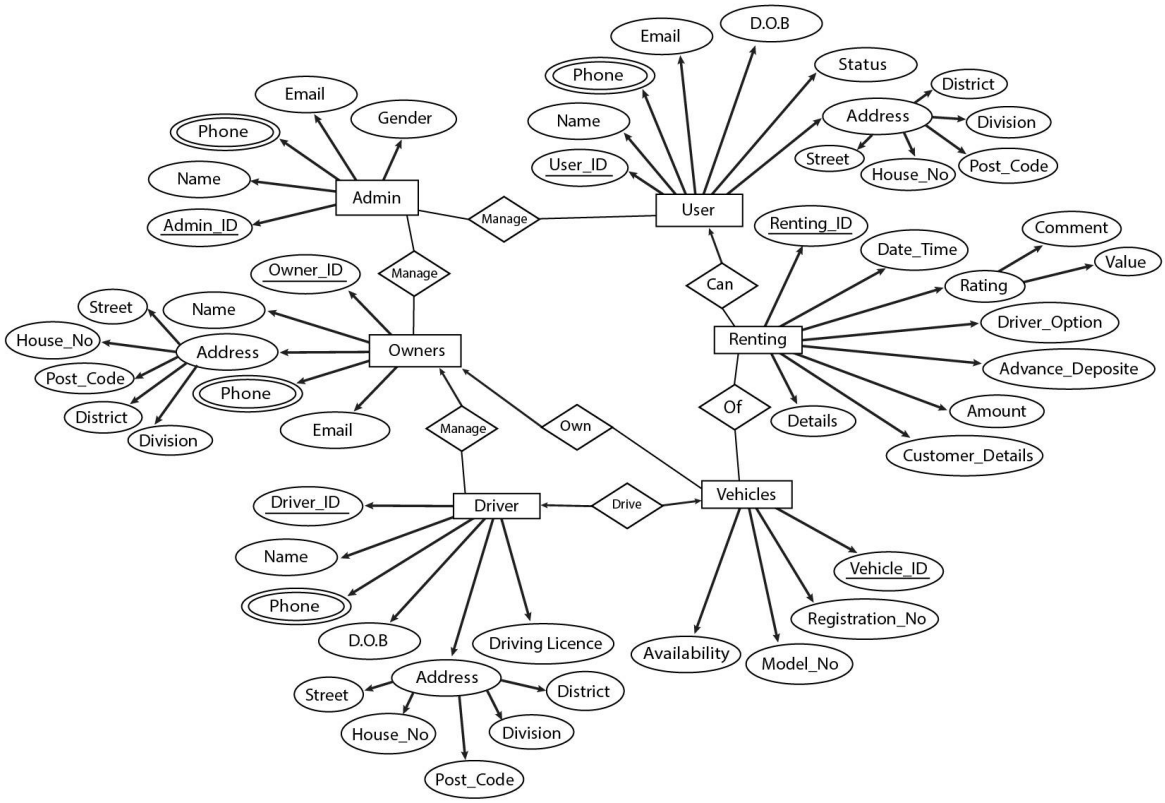
  Personal information including the drivers' name, contact info i.e. cell phone number, mail etc., photo, driving license and national ID card. It also includes ratings and feedback from previous car owners and renters.

- **Payment information**

  Payment information including the amount, date-time, payment method, recipients' and senders' information, due information and payment history.

2.2 Product Features

The major features of the Car Rental Management System as shown in the below Entity-Relationship model (ER model)



E-R Model

2.3 User Characteristics

There will be accounts for car owners, car renters and administrators. The system will support different levels of functionality such as car listing by car owners, searching and reserving cars by car renters, payments management and support from administrators.

The Car owner should be able to do the following functions:

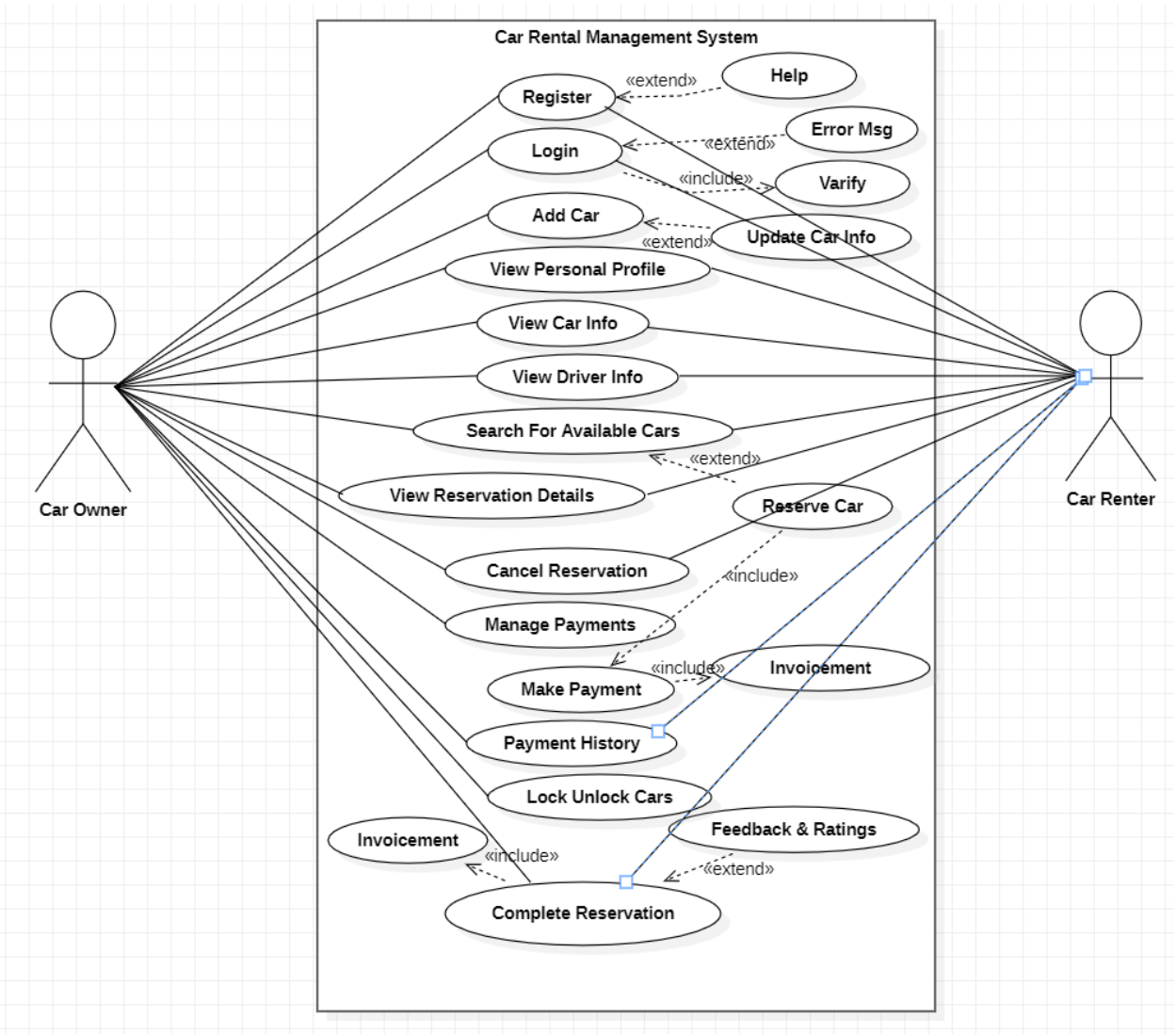
- Create and manage account
- Login with username and password
- Add car, manage and update car information
- Manage payments
- Send notifications to car renter
- Give ratings and feedback

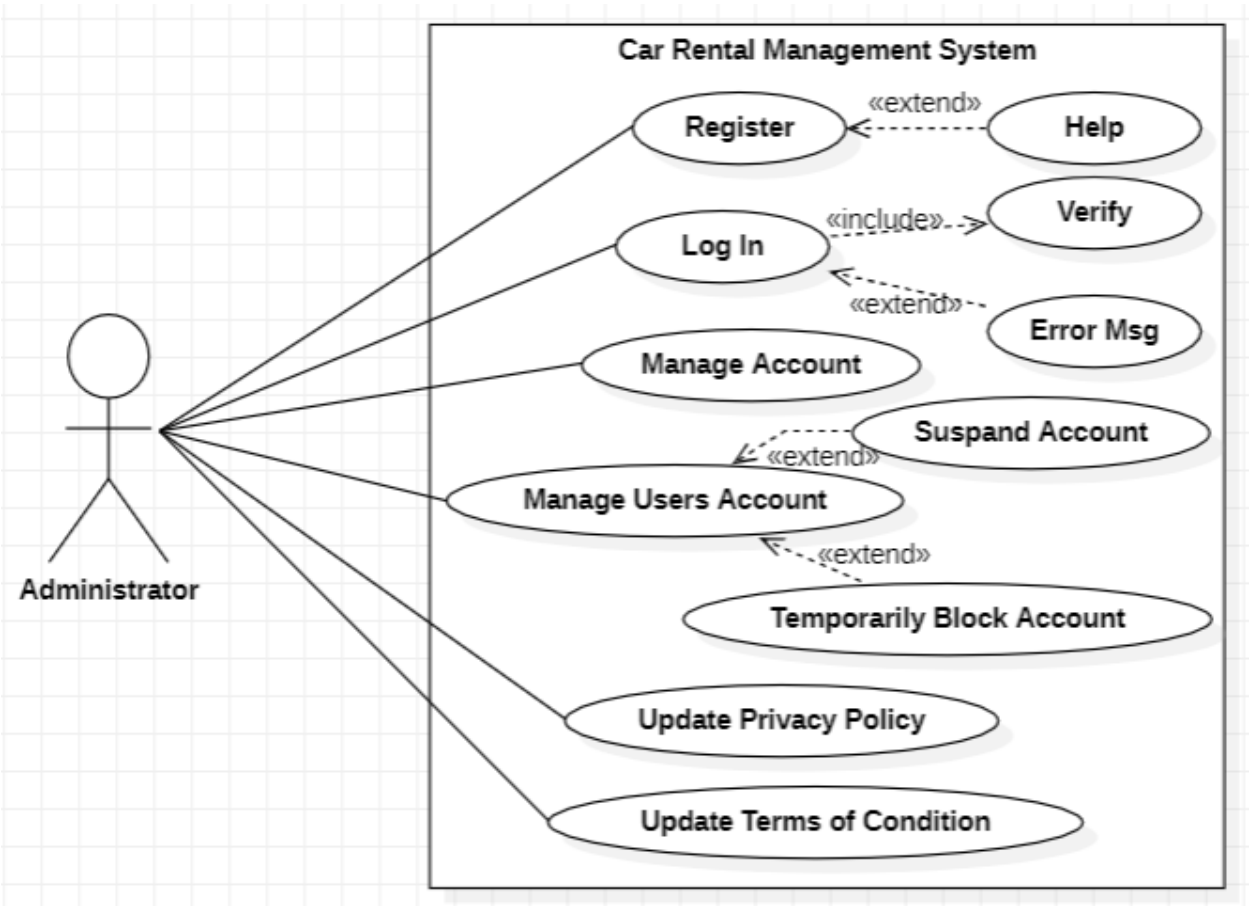
The Car renter should have the following management functions:

- Create and manage account
- Login with username and password
- Search and view cars
- Car booking and car rent agreements
- Make payment and view payment history
- Booking cancellation
- Give ratings and feedback

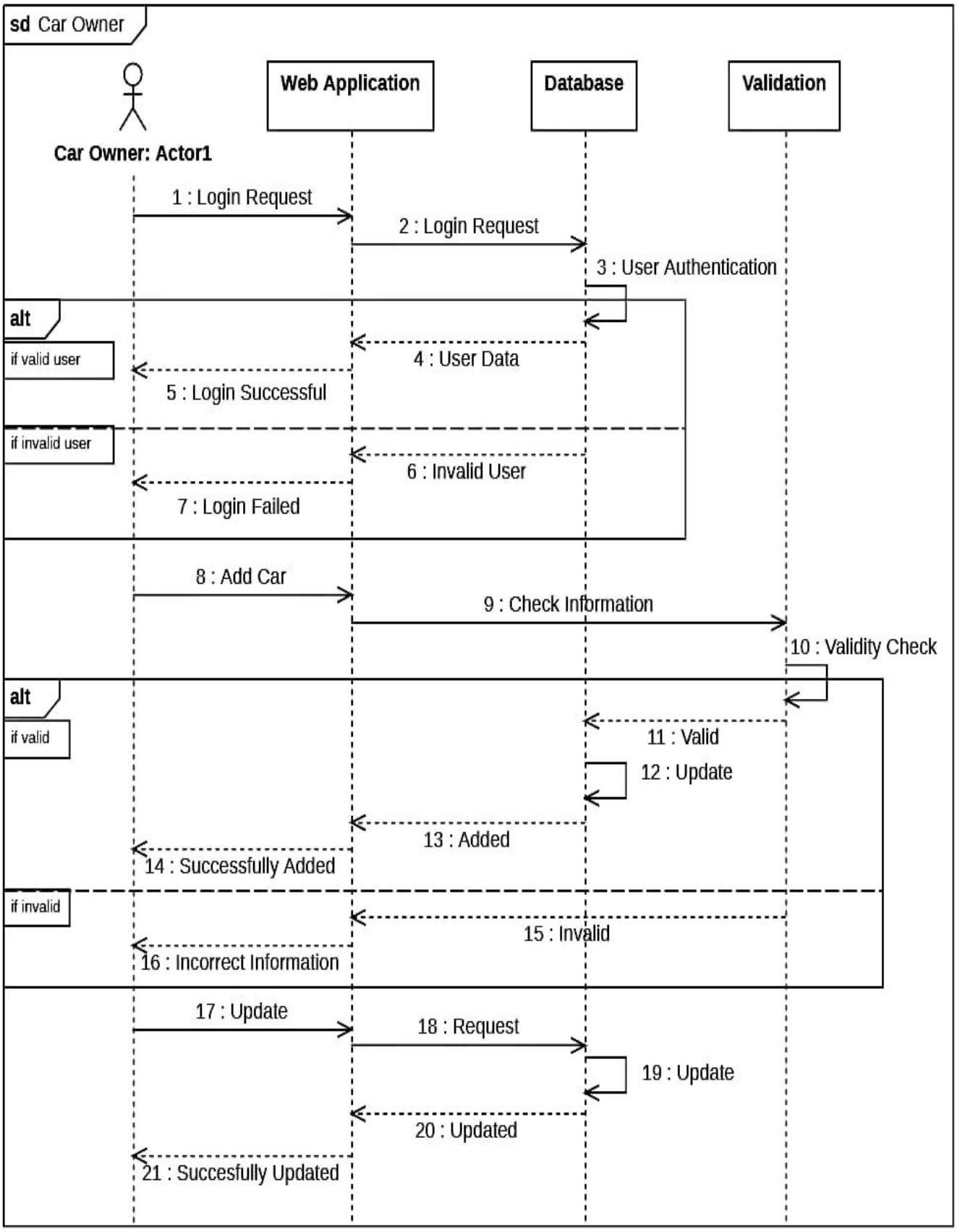
The Admin should have following management functions:

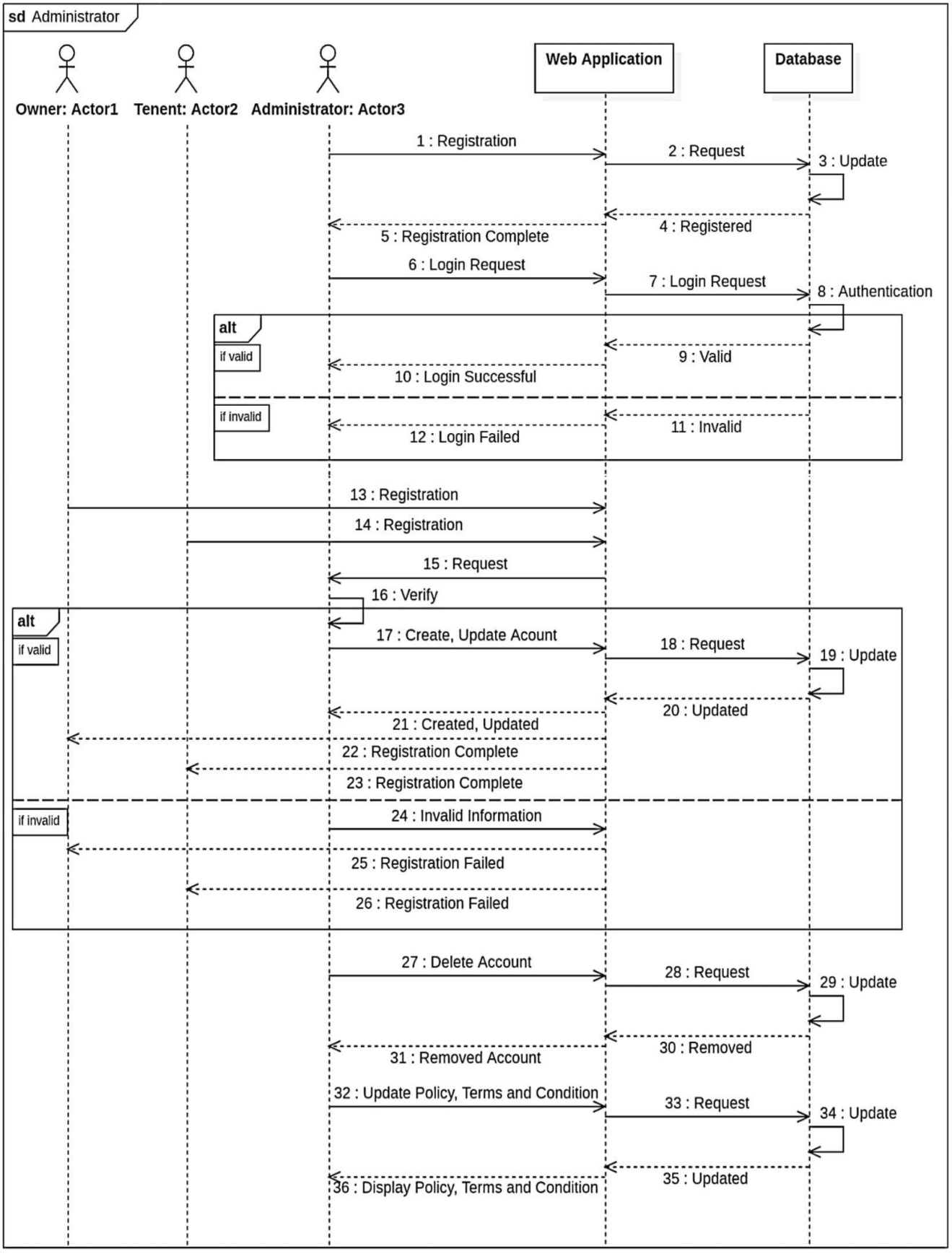
- Create and manage account
- Login with username and password
- Update privacy policy and terms & conditions
- Manage user account i.e. suspend or temporarily block user account

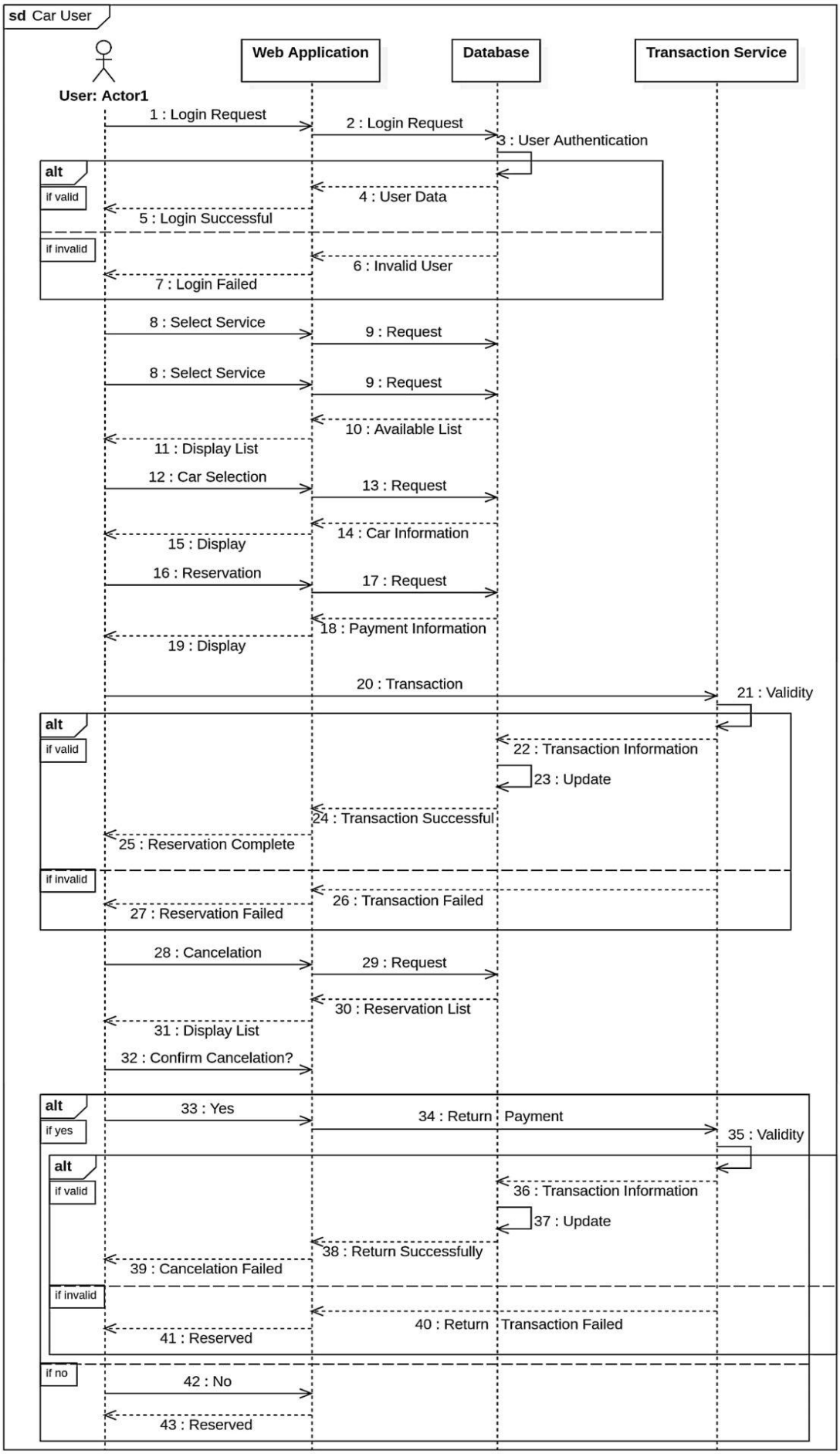

Car Rental Management System
Register
Login
Add Car
View Personal Profile
View Car Info
View Driver Info
Search For Available Cars
View Reservation Details
Cancel Reservation
Manage Payments
Make Payment
Payment History
Lock Unlock Cars
Invoicement
Complete Reservation
Help
Error Msg
Varify
Update Car Info
Reserve Car
Invoicement
Feedback & Ratings
«extend»
«include»
Car Owner
Car Renter



UML (Use Case) Diagram


sd Car Owner
Car Owner: Actor1
Web Application
Database
Validation
1 : Login Request
2 : Login Request
3 : User Authentication
alt
if valid user
4 : User Data
5 : Login Successful
if invalid user
6 : Invalid User
7 : Login Failed
8 : Add Car
9 : Check Information
10 : Validity Check
alt
if valid
11 : Valid
12 : Update
13 : Added
14 : Successfully Added
if invalid
15 : Invalid
16 : Incorrect Information
17 : Update
18 : Request
19 : Update
20 : Updated
21 : Succesfully Updated


sd Administrator
Owner: Actor1
Tenent: Actor2
Administrator: Actor3
Web Application
Database
1 : Registration
2 : Request
3 : Update
4 : Registered
5 : Registration Complete
6 : Login Request
7 : Login Request
8 : Authentication
alt
if valid
9 : Valid
10 : Login Successful
if invalid
11 : Invalid
12 : Login Failed
13 : Registration
14 : Registration
15 : Request
16 : Verify
alt
if valid
17 : Create, Update Acount
18 : Request
19 : Update
20 : Updated
21 : Created, Updated
22 : Registration Complete
23 : Registration Complete
if invalid
24 : Invalid Information
25 : Registration Failed
26 : Registration Failed
27 : Delete Account
28 : Request
29 : Update
30 : Removed
31 : Removed Account
32 : Update Policy, Terms and Condition
33 : Request
34 : Update
35 : Updated
36 : Display Policy, Terms and Condition


sd Car User
User: Actor1
Web Application
Database
Transaction Service
1 : Login Request
2 : Login Request
3 : User Authentication
alt
if valid
4 : User Data
5 : Login Successful
if invalid
6 : Invalid User
7 : Login Failed
8 : Select Service
9 : Request
8 : Select Service
9 : Request
10 : Available List
11 : Display List
12 : Car Selection
13 : Request
14 : Car Information
15 : Display
16 : Reservation
17 : Request
18 : Payment Information
19 : Display
20 : Transaction
21 : Validity
alt
if valid
22 : Transaction Information
23 : Update
24 : Transaction Successful
25 : Reservation Complete
if invalid
26 : Transaction Failed
27 : Reservation Failed
28 : Cancelation
29 : Request
30 : Reservation List
31 : Display List
32 : Confirm Cancelation?
alt
if yes
33 : Yes
34 : Return Payment
35 : Validity
alt
if valid
36 : Transaction Information
37 : Update
38 : Return Successfully
39 : Cancelation Failed
if invalid
40 : Return Transaction Failed
41 : Reserved
if no
42 : No
43 : Reserved

2.4 Operating Environment

- Client-Server system
- Operating system: Windows, Android, iOS, MacOS
- Database: SQL + MariaDB
- Tools & Technologies: Nodejs, Flutter and Dart

2.5 Design and Implementation Constraints

1. The system will be developed using Nodejs, MariaDB, Flutter and Dart.
2. Implement the database at least using a centralized database management system.
3. The development of the system must be completed by 12 April 2023.5.
4. The development of the system may require certain resources such as servers, software tools, and hardware.
5. The system must be secure and protect sensitive data such as tenant and car owner information.
6. The system should be designed in such a way that it can handle a large number of users and properties.
7. The system should comply with relevant laws and regulations related to rental management.

2.6 Assumptions and Dependencies

It is assumed that users will have access to the internet and will be able to use the CRMS without significant technical difficulties.

3. System Features

3.5.1 Functional Requirements

- **Car Listings**

  The CRMS shall allow car owners to create and manage car listings. Each listing shall include the following details:

  1. **Car information:** Car model, type, size, no of seats, registration papers, and availability.
  2. **Rental price:** The rental price for the car.
  3. **Description:** Color, AC/Non-AC, condition, mileage.
  4. **Insurance information:** Details about the cars insurance policy such as insurance company name, policy number and expiration date.

- **Car Owner Info**

  The CRMS shall allow carowners' to create and manage a profile. The profile shall include the following details:

  1. **Personal information:** Name, address, phone number, and any other relevant contact information.
  2. **Cars:** Information about cars and pricing.

- **Car Renters Info**

  The CRMS shall allow car renters to create and manage a profile. The profile shall include the following details:

  1. **Personal information:** Name, address, phone number, and any other relevant contact information.
  2. **Rental preferences:** Location, rental price range, car type, number of seats, and any other relevant search criteria.

- **Rent Management**

  The CRMS may allow car owners to create and manage rents. Each rent must include the following details:

  1. **Rent start and end time:** The start and end time for the rent.
  2. **Rent amount:** The rental amount for the car.
  3. **Security deposit amount:** The amount of the security deposit required for the car.
  4. **Renters information:** The name and contact information of the renter on the rent.
  5. **Rent status:** The status of the rent, such as active or terminated.

- **Payment Processing**

  The CRMS shall allow car renters to make rental payments through the system, and car owners to track payments and issue invoices. The system shall include the following features:

  1. **Payment processing:** The CRMS shall accept rental payments from car renters via credit card, debit card, mobile banking or bank transfer.
  2. **Payment tracking:** The CRMS shall track all rental payments made by car renter and display the payment history for each rent.
  3. **Invoicing:** Property owners shall be able to generate and send invoices for rental payments due from car renters.

- **Reporting**

  The CRMS shall provide reporting features for car owners to track rental income, expenses, and rental rates. The system shall include the following reports:

  1. **Rental income report:** A report that shows the total rental income earned by the car owner over a specified period.

2. **Expense report:** A report that shows the total expenses incurred by the car owner for a specified period, including maintenance, repairs and other costs. It also includes a report of the car renter's expenses.

4. External Interface Requirements

4.1 User Interfaces

The CRMS will have a web-based and application-based user interface that is minimal and user-friendly. The interface shall be accessible on desktop and mobile devices.

4.2 Hardware Interfaces

The CRMS will require an internet connection, a device(computer/smartphone) and a web browser or app to access the system.

4.3 Software Interfaces

Software used	Description
Operating system	We have used the most popular operating systems i.e. Windows, Android, iOS and MacOS which are easily reachable by our user base.
MariaDB	To save information about the car, carowner, car renter and admins.
Flutter and Dart	To implement the project in an android and web platform we have chosen this language and framework for its more interactive and cross-platform support.
Nodejs	To implement the backend of this project we have chosen Nodejs to create restAPI.

4.4 Communications Interfaces

This project will support all types of web browsers and devices(computers and smartphones).

Other Nonfunctional Requirements

4.5 Performance Requirements

The steps involved to perform the implementation of the database are listed below:

A. E-R diagram

The E-R Diagram constitutes a technique for representing the logical structure of a database in a pictorial manner. This analysis is then used to organize data as a relation, normalize relation and finally obtain a relation database.

B. Normalization

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to a wastage of storage space and an increase in the total size of the data stored.

Normalization is the process of breaking down a table into smaller tables. So that each table deals with a single theme. There are three different kinds of modifications of anomalies and formulating the first, second and third normal forms is considered sufficient for most practical purposes. It should be considered only after a thorough analysis and complete understanding of its implications.

4.6 Safety Requirements

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

4.7 Security Requirements

Security systems need database storage just like many other applications.

4.8 Technical Constraints

The CRMS shall be developed using Nodejs for the back-end, Flutter and Dart for the front-end and MariaDB for the database. The system shall be hosted on Amazon Web Services (AWS) using a Linux-based server.

4.9 Business Constraints

The CRMS shall comply with all applicable laws and regulations related to rental car management and payment processing. The system shall be compatible with major web browsers, including Google Chrome, Mozilla Firefox, Safari etc and popular Android and iOS versions.

4.10 Software Quality Attributes

- **Availability:** The CRMS should be available 24/7.
- **Correctness:** The system must provide accurate search results.
- **Maintainability:** The administrators should maintain correct schedules for updating the system.
- **Usability:** The system should satisfy a maximum number of customers' needs.
- **Scalability:** The CRMS shall be scalable to handle an increasing number of users.

5. Glossary

- **Use Case Diagram:** A use case diagram is a way to summarize details of a system and the users within that system.
- **E-R Diagram:** E-R diagram stands for an Entity-Relationship diagram. It is a high-level data model. This model is used to define the data elements and relationships for a specified system.
- **Operating System:** An operating system (OS) is system software that manages computer hardware and software resources, and provides common services for computer programs.
- **SQL:** SQL(Structured Query Language) is a standard language for accessing and manipulating databases.
- **Distributed Database:** A distributed database is a database that consists of two or more files located on different sites either on the same network or on entirely different networks.
- **Front-end:** The layer above the back end is the front end and it includes all software or hardware that is part of a user interface.
- **Back-end:** The back end refers to parts of a computer application or a program's code that allow it to operate and that cannot be accessed by a user.

3.5 Use Case and Data Flow Description

3.5.1 Registration

Use Case ID: USC001

Description: This use case describes the process of registration.

Actor: All kinds of users.

Precondition:

- The actor must have a phone number and email address.
- The actor must input the required information for registration.

Post Condition:

- The system will save the registration information to the database.
- The actor will receive a confirmation email.
- The actor will activate the account by clicking the link in the email.
- The actor will log in to the system.

Normal Flows:

- **Actor:** The actor will input the required information for the registration and click on the “Register” button.
- **System:** The system will validate the registration information and save it to the database. The system will also send a confirmation email to the actor.
- **Actor:** The actor will receive the confirmation email and click on the link in the email to activate their account.
- **System:** The system will verify the activation link and show a success message to the actor. The system will also redirect the actor to the login page.

3.5.2 Login

Use Case ID: USC002

Actor: All kinds of users.

Description: This use case describes the process of login.

Precondition:

- The actor must have the user id and password.

Post Condition:

- The system will check the user id and password from the system database.
- If the user id and password are correct, the system will log in the user to the system according to their user type and show their profile page.
- If the user id and password are incorrect, the system will show an error message.

Normal Flows:

- **Actor:** The actor will enter the user id and password to the system and click on the “Login” button.
- **System:** The system will check the user id and password from the database system. If the user id and password are correct, the system will allow the user to log in and show their home page. If the user id and password are incorrect, the system will show an error message.

3.5.3 View Personal profile

Use Case ID: USC003.5

Description: This use case describes the process of viewing personal profile.

Actor: Car owner, Car renter

Precondition:

- The actor must be an authorized user and have to log in.
- The actor must have permission to view his personal profile.

Postcondition:

- The system must show the personal profile information.

Normal flow:

- **Actor:** At first actor will press the ‘profile’ button.
- **System:** Then the system will show the personal profile information, such as name, email, phone number, address, rating, reviews, etc.

3.5.4 View Car Info

Use Case ID: USC004

Description: This use case describes the process of viewing car information.

Actor: Car owner, Car renter

Precondition:

- The actor must be an authorized user and have to log in.
- The actor must have permission to view car information.

Postcondition:

- The system must show the car information, such as model, type, color, price, location, availability, rating, reviews, etc.

Normal flow:

- **Actor:** At First actor will press the ‘car info’ button on the car list page or the reservation details page.
- **System:** Then the system will show the car information.

3.5.5 View Driver Info

Use Case ID: USC003.5

Description: This use case describes the process of viewing driver information.

Actor: Car owner, Car renter

Precondition:

- The actor must be an authorized user and have to log in.
- The actor must have permission to view driver information.

Postcondition:

- The system must show the driver's information, such as name, license number, phone number, rating, reviews, etc.

Normal flow:

- **Actor:** At first actor will press the ‘driver info’ button on the car list page or the reservation details page.
- **System:** Then the system will show the driver's information.

3.5.6 Make Payment

Use Case ID: USC006

Description: This use case describes the process of making a payment.

Actor: Car renter

Precondition:

- The actor must be a car renter and have to log in.
- The actor must have reserved a car and confirmed the reservation.
- The actor must have a valid payment method.

Postcondition:

- The system must deduct the payment amount from the actor's account or card.
- The system must send an invoice to the car owner and the car renter.
- The system must update the payment status of the reservation.

Normal flow:

- **Actor:** At first actor will press the 'pay' button on the reservation details page.
- **System:** Then the system will show the payment amount and ask the actor to choose a payment method, such as credit card, debit card, mobile banking, etc.
- **Actor:** The actor will select a payment method and enter the required information, such as card number, expiry date, CVV, etc.
- **System:** The system will verify the payment information and process the payment. If the payment is successful, the system will show a confirmation message and send an invoice to the car owner and the car renter via email. If the payment fails, the system will show an error message and ask the actor to try again or choose another payment method.

3.5.7 Search for Available Car

Use Case ID: USC007

Description: This use case describes the process of searching for cars.

Actor: Car Renter, Car Owner

Precondition:

- The actor must be an authorized user and have to log in (except for an unauthorized renter).
- The actor must have permission to search cars (except for unauthorized renters).
- The actor must categorize the car by selecting a size, type, price, location, or color.

Post Condition:

- The system must show the car information.

Normal Flows:

- **Actor:** The actor will input the car information (such as name and type) and click on the “Search” button.
- **System:** The system will show the car details that match the search criteria.

3.5.8 Add Car

Use Case ID: USC008

Description: This use case describes the process of adding a car to the system.

Actor: Car Owner

Precondition:

- The actor must be a car owner and have to log in.
- The actor must have permission to add a car to the system.
- The actor must have the necessary information about the car, such as model, type, color, price, location, etc.

Post Condition:

- The system will save the car information to the database.
- The system will display the car information on the car owner’s profile page.

Normal Flows:

- **Actor:** The actor will click on the “Add Car” button on their profile page.
- **System:** The system will show a car form with fields for entering the car information.

- **Actor:** The actor will fill in the car form with the required information and click on the “Submit” button.
- **System:** The system will validate the car information and save it to the database. The system will also show a confirmation message to the actor.
- **System:** The system will display the car information on the car owner’s profile page. The system will also make the car available for searching and reservation by other actors.

3.5.9 Payment History

Use Case ID: USC009

Description: This use case describes the process of seeing payment history.

Actor: Car Renter, Car Owner

Precondition:

- The actor must have to log in.
- The actor must have permission to see history.

Post Condition:

- The system must show the payment history information.

Normal Flows:

- **Actor:** The actor will click on the “History” button on their profile page.
- **System:** The system will show the payment history information of the actor.

3.5.10 Reserve a Car

Use Case ID: USC010

Description: This use case describes the process of reservation.

Actor: Car Renter

Precondition:

- The actor must be a car renter and have to log in.
- The car must be available.

- The actor must select a car and make payment for the reservation.

Post Condition:

- The system will save the reservation information to the database.
- The system will send an invoice to the car owner.
- The system will show a confirmation message to the customer. The system also provides a cancel reservation method.

Normal Flows:

- Actor: The actor will select a car and click on the “Reserve” button.
- System: The system will show the payment gateway page.
- **Actor:** The actor will perform all the payment-related tasks and click on the “Confirm” button.
- **System:** The system will validate the payment information and save it to the database. The system will also send an invoice to the car owner and a confirmation message to the customer.

3.5.11 Cancel Reservation

Use Case ID: USC011

Description: This use case describes the flows of cancellation of a reservation.

Actors: Car Renter, Car Owner.

Precondition:

- Actors must be authorized users and have to log in to the system.
- Actors must have permission to cancel a reservation.
- Actors must have a valid reservation id for cancellation.

Post Condition:

- The system will update the information in the database and show a confirmation message. The system will also send a cancel reservation message to the other actor.

Normal Flows:

- **Actor:** The actor will input the car name and corresponding reservation id for cancellation and click on the “Cancel” button.

- **System:** The system will check the reservation id and update the information in the database. The system will also show a confirmation message to the actor and send a cancel reservation message to the other actor.

3.5.12 Feedback and Ratings

Use Case ID: USC012

Description: This use case describes the process of giving and receiving feedback and ratings.

Actor: Car Renter

Precondition:

- The actor must be a car renter and have to log in.
- The actor must have to complete a reservation.
- The actor must have permission to give and receive feedback and ratings.

Post Condition:

- The system will save the feedback and ratings information to the database.
- The system will display the feedback and ratings information on the profile page of the car owner.

Normal Flows:

- **Actor:** The actor will select a completed reservation transaction and click on the “Give Feedback” button.
- **System:** The system will show a feedback form with a rating scale and a comment box.
- **Actor:** The actor will rate the other actor on a scale of 1 to 3.5 stars and write a comment about their experience.
- **System:** The system will validate the feedback and ratings information and save it to the database. The system will also show a confirmation message to the actor.
- **System:** The system will display the feedback and ratings information on the profile page of the actors. The system will also calculate the average rating of each actor based on their feedback history.

3.5.13 Lock and Unlock Cars

Use Case ID: USC013.5

Description: This use case describes the process of locking and unlocking cars.

Actor: Car Owner

Precondition:

- The actor must be a car owner and have to log in.
- The actor must have permission to lock and unlock cars.
- The actor must have access to the car database.

Post Condition:

- The system will update the car status in the database and show a confirmation message.

Normal Flows:

- **Actor:** The actor will select a car from the car database and click on the “Lock” or “Unlock” button.
- **System:** The system will check the car status and perform the action. The system will also update the car status in the database and show a confirmation message to the actor.

3.5.14 Manage User Account

Use Case ID: USC014

Description: This use case describes the process of managing user accounts.

Actor: Administrator

Precondition:

- The actor must be an administrator and have to log in.
- The actor must have permission to manage user accounts.
- The actor must have access to the user database.

Post Condition:

- The system will update the user account information in the database and show a confirmation message.

Normal Flows:

- **Actor:** The actor will select a user account from the user database and click on the “Manage” button.

- **System:** The system will show the user account details and the available actions, such as suspend, block, activate, or delete.
- **Actor:** The actor will choose an action and click the “Confirm” button.
- **System:** The system will perform the action and update the user account information in the database. The system will also show a confirmation message to the actor and send a notification to the affected user.

3.5.15 Update Privacy Policy

Use Case ID: USC013.5

Description: This use case describes the process of updating the privacy policy of the car rental system.

Actor: Administrator

Precondition:

- The actor must be an administrator and have to log in.
- The actor must have permission to update the privacy policy.

Postcondition:

- The system must save the updated privacy policy to the database.
- The system must notify the users about the changes in the privacy policy.

Normal flow:

- **Actor:** At first actor will press the ‘privacy policy’ button on the admin dashboard page.
- **System:** Then the system will show the current privacy policy and allow the actor to edit it.
- **Actor:** The actor will make the necessary changes to the privacy policy and press the ‘save’ button.
- **System:** The system will validate the changes and save them to the database. The system will also send an email notification to all the users about the updated privacy policy.