Software Design Specification (SDS)

Project Name: Rent-A-Ride car rental platform: Your Go-To Car Rental Hub
Prepared By: Hania khaled
Date: 21-12-2024

---

1. Introduction

1.1 Purpose

The purpose of this document is to describe the design, architecture, technical specifications, and applied design patterns for the Rent-A-Ride car rental platform: Your Go-To Car Rental Hub. It outlines the functionality, system components, design decisions, and how design patterns enhance the platform's architecture.

1.2 Scope

This SDS covers the design and implementation details of the Rent-A-Ride car rental platform: Your Go-To Car Rental Hub. The software will perform the following major tasks :

- car search and booking
- offers
- user authentication
- review and rating system
- search
- checking availability and booking
- dashboard

---

2. System Overview

The system consists of the following components:

- **Frontend**: JavaScript, HTML, and CSS.
- **Backend**: Flask.
- **Database**: MySQL.

---

3. System Architecture

3.1 Architectural Design

This project follows the client-server architecture, where:

- **The frontend** communicates with the backend via RESTful APIs to handle data requests and user actions.
- **Backend** processes requests, manages the messaging system, and handles user authentication. It interacts with the database to manage and retrieve data.

3.2 Data Flow

1. **User Interaction**: The user interacts with the UI to perform an action search for cars, book a rental,check offers or leave a review
2. **Request Processing**: The frontend sends a RESTful API request to the backend server.
3. **Data Handling**: The backend processes the request, interacts with the database, and fetches or updates the necessary data.
4. **Response**: The backend sends a JSON response with the requested data back to the frontend, updating the UI.

---

4. Database Design

The database used for this system is MySQL, a relational database chosen for its benefits like scalability, performance, ease of integration with our backend technology with the following entities and relationships:

Table 1:

The **Users** table stores information about customers who will rent cars.

- **UserID**: Integer, primary key, unique identifier for each user.
- **Name**: String, full name of the customer.
- **Email**: String, email address, must be unique for each user.
- **Password**: String, encrypted password for security purposes.
- **Phone**: String, contact number for the customer.
- **Active**: TINYINT, default value 0 (indicates whether the user is active or not).

Table 2:

The **Owners** table stores information about the owners who offer cars for rent.

- **OwnerID**: Integer, primary key, unique identifier for each owner.
- **Name**: String, full name of the car owner.
- **Email**: String, email address, must be unique.
- **Password**: String, encrypted password for security.
- **Phone**: String, optional contact number for the owner.
- **Address**: String, optional address of the owner.

Table 3: cars table

The **Cars** table stores details about the cars available for rent.

- **CarID**: Integer, primary key, unique identifier for each car.
- **Name**: String, model name of the car.
- **Year**: Integer, year of manufacture.
- **Capacity**: Integer, seating capacity of the car.
- **Fuel_Type**: String, type of fuel the car uses (e.g., petrol, electric).
- **Mileage_Per_Litre**: Decimal, fuel efficiency in terms of mileage per litre.
- **Transmission**: String, type of transmission (e.g., automatic, manual).
- **Price_Per_Day**: Decimal, daily rental price for the car.
- **Photo_URL**: String, URL of the car's image.
- **OwnerID**: Foreign key, references the **Owners** table.
- **Offers**: String, promotional offers available for the car.
- **New_Price**: Decimal, updated rental price for the car after any offer.
- **Old_Price**: Decimal, original rental price for the car before any offer.

Table 4: Rentals table

The **Rentals** table stores data about rental transactions between users and cars.

- **RentalID**: Integer, primary key, unique identifier for each rental transaction.
- **UserID**: Foreign key, links to the **Users** table, indicating which user rented the car.
- **CarID**: Foreign key, links to the **Cars** table, indicating which car was rented.
- **RentalStartDate**: Date, start date of the rental period.
- **RentalEndDate**: Date, end date of the rental period.

---

5. Technology Stack

- **Frontend**: The frontend is built with JavaScript for efficient, dynamic user interfaces. For responsive design, we utilize Bootstrap. This setup ensures compatibility with various devices, providing a seamless user experience on both desktop and mobile.

- **Backend**: The backend is developed using Flask, which allows us to handle user requests, process business logic, and communicate with the database securely and efficiently.

- **Database**:The system uses MySQL as the database, selected for its ability to manage relational/non-relational data efficiently and its compatibility with the chosen backend technology.

- **Hosting**: We will deploy the website on Heroku/Google Cloud for reliable, scalable hosting with easy management of resources, automatic scaling options, and integrated security features.

---

6. Testing Plan

6.1 Unit Testing
Each individual module and function will undergo unit testing to confirm correct functionality. For example, the function handling user registration will be tested to verify it stores data correctly, checks for email uniqueness, and encrypts passwords securely.

6.2 Integration Testing
Integration tests will verify interactions between different components. We will test the frontend's car search functionality to ensure it successfully retrieves data from the backend and displays it accurately. API calls between frontend and backend and data exchanges between backend and database will be tested to ensure smooth functionality.

6.3 User Acceptance Testing (UAT)
End users will participate in UAT to ensure the system meets their needs and expectations. They will test the car search, booking, and checkout processes. Feedback from UAT will guide adjustments to improve the user experience.

6.4 Performance Testing
Performance testing, including load and stress testing, will assess the system's stability and responsiveness under high traffic. The goal is to ensure that the website can handle a minimum of 500 of concurrent users or requests per second.

---

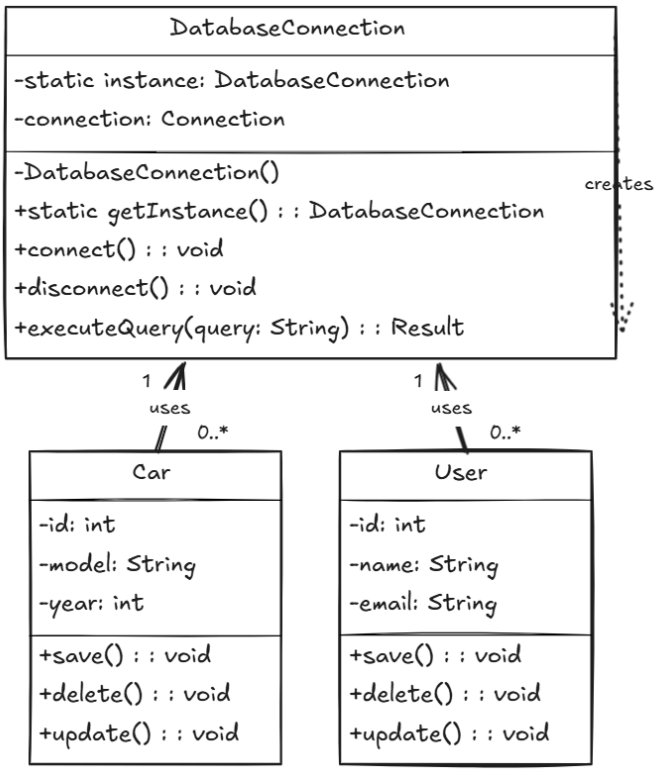
7. Diagrams

Applied Design Patterns

To enhance the maintainability, scalability, and flexibility of the system, the following design patterns have been implemented:

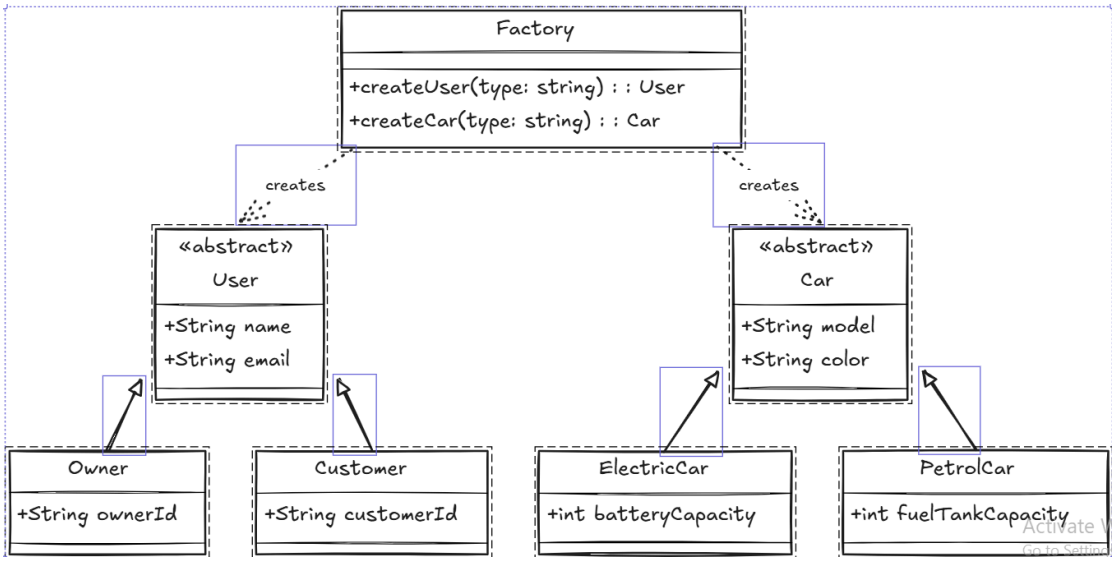
3.1 Singleton Pattern

- **Application:** Ensures a single instance of the MySQL database connection exists throughout the application. **Purpose:** Prevents multiple connections being created and improves resource efficiency.



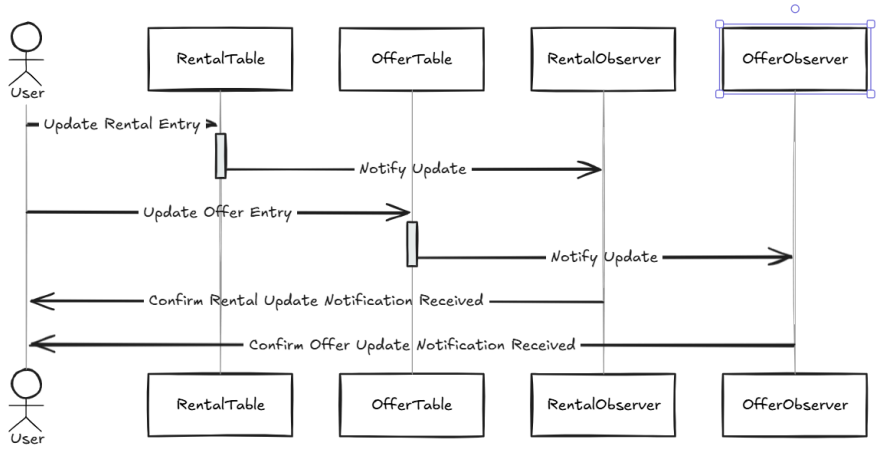
3.2 Factory Method Pattern

- **Application:** Encapsulates the creation of `User` and `Car` objects based on their attributes. **Purpose:** Simplifies object creation logic and promotes extensibility for future requirements.



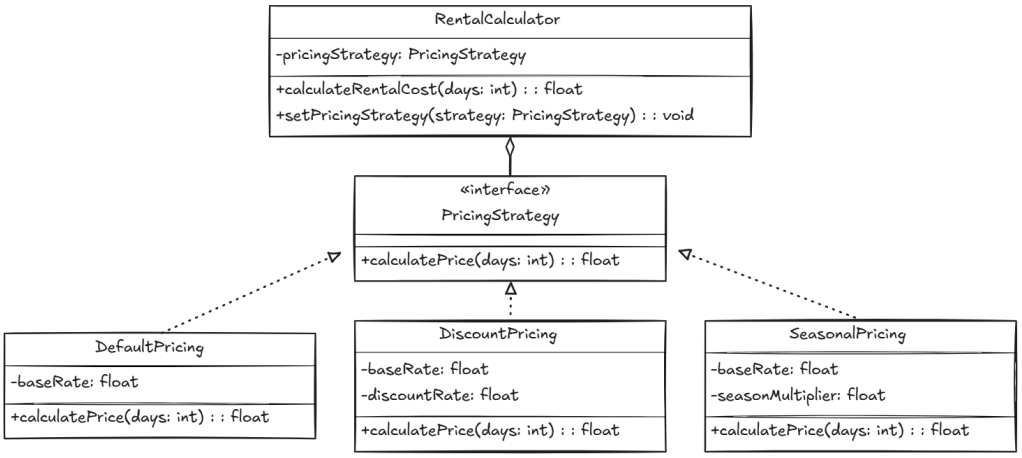
3.3 Observer Pattern

- **Application:** Automates notifications for booking confirmations and promotional offers. **Purpose**: Keeps users informed about updates and ensures real-time communication.



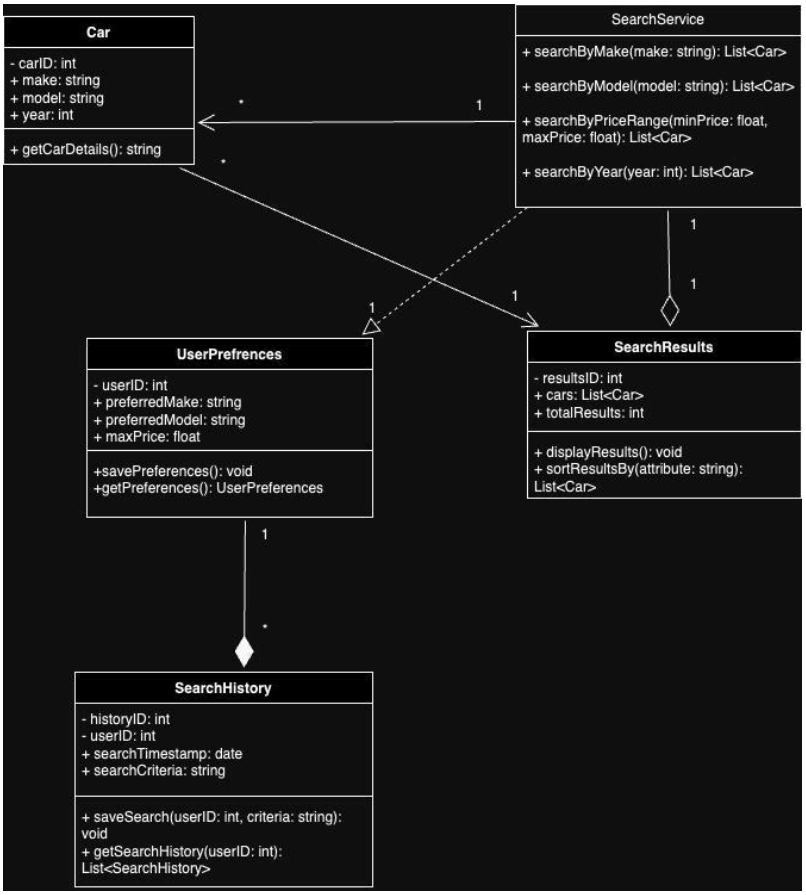
3.4 Strategy Pattern

- **Application:** Modularizes pricing calculations for rentals. **Purpose:** Allows flexibility in applying different pricing strategies (e.g., discounts, seasonal adjustments).

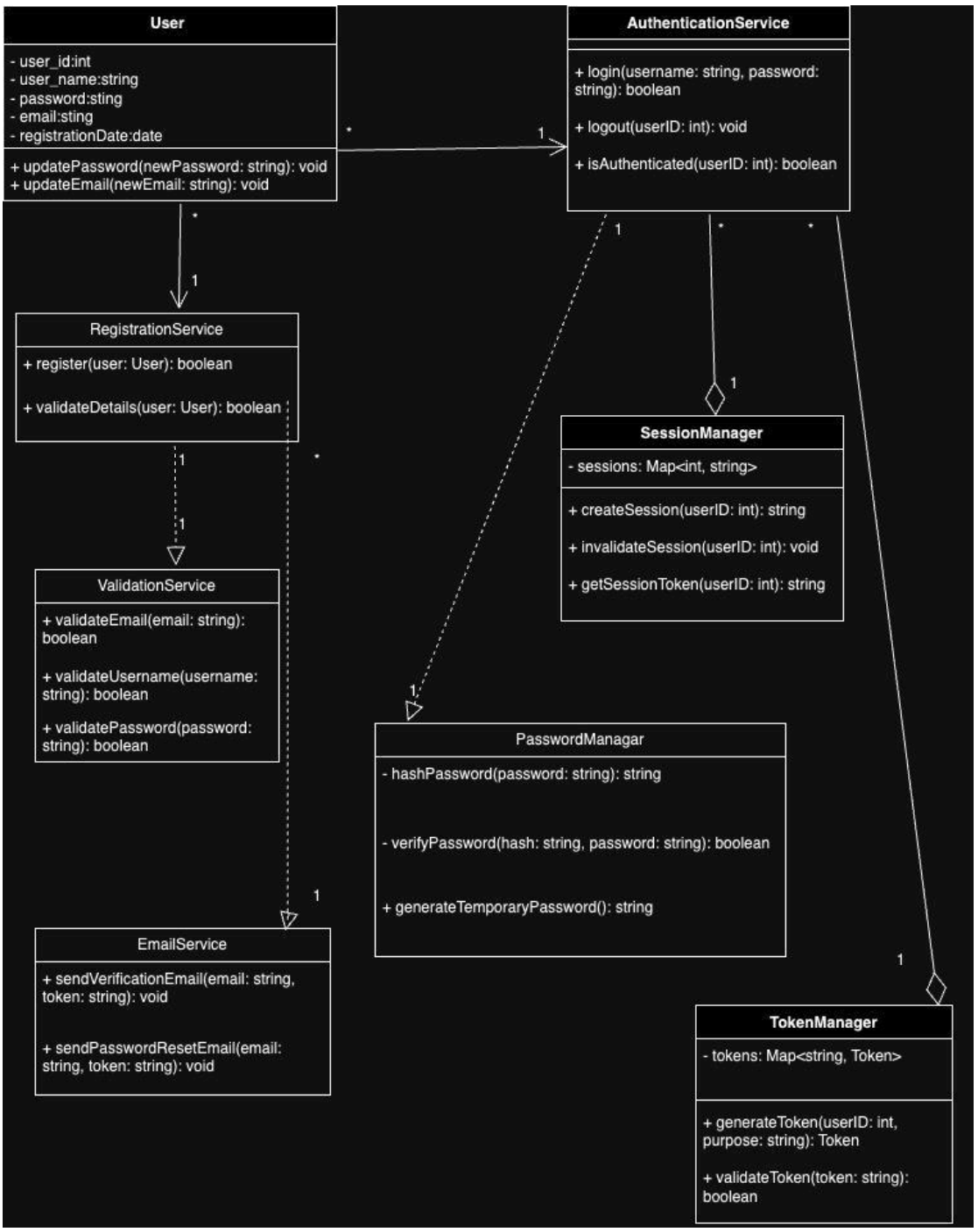


a) Class Diagram

The first module is for the Authentication :

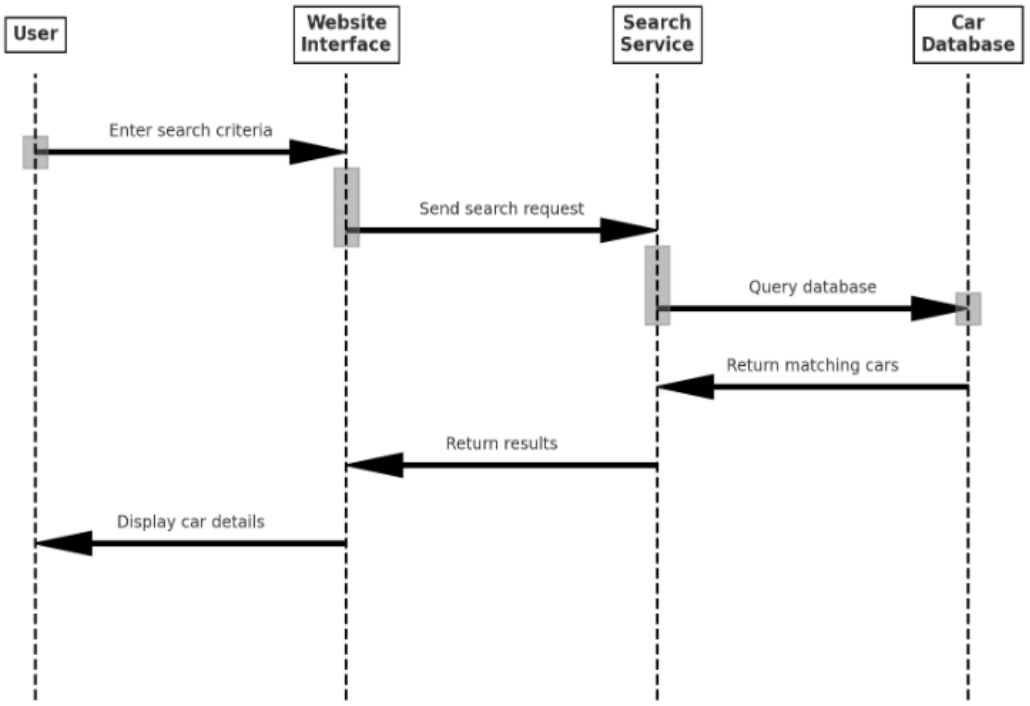


The second module is for the car search



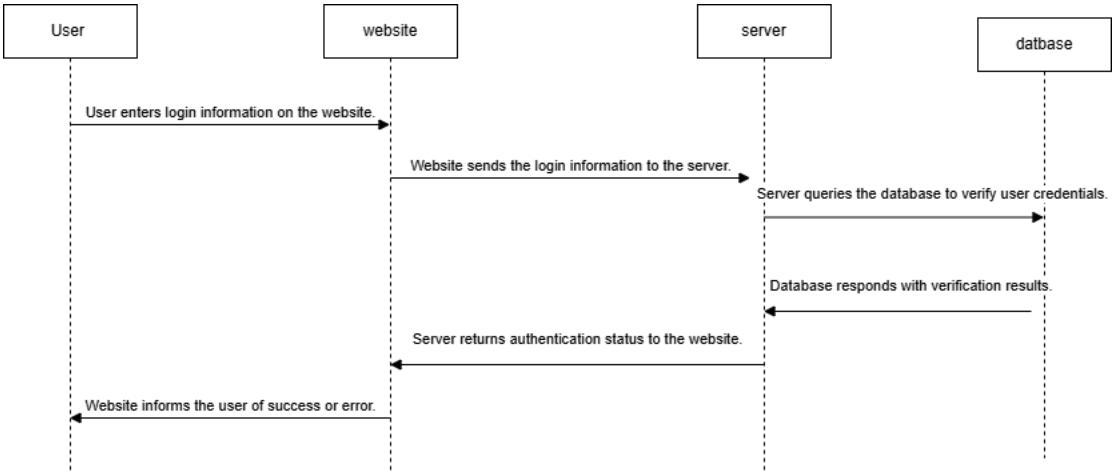
b. Sequence diagrams :

1) Car Search :

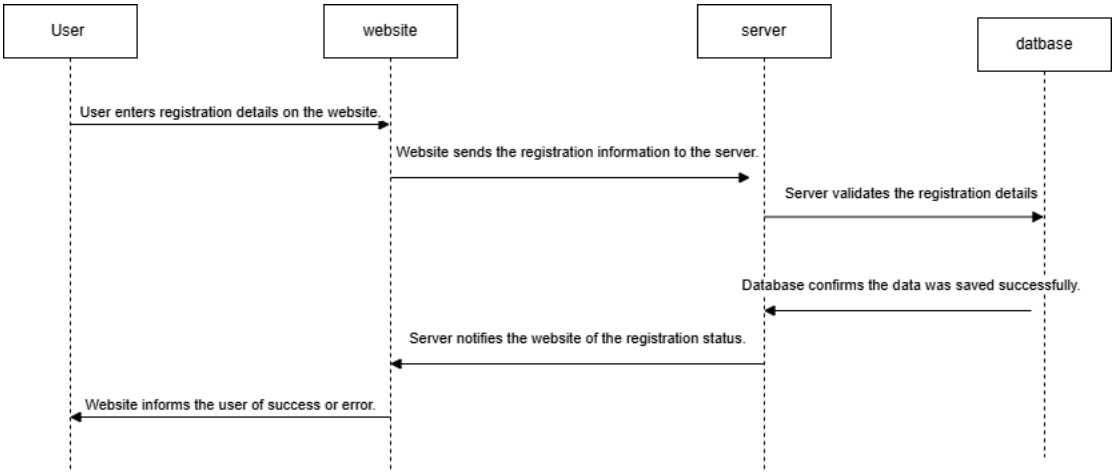


2) Authentication :

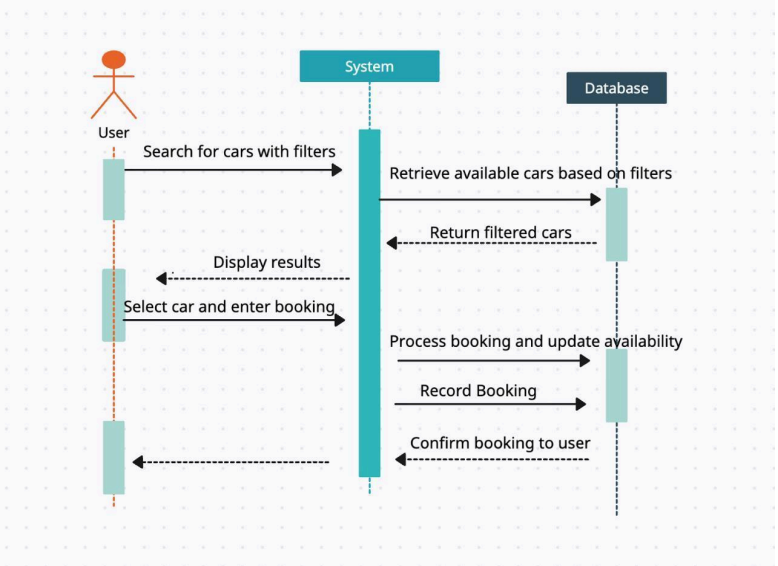
- **login**



- **Registration**



3) Car Booking :



8. Conclusion

This document outlines the technical design for the “Rent A Ride“ car rental website. With a robust architecture, efficient database design, and a thorough testing strategy, the system is expected to meet all specified functional and non-functional requirements. By following the SDS guidelines, the development team will create a reliable, scalable, and user-friendly platform for customers to rent cars easily based on their preferences.