Software Requirements Specification (SRS) for Car Rental System

Team :

script squad

Team members :

(Abdullah Ismael Ahmed) , Mayar Mohamed Ebrahim ,Mohab Marzouk Gaber , Ibrahim Hussien AbdelRazek , Kerlous Nasser Shehata

1. Introduction

- **Purpose**:
  The purpose of this document is to define the functional and non-functional requirements for the Car Rental System. This system will provide a platform for customers to search, filter, and rent cars while offering administrators a dashboard to manage cars, customers, and rental contracts.
- **Scope**:
  This system will serve two types of users: customers and admins. Customers will be able to search for cars, view details, and make bookings. Admins will manage car listings, customer data, and rentals. The project will be built using the MERN stack (MongoDB, Express.js, React.js, Node.js) and designed with Figma. The system will be modular to allow for future integrations such as payment gateways and notifications.
- **Definitions and Acronyms**:
  - MERN: MongoDB, Express.js, React.js, Node.js
  - CRUD: Create, Read, Update, Delete
  - API: Application Programming Interface
  - UI: User Interface
  - Admin: Administrator of the system
- **References**:
  - Project Proposal: Car Rental System

---

2. Overall Description

- **Product Perspective**:
  The Car Rental System will be a web-based platform developed using the MERN stack. It will consist of two main interfaces: a customer-facing side for car rentals and an admin dashboard for management. The system will be designed to handle real-time updates for car availability and rental statuses using Redux for state management.
- **Product Features**:
  - **Customer Features**:
    - Car Search & Filtering (e.g., by price, type, and availability)
    - Car Selection & Reservation
    - Customer Account Management (view rental history, invoices, etc.)
  - **Admin Features**:
    - Car Management (CRUD operations)
    - Customer Management (CRUD operations)
    - Rental Contract Management
    - Admin Dashboard with key metrics (active rentals, pending maintenance)
  - **Additional Features (V2)**:
    - Payment Integration
    - Notifications for car returns or maintenance
- **User Characteristics**:
  - **Customers**: Users who will search, book, and rent cars.
  - **Admins**: Users who manage cars, customers, and rental contracts.
- **Constraints**:
  - The system must support real-time updates for car availability.
  - Initially, payments will not be integrated but will be added in a future version.

---

3. Functional Requirements

3.1. Customer Side

1. **Car Search & Filtering**:
   - ○ The system shall allow customers to search for cars using filters such as price, brand, car type, and availability.
   - ○ The search results shall dynamically update based on real-time availability.
2. **Car Selection & Reservation**:
   - ○ Customers shall be able to view detailed information about each car (description, price, availability).
   - ○ The system shall allow customers to reserve a car for a specific period.
3. **User Authentication**:
   - ○ Customers shall be able to create accounts, log in securely, and manage their profile.
   - ○ The system shall provide options for users to reset forgotten passwords.
4. **Reservation Management**:
   - ○ Customers shall be able to view their reservation history and download invoices.

3.2. Admin Side

1. **Car Management**:
   - ○ The system shall allow admins to add, update, delete, and view car listings.
   - ○ Admins shall be able to track car maintenance schedules and mark cars as under maintenance when needed.
2. **Rental Contract Management**:
   - ○ Admins shall manage rental contracts, including customer details, car information, and rental periods.
   - ○ The system shall allow admins to track ongoing rentals and access the full rental history.
3. **Customer Management**:
   - ○ Admins shall manage customer profiles (CRUD operations).
4. **Dashboard Overview**:
   - ○ The admin dashboard shall display key metrics like active rentals, available cars, and maintenance alerts.
5. **Real-Time Updates**:
   - ○ The system shall ensure that any changes in car availability or rental status are updated across the application in real-time using Redux.

---

4. Non-Functional Requirements

1. **Performance**:
   - The system shall handle at least 100 concurrent users without significant performance degradation.
2. **Scalability**:
   - The system shall be scalable to accommodate an expanding number of cars and users.
3. **Security**:
   - The system shall use secure authentication for both customer and admin logins.
   - All sensitive data, such as passwords, shall be encrypted.
4. **Usability**:
   - The UI/UX of the system shall be intuitive and easy to navigate for both customers and administrators, with minimal learning required.
5. **Maintainability**:
   - The code shall be modular, allowing for easy maintenance and future feature expansions (e.g., payment gateways, notifications).

---

5. System Architecture

- **Frontend**:
  The frontend will be built using React.js with Redux for state management. It will consist of separate components for the customer side and the admin side.
- **Backend**:
  The backend will be developed using Node.js and Express.js. APIs will be created for managing car listings, customer accounts, and rental contracts.
- **Database**:
  MongoDB will be used to store data such as car listings, customer information, and rental contracts. Mongoose will serve as the ODM for MongoDB interactions.

---

6. External Interface Requirements

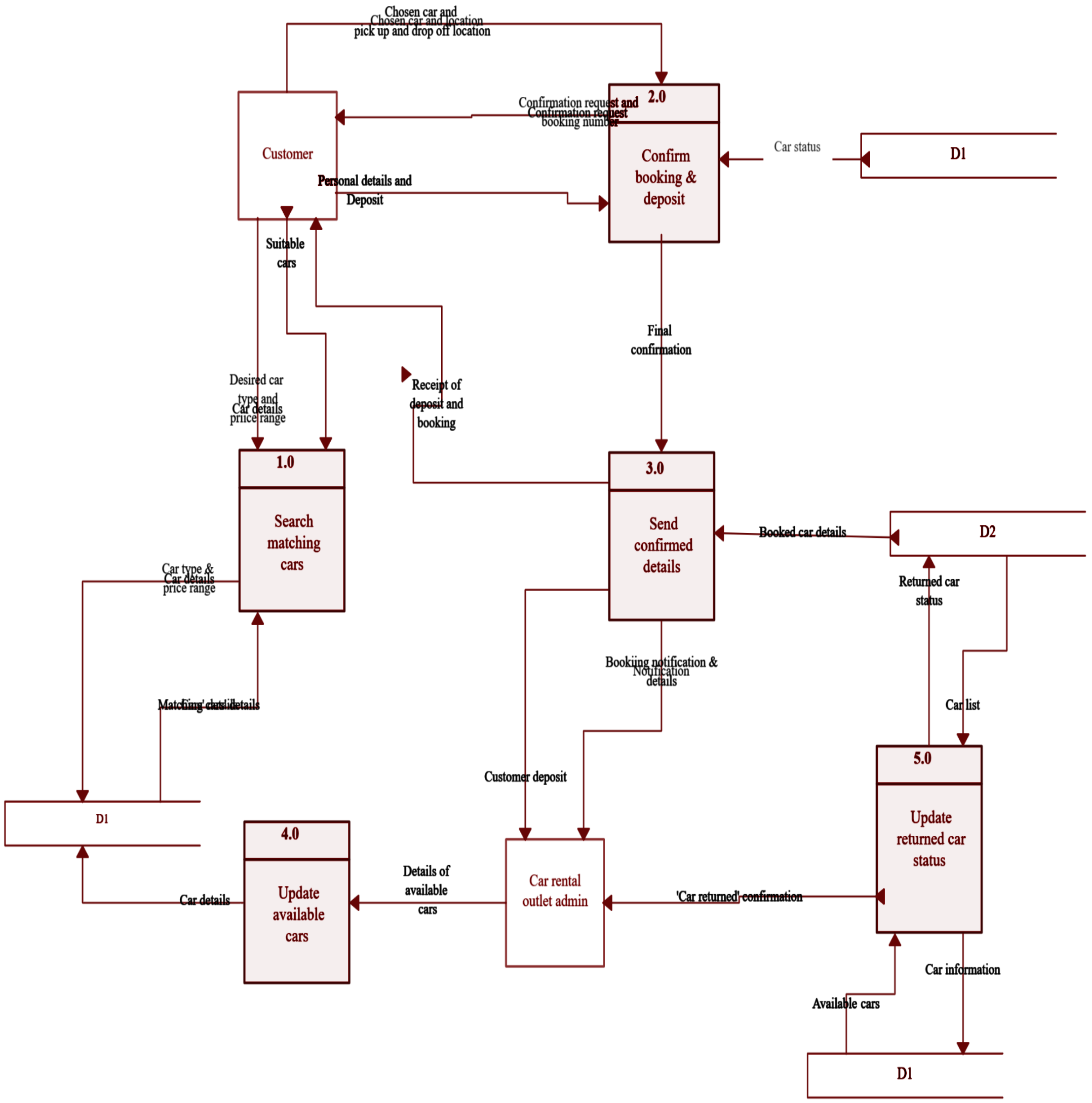
- **Figma**:
  Figma will be used to design the user interface for both the client and admin sides.
- **Payment Gateway (V2)**:
  In a future version, third-party payment gateways will be integrated for online transactions.

---

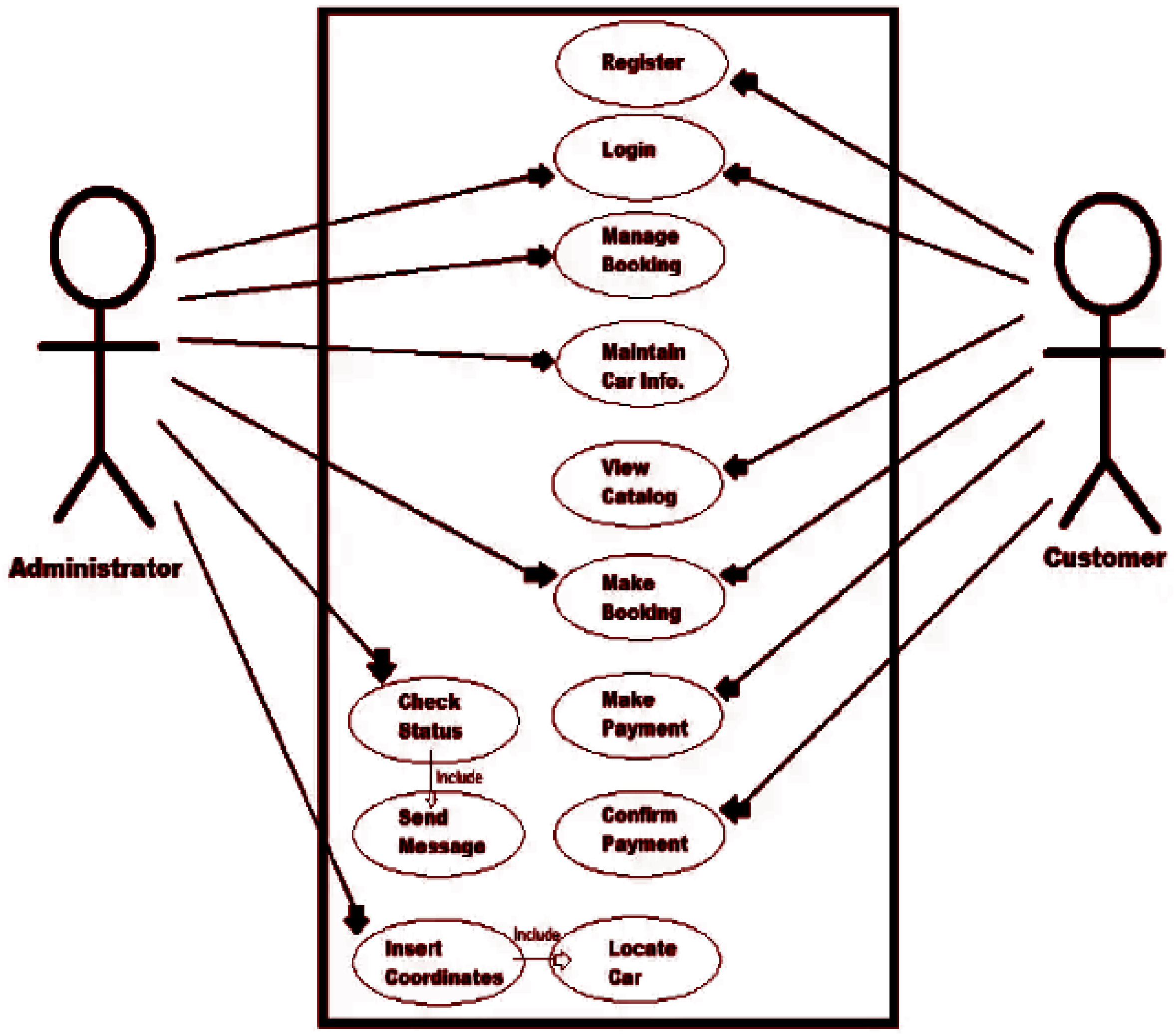
7. Appendices

- **Phase-wise Development Timeline**:
  1. **Phase 1**: Static prototype using Vanilla JavaScript.
  2. **Phase 2**: React frontend with Redux for dynamic features like car filtering.
  3. **Phase 3**: Backend integration with Node.js, MongoDB, and Mongoose for API development.

Dfd diagram level one



Use case :


Register
Login
Manage Booking
Maintain Car Info.
View Catalog
Make Booking
Check Status
Make Payment
Include
Send Message
Confirm Payment
Include
Insert Coordinates
Locate Car
Administrator
Customer