Software Requirements Specification

for

VEHICLE BOOKING SERVICE

Version 1.0

Prepared by

Group Name: DBMS_CS04_GRP1

Sahil Muhammed	B230089CS	sahil_b230089cs@nitc.ac.in
Sreehari Sanjeev	B230574CS	sreehari_b230567cs@nitc.ac.in
Nishanth K Nizar	B230471CS	nishanth_b230471cs@nitc.ac.in
Mohammed Omar	B230426CS	mohammed_b230426cs@nitc.ac.in

Instructor: Professor Prabhu Mohandas

Course: CS2011E DBMS

Date: 28/02/2025

TABLE OF CONTENTS

1 Introduction

Our service aims to be a vehicle booking application that allows its users to book vehicle rides and avail transportation at fair costs. The web-app gives the passengers the ability to book hassle free rides and drivers to sign up with us and earn money. This section of the SRS document directs the reader to the purpose, scope, definitions and conventions involved in our service.

1.1 Document Purpose

The purpose of this document is to give a description of the software requirements for the Vehicle booking service/application which allows users to book vehicles and avail transportation at cheap and reasonable prices. The Software Requirement Specification document covers the entire service's system of operations, capturing both the passenger and driver functionalities, thereby ensuring an efficient travel experience.

1.2 Product Scope

The service exists as a web-app that connects users with drivers. The system is guaranteed to give the user a hassle-free ride booking experience. The web-app allows the passengers to book rides using a simple web interface and can proceed to rate/review once their journey has concluded. The drivers can register on our platform and accept ride requests when they deem fit according to their schedule.

Software used:

1. **Frontend:** Next.js (React.js)
2. **Backend:** Node.js with Express.js
3. **Database:** SQL
4. **Deployment**: Vercel and Azure
5. **Online Payment facilitator:** Razorpay

The purpose of this service is to provide a platform for passengers and drivers to connect with one another for the purpose of transportation. The benefit of using a platform like ours lies in the fact that we don't own the vehicles or provide them, we simply serve as middlemen to facilitate your freight. Our primary goal is to ensure the passengers' flawless transit from location 'A' to location 'B' while also guaranteeing fair payments per ride for the drivers.

1.3 Intended Audience and Document Overview

The SRS document is intended for the development team, project managers, marketing staff, users, testers and Professor Prabhu Mohandas. The document gives an outline over the service's functional and non-functional requirements and project scope. Those that wish to read this document should do this with relevance to their role and should start with the overview sections that describe the overall project scope.

1.4 Definitions, Acronyms and Abbreviations

API: Application Programming Interface
CRUD: Create, Read, Update and Destroy
DBMS: Database Management System
UI/UX: User Interface/User Experience
ETA: Estimated time of Arrival

1.5 Document Conventions

This document follows the IEEE formatting requirements. The text is written in Arial with a font size of 11 or 12, with spacing and 1-inch margins. Italics are used for comments and notes.

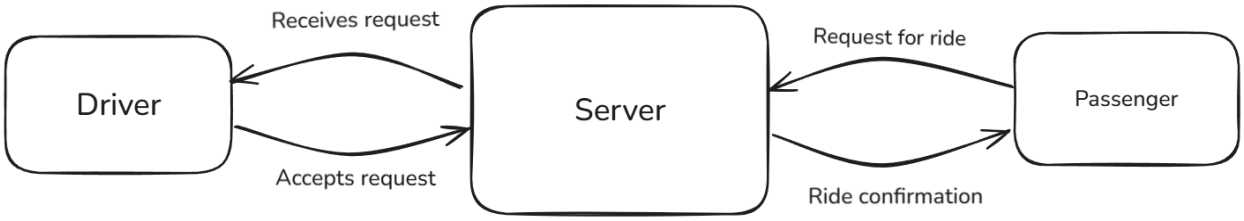
1.6 References and Acknowledgments

1. IEEE Standard for Software REquirement Specifications
2. React.js and Node.js Official Documentation
3. SQL Documentation

2 Overall Description

2.1 Product Overview

The Vehicle Booking Service Web-app provides seamless transportation services to passengers while also offering the drivers a platform to earn money by facilitating the passengers' requests. The entire system doesn't rely on other existing systems/services, although it may be similar to them. The service does rely on mapping services and payment gateways externally, whose APIs will be used. An administrative panel is also integrated within to help the development team manage platform operations.



2.2 Product Functionality

The core functionality of the system includes:

- The creation of the personal account of the passenger.
- Booking rides
- Real-time connection between available drivers and logged in passengers
- Real-time ETA of the drivers
- The creation of the personal account of the driver
- Automatic fare calculation
- Updation of the fare in the case of busy roads, rainy weather etc.
- Payment processing using Razorpay
- Ability to rate and review for both the driver and passenger
- Admin panel for monitoring accounts

2.3 Design and Implementation Constraints

Building a service like this factors in several constraints that influence the development of the web-app.

- **Tech-Stack:** The web-app will be developed using React for the frontend, Node.js with Express for the backend and SQL for the database.
- **Logical ER Modeling:** The database's architecture will be based off of an ER diagram handling the various entities and relationships.
- **Hardware Limitations:** The system would need to be optimized for cloud deployment and should also support mobile devices with reasonable performance.

2.4 Assumptions and Dependencies

- The service requires the user to have internet access for operations to begin.

- The web-app depends on third-party services such as Google Maps API for the location tracking and Razorpay API for payment processing.
- The backend leverages scalability through cloud hosting.
- The service assumes that the drivers will adhere to transportation laws and regulations.
- The system heavily relies on the availability of a steady network connection to ensure real-time tracking.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The web-app provides an easy-to-use interface for both the passengers and the drivers. Users can interacts with the application through:

- **The Passenger's Dashboard:** Once logged in, the users are allowed to book rides. Once booked, the ETA is displayed for the user to see and await their pick-up.
- **The Driver's Dashboard:** Allows the drivers to accept ride requests if a passenger nearby has requested for one. Gives the drivers the ability to toggle their availability and view their earnings for the day.
- **The Booking Interface:** The user can input their pickup and drop-off locations. They can also select the type of transportation they'd like to avail. We offer a wide range of vehicles to choose from:
    - **Bikes:** Your go-to vehicle to tackle the busy roads on a sunny day.
    - **Autos:** Avail cheap transportation for you and your friends.
    - **General Cabs:** Book these to experience a hassle free ride on those long journeys which demand a peace of mind.
    - **Premium Cabs:** The most luxurious of them all. With multiple seats and refreshments in the vehicle, these cabs are for those who desire an unforgettable experience.
- **The Payment Interface:** Secure payments either through Razorpay or through fiat currency.

3.1.2 Hardware Interfaces

The web-app expects the availability of the following hardware components for smooth functioning:

- **GPS Receiver inside the device:** To 'get' the current location of both the driver and passenger.
- **Network Card (NIC):** This facilitates communication with the public network.
    - Furthermore, this is the 'gateway' via which most of the traffic to and from the servers will be directed.

3.1.3 Software Interfaces

The web-app relies on third-party software to ensure compartmental functioning:

- **Google Maps API:** To infer the current position of the driver and passenger.
- **Razorpay API:** To process payments made using the internet.
- **Database Management System:** This is crucial to maintain history and records.

3.2 Functional Requirements

3.2.1 **F1:** The system shall allow the passengers and the drivers to create an account and log in.

3.2.2 **F2:** The system shall allow the passengers to select their pickup and drop-off locations.

3.2.3 **F3:** The system shall allow the passengers to choose from the different vehicles available for transportation.

3.2.4 **F4:** The system shall also display the prices for each of these vehicles of transportation. Additionally, the system must also update these prices based on environmental conditions.

3.2.5 **F5:** The system shall allow the drivers to toggle between being available and not being available.

3.2.6 **F6:** The system shall allow the drivers to accept or decline ride requests.

3.2.7 **F7:** The system shall display the ETA of the driver in the passenger's interface.

3.2.8 **F8:** The system shall store ride history for both the passengers and the drivers.

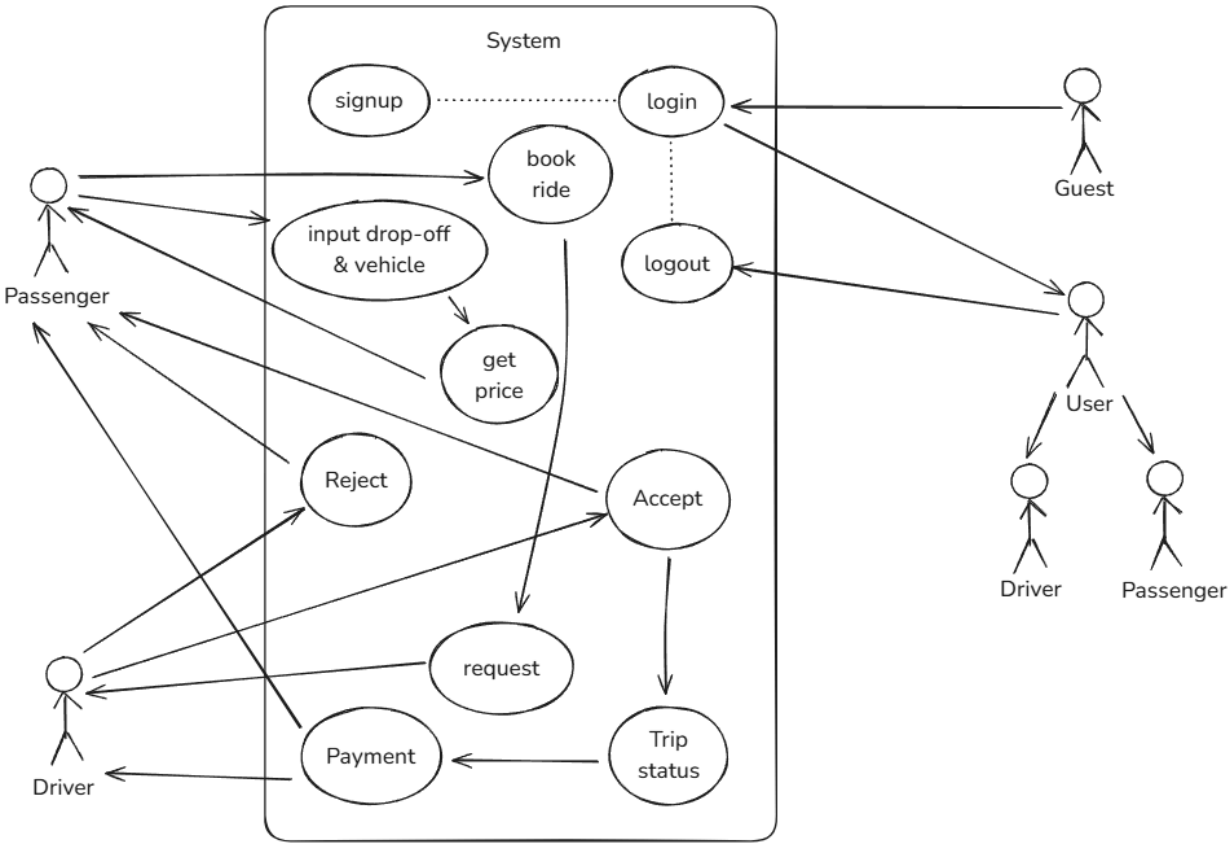
3.2.9 **F9:** The system shall send confirmations upon ride acceptance and completion to any of the concerned entities.

3.2.10 **F10:** The system shall allow the passengers to earn rewards by the use of our web-app for their transit.

3.2.11 **F11:** The system shall allow both the passengers and the drivers to rate/review each other after the completion of every ride.

3.3 Use Case Model

The below drawn use case model illustrates a high-level overview of the workflow involved in the usage of our web-app.



3.3.1 U1: Sign Up / Log in

Author: Sreehari
Purpose: Allows the passengers and drivers to create an account. If already registered, they're allowed to log in and access the web-app.
Requirements Traceability: The system must store the user credentials securely in the database.
Priority: High.
Pre-conditions: The user must be connected to the internet.
Post-conditions: The user should now be logged in and be able to access the web-app's core functionality.
Actors: Passenger, Driver.
Extends: None
Flow of Events:

- **Basic Flow:**
  - User enters an email address and password.
  - The database validates the entered credentials.
  - If valid, the user is sent to the relevant dashboard.
- **Alternative Flow:**
  - If the user hasn't ever registered with the web-app, then they can register by providing the relevant details.
- **Exceptions:**
  - An error message is displayed if incorrect credentials are entered.

Includes: None
Notes/Issues: None

3.3.2 U2: Book Ride

Author: Sahil
Purpose: To allow the passengers to book a vehicle of their choice for transit.
Requirements Traceability: Should include the display of the fare while also confirming the booking if a driver does accept the ride request.
Priority: High.
Pre-conditions: Passenger must be logged in and these must be drivers available in the vicinity of the passenger.
Post-conditions: The ride request has been sent and the driver has been assigned to this user.
Actors: Passenger, Driver, Server.
Extends: None.
Flow of Events:

- **Basic Flow:**
  - Passenger inputs a pickup and drop-off location.
  - Passenger selects the vehicle they'd like to transit in.
  - The system responds with an estimate for the fare.
  - Passenger confirms the booking.

        - The system requests drivers in the vicinity to either accept or deny the passenger's request.
        - One of the drivers accepts the request.
    - **Alternative Flow:**
        - If there aren't any drivers available, the passenger is notified relevantly.
    - **Exceptions:**
        - Network Failure.

Includes: U3 and U4.

Notes/Issues: Drivers must be assigned properly. (the same passenger must not be assigned 2 different drivers)

3.3.3 U3: View Ride Fare Estimate

Author: Omar

Purpose: To allow the passengers to see an estimated cost of their ride before they book the ride.

Requirements Traceability: Should factor in the distance of the transit, traffic and the environmental conditions.

Priority: Medium

Pre-conditions: Passenger must have entered the pickup and drop-off locations. Additionally, the passenger must also specify the vehicle of transit.

Post-conditions: The fare price is displayed.

Actors: Passenger, Server.

Extends: None.

Flow of Events:

- **Basic Flow:**
    - Passenger enters the pickup and drop-off locations.
    - The system responds with the estimated fare price.
- **Alternative Flow:**
    - Additional costs might add onto the base price depending on availability of drivers, traffic or rainy weather.

Includes: None.

Notes/Issues: Must always ensure fair pricing.

3.3.4 U4: Display the ETA of the Ride

Author: Nishanth

Purpose: Allows the passenger to view the ETA of their booked ride.

Requirements Traceability: Utilizes GPS to deliver the ETA.

Priority: High

Pre-conditions: The ride must be booked. The driver must be assigned.

Post-conditions: Passengers should now be able to see the ETA.

Actors: Passenger, Server, Driver.

Extends: None

Flow of Events:

- **Basic Flow:**
  - The driver starts their journey towards the passenger.
  - The passenger can view the ETA on their screen.
- **Exceptions:**
  - GPS failure from the driver's end.

Includes: None
Notes/Issues: None

3.3.5 U5: Cancel Ride

Author: Omar
Purpose: To allow the passengers to cancel the requested ride.
Requirements Traceability: Cancellation Policy.
Priority: Medium.
Pre-conditions: The ride must have been booked and not completed yet.
Post-conditions: Confirmation that the ride has been canceled must be sent to both the driver and the passenger.
Actors: Passenger, Server, Driver.
Extends: None.
Flow of Events:

- **Basic Flow:**
  - Passenger requests to cancel the ride.
  - The server processes the cancellation request.
- **Alternative Flow:**
  - To ensure fairness, if the passenger requests for cancellation after the driver has moved a significant amount towards their pick-up location, the passenger is penalized.
- **Exceptions:**
  - Network failure.

Includes: None.
Notes/Issues: A proper penalized amount must be set.

3.3.6 U6: Make payments

Author: Sreehari
Purpose: To give the ability to the passengers to pay for the rides they avail.
Requirements Traceability: Secure online transactions.
Priority: High.
Pre-conditions: The passenger must have reached their mentioned drop-off location.
Post-conditions: The driver has been paid.
Actors: Passenger, Server, Razorpay API.
Extends: None.
Flow of Events:

- **Basic Flow:**
  - The passenger requests to complete the payment.
  - The request is redirected to Razorpay.

        - If the payment was successful, the driver and passenger can rate/review each other.
    - **Alternative Flow:**
        - The passenger may choose to pay using cash.
    - **Exceptions:**
        - Payment failure.

Includes: None.

3.3.7 U7: Accept Ride Requests as a Driver.

Author: Sahil
Purpose: To allow the drivers to accept or decline ride requests.
Requirements Traceability: The driver must receive the notification followed by matching the passenger to this driver.
Priority: High.
Pre-conditions: The passenger requests for a ride.
Post-conditions: The ride has been accepted by the driver and the driver begins to navigate their way towards the passenger.
Actors: Driver, Server.
Extends: None.
Flow of Events:

- **Basic Flow:**
    - The driver receives the ride request.
    - The driver accepts the ride.
    - The passenger is notified of the confirmation of their ride.
- **Alternative Flow:**
    - If the driver declines the request, the server tries matching the passenger with some other driver.

Includes: None.

3.3.8 U8: Complete Trip and Update Status

Author: Nishanth
Purpose: Mark a trip as complete and initiate the payment protocol.
Requirements Traceability: Updation of the trip's completion.
Priority: High.
Pre-conditions: Ride must be almost complete.
Post-conditions: Trip has been updated to be complete.
Actors: Driver, Server, Razorpay API.
Extends: None.
Flow of Events:

- **Basic Flow:**
    - Driver reaches the drop-off location specified by the passenger.
    - Passenger confirms the completion of the trip.

Includes: U6.

4 Other Non-functional Requirements

4.1 Performance Requirements

- **P1:** The system shall return the ride fare within 5 seconds of the passenger entering the pickup, drop-off locations and the choice of vehicle.
- **P2:** The system shall notify the passenger of an assigned driver within 5 seconds of acceptance.
- **P3:** The system shall accurately display the ETA of the driver.

4.2 Safety and Security Requirements

- **S1:** All user and driver sensitive data must be encrypted.
- **S2:** The system shall not abuse its power of having access to the location of the driver and passenger.
- **S3:** The system must flag and report suspicious behavior from either end.
- **S4:** In case of an emergency, the system must provide a SOS button.

Appendix A – Data Dictionary

Variable Name	Type	Description	Example values	Related operations
user_id	Integer	Uniquely identifies a user	>=0	CRUD
driver_id	Integer	Uniquely identifies a driver	>=0	CRUD
ride_id	Integer	Uniquely identifies a ride	>=0	CRUD
ride_status	String	Status of a ride	“Requested”, “Accepted”, “Completed”, ”Canceled”	Updation of the ride status
fare_amount	Float	Total cost of the ride	>=0	Calculation of the fare
payment_status	String	Status of the payment	“Pending”, “Paid”, “Failed”	Processing the payment using the API
passenger_rating	Float	Rating given by the driver	>=0	Update the Database with the rating
driver_rating	Float	Rating given by the passenger	>=0	Update the Database with the rating
vehicle_type	String	The type of the vehicle	Different types of vehicles	Vehicle Selection
pickup_location	String	The start point of the ride	Any valid coordinates	Geolocation from the API
dropoff_location	String	The end point of the ride	Any valid coordinates	Geolocation from the API

Appendix B - Group Log

Date	Attendees	Topics Discussed	Actions taken
25/02/2025	Sahil, Nishanth, Sreehari, Omar	Initial planning and the preparation of the SRS document was started.	Defining roles.
26/02/2025	Sahil, Nishanth, Sreehari, Omar	ER diagram and database schema design.	Working on the ER diagram.
27/02/2025	Sahil, Nishanth, Sreehari, Omar	Use case model diagram and product overview diagram	Completed the SRS document