Bus Reservation and Locator System

Software Requirement Specification

BY

LAKESH KUMAR, CMS. ID: 053-24-0036

MUNESH KUMAR, CMS. ID: 053-24-0037

YOUGESH KUMAR, CMS. ID: 053-24-0026

Submitted To: DR. KHURSHED ALI

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF SCIENCE AND INFORMATION TECHNOLOGY

SUKKUR IBA UNIVERSITY

DECEMBER 2025

1. *Introduction*

1.1. Purpose of Document

The purpose of this Software Requirements Specification (SRS) document is to describe the functional and non-functional requirements of the Bus Reservation and Locator System. It serves as a reference for developers, testers, project managers, and other stakeholders.

1.2. Intended Audience

Project Owner: Primary stakeholder, typically representing the bus company, who defines the business goals and has final approval authority.
Project Managers: To understand the project scope, plan resources, and track progress.
Software Developers & Engineers: The backend, frontend, and mobile teams who will use these requirements to design and build the system.
UI/UX Designers: To understand the required user interactions, features, and workflows to create an effective user interface.

1.3. Document Convention, abbreviations, and acronyms

Document Formatting Standards

Font Family: *Times New Roman*
Font Size (Body Text): *10 pt*
Font Size (Main Headings – e.g., 1, 2, 3): *16 pt, Bold*
Font Size (Sub-Headings – e.g., 1.1, 1.1.1): *12 pt, Bold*
Line Spacing: *1.0*
Paragraph Alignment: *Justified*
Table Text: *Arial 10–12 pt*
Page Margins: *Normal (1 inch on all sides)*
Document Format: *IEEE-style SRS/SDS structure*

Acronyms Used:

BRLS: Bus Reservation and Locator System
SRS: Software Requirements Specification
SDS: Software Design Specification
GPS: Global Positioning System
API: Application Programming Interface
DB: Database
UI: User Interface
UX: User Experience

2. Overall System Description

2.1. Project Background

- In many cities, bus services are still managed manually.
  Passengers face problems like unclear schedules, no seat information, and no updates on delays.
  This causes long waiting times and inconvenience for travellers.
  Bus companies also struggle with managing routes and seat availability efficiently.
- To solve these problems, a digital **Bus Reservation and Locator System** is needed.
  Passengers can book seats online from their phones or computers.
  They can see live bus locations on a map and get instant booking confirmation.
  Bus companies can manage routes, schedules, and seat availability easily.
  GPS tracking ensures passengers know real-time bus locations.

2.2. Project Scope

- The **Bus Reservation and Locator System** aim to make bus booking easier and more reliable for both passengers and service providers. It will be available as a **web and Android-based application**. Passengers can use it to **book seats online**, **check bus timings**, **view available seats**, and **track the bus location in real time**.
- Bus companies can use the system to **manage routes, schedules, and seat availability** efficiently. The system will also show the **expected arrival time**, **bus speed**, and **route details** through GPS and Google Maps integration.

2.3. Not in Scope

The current version of the **Bus Reservation and Locator System** will focus only on the core booking and tracking features. The following functionalities are **not included in this phase** of the project:

- Online payment gateway integration (credit/debit cards, mobile wallets, etc.)
- Automated refund or rescheduling system
- Driver performance analytics and report generation
- Integration with third-party transport services

2.4. Project Objectives

The main objective of the Bus Reservation and Locator System is to improve the current bus management process by using modern technology. The system aims to make booking, tracking, and managing bus services easier for both passengers and operators.

The main objectives of the Bus Reservation and Locator System are:

- To provide passengers with an easy way to book bus seats online.
- To give real-time information about bus locations and delays.
- To improve communication between passengers and bus service providers.
- To help bus companies manage routes, schedules, and seat availability efficiently.
- To reduce waiting times and make public transport more convenient and reliable.
- To increase transparency and customer satisfaction in bus services.

2.5. Stakeholders

- Bus Passengers/Users: Book tickets, check schedules, routes, and bus status.
- Bus Service Providers/Operators: Manage schedules, routes, seat availability, and monitor buses.
- Project Development Team: Frontend and backend developers, database administrators, QA testers, and project managers responsible for building and maintaining the system.
- Technical Service Providers: GPS/Mapping API providers for bus tracking.
- Indirect Stakeholders: Government transport authorities and maintenance staff benefiting from operational data.

2.6. Operating Environment

- Hardware Platform: Desktop/laptop computers, Android mobile devices, and servers to host the web application.
- Operating System: Windows, macOS, Linux for web access; Android for mobile application.
- Network Environment: Internet connectivity required for real-time bus tracking, seat booking, and data synchronization between users and servers.
- Software Components:
- Frontend: React JS, CSS
- Backend: Node.js (Express.js), MySQL (SQL Database)
- APIs/Services: Google Maps API for GPS tracking, AJAX for improved performance and responsiveness.
- Coexistence Requirements: Must operate alongside standard web browsers, mobile applications, and GPS services without conflicts.

2.7. System Constraints

- Software Constraints: Must use React.js, Node.js, MySQL, and Android (Java/Kotlin); dependent on Google Maps API for GPS functionality.
- Hardware Constraints: Limited to desktop/laptop computers, Android devices, and available server specifications.
- Cultural Constraints: System interface in English (can be extended to Urdu for local users).
- Legal Constraints: Compliance with data privacy regulations and local transport laws.
- Environmental Constraints: System must operate in areas with variable internet connectivity; mobile users may access it in moving buses.
- User Constraints: Designed for general passengers; interface should be simple, intuitive, and mobile-friendly.
- Off-the-Shelf Components: Limitations of third-party APIs (Google Maps) and libraries may affect performance or availability.

2.8. Assumptions & Dependencies

- **Assumptions:**
    - Users have basic familiarity with mobile apps and web applications.
    - Internet connectivity is available for real-time tracking and booking.
    - Buses are equipped with GPS devices or compatible sensors for location tracking.
    - Users and bus operators will provide accurate data for booking and scheduling.
- **Dependencies:**
    - Google Maps API for real-time bus location and route information.
    - Android and web platforms for application deployment.
    - Database and server infrastructure for storing and managing data.
    - Third-party libraries and frameworks (React JS, Node JS) for system functionality.

3. External Interface Requirements

The Bus Reservation and Locator System interact with various external components to ensure smooth operation and real-time functionality.

3.1. Hardware Interfaces

- Desktop/Laptop Computers: Users access the web application via standard input/output devices (keyboard, mouse, display).
- Android Mobile Devices: Mobile app interacts with touchscreen input, GPS sensors, and motion sensors for real-time tracking.
- Servers: Communicate with client devices over the internet; process requests, store data, and manage bookings.
- GPS Devices: Provide real-time location data to the system for bus tracking.

3.2. Software Interfaces

- Database: MySQL stores users, operators, buses, routes, bus stops, trip schedules, seats, bookings, bus locations, and notifications; backend interacts via Node.js.
- Frontend Frameworks: React JS and CSS handle UI rendering and user interaction.
- Backend Framework: Node JS manages business logic, APIs, and server-client communication.
- Google Maps API: Supplies map, route, and GPS data; interacts with frontend and backend to display bus location.
- Operating Systems: Windows, macOS, Linux (web access) and Android (mobile) host the applications.

3.3. Communications Interfaces

- Network Protocols: HTTP/HTTPS for web requests; RESTful APIs for server-client communication.
- Data Exchange: JSON format for sending/receiving data between frontend, backend, and APIs.
- Security/Encryption: HTTPS ensures encrypted communication; secure authentication for users and bus operators.
- Synchronization: Real-time updates for bus location, seat availability, and booking confirmations.

4. Functional Requirements

4.1. Functional Hierarchy

i. User Module

a) Sign Up / Login: User registration and authentication.

b) Search Buses: View available buses by route, date, and time.

c) Seat Selection: View seat layout and choose available seats.

d) Seat Booking: Select and book available seats.

e) Booking History: View past and upcoming bookings.

f) Bus Tracking: Real-time location and status of booked bus.

g) Notifications: Alerts for bus arrival, delays, or cancellations.

ii. Bus Operator Module

a) Bus Registration: Add new buses and their details.

b) Route Management: Define bus routes and stops.

c) Trip Schedule Management: Define and manage trip timings and availability.

d) Booking Management: View and manage seat bookings.

e) Bus Status Update: Indicate bus in motion, stopped, or delayed.

iii. System Administration / Backend Module

a) User & Operator Management: Manage accounts and permissions.

b) Database Management: Store and manage data (users, buses, bookings).

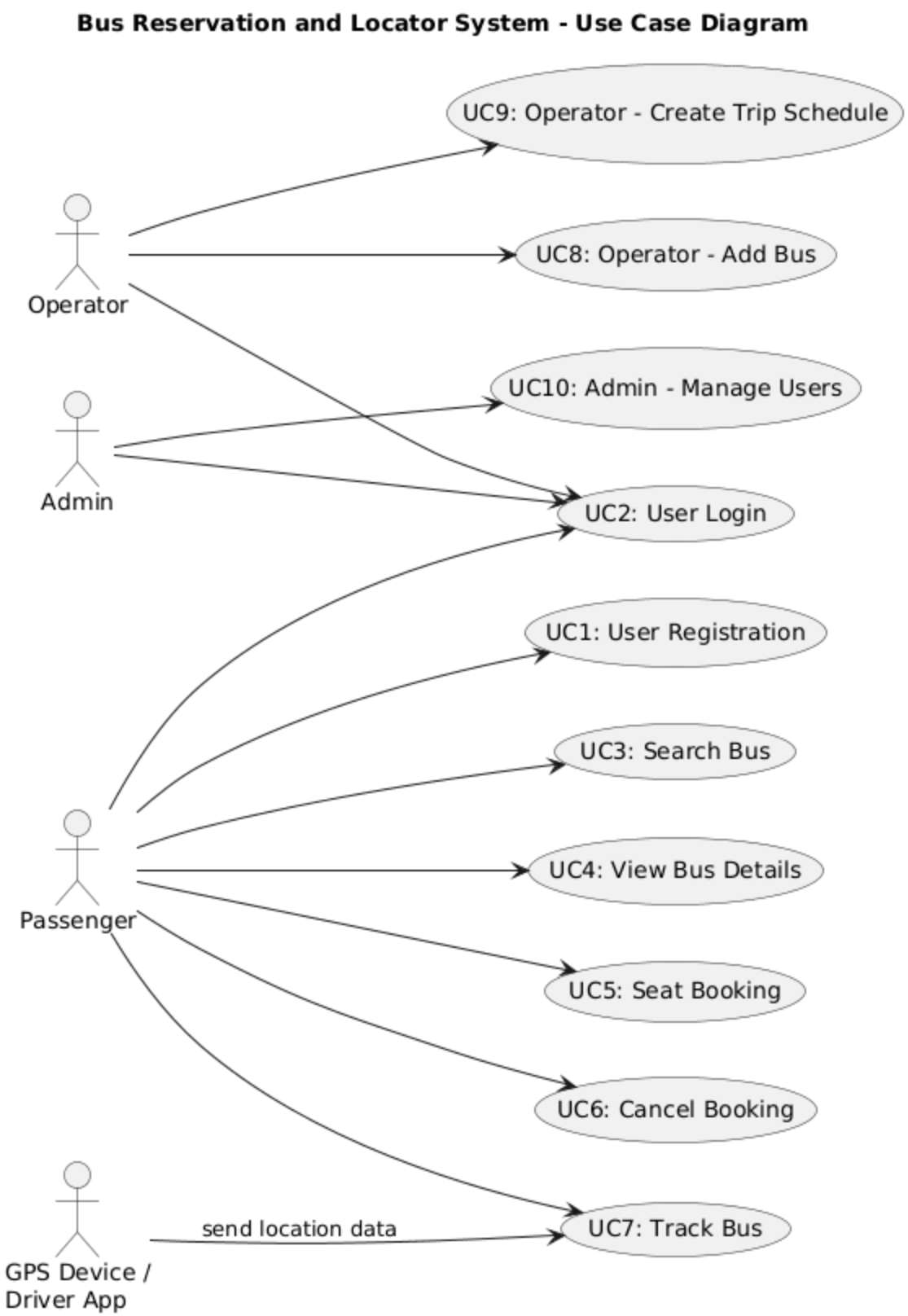
c) API Integration: Connect with Google Maps API for GPS tracking.

d) Reporting & Logs: Monitor system activity and errors.

4.2. Use Cases

- The Bus Reservation and Locator System includes a set of core use cases that represent the main interactions between the users (Passenger, Operator, Admin) and the system.
  These use cases describe how users perform key actions such as registering, logging in, searching for buses, viewing trip details, booking seats, cancelling bookings, and tracking buses in real time.
- A complete Use Case Diagram illustrates these interactions and the relationship between actors and system functionalities.

UC-01 – User Registration	**UC-06 – Cancel Booking**
UC-02 – User Login	**UC-07 – Track Bus**
UC-03 – Search Bus	**UC-08 – Operator – Add Bus**
UC-04 – View Bus Details	**UC-09 – Operator – Create Trip Schedule**
UC-05 – Seat Booking	**UC-10 – Admin – Manage Users**


Bus Reservation and Locator System - Use Case Diagram
UC9: Operator - Create Trip Schedule
UC8: Operator - Add Bus
Operator
UC10: Admin - Manage Users
Admin
UC2: User Login
UC1: User Registration
UC3: Search Bus
Passenger
UC4: View Bus Details
UC5: Seat Booking
UC6: Cancel Booking
send location data
UC7: Track Bus
GPS Device /
Driver App

4.2.1. USER REGISTRATION

This use case describes how a new Passenger creates an account in the system by providing their personal details and login credentials.

UC-01: User Registration		
Use case Id:	UC-01	
Actors:	Passenger (New user)	
Feature:	Account Creation / User Registration	
Pre-condition:	• User is not already registered. • Registration page must be available.	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	User opens "Sign Up / Register" page.	System displays registration form.
2.	User enters Name, Email, Phone, Password.	System validates input format.
3.	User clicks "Register".	System checks if email/phone already exist.
4.		If unique → system creates new user record.
		System shows success message and redirects to Login.
Alternate Scenarios:		
1a: User leaves fields empty → System highlights missing fields. **2a: Email already exists → System shows "Email already registered".**		
Post Conditions		
Step#	**Description**	
	A new Passenger account is created in the database.	
	User can log in using email and password**.**	
Use Case Cross referenced	UC-02 (User Login)	

4.2.2. USER LOGIN

This use case explains how a Passenger, Operator, or Admin logs into the system using valid credentials to access authorized features.

UC-02: User Login		
Use case Id:	UC-02	
Actors:	Passenger, Operator, Admin	
Feature:	User Authentication	
Pre-condition:	• User must already be registered. • Account must be active.	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	User opens Login page.	System displays login form.
2.	User enters Email & Password.	System validates input format.
3.	User clicks "Login".	System checks credentials.
4.		If valid → system creates session/JWT.
		System redirects user based on role.
Alternate Scenarios:		
1a: Invalid credentials → System shows "Incorrect email or password". **2a: Account blocked → System shows "Account disabled".**		
Post Conditions		
Step#	**Description**	
	User session is created.	
	User is logged in and redirected to home/dashboard.	
Use Case Cross referenced	UC-01 (User Registration), UC-04 (View Bus Details), UC-06 (Cancel Booking), UC-07 (Track Bus)	

4.2.3. CANCEL BOOKING

This use case explains how a Passenger cancels an already booked seat for an upcoming trip and updates the booking status.

UC-06: Cancel Booking		
Use case Id:	UC-06	
Actors:	Passenger	
Feature:	Booking Management	
Pre-condition:	• Passenger must be logged in (UC-02). • Booking must be in status "Booked".	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	User opens "My Bookings".	System shows booking list.
2.	User selects a booked trip.	System displays booking details.
3.	User clicks "Cancel Booking".	System displays confirmation popup.
4.	User confirms cancellation.	System checks cancellation rules.
5.		System updates booking to "Cancelled".
6.		System sends cancellation confirmation.
Alternate Scenarios:		
1a: Booking already cancelled → System disables cancel option. **2a: Cancellation not allowed (too close to departure) → Show error.**		
Post Conditions		
Step#	**Description**	
	Booking status updated to "Cancelled".	
	Seat becomes available for the same trip.	
Use Case Cross referenced	UC-02 (User Login), UC-07 (Track Bus)	

4.2.4. TRACK BUS

This use case describes how a Passenger views the live GPS location of the bus assigned to their booked trip, displayed on a real-time map.

UC-07: Track Bus		
Use case Id:	UC-07	
Actors:	Passenger	
Feature:	Real-Time Bus Tracking	
Pre-condition:	• User must be logged in (UC-02). • Bus must be sending GPS updates. • User must have an active booking.	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	User selects a booking and clicks "Track".	System requests latest GPS location.
2.		System fetches location from database.
3.		System shows bus on map.
4.		System refreshes location every 10–30 sec
Alternate Scenarios:		
1a: No GPS data available → System shows "Location unavailable". **2a: Network error → System shows retry message.**		
Post Conditions		
Step#	**Description**	
	Latest bus location displayed.	
	Passenger stays updated on bus movement.	
Use Case Cross referenced	UC-06 (Cancel Booking), UC-02 (Login)	

4.2.5. VIEW BUS DETAILS

This use case describes how a Passenger views detailed information about a selected bus trip, including schedule, route, fare, and seat layout.

UC-04: View Bus Details		
Use case Id:	UC-04	
Actors:	Passenger	
Feature:	View Trip Details & Seat Layout	
Pre-condition:	• User must be logged in (UC-02). • Bus search results must already be displayed.	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	User selects a trip and clicks "View".	System requests trip & seat details.
2.		System loads bus, route, fare info.
3.		System displays seat layout (available/booked).
4.	User clicks "Proceed to Book".	System forwards seat/trip to booking flow.
Alternate Scenarios:		
1a: Unable to load data → System shows "Error loading details". **2a: No seats available → Booking button disabled.**		
Post Conditions		
Step#	**Description**	
	User views necessary trip and seat information.	
	User proceeds toward booking (UC-03).	
Use Case Cross referenced	UC-02 (User Login), UC-03 (Seat Booking)	

5. *Non-functional Requirements*

5.1. Performance Requirements

- The system shall respond to all user actions within **2–3 seconds** under normal load.
- The system shall support **at least 500 concurrent users** without performance degradation.
- Real-time bus tracking data shall refresh every **10–15 seconds**.
- The database shall handle **high-volume booking operations** without delays or data loss.
- System uptime shall be **99% or higher** to ensure high availability and reliability.

5.2. Safety Requirements

- The system shall prevent **accidental or duplicate seat bookings**.
- The system shall ensure **accurate display** of bus status (in motion, stopped, delayed).
- System design shall comply with **local transport safety regulations**.
- Proper fail-safes shall be implemented to prevent **data loss during system crashes**, including automatic backups and rollback mechanisms.

5.3. Security Requirements

- Users shall authenticate using username/password; OTP verification may be used for enhanced security.
- Role-based access control (RBAC) shall ensure different permissions for **Passengers, Operators, and Admins**.
- All data transmission shall be encrypted using **HTTPS/TLS**.
- User data, booking data, and bus location data shall comply with **data privacy regulations**.
- The system shall be protected against common cyber threats such as **SQL injection, brute-force attacks, and unauthorized access**.

5.4. User Documentation

- A comprehensive **User Manual** shall be provided for passengers, operators, and administrators.
- **Online Help** and contextual tooltips shall guide users within the application.
- **Tutorials** shall cover essential tasks such as account creation, booking, and bus tracking.
- A **FAQs and Troubleshooting** section shall be available to assist users with common issues.

6. References

[1] IEEE, IEEE Std 830-1998 – IEEE Recommended Practice for Software Requirements Specifications, IEEE, 1998. Available at: https://standards.ieee.org

[2] React Team, React Documentation, Meta Platforms (Accessed 2025). Available at: https://react.dev

[3] Node.js Foundation, Node.js Documentation, OpenJS Foundation (Accessed 2025). Available at: https://nodejs.org/en/docs

[4] Express.js Foundation, Express.js Web Framework Documentation (Accessed 2025). Available at: https://expressjs.com

[5] Oracle Corporation, MySQL 8.0 Reference Manual (Accessed 2025). Available at: https://dev.mysql.com/doc

[6] Google Developers, Google Maps Platform Documentation (Accessed 2025). Available at: https://developers.google.com/maps

[7] Android Developers, Android Developer Documentation (Accessed 2025). Available at: https://developer.android.com

[8] Auth0 Engineering, JSON Web Token (JWT) Introduction and Usage Guide (Accessed 2025). Available at: https://jwt.io

[9] OWASP Foundation, OWASP Cheat Sheet Series – Authentication, Input Validation, and Secure Communication (Accessed 2025). Available at: https://cheatsheetseries.owasp.org

[10] PlantUML Team, PlantUML Documentation (Used for UML diagrams) (Accessed 2025). Available at: https://plantuml.com

7. Appendices

Appendix A – Glossary of Terms

This appendix provides definitions of common terms used throughout the document.

Term	Definition
Passenger	End-user who searches buses, selects seats, books tickets, and tracks buses.
Operator	Bus company staff who manage buses, routes, and trip schedules.
Admin	System administrator who manages users, operators, and system settings.
Trip Schedule	A scheduled journey for a specific bus on a specific date and time.
GPS	Global Positioning System used to track real-time bus location.
API	Application Programming Interface used to communicate between frontend and backend.
JWT	JSON Web Token used for secure user authentication.
UI	User Interface screens used by Passengers, Operators, and Admins.
Database	MySQL database storing persistent system data.

Appendix B – Assumptions

The following assumptions were made during the development of this SRS:

- All users have access to stable internet connectivity.
- GPS devices or driver mobile apps always send accurate, timely location data.
- Bus Operators maintain correct schedule information.
- Server hosting and cloud services remain active during system usage.
- Passengers use modern browsers or updated Android devices.

Appendix C – Constraints

This system is developed under the following constraints:

- Real-time tracking accuracy depends on external GPS hardware and Google Maps API.
- System performance depends on server capacity and database optimization.
- External APIs (Google Maps) may have usage limits or billing requirements.
- System cannot control physical bus delays, network failures, or GPS signal loss.

Appendix D – Tools Used

- **StarUML / PlantUML** for UML diagrams
- **Draw.io / Figma** for UI sketches
- **VS Code / Android Studio** for development
- **GitHub** for version control