Practical 4

Software Requirement Specification (SRS) Development for the Selected System

Aim: Perform a requirement analysis and develop a Software Requirement Specification.

1. Functionality: Describe what the software is supposed to do.
2. External Interfaces: Explain how the software interacts with people, the system's hardware, other hardware, and other software.
3. Performance: Outline the expected speed, availability, response time, recovery time, and other performance-related characteristics of the software functions.
4. Attributes: Define considerations related to portability, correctness, maintainability, security, and other relevant attributes.
5. Design Constraints: Specify any design constraints imposed on the implementation, such as required standards, implementation languages, database integrity policies.

4. Software Requirement Specification (SRS)

A Software Requirement Specification (SRS) is a structured document that defines the functionalities, constraints, and requirements of a software system. It serves as a foundation for development, testing, and stakeholder communication.

4.1 Overview

A **Bus Management System (BMS)** is a digital solution designed to streamline and automate bus transportation operations. It replaces traditional manual record-keeping with an efficient, organized system that enables seamless management of buses, passenger data, ticket bookings, route planning, and trip scheduling functionalities. The BMS aims to enhance the user experience by providing faster access to services, improving accountability, and reducing human errors.
This Software Requirement Specification (SRS) document follows the IEEE SRS standard and outlines the functional and non-functional requirements of the system, as well as external interfaces, constraints, and dependencies.

4.2 Purpose

The primary purpose of this **Bus Management System (BMS)** is to provide an automated platform for managing bus transportation services, bus availability, and passenger transactions. The system enables users to search for routes, book tickets, and manage travel schedules efficiently, while allowing administrators to monitor trip activities, track bus inventory, and generate operational reports. It also supports notifications for delays, maintenance schedules, and policy enforcement.

4.3 Document Conventions

This document follows structured formatting using numbered sections, bullet points, and industry-standard terminology for clarity and ease of reference. This document is intended mainly for users like **Bus Management Administrators** who manage passenger accounts, bus records, route schedules, and operational policies, as well as others related to the project such as **Developers** who understand and implement system requirements, **Testers** who validate system functionality and performance, **Stakeholders** who review system capabilities and constraints, etc. This document follows **IEEE 830-1998: Recommended Practice for Software Requirements Specifications**.

4.4 Overall Description

This section provides an overview of the **Bus Management System's** purpose, core functions, and operational constraints.

Product Perspective

The **Bus Management System (BMS)** is a standalone software system designed to digitize bus transportation operations. It integrates with existing transport infrastructure, replacing traditional record-keeping methods with a centralized database.

Product Functions

The system will provide:

- **User Account Management:** Role-based access control for passengers, bus staff, and system administrators.
- **Bus Management:** Add, update, delete, and search bus records including route, capacity, and availability.
- **Trip & Ticket Management:** Track ticket bookings, travel schedules, and passenger details.
- **Notifications & Alerts:** Alerts via email/SMS for trip reminders, delays, or schedule changes.
- **Report Generation:** Statistical insights on bus utilization, passenger traffic, and route performance.

Users of the BMS include:

- **Bus Staff:** Administrators who manage bus records, passenger information, and trip schedules.
- **Passengers:** Registered users who search for routes, book tickets, and view schedules.
- **System Administrators:** IT personnel responsible for system setup, configuration, and maintenance.

General Constraints

The system must comply with data privacy regulations. It should support at least **10,000 bus records**. The system must be operational **24/7 with minimal downtime** to ensure uninterrupted access to bus services and scheduling.

4.5 Specific Requirements

The **Specific Requirements** section of the SRS outlines detailed functional and non-functional needs of the **Bus Management System (BMS)**. Functional requirements define what the system should do, describing specific actions and behaviors. These requirements directly contribute to the core functionalities of the software. Non-functional requirements define how the system performs its functions, focusing on system attributes like **security, performance, usability**, and **scalability**.

The **External Interfaces** section of an SRS describes how the **Bus Management System (BMS)** interacts with users, hardware, and other software. These interfaces ensure seamless operation and integration within the existing transportation environment.

4.5.1 Functional Requirements

Functional requirements describe what the system should do:

1. **User Authentication & Authorization:** Secure login and role-based access for passengers, bus staff, and administrators.
2. **Bus Management:** CRUD (Create, Read, Update, Delete) operations for bus records including route, schedule, and capacity.
3. **Trip & Ticket Management:** Tracking ticket bookings, travel dates, passenger details, and trip completion status.
4. **Delay & Maintenance Alerts:** Automatic notifications and status updates for delayed trips or buses under maintenance.
5. **Search & Filter:** Advanced search options for routes, schedules, and available seats based on location, date, and time.
6. **Reports & Analytics:** Generate reports on system usage, passenger volume, route performance, and bus utilization.
7. **Notifications:** Email/SMS alerts for ticket confirmations, trip reminders, schedule changes, and important announcements.

4.5.2 External Interfaces

External interfaces define how the system interacts with users, hardware, and other software components:

1. **User Interface (UI):** A web-based interface that provides an intuitive dashboard for users to interact with the system, including booking tickets, viewing schedules, and managing routes.
2. **Hardware Interface:** The system must support barcode or QR code scanners for ticket validation, printers for issuing tickets and reports, and mobile devices for on-the-go access by staff and passengers.
3. **Software Interfaces:** The system integrates with SQL databases for data storage and third-party payment gateways for secure fare transactions.
4. **Communication Interfaces:** The system supports email, SMS, and push notifications to alert users about booking confirmations, trip reminders, delays, and system updates.

4.5.3 Non-Functional Requirements

Non-functional requirements describe system performance, security, and usability expectations:

1. **Performance:** The system must handle a large number of concurrent users (e.g., passengers and staff) with response times under 2 seconds.
2. **Security:** Implements encryption, multi-level authentication, and role-based access control to prevent unauthorized access to passenger data and system functions.
3. **Usability:** The user interface should be intuitive, user-friendly, and accessible, following **WCAG** (Web Content Accessibility Guidelines).
4. **Scalability:** The system must support growth in users, bus records, and transactions without performance degradation.
5. **Availability:** Ensures high availability and uptime, with automated backups and disaster recovery mechanisms for data protection.

4.5.4 Design Constraints

Design constraints are limitations that affect the implementation and architecture of the Bus Management System (BMS). These constraints ensure that the system is developed within predefined technical, security, and operational guidelines. Design constraints define system limitations:

1. **Database Management:** The system should support both **SQL** and **NoSQL** databases depending on scalability and performance needs for storing bus schedules, passenger data, and ticketing information.
2. **Technology Stack:** The system should be built using **Node.js** for the backend, **React.js** for the frontend, and **MongoDB** as the primary database.
3. **Security Compliance:** The system must encrypt user and operational data, comply with **GDPR** or relevant regional standards, and implement **multi-factor authentication (MFA)**.
4. **Performance Optimization:** The system should implement **load balancing** and **caching mechanisms** to ensure high availability and smooth performance during peak usage.

4.5.5 System Attributes

1. **Portability:** The system must be accessible on **Windows, macOS, Linux**, and **mobile devices** to accommodate different user environments.
2. **Maintainability:** Designed with a **modular architecture** to allow easy updates, bug fixes, and enhancements.
3. **Reliability:** Ensures consistent and stable operation, with built-in **failover mechanisms** to handle unexpected system failures.
4. **Scalability:** The system must efficiently handle an increasing number of **users, buses, routes**, and **ticket transactions** without performance degradation.
5. **Interoperability:** Capable of integrating with **third-party systems** such as payment gateways, GPS tracking, or government transit APIs for extended functionality.
6. **Extensibility:** The system should support **future enhancements** and new features with minimal changes to the existing codebase.

4.5.6 Implementation Details and Hardware Requirements

Implementation Details:

The Bus Management System (BMS) will be implemented using Node.js for backend development, React.js for the frontend, and MongoDB as the database. It will use RESTful APIs for communication, integrate third-party APIs for payments, notifications, and route tracking, and will be deployed on cloud platforms like AWS or Google Cloud to ensure scalability, reliability, and high availability.

Hardware Requirements:

- Server: Quad-core CPU (2.5 GHz), 16 GB RAM, 500 GB SSD, Linux OS, 100 Mbps internet.
- Client: Dual-core CPU (1.8 GHz), 4 GB RAM, 10 GB free space, Windows/macOS/Linux OS, modern browsers.
- Additional: Barcode/QR scanners, printers, and optional mobile devices for remote access