Software Requirement and Design Specifications

Rental Car Hub

Course Code	CS 3009
Instructor	Miss Noureen Fatima
Project Team	Syed Muhammad Zaid (21k3348) Hassan Abbas (21K3286)
Submission Date	12/05/2024

Table of Contents

1. Introduction

1.1. Purpose of Document

This document serves as a comprehensive guide and reference for the Rental Car Hub project. It outlines the key aspects, functionalities, and components of the system to facilitate understanding, implementation, and maintenance.

The primary purposes of this document are:

- Provide Clarity: Clearly define the objectives, scope, and requirements of the Rental Car Hub .
- Guidance: Offer guidance for developers, administrators, and stakeholders involved in the project.
- Reference: Serve as a reference document throughout the development, testing, and maintenance phases.
- Communication: Facilitate effective communication among project team members and stakeholders.

1.2. Intended Audience

This document is intended for individuals and groups involved in the Rental Car Hub project. The primary audience includes:

- Developers: Those responsible for designing, coding, testing, and implementing the RCH.
- Administrators: System administrators and IT personnel responsible for deploying and maintaining the system.
- Stakeholders: Individuals and groups with an interest in the RCH project, including Hub Admin, Partner, and user.
- Quality Assurance (QA) Teams: Teams responsible for testing and ensuring the quality and reliability of the system.
- Project Managers: Those overseeing the planning, execution, and delivery of the RCH project.

It is essential for the intended audience to review and understand this document to ensure alignment with project goals and requirements.

2. Overall System Description

2.1. Project Background

The Car Rental Hub project aims to develop a web-based application for car rentals and management. The project originated from the need to streamline car rental operations, improve customer experience, and enhance administrative efficiency. The existing challenges in manual car rental management, payment processing, and inventory management prompted the development of this system. Additionally, the incorporation of multiple payment options, including credit cards, online payment, and cash, aims to cater to diverse customer needs and preferences.

2.2. Project Scope

The project scope encompasses the development of a comprehensive Car Rental Hub system with the following main functionalities:

Customer Module:

- Search and book cars
- View car details and availability
- Make payments and manage bookings

Admin Module:

- Add and delete car records
- Manage car inventory and availability
- Handle customer bookings and payments

Partner Module:

- Manage partner car rentals and availability
- Receive payments and manage bookings

The system will not include advanced car management features, external integrations beyond the car rental hub's immediate needs, or functionalities unrelated to core car rental management.

2.3. Not In Scope

- The following functionalities are explicitly excluded from the project scope:
- Advanced car management features beyond car rental and inventory management
- Integration with external systems or services beyond the immediate scope of the car rental hub
- Any features not directly related to core car rental management and administration

2.4. Project Objectives

The primary objectives of the project are to:

- Enhance customer experience through streamlined car rental management and diverse payment options
- Improve operational efficiency for administrators by providing a centralized system for car management and customer bookings
- Empower partners with tools for efficient car rental management and payment processing
- Facilitate diverse payment methods (credit cards, online payment, cash) for a seamless financial transaction experience

2.5. Stakeholders

Stakeholders in the system include:

- Customers: Interact with the system for car rental bookings, payments, and management
- Administrators: Oversee the entire car rental hub system, including car inventory, customer bookings, and payments
- Partners: Utilize the system for car rental management and payment processing
- Developers/Technical Team: Involved in the software development and management of the system

2.6. Operating Environment

- The software will operate in a standard car rental hub environment with the following specifications:
- Hardware: Standard server infrastructure
- Operating System: Compatible with major operating systems such as Windows, Linux, and macOS
- Network Environment: Requires a stable network connection for real-time data access

2.7. System Constraints

The successful implementation of the Car Rental Hub system is subject to various constraints imposed by the

external environment. These constraints, originating from stakeholders, business conditions, technical considerations, legal requirements, and other factors, include:

Software Constraints:

- Compatibility with existing software infrastructure, adherence to industry standards

Hardware Constraints:

- Meeting minimum hardware requirements, compatibility with server infrastructure

Cultural Constraints:

- Language considerations, user interface sensitivity to cultural preferences

Legal Constraints:

- Compliance with data privacy laws and regulations, adherence to legal standards

Environmental Constraints:

- Operating in designated environment, considering noise levels and potential disruptions

User Constraints:

- User interface design catering to varying technical expertise, consideration of user demographic

3. External Interface Requirement

3.1. Hardware Interfaces

The Car Rental Hub will interact with various hardware components to ensure seamless operation. Key hardware interfaces include:

Server Infrastructure:

- The system will be hosted on a cloud-based server infrastructure, requiring compatibility with server hardware specifications.

Client Devices:

- Customer, administrator, and partner interactions will occur through desktop computers, laptops, tablets, and smartphones.

Payment Terminals:

- Interfaces with payment terminals for online payment and cash transactions.

3.2. Software Interfaces

The Car Rental Hub will interface with various software components to facilitate data exchange and interoperability. Software interfaces include:

Database Management System (DBMS):

- The system will interact with a relational database to store and retrieve car rental information, customer data, and payment records.

Operating System:

- Compatibility with major operating systems, including Windows, Linux, and macOS.

Web Browsers:

- The system will be accessible through standard web browsers such as Google Chrome, Mozilla Firefox, and Safari.

Payment Gateway Services:

- Integration with payment gateway services for online transactions.

Third-Party Libraries and Frameworks:

- Utilization of Java framework and other libraries for efficient development.

3.3. Communications Interfaces

Efficient communication is essential for the Car Rental Hub to function effectively. Communication interfaces include:

Network Communication Standards:

- The system will adhere to standard network communication protocols such as HTTPS for secure data transmission.

Web Services:

- APIs for integrating with external systems and services.

Data Transfer Rates:

- Minimum data transfer rates to ensure timely access to information.

Security Protocols:

- Implementation of encryption protocols to secure sensitive data during transmission.

Synchronization Mechanisms:

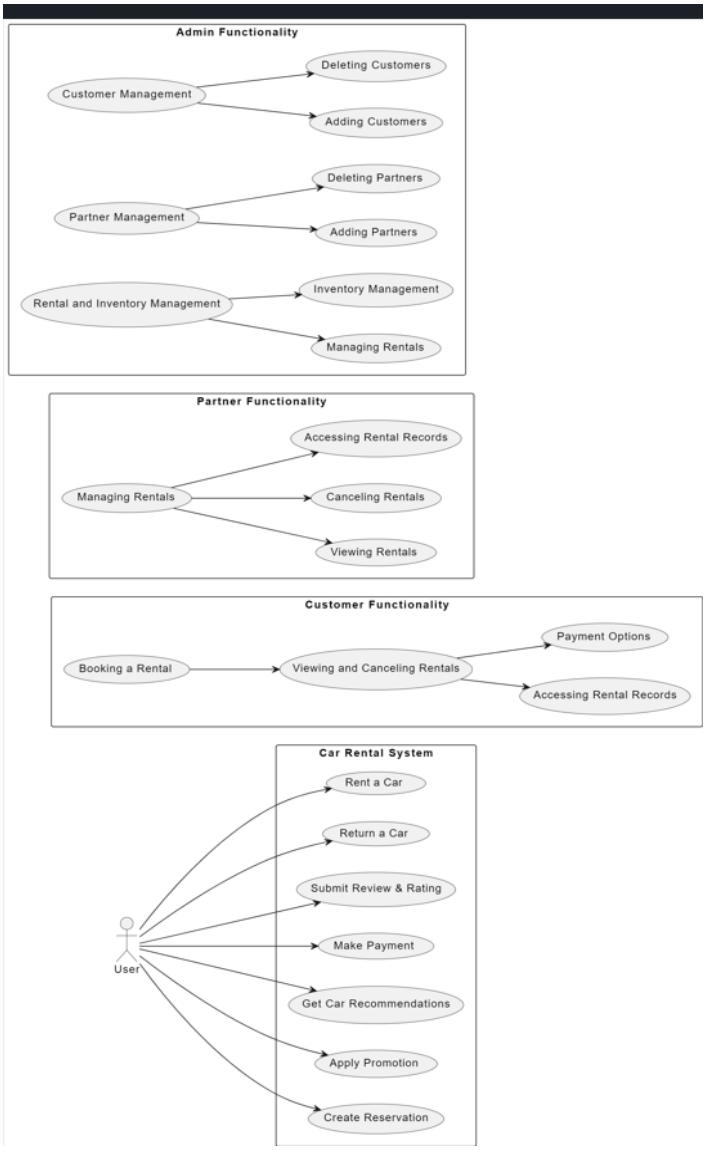
- Real-time synchronization of data across different modules to maintain consistency.

4. Functional Requirements

4.1. Functional Hierarchy

The Car Rental Hub is structured into distinct modules, each catering to specific functionalities:

4.2. Use Cases



5. Non-functional Requirements

5.1. Performance Requirements

The Car Rental Hub is expected to meet the following performance characteristics:

- Concurrency: The system must support concurrent access by multiple users without performance degradation, accommodating the car rental hub's operational demands.
- Reliability: The system should achieve an uptime of at least 99.9%, ensuring continuous availability for users.
- Scalability: The architecture should be scalable to accommodate future increases in data volume and user base.

5.2. Safety Requirements

To ensure the safety of users and data, the following requirements are established:

Data Integrity:

- Safeguards should be in place to prevent data corruption or loss, ensuring the accuracy of car rental records and financial transactions.

Error Handling:

- The system should provide clear error messages to users, guiding them in case of mistakes to prevent accidental actions that may lead to harm (e.g., entering incorrect payment information).

5.3. Security Requirements

Security is a top priority for the Car Rental Hub. The following security requirements are identified.

User Authentication:

- Implement secure user authentication mechanisms, such as username/password authentication, to ensure authorized access.

User Authorization:

- Define role-based access control to restrict access to specific functionalities based on user roles (customer, admin, partner).

Data Encryption:

- Use encryption protocols for secure data transmission over the network.

Privacy Compliance:

- Ensure compliance with data privacy regulations, safeguarding customer information.

5.4. User Documentation

Customer, Admin, Partner Manuals:

- Step-by-step guides for users and administrators, including screenshots and examples.

Online Help:

- Context-sensitive assistance, including integrated tooltips and pop-up guides.

Tutorials:

- Video demos and practice exercises.

FAQ Section:

- Common queries with quick solutions.

Contact Information:

- Support channels, including technical support details.

6. System Architecture

6.1. System Level Architecture

The Car Rental Hub is decomposed into the following elements:

Customer Module:

- Handles customer-related functionalities such as car rental booking, cancellation, viewing rental records, and payment.

Admin Module:

- Manages administrative tasks, including car management, customer management, booking handling, and inventory management.

Partner Module:

- Governs partner-related activities, including car rental management, booking management, and payment processing.

The relationships between the modules are defined as follows:

Customer Module:

- Interacts with the Admin Module for car rental booking, cancellation, and account management.

Admin Module:

- Manages both Customer and Partner Modules. Coordinates car rental booking, cancellation, and inventory management.

Partner Module:

- Collaborates with the Admin Module for car rental management and payment processing.

The Car Rental Hub interfaces with external systems for the following purposes:

Payment Gateway:

- Handles online payment transactions for car rental bookings.

External Databases:

- Interfaces with external databases for data exchange and synchronization.

Major physical design considerations include:

Server Deployment:

- Determines where each module will execute, whether on a centralized server or distributed across multiple servers.

Load Balancing:

- Considers load balancing strategies for optimizing system performance during peak times.

Global design strategies encompass:

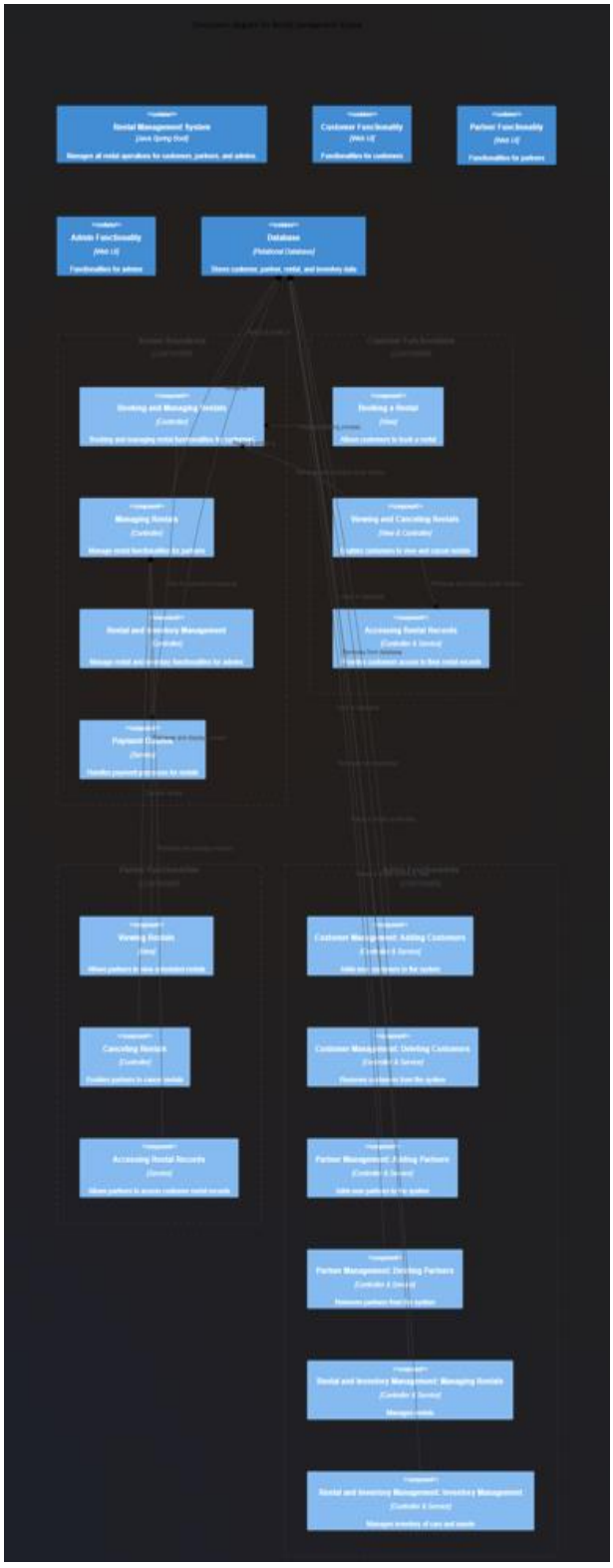
Error Handling:

Establishes strategies for handling errors at the system level, ensuring graceful degradation and user-friendly error messages.

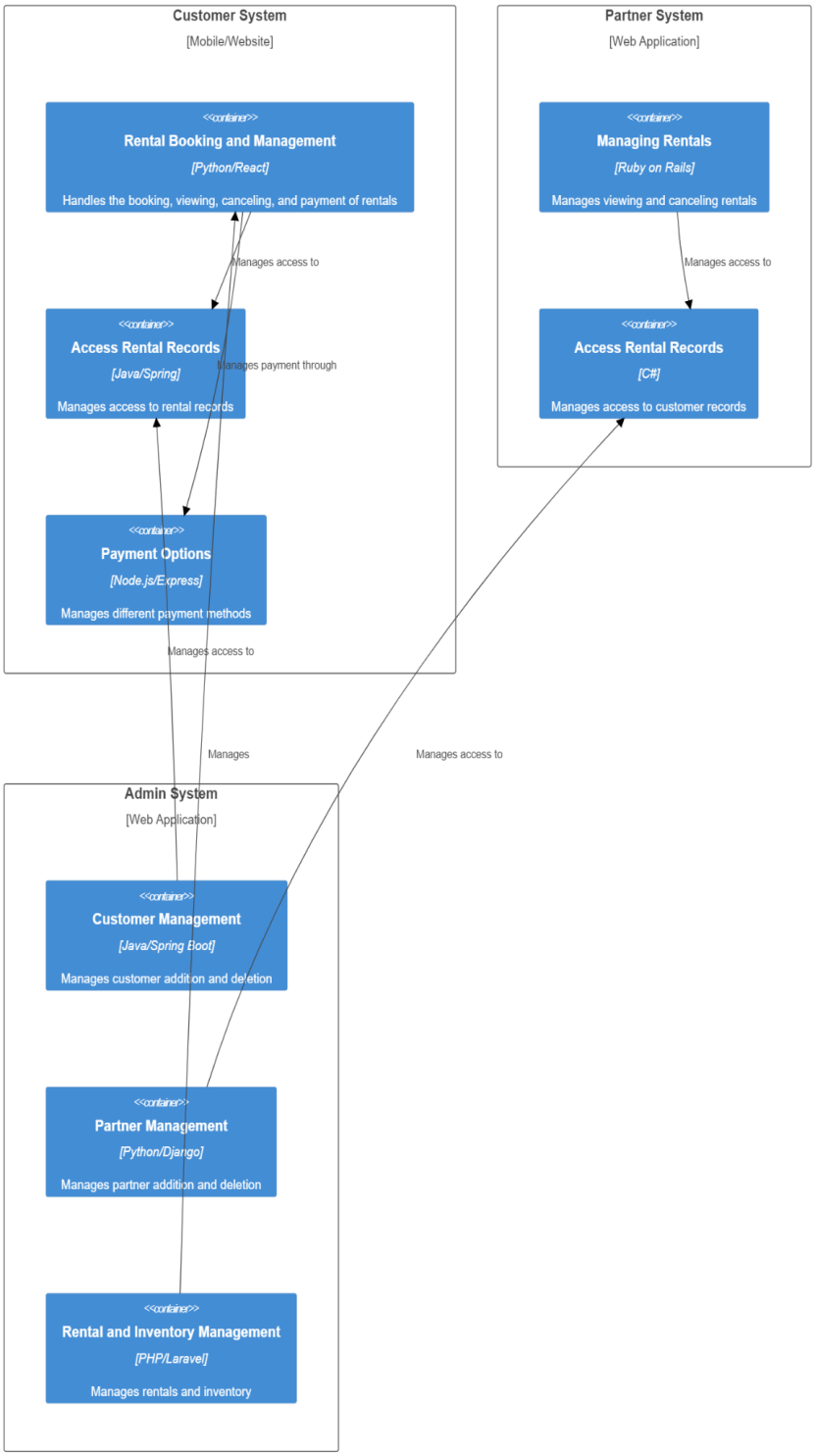
Security:

- Implements security protocols to protect sensitive data and prevent unauthorized access.

Component Diagram:



Deployment Diagram:



6.2. Software Architecture

User Interface Layer

The User Interface Layer is responsible for presenting information to users and receiving user inputs. It includes the graphical user interface (GUI) components for patients, doctors, and administrators.

Middle Tier

The Middle Tier serves as the intermediary layer between the user interface and the data access layer. It manages business logic and facilitates communication between the User Interface Layer and the Data Access Layer.

Data Access Layer

The Data Access Layer handles interactions with the database, retrieving and storing data as required by the application. It ensures data integrity and security.

Interaction Between Layers

This diagram illustrates the interaction between the User Interface Layer, Middle Tier, and Data Access Layer, demonstrating the flow of data and control through the system.

7. Design Strategy

1. Future System Extension or Enhancement:
    Design Strategy:
    Employ a modular and scalable architecture to facilitate future system extensions or enhancements.
    Utilize design patterns such as the modular monolith or microservices, allowing for the addition of new features or components without major disruptions to existing functionality.

    Reasoning:
    Facilitates adaptability to evolving requirements and technological advancements.
    Supports incremental development, making it easier to integrate new features without affecting the entire system.

    Trade-offs:
    Requires thoughtful design to define clear module boundaries and interfaces.
    Overly modular architectures may introduce complexity.

2. System Reuse:
    Design Strategy:
    Encourage code reusability through the application of object-oriented principles and modular design.
    Utilize established design patterns and frameworks to leverage existing solutions.

    Reasoning:
    Reduces development time and effort by building on proven solutions.
    Enhances maintainability and consistency across the system.

    Trade-offs:
    May require a learning curve for developers new to specific frameworks.
    Careful consideration needed to balance generic solutions with application-specific requirements.

3. User Interface Paradigms:
    Design Strategy:
    Adopt a user-centered design approach to create an intuitive and responsive user interface.
    Utilize a consistent design language and layout across the system for a cohesive user experience.

    Reasoning:
    Enhances user satisfaction and productivity.
    A consistent and well-designed interface promotes ease of use and reduces user errors.

    Trade-offs:
    Balancing aesthetic considerations with functional requirements.
    Ensuring compatibility with different devices and accessibility standards.

4. Data Management (Storage, Distribution, Persistence):
    Design Strategy:
    Implement a robust data management strategy, including efficient storage, distribution, and persistence mechanisms.
    Consider a relational or NoSQL database based on specific data requirements.

    Reasoning:
    Ensures data integrity, consistency, and availability.
    Allows for optimal performance and scalability based on the nature of data interactions.

    Trade-offs:
    Choosing the right database technology may involve trade-offs between consistency and availability (CAP theorem).
    Balancing data normalization for relational databases with denormalization for performance.

5. Concurrency and Synchronization:
    Design Strategy:
    Implement concurrency control mechanisms to manage simultaneous access to shared resources.

Utilize multi-threading, locks, or transactional processing based on the nature of operations.

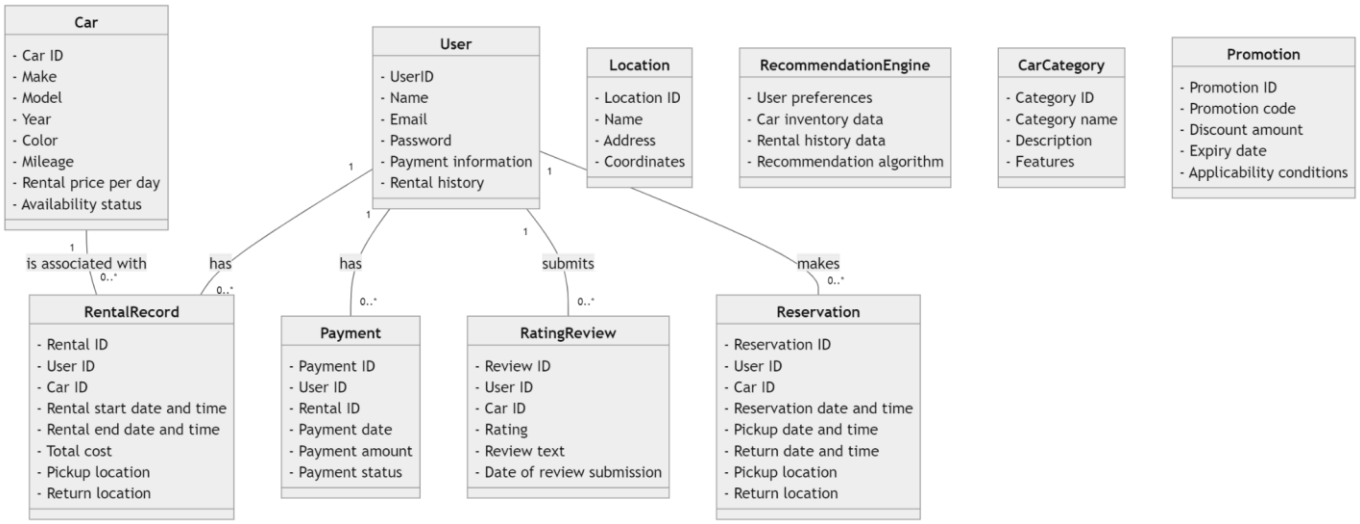
Reasoning:
Prevents data corruption and ensures consistent system behavior in multi-user environments.
Enhances system performance by allowing parallel execution of tasks.

Trade-offs:
Introducing complexity in managing concurrent access.
Careful consideration needed to avoid deadlocks and contention issues.

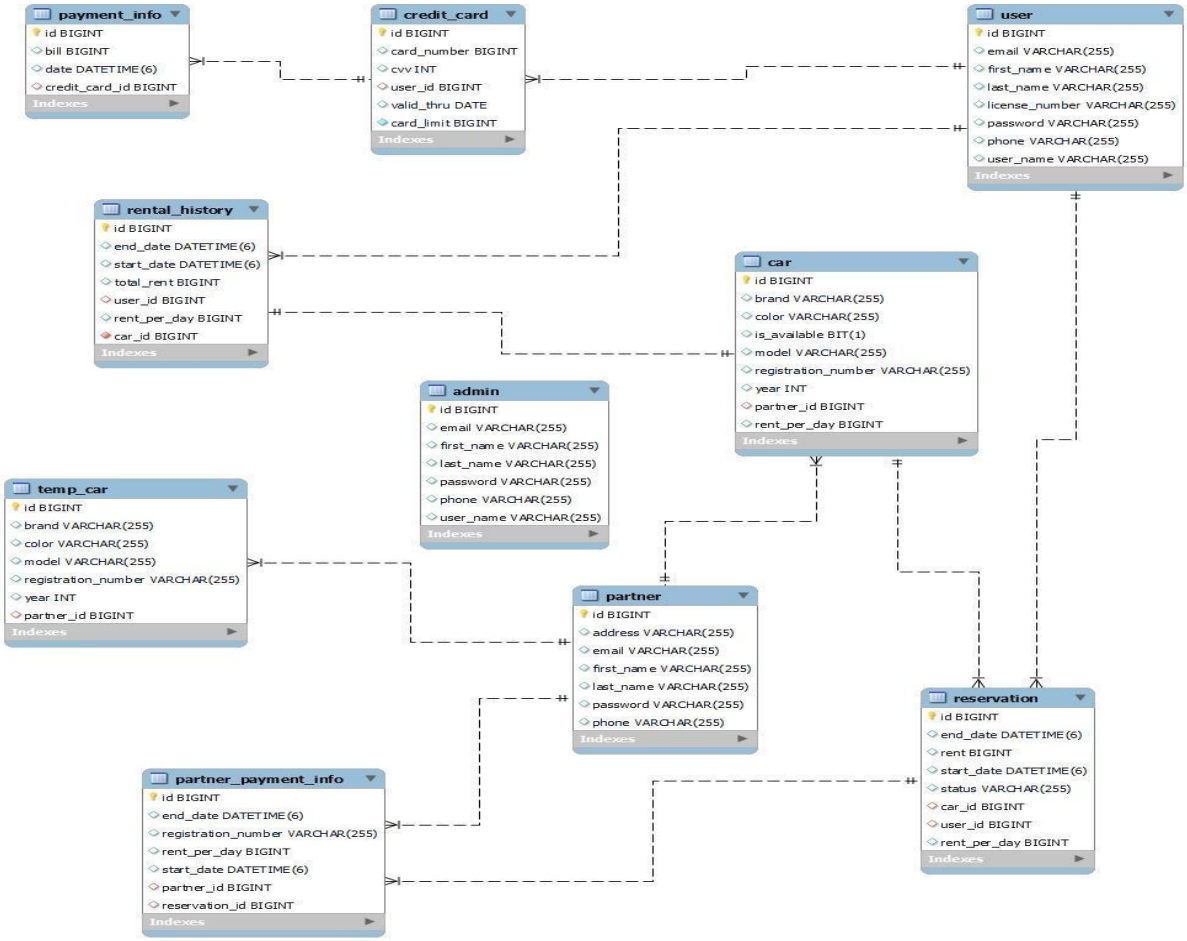
8. Detailed System Design

8.1. Database Design

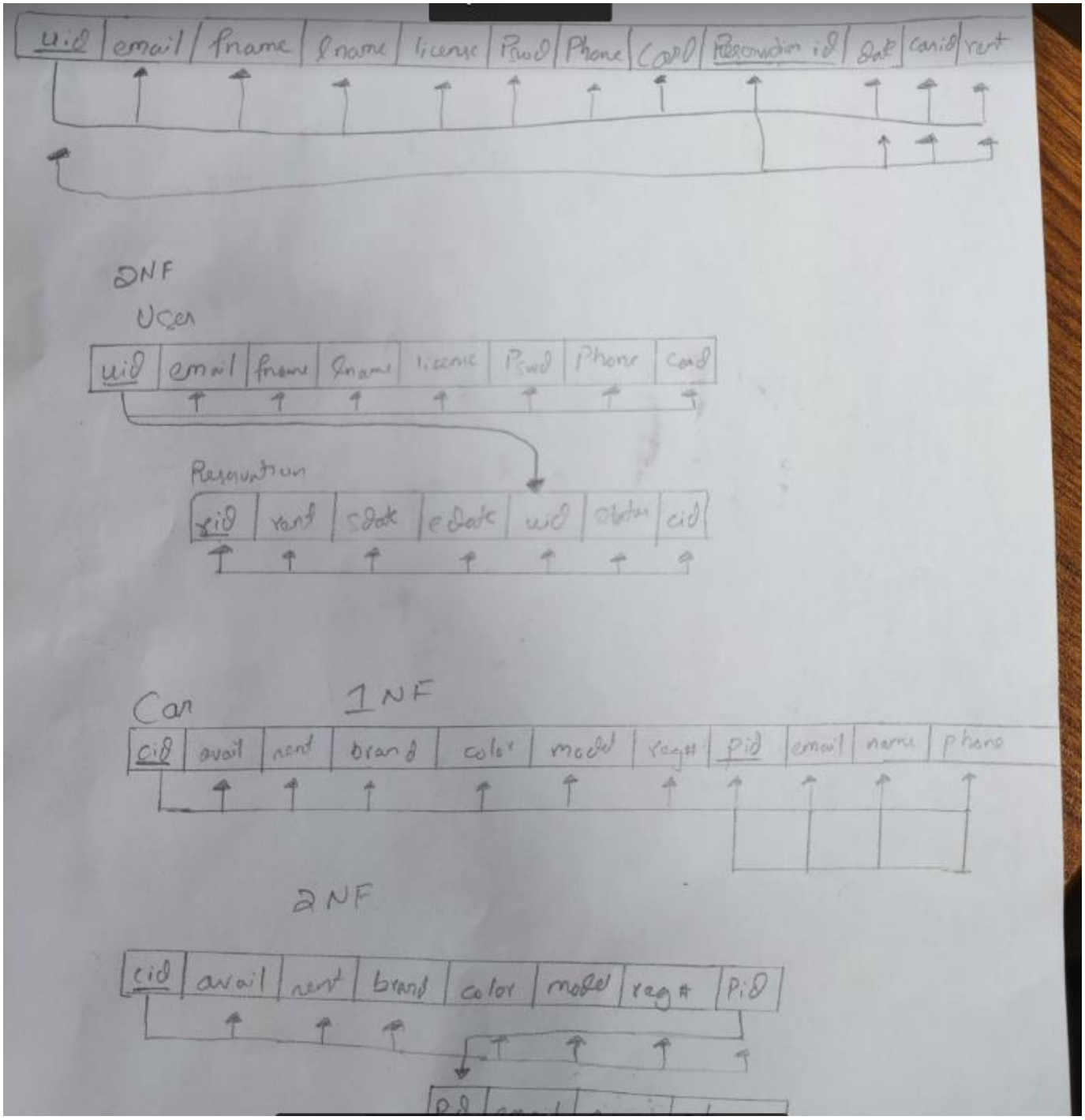
Class Diagram:



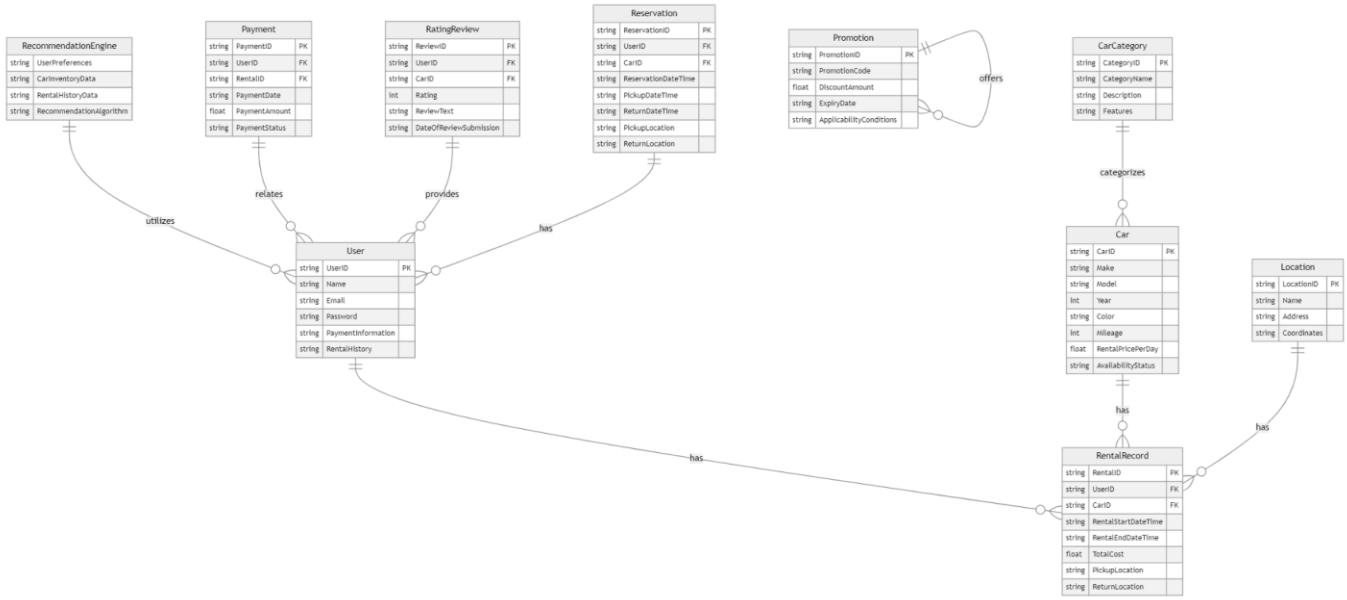
Logical Model:



Normalization:



8.1.1. ER Diagram



8.2 Application Design

Explanation:

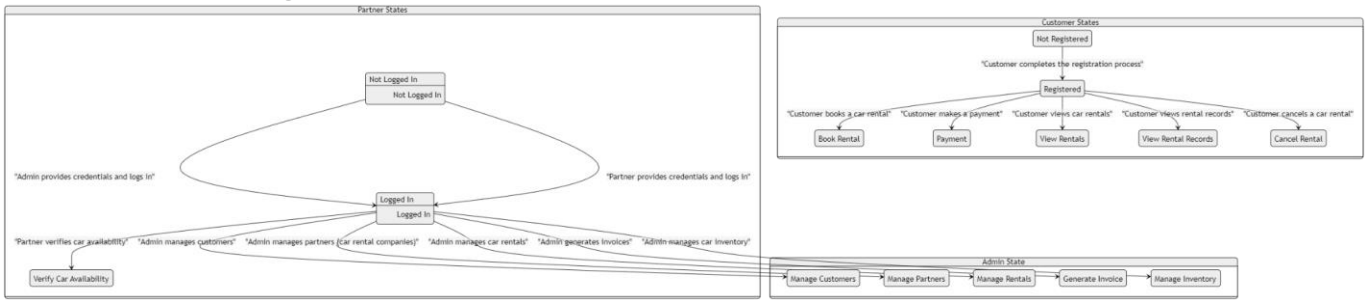
1. Customer Functionality:

- Booking and Managing Rentals:
  - Booking a Rental:
    - Action: The customer initiates the rental booking process by sending a request through the system.
    - Interaction: Customer requests a rental. The system retrieves available cars and displays options.
  - Viewing and Canceling Rentals:
    - Action: Customers can view their scheduled rentals and cancel them if necessary.
    - Interaction: Customer requests to view their rentals. The system retrieves and displays the rentals. Customer requests to cancel a rental. The system cancels the rental and notifies the customer.
- Accessing Rental Records:
  - Viewing Rental Records:
    - Action: Customers can access their rental records through the system.
    - Interaction: Customer requests to view their rental records. The system retrieves and displays the relevant rental records.
- Payment Options:
  - Payment Process:
    - Action: Customers can make payments for rentals through different options.
    - Interaction: Customer selects a payment method (credit card, online

payment, cash). The system processes the payment accordingly.

2. Partner Functionality:
    - Managing Rentals:
        - Viewing Rentals:
            - Action: Partners can view their scheduled rentals.
            - Interaction: Partner requests to view upcoming rentals. The system retrieves and displays the rentals.
        - Canceling Rentals:
            - Action: Partners can cancel rentals if necessary.
            - Interaction: Partner requests to cancel a rental. The system cancels the rental and notifies the customer.
    - Accessing Rental Records:
        - Viewing Customer Rental Records:
            - Action: Partners can access customer rental records.
            - Interaction: Partner requests to view a customer's rental records. The system retrieves and displays the relevant rental records.
3. Admin Functionality:
    - Customer Management:
        - Adding Customers:
            - Action: Admin can add new customers to the system.
            - Interaction: Admin inputs customer details. The system adds the customer to the database.
        - Deleting Customers:
            - Action: Admin can remove customers from the system.
            - Interaction: Admin selects a customer for deletion. The system removes the customer's records.
    - Partner Management:
        - Adding Partners:
            - Action: Admin can add new partners to the system.
            - Interaction: Admin inputs partner details. The system adds the partner to the database.
        - Deleting Partners:
            - Action: Admin can remove partners from the system.
            - Interaction: Admin selects a partner for deletion. The system removes the partner's records.
    - Rental and Inventory Management:
        - Managing Rentals:
            - Action: Admin can view and manage rentals.
            - Interaction: Admin requests to view or manage rentals. The system provides relevant information and options.
        - Inventory Management:
            - Action: Admin can manage the inventory of cars and other rental assets.
            - Interaction: Admin requests to manage inventory. The system provides tools to update and monitor inventory.

8.1.2 State Diagram



Explanation:

Admin State:

- Not Logged In: The initial state when the admin has not yet logged into the Car Rental Hub system.
  - Transitions:
    - Trigger: Admin provides credentials and logs in.
    - Action: Transition to the "Logged In" state.
- Logged In: The admin has successfully logged into the Car Rental Hub system.
  - Transitions:
    - Trigger: Admin manages customers.
    - Action: Transition to the "Manage Customers" state.
    - Trigger: Admin manages partners (car rental companies).
    - Action: Transition to the "Manage Partners" state.
    - Trigger: Admin manages car rentals.
    - Action: Transition to the "Manage Rentals" state.
    - Trigger: Admin generates invoices.
    - Action: Transition to the "Generate Invoice" state.
    - Trigger: Admin manages car inventory.
    - Action: Transition to the "Manage Inventory" state.

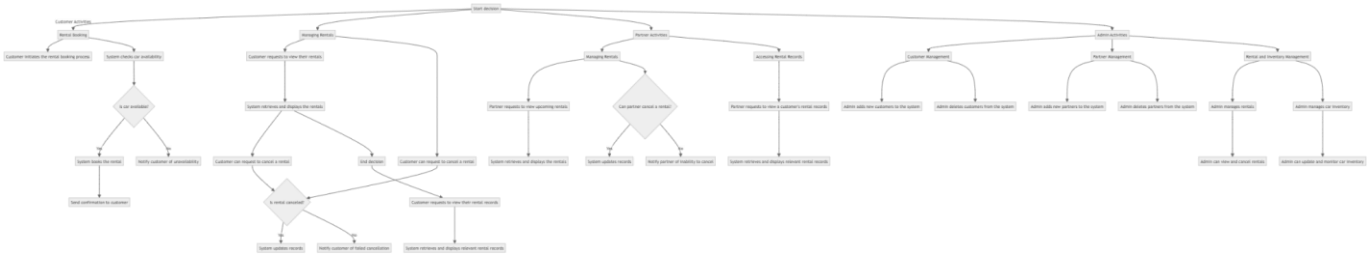
Partner States

- Not Logged In: The initial state when the partner (car rental company) has not yet logged into the Car Rental Hub system.
  - Transitions:
    - Trigger: Partner provides credentials and logs in.
    - Action: Transition to the "Logged In" state.
- Logged In: The partner has successfully logged into the Car Rental Hub system.
  - Transitions:
    - Trigger: Partner verifies car availability.
    - Action: Transition to the "Verify Car Availability" state.

Customer States:

- Not Registered: The initial state when the customer is not registered in the Car Rental Hub system.
  - Transitions:
    - Trigger: Customer completes the registration process.
    - Action: Transition to the "Registered" state.
- Registered: The customer is successfully registered in the Car Rental Hub system.
  - Transitions:
    - Trigger: Customer books a car rental.
    - Action: Transition to the "Book Rental" state.
    - Trigger: Customer makes a payment.
    - Action: Transition to the "Payment" state.
    - Trigger: Customer views car rentals.
    - Action: Transition to the "View Rentals" state.
    - Trigger: Customer views rental records.
    - Action: Transition to the "View Rental Records" state.
    - Trigger: Customer cancels a car rental.
    - Action: Transition to the "Cancel Rental" state.

8.1.3 Activity Diagram



Explanation:

Customer Activities:

- Rental Booking:
  - Activity: Customer initiates the rental booking process.
  - System checks car availability.
  - If available, the system books the rental.
  - Confirmation is sent to the customer.
- Managing Rentals:
  - Activity: Customer requests to view their rentals.
  - System retrieves and displays the rentals.
  - Customer can request to cancel a rental.

- If canceled, the system updates the records and notifies the customer.
- Accessing Rental Records:
  - Activity: Customer requests to view their rental records.
  - System retrieves and displays the relevant rental records.
- Payment Process:
  - Activity: Customer selects a payment method (credit card, online payment, cash).
  - System processes the payment accordingly.

Partner Activities:

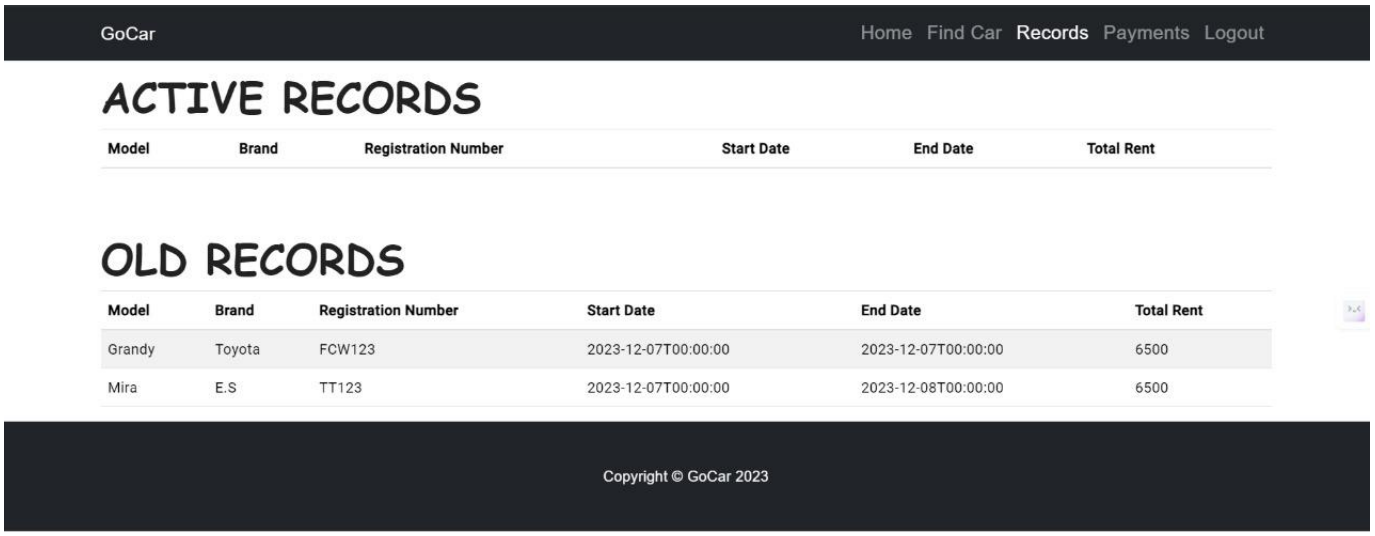
- Managing Rentals:
  - Activity: Partner requests to view upcoming rentals.
  - System retrieves and displays the rentals.
  - Partner can request to cancel a rental.
  - If canceled, the system updates the records and notifies the customer.
- Accessing Rental Records:
  - Activity: Partner requests to view a customer's rental records.
  - System retrieves and displays the relevant rental records.

Admin Activities:

- Customer Management:
  - Activity: Admin adds new customers to the system.
  - Admin deletes customers from the system.
- Partner Management:
  - Activity: Admin adds new partners (car rental companies) to the system.
  - Admin deletes partners from the system.
- Rental and Inventory Management:
  - Activity: Admin manages rentals (view, cancel).
  - Admin manages car inventory (update, monitor).

9. GUI Design:


GoCar
Home Find Car Records Payments Logout
WELCOME HASSAN ABBAS!
Copyright © GoCar 2023


GoCar
Home Find Car Records Payments Logout
ACTIVE RECORDS
Model Brand Registration Number Start Date End Date Total Rent
OLD RECORDS
Model Brand Registration Number Start Date End Date Total Rent
Grandy Toyota FCW123 2023-12-07T00:00:00 2023-12-07T00:00:00 6500
Mira E.S TT123 2023-12-07T00:00:00 2023-12-08T00:00:00 6500
Copyright © GoCar 2023

10. Test Cases:

Test Case 1: Registering a User

- Input: A new user's details (name, email, password, etc.).
- Expected Output: Successful registration of the user.

Test Case 2: Logging in as Admin

- Input: Admin credentials (email, password).
- Expected Output: Successful login as admin.

Test Case 3: Adding a New Car

- Input: Details of a new car (brand, model, rent per day, etc.).
- Expected Output: Successful addition of the new car to the system.

Test Case 4: Reserving a Car by a User

- Input: User ID, Car ID, Reservation details (start date, end date, etc.).
- Expected Output: Successful reservation of the car by the user.

Test Case 5: Updating User's Password

- Input: User ID, Old Password, New Password.
- Expected Output**: Successful update of the user's password.

Test Case 6: Adding a Credit Card to User's Account

- Input: User ID, Credit Card details (card number, expiry date, etc.).
- Expected Output: Successful addition of the credit card to the user's account.

Test Case 7: Partner Login

- Input: Partner credentials (email, password).
- Expected Output: Successful login as a partner.

Test Case 8: Deleting a Car

- Input: Car ID of the car to be deleted.
- Expected Output: Successful deletion of the car from the system.

Test Case 9: Retrieving Rental History for a User

- Input: User ID.
- Expected Output: Successful retrieval of the rental history for the user.

Test Case 10: Adding a Temp Car by a Partner

- Input: Partner ID, Temp Car details (brand, model, etc.).
- Expected Output: Successful addition of the temporary car by the partner.