Software Requirements Specification

for

Rent A Car

**Version 1.0 approved**

**Prepared by, Ayesha AbdulRahman (22k-4591),Muhammad Yameen(21k-3452), M. Affnan Soomro(21k-4846)**

**FAST NUCES**

Contents

[1. Introduction 1](#_Toc197980854)

[1.1. Purpose 1](#_Toc197980855)

[1.2. Document Conventions 1](#_Toc197980856)

[1.3. Intended Audience and Reading Suggestions 1](#_Toc197980857)

[1.4. Product Scope 1](#_Toc197980858)

[1.5 References 1](#_Toc197980859)

[2. Overall Description 2](#_Toc197980860)

[2.1.1. Product Perspective 2](#_Toc197980861)

[2.2 Product Functions 2](#_Toc197980862)

[2.3 User Classes and Characteristics 2](#_Toc197980863)

[2.4 Operating Environment 2](#_Toc197980864)

[2.5 Design and Implementation Constraints 2](#_Toc197980865)

[2.6 User Documentation 2](#_Toc197980866)

[2.7 Assumptions and Dependencies 2](#_Toc197980867)

[3. Specific Requirements 3](#_Toc197980868)

[3.1 External Interface Requirements 3](#_Toc197980869)

[3.1.1 User Interfaces 3](#_Toc197980870)

[3.1.2 Hardware Interfaces 3](#_Toc197980871)

[3.1.3 Software Interfaces 3](#_Toc197980872)

[3.1.4 Communication Interfaces 3](#_Toc197980873)

[3.2 Functional Requirements 3](#_Toc197980874)

[3.3 System Features 3](#_Toc197980875)

[4. Other Non-Functional Requirements 4](#_Toc197980876)

[4.1 Performance Requirements 4](#_Toc197980877)

[4.2 Safety Requirements 4](#_Toc197980878)

[4.3 Security Requirements 4](#_Toc197980879)

[4.4 Software Quality Attributes 4](#_Toc197980880)

[4.5 Business Rules 4](#_Toc197980881)

[5. Other Requirements 5](#_Toc197980882)

[Appendix A: Glossary 5](#_Toc197980883)

[Appendix B: Analysis Models 5](#_Toc197980884)

[Class Diagram: 5](#_Toc197980885)

[ER Diagram: 6](#_Toc197980886)

[System level Architecture 7](#_Toc197980887)

[Design Strategy: 8](#_Toc197980888)

[Sequence Diagram (Behavior Model): 9](#_Toc197980889)

[Collaboration Diagram: 10](#_Toc197980890)

[State Diagram (Behavior Model): 11](#_Toc197980891)

[Activity Diagram (Flow-based Model): 13](#_Toc197980892)

[Appendix C: To Be Determined List 14](#_Toc197980893)

# Introduction

## Purpose

This Software Requirements Specification (SRS) provides a detailed description of the “Rent A Car” system. The application facilitates car rental services with support for admin and customer roles. It aims to ensure secure and efficient car bookings and management.

## Document Conventions

Requirement priorities are noted as:

* H: High
* M: Medium
* L: Low

Requirement IDs follow the format: REQ-1, REQ-2, etc.

## Intended Audience and Reading Suggestions

This document is intended for software developers, project managers, QA testers, and stakeholders. Readers looking for a general understanding should start with the Product Scope and Overall Description.

## Product Scope

The Rent A Car application will enable users to register as Admin or Customer. It provides secure login, car listings, booking functionality, and dashboards tailored to each role. The system is built with Angular and Spring Boot and backed by a MySQL database.

## 1.5 References

* Spring Boot Documentation: https://spring.io/projects/spring-boot
* Angular Documentation: https://angular.io
* MySQL Documentation: https://dev.mysql.com/doc/
* Ng-Zorro Documentation: https://ng.ant.design

───────────────────────────────────────────────

# Overall Description

## Product Perspective

Rent A Car is a web-based application consisting of a client interface (Angular), backend APIs (Spring Boot), and a MySQL database. It is a standalone product, intended to streamline car rental operations.

## 2.2 Product Functions

* User registration and login
* Role-based dashboards
* Admin: car management and booking control
* Customer: car browsing, booking requests, booking history
* Encrypted password handling
* Search and filter cars

## 2.3 User Classes and Characteristics

* Admin: Manages cars and bookings. Technically proficient.
* Customer: General user, can browse and rent cars.

## 2.4 Operating Environment

* Frontend: Angular with Ng-Zorro
* Backend: Spring Boot with Maven
* Database: MySQL
* Platform: Any modern browser (Chrome, Firefox, Edge)

## 2.5 Design and Implementation Constraints

* Must follow REST API principles
* Must ensure encrypted password storage
* Use of Angular for UI and Spring Boot for backend logic
* Responsive layout for desktop and mobile

## 2.6 User Documentation

* Admin and Customer user manuals (to be created)
* Help tooltips and UI guides embedded in frontend

## 2.7 Assumptions and Dependencies

* Internet connectivity is required
* Compatible with modern browsers
* Java, Node.js, and MySQL should be installed in development environment

───────────────────────────────────────────────

# 3. Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

* Responsive web interface using Angular
* Separate dashboards for Admin and Customer
* Search bars, form inputs, and booking views

### 3.1.2 Hardware Interfaces

* **Client:** Devices with browser (PC, laptop, mobile)
* **Server:** Hosted backend with database and APIs

### 3.1.3 Software Interfaces

* Frontend: Angular with Ng-Zorro
* Backend: Spring Boot (Java, Maven)
* Database: MySQL

### 3.1.4 Communication Interfaces

* Secure HTTP (HTTPS) for all data transmission
* RESTful API for frontend-backend communication

## 3.2 Functional Requirements

REQ-1: Users must be able to sign up with a role (Admin or Customer)

REQ-2: Users must log in with secure password validation

REQ-3: Passwords must be stored in encrypted format

REQ-4: Admins can add, update, and delete car entries

REQ-5: Admins can approve/reject booking requests

REQ-6: Customers can view available cars

REQ-7: Customers can submit booking requests

REQ-8: Both users should access a search function

REQ-9: Customers can view booking history

### 3.3 System Features

System Feature: User Authentication

Description: Sign up, login, and role management

Priority: High

Inputs: Name, email, password

Outputs: Role-based dashboard redirection

System Feature: Car Management (Admin)

Description: Manage car listings and availability

Priority: High

Inputs: Car details (name, brand, model, price, etc.)

Outputs: Car list shown to customers

System Feature: Booking Management

Description: Submit and review bookings

Priority: High

Inputs: Car ID, booking duration, customer ID

Outputs: Booking status (Pending/Approved/Rejected)

───────────────────────────────────────────────

# 4. Other Non-Functional Requirements

## 4.1 Performance Requirements

* The system should support 500 concurrent users
* Page loads should not exceed 2 seconds under normal conditions

## 4.2 Safety Requirements

* Regular backups of user and booking data
* Failover support in case of backend crash

## 4.3 Security Requirements

* Passwords stored using encryption (e.g., bcrypt)
* HTTPS for secure data transfer
* Role-based access control

## 4.4 Software Quality Attributes

* Usability: Clean and intuitive UI with tooltips
* Reliability: 99.9% uptime goal
* Scalability: Scalable APIs and modular frontend
* Maintainability: Proper documentation and modular code

## 4.5 Business Rules

* A car cannot be double-booked for the same duration
* Booking can only be placed if the car is available
* Admin must approve bookings before confirmation

───────────────────────────────────────────────

# 5. Other Requirements

* Payment integration may be added in the future
* Email notifications may be integrated for booking confirmation
* Image uploads for car listings

───────────────────────────────────────────────

# Appendix A: Glossary

CRUD: Create, Read, Update, Delete

ERD: Entity Relationship Diagram

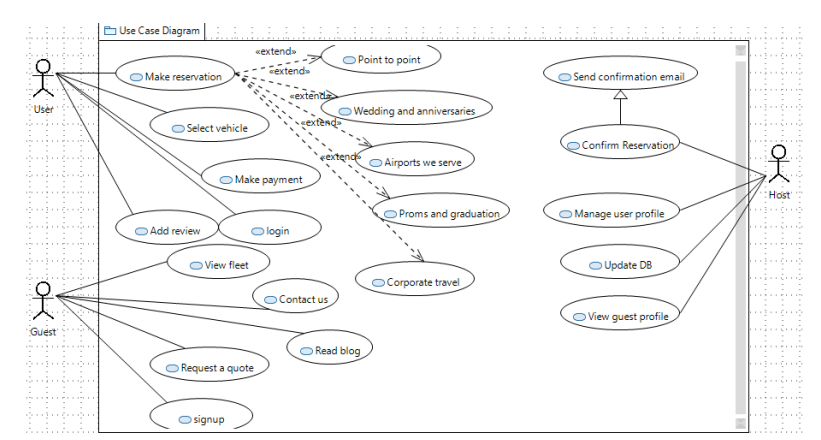
JWT: JSON Web Token

REST: Representational State Transfer

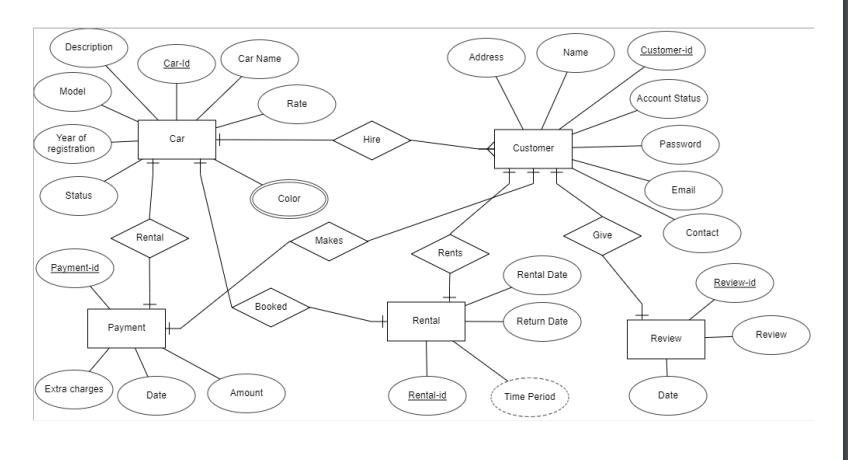
UI: User Interface

# Appendix B: Analysis Models

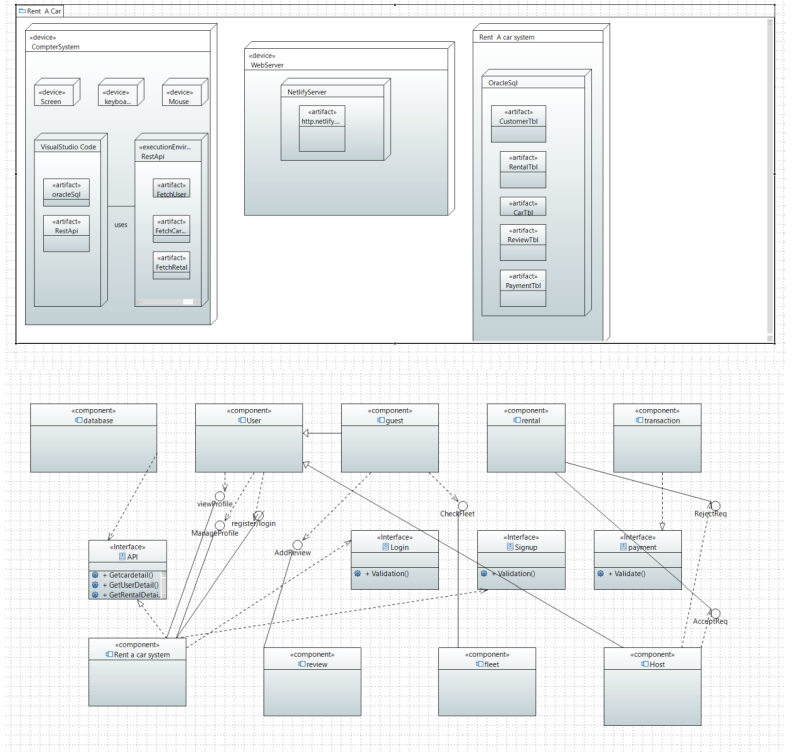
## Class Diagram:



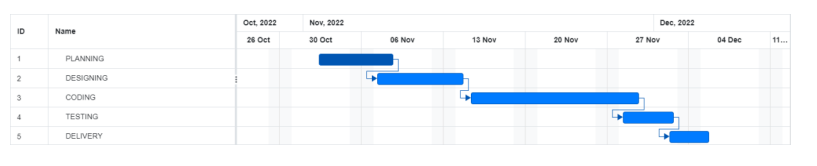
## ER Diagram:



## System level Architecture

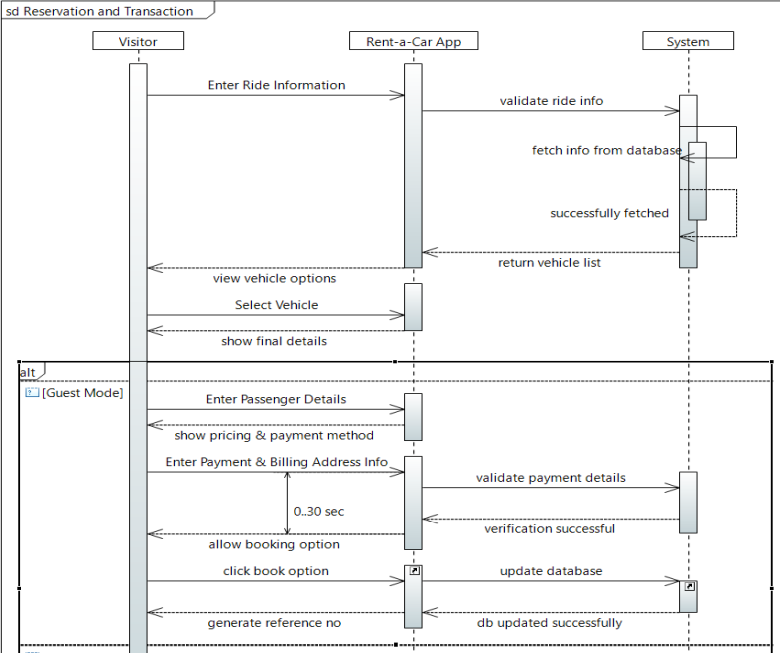


## Design Strategy:

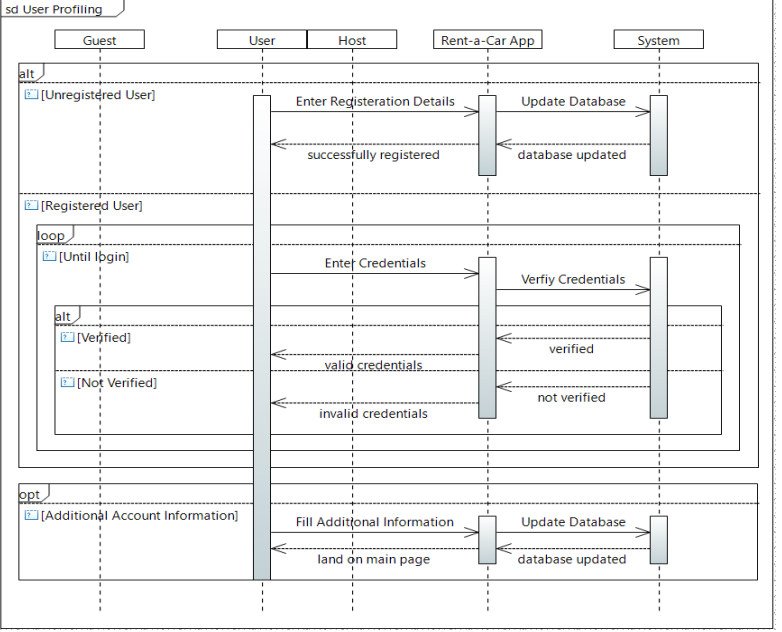


During first 6 days we collected the information related to our project and problem statement and design a proposal which work as a ground work of what our project will do and what technology will be used. Then UML diagrams were generated to get an idea of how our system will interact with different hardware, software and its user and what features it will provide. Next was the largest phase of coding where we setup work environments and database were built and connected using API to the frontend and then testing was done with dummy data and next project was delivered.

## Sequence Diagram (Behavior Model):



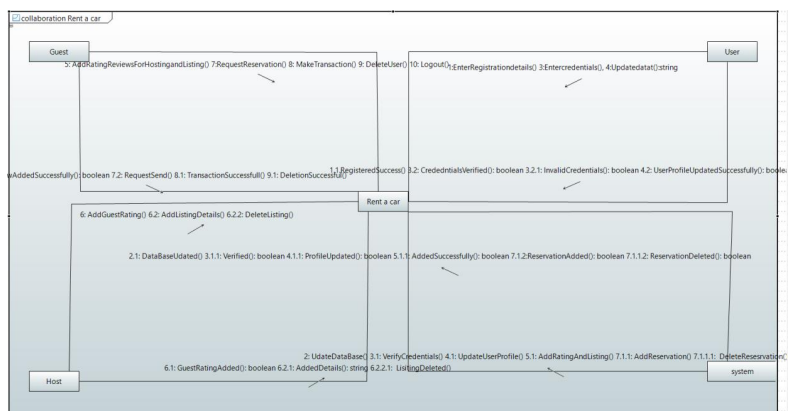
A visitor arrives at the webpage and feeds the ride information to the web application which is then in return verified by the system by fetching the information from the database and the list of vehicles is returned. The visitor will then view the options available to select the vehicle from. He then selects the vehicle and the final details are displayed to them through a GUI. For booking purposes, the guest will enter the passenger details and an invoice will be generated. The guest enters the credentials for the credit card. There is a 30 second time window acting as the captcha meanwhile, the system verifies the provided credentials at the backend server. If the approval is received, the database is updated and the ride is booked successfully



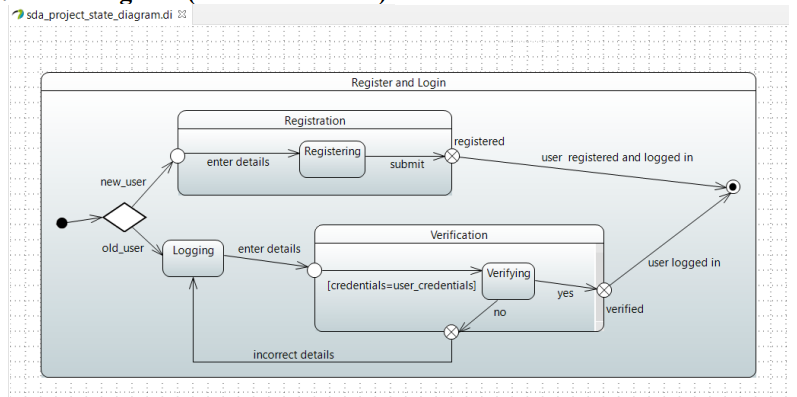
The user profiling section presents a more advanced overview of the system in general. The guest and host timeline are not as interactive. This diagram is a basic representation of the functionalities occurring at the backend such as verification of the registration details, credentials and updating of the database in case of additional information being received

## Collaboration Diagram:

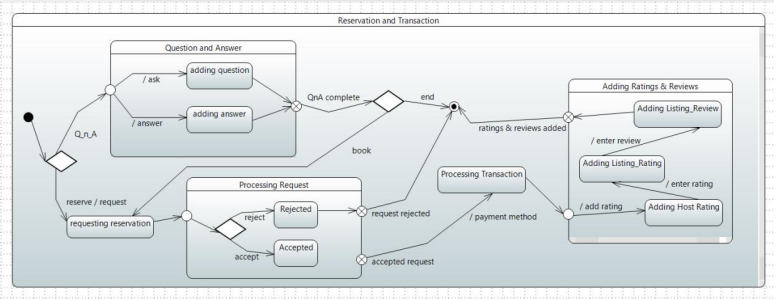
The collaboration diagram shows 4 actors in guest, user, system and the host itself. The functionalities and features that can be availed by the following is depicted through the functions listed beside them. The numbering before the name of the function represents the flow of the website and the order in which they may be accessed.



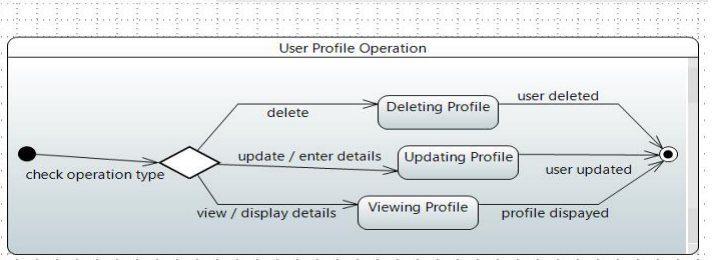
## State Diagram (Behavior Model):



The state diagram above refers to the register and login scenario. When a user first arrives at the website, he may be new to the system or a regular customer. In case of a new user, he first has to follow the signup procedure and register with the system. Whereas, the already registered visitor will have to log in to the system, where his provided credentials are verified. If the credentials entered were correct, the user will be given access to the features of the web application. Or else he will be redirected to the login page to reenter the correct credentials.

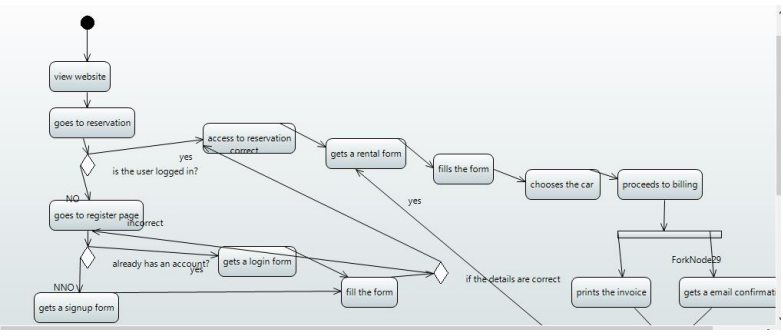


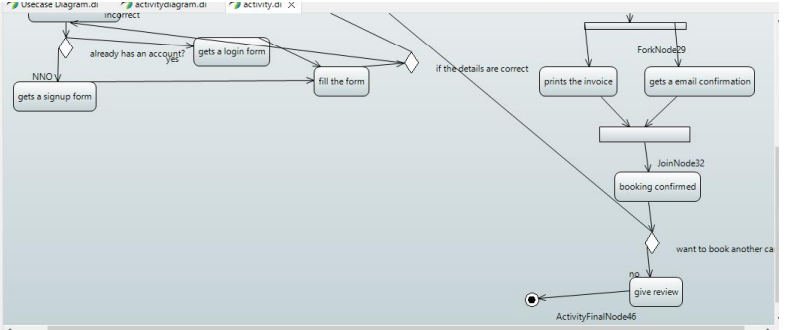
Before reservation, a customer may have questions and answers regarding the services provided, such as is the desired car available, so a separate page caters to this need of the clients. If the customer is satisfied and wants to make a reservation, he sends a request, if the request is rejected, he cannot book a ride due to the website’s safety protocols. However, if the request is processed and approved, the transaction is made by the customer to receive a confirmation of the booking via email. In the end before leaving the system, they are required to leave a feedback through reviews and rating the overall experience.



The above state diagram is a simple model of the operations the system undergoes at the backend when a user requests to delete, to update or to view their previous entered profile information while setting up their accounts.

## Activity Diagram (Flow-based Model):





A visitor to the website first arrives at the reservation page where he is asked to either login or signup to the system to continue. If the user is not registered, he will have to fill up the signup form and then is granted access to the reservation page. If the user was already registered with the system, he logs in and is directed to the reservation page. The user is then specified to fill the rental form which includes details such as selecting the car required by the user. He is then navigated to the billing page where an invoice is generated. After the user completes the transaction, he receives a confirmation email for the booking. Later on, if the customer wants to book another car as well, he is redirected to the make reservation page; else, he is sent to the page where he will leave the review for the experience

# Appendix C: To Be Determined List

Payment Gateway Integration

Booking Notification via Email

Review and Rating System for Cars

───────────────────────────────────────────────

## 