Software Requirements Specification (SRS)

Project Title: Car Rental Management System (SaaS)

Team:

- **Team Leader:** Salah
- **Frontend Developers:** Salah, Hassib
- **Backend Developers:** Badro, Khaled, Salah
- **UI/UX Designer:** Ghofrane

---

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for the **Car Rental Management System (CRMS)**.
The system will be offered as a **Software as a Service (SaaS)** platform for car rental businesses. Each client (a rental store) will have its own hosted dashboard to manage cars, users, and reservations.

1.2 Scope

The CRMS will allow:

- **Clients (Car Rental Companies)** to manage cars, view reservations, analyze statistics, and handle payments.
- **End Users (Customers)** to browse available cars, filter results, compare cars, make reservations, and complete payments online.

The system will support **multi-tenancy**, where each client has a unique portal under the main SaaS platform.

1.3 Objectives

- Provide a simple and intuitive interface for clients to manage their car fleet.
- Enable customers to easily find, compare, and reserve vehicles.
- Offer insightful analytics for business owners.
- Ensure secure authentication and payment integration.

---

2. Overall Description

2.1 Product Perspective

This product will be a **web-based application** accessible from any browser. It consists of two main modules:

1. **Admin Dashboard (for clients)**
2. **User Portal (for end-users/customers)**

Both modules will communicate through a secure backend API.

2.2 Users and Characteristics

User Type	Description	Privileges
System Admin (SaaS owner)	Manages all clients, handles subscriptions	Full access
Client (Car Rental Store)	Manages their own cars, customers, reservations, and payments	CRUD access to their data
End User (Customer)	Browses, filters, compares, and reserves cars	Limited access (registration required for booking)

---

3. Functional Requirements

3.1 Client Dashboard

- **Authentication & Authorization**
  - Secure login for clients (store owners)
  - Only one account shared between employes, no need to create many roles
  - **Subject to the client needs later on**
- **Car Management**
  - Add/Modify/Delete car entries
  - View all cars in card/grid format

    - View each car’s details (availability, reservations, rental history)
- **Reservation Management**
    - View who reserved which car
    - Track status (pending, confirmed, returned)
    - Modify or cancel reservations
- **Statistics & Insights**
    - Display data visualization on:
        - Most rented cars
        - Rentals by brand, color, location…
        - Revenue and occupancy rates

3.2 End-User Portal

- **Authentication**
    - Sign up / Login / Password recovery
- **Car Browsing**
    - View available cars in card view
    - Filter by brand, type, price range, etc.
    - Highlight "🔥 On Fire" cars (most rented)
- **Car Comparison**
    - Compare multiple cars side by side (specs, price, availability)
- **Reservation & Payment**
    - Select car and rental duration via calendar
    - Pay a percentage (defined by the client) using integrated payment API (later stage - Baridi mob API)
    - Receive confirmation and notifications
- **Notifications**
    - Email or in-app notifications for reservation updates, payments, etc.

---

4. Non-Functional Requirements

Category	Description
Performance	The system should handle up to 500 concurrent users per client.
Security	All data transmitted via HTTPS; passwords encrypted.
Availability	99% uptime SLA for hosted clients.
Scalability	Multi-tenant architecture to allow growth of client base.

Usability	Intuitive UI with responsive design.
Maintainability	Modular code architecture for easy updates.
Payment Integration	Use secure APIs (e.g., Stripe, PayPal).

---

5. System Design Overview

5.1 Architecture

- **Frontend:** React / tailwind Web
- **Backend:** Nest.js / Node.js / Express (or Laravel / Django depending on team decision)
- **Database:** PostgreSQL / MongoDB
- **Hosting:** Cloud-based (e.g., AWS, Render, or Vercel for frontend)

5.2 Modules Diagram (Simplified)

```
        ┌──────────────────────────┐
        │    SaaS Main Platform    │
        └────────────┬─────────────┘
                     │
        ┌────────────┴────────────┐
        │                         │
   ┌────▼────┐              ┌─────▼───┐
   │ Client  │              │ Customer│
   │Dashboard│              │ Portal  │
   └─────────┘              └─────────┘
```

---

6. Project Plan

6.1 Task Division

Team Member	Responsibilities
Salah (Leader)	Project management, frontend integration, backend integration
Ghofrane	UI/UX design (dashboard, user portal, branding)

Hassib	Frontend integration
Badro	Backend (database models, authentication, reservations API)
Khaled	Backend …

6.2 Development Phases and Duration (Estimated)

Phase	Description	Duration
1. Requirement Analysis & SRS	Define system requirements	Done
2. UI/UX Design	Wireframes, prototypes (Figma)	2 weeks
3. Backend Development	Database setup, API endpoints	3 weeks (in parallel)
4. Frontend Development	Dashboard + user interfaces	3 weeks (in parallel)
5. Integration	Connect frontend & backend, authentication	1 week
6. Testing & Debugging	Unit + integration testing	1 week
7. Deployment	Host and launch the system	0.5 week
Total Duration	—	**~6 weeks (1.5 month)**

6.3 Tools & Technologies

Category	Tool
Design	Figma
Frontend	React / Tailwind
Backend	Nest.js / Express (you will decide and update it here)
Database	PostgreSQL / MongoDB (you will decide and update it here)

Version Control	GitHub
Project Management	Trello / Notion / Telegram / Jira / Slack
Hosting	Vercel (frontend), Render/AWS (backend)

---

7. Future Enhancements

- Advanced analytics (AI-based recommendations for pricing or maintenance)
- Integration with car tracking IoT systems
- Multi-language and multi-currency support

---

8. Conclusion

This SRS outlines the structure, features, and plan for building our SaaS-based Car Rental Management System.
The project aims to simplify rental operations for clients and provide a smooth, modern experience for customers, combining efficiency, usability, and scalability.