Software Requirements Specification

for

Smart Traffic Routing

Version 1.0

Prepared by

K.Prabakaran
M.S.Sukil Raj
K.N.Sanjay

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for the Smart Traffic Management System, Version 1.0. Its purpose is to provide a detailed overview of the system, its features, and its intended behavior to guide the development and testing process.

1.2 Document Conventions

This document follows a standard SRS template. Standard font and formatting conventions are used throughout.

1.3 Intended Audience and Reading Suggestions

The intended audience for this document includes project managers, software developers, testers, and system administrators. It is recommended to read the document sequentially to gain a comprehensive understanding of the project.

1.4 Product Scope

The Smart Traffic Management System is a web-based application designed to optimize urban traffic flow by allowing users to pre-book a travel path. The system will use Dijkstra's algorithm to calculate the shortest available route based on the desired start time and existing bookings. It will feature two distinct user roles: **Users**, who can book and manage their trips, and **Admins**, who can monitor system activity and manage user appointments. The primary goals are to reduce traffic congestion, minimize travel time, and provide a centralized platform for traffic management.

1.5 References

This document is the primary source of requirements for the project.

2. Overall Description

2.1 Product Perspective

The Smart Traffic Management System will be a new, self-contained product. It is intended to function as a standalone web application, providing a centralized solution for intelligent traffic routing.

2.2 Product Functions

The major functions of the system are:

- Secure user authentication for both User and Admin roles.
- A User dashboard for booking road paths, viewing scheduled trips, and cancelling trips.
- Path calculation using Dijkstra's algorithm to determine the shortest route.
- An Admin dashboard to monitor a live map of user travel, view user timestamps, and cancel appointments.
- A traffic visualization feature for users.

2.3 User Classes and Characteristics

There are two main classes of users:

- **User:** General users who want to book a travel path. They can register, log in, book trips by providing a start/end point and time, view their booked trips, and cancel them.
- **Admin:** System administrators responsible for monitoring and managing the system. Admins can view all user activity on a map, check the allocated time for each user's travel, and have the authority to cancel any user's booking if necessary

2.4 Operating Environment

Client-Side: The application will be accessible through modern web browsers (e.g., Chrome, Firefox, Safari, Edge) on standard desktop and mobile operating systems.
Server-Side: The backend will run on a Node.js environment. The system will utilize a MongoDB database for data storage and a Python environment to execute the pathfinding algorithm.

2.5 Design and Implementation Constraints

Technology Stack: The system must be developed using React for the frontend, Node.js for the backend, and MongoDB as the database.
Algorithm: Pathfinding and time-slot allocation must be implemented using Dijkstra's algorithm, executed via a Python script.
Protocols: Communication between the client and server will be handled via standard HTTP/S protocols

2.6 User Documentation

User documentation will include online help sections and tutorials to guide both Users and Admins on how to use the system effectively.

2.7 Assumptions and Dependencies

It is assumed that accurate and up-to-date map data (road networks, distances) is available for the algorithm.
The system depends on the successful integration of the Python-based Dijkstra's algorithm with the Node.js backend.
Users are assumed to have a stable internet connection to access the web application.

3. External Interface Requirements

3.1 User Interfaces

The system will feature a clean, intuitive, and responsive web-based graphical user interface (GUI). Key UI elements will include:

- A login page with separate entry points or role selection for Users and Admins.
- A User Dashboard with controls for entering start/destination points, selecting a time, viewing booking results, and managing trips.
- An Admin Dashboard featuring an interactive map for real-time monitoring and a table view for appointment details

3.2 Hardware Interfaces

No special hardware interfaces are required beyond standard client devices (PCs, tablets, smartphones) and a server to host the application.

3.3 Software Interfaces

- Frontend-Backend API: The React frontend will communicate with the Node.js backend via a RESTful API to send requests and receive data.
- Database: The Node.js server will interface with a MongoDB database to store and retrieve user data, bookings, and road network information.
- Python Script Execution: The Node.js backend will execute the Python script containing Dijkstra's algorithm to perform path calculations

3.4 Communications Interfaces

All communications between the client and server will be conducted over standard TCP/IP and HTTP/S protocols to ensure secure data transfer.

4. System Features

4.1 Feature 1: User and Admin Authentication

4.1.1 Description and Priority

High. The system must provide secure login capabilities for both User and Admin roles.

4.1.2 Stimulus/Response Sequences

A user enters their credentials and selects a role. The system validates the credentials against the database and directs the user to the appropriate dashboard.

4.1.3 Functional Requirements

REQ-1: The system shall provide separate login forms or a role-selection option for Users and Admins.

REQ-2: User passwords must be securely hashed and stored in the database.

REQ-3: The system shall manage user sessions to maintain login status.

4.2 Feature 2: User Module

4.2.1 Description and Priority

High. This module contains all functionalities available to the standard user.

4.2.2 Functional Requirements:

REQ-4 (Path Booking): The user shall be able to input a starting point, a destination, and a desired start time for their trip.

REQ-5 (Shortest Path Calculation): Upon receiving a booking request, the system shall execute Dijkstra's algorithm to calculate the shortest available path.

REQ-6 (Road Availability): The system shall check if the calculated path is free at the requested time. If available, the user can book the path.

REQ-7 (Alternative Paths): If the primary shortest path is already booked, the system shall show alternative available paths.

REQ-8 (Display Trips): The user shall be able to view a list of their upcoming and past trips with details.

REQ-9 (Cancel Trip): The user shall be able to cancel a booked trip.
REQ-10 (View Traffic): The user shall be able to view a map displaying real-time traffic conditions, if possible

4.3 Feature 3: Admin Module

4.3.1 Description and Priority:

High. This module provides system monitoring and management capabilities to the admin.

4.3.2 Functional Requirements:

REQ-11 (Map View): The admin shall be able to see a real-time map displaying the locations of all users currently on their booked trips.
REQ-12 (View Timestamps): The admin shall be able to view the allocated start and end times for each user's booked appointment.
REQ-13 (Cancel Appointment): The admin shall have the authority to cancel any user's booked trip

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The pathfinding algorithm should return a result within 5 seconds of the user's request.
- The web interface should load in under 3 seconds on a standard broadband connection.
- The system should support at least 100 concurrent users without significant degradation in performance.

5.2 Safety Requirements

- All user data, especially personal information and location data, must be encrypted both in transit and at rest.
- The system must implement role-based access control (RBAC) to ensure users can only access functionalities permitted for their role.
- The system should be protected against common web vulnerabilities such as SQL injection and Cross-Site Scripting (XSS).

5.3 Software Quality Attributes

Reliability: The system should have an uptime of 99.5% and ensure the accuracy of path calculations.
Usability: The user interface should be intuitive and easy to navigate for both technical and non-technical users.

Maintainability: The code should be well-documented, modular, and easy to modify or extend in the future.

5.4 Business Rules

A specific road segment at a given time slot can only be allocated to one user to prevent overlaps. Admins are not permitted to book trips for themselves through the admin dashboard.

6. Other Requirements

(This section can be used for any requirements not covered above, such as legal or internationalization needs. Currently, there are none.)

Appendix A: Glossary

- Dijkstra's Algorithm: An algorithm for finding the shortest paths between nodes in a graph.
- GUI: Graphical User Interface.
- RBAC: Role-Based Access Control.
- SRS: Software Requirements Specification.

Appendix B: Analysis Models

(This section would optionally include diagrams like Use Case Diagrams, Data Flow Diagrams, or Class Diagrams to further illustrate the system's design.)

Appendix C: To Be Determined List

- *Specific details of the third-party map data provider.*
- *The exact refresh rate for real-time traffic data visualization.*
- *The specific criteria for suggesting "alternative paths."*