NATIONALUNIVERSITY OF MODERN LANGUAGES
ISLAMABAD

Github linkedin

# SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

**Project Title: CarRental-Software**

Submitted By: **Obaid Ullah**
Department: **BS Information Technology**
Session: 20**23** – 20**27**

Github-Project-Link

# SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

## CAR RENTAL SOFTWARE:

### 1. Introduction:

#### 1.1 Purpose:

Car rental software is a specialized application designed to streamline and automate the management of a car rental business. It serves as a comprehensive solution to handle various aspects of renting and managing cars, customers, rentals, and returns. The primary purpose of this software is to provide a user-friendly platform that facilitates efficient and organized operations for car rental businesses.

#### 1.2 Scope:

The scope of the car rental software extends to the entire car rental process, from user authentication to car management, customer registration, rental transactions, and car returns. It aims to enhance the overall efficiency of a car rental business by automating manual processes, ensuring accurate data management, and providing a seamless experience for both administrators and users.

#### 1.3 General Description:

The car rental software comprises several interfaces that cater to specific functionalities within the car rental business. Each interface addresses a key aspect of the process, ensuring a systematic and well-organized workflow.

### 2. Specific Requirements:

### 3. Functional Requirements:

#### 1. User Authentication:

- The system must authenticate users based on provided credentials.
- Admins have special privileges.

#### 2. Car Management:

- Admins can add, edit, and delete car information.
- The system must keep track of available and booked cars.

author: Obaid-Ullah

### 3. Customer Management:

- Admins can register and delete customers.
- Customer details are stored in the database.

### 4. Rental Management:

- Users can rent a car by providing necessary details.
- The system updates the car tables accordingly.

### 5. Return Management:

- Users can return cars, and the system updates the car status.
- Fines are calculated for delayed returns.

### 6. Fine Generation and Slip:

- Users can generate fines based on return information.
- Slip information is stored and displayed.

## 4. Non-Functional Requirements:

### 1. Security:

- User passwords must be securely stored.
- Access to certain functionalities is restricted to admins.

### 2. Performance:

- The system should handle a reasonable number of concurrent users.
- Database queries for retrieving and storing information should be optimized.

### 3. Usability:

- User interfaces should be intuitive.
- Error messages should be clear and informative.

### 4. Reliability:

- The system should be available and reliable for use during operational hours.
- Regular backups of the database should be performed.

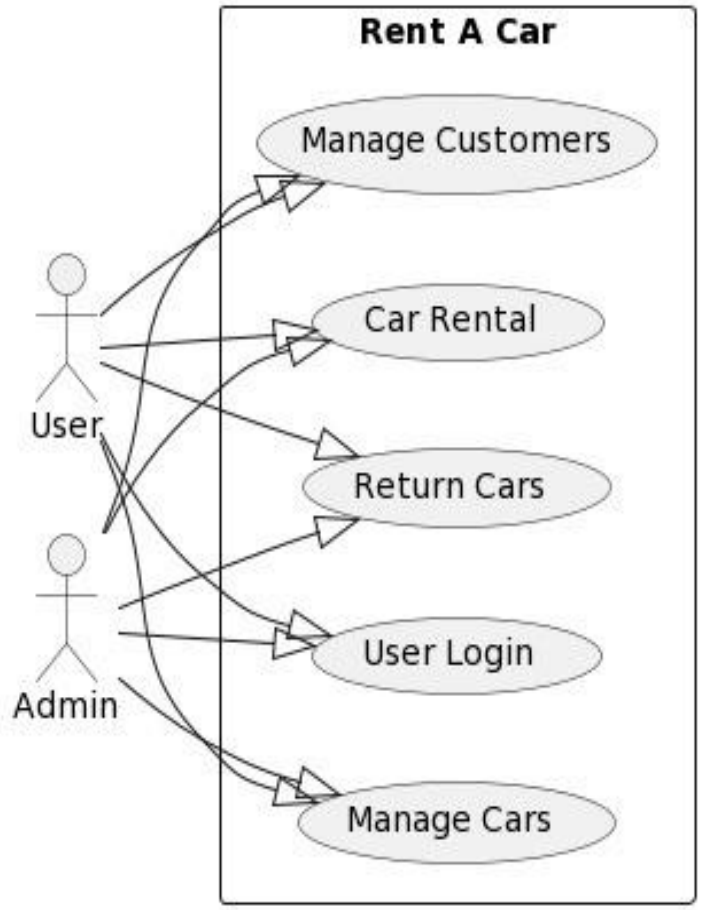
### 5. Scalability:

 The system should be able to scale with an increasing number of cars, customers, and rentals.

**6. Data Integrity:**

- The data in the database should be accurate and consistent.
- Proper validation should be in place to prevent data corruption.

# 5. Use cases Diagram




**Use cases**

## 1. User Login

**Preconditions:** 

**Actor:** System User

**Preconditions:**

- The car rental software is installed and operational.
- User credentials (username and password) are set by the user requirement.

**Description:**

1. The user launches the car rental software.
2. The system presents a login interface with fields for username and password.
3. The user enters their valid credentials.

**Postconditions:**

- The system verifies the entered credentials by querying the user name
- If the credentials are valid:
- The system grants access to the Manage Cars interface.
- If the credentials are invalid:
- The system displays an error message, prompting the user to enter correct credentials.

## 2. Manage Cars

- **Actor:** Manager Admin

**Preconditions:**

- User is logged into the system.
- Car data is stored in the system database.

**Description:**

1. The admin navigates to the "Manage Cars" interface.
2. The admin selects the option to add a new car.
3. The system prompts the user to enter details for the new car (e.g., car model, registration number, etc.).
4. The admin submits the new car details.

**Postconditions:**

- The system adds the new car to the car database.
- The "Available Cars" table is updated by querying the car database to reflect the addition of the new car.

## 3. Register a Customer

### Preconditions:

- **Actor:** System User

### Preconditions:

- User is logged into the system.
- Customer data is stored in the customer database.

### Description:

1. The user navigates to the "Manage Customer" interface.
2. The user selects the option to register a new customer.
3. The system prompts the user to enter details for the new customer (e.g., customer name, address, phone number, etc.).
4. The user submits the new customer details.

### Postconditions:

- The system adds the new customer to the customer database.
- The "Registered Customers" table is updated by querying the customer database to reflect the addition of the new customer.

## 4. Rent a Car

### Preconditions:

- **Actor:** System User

### Preconditions:

- User is logged into the system.
- Car and customer data are stored in the respective databases.

### Description:

1. The user navigates to the "Car Rental" interface.
2. The user selects the option to rent a car.
3. The system prompts the user to enter details for the rental (e.g., customer name, rental date, return date, etc.).
4. The user submits the rental details.

**Postconditions:**

- The system records the rental information in the rental database.
- The "Available Cars" and "Rented Cars" tables are updated by querying the car and rental databases to reflect the changes.

## 5. Return Cars

**Preconditions:**

- **Actor:** System User

**Preconditions:**

- User is logged into the system.
- Rental data is stored in the rental database.

**Description:**

1. The user navigates to the "Return Cars" interface.
2. The user selects a rented car from the table for return.
3. The system displays details of the selected rental in text fields.
4. The user confirms the return.

**Postconditions:**

- The system updates the status of the returned car in the car database.
- If the return is within the specified time, no fines are generated.
- If there's a delay, the system calculates fines and updates the fine information in the fine database.

## 6. Fine Generation and Slip

**Preconditions:**

- **Actor:** System User

**Preconditions:**

- User is logged into the system.
- Car return information is available in the rental and fine databases.

**Description:**

1. The user navigates to the "Fine Generation and Slip" interface.

2. The system displays the details of the returned car, including return date and any delay.
3. The user enters the delay (if any) and presses the "GENERATE" button.

**Postconditions:**

- The system calculates and displays fines (if applicable) by querying the rental and fine databases.
- The generated fine information is stored in the fine database.
- The system updates the slip information in the slip table by querying the fine database.