Software Requirements Specification (SRS)

Project Title: Hospital Management System (HMS)

Prepared by: Sanjaya Samudra – PR24108458
Program: ICM108
Institution: Institute of Computer Engineering Technology

Date: 20/10/2024

1. Introduction

1.1. Purpose

- The Hospital Management System (HMS) aims to provide a centralized and efficient management solution for hospitals. It manages patient records, doctor schedules, appointments, prescriptions, billing, and other hospital activities in an integrated system. This document outlines the system's functional and non-functional requirements, system architecture, and overall design.

1.2. Scope

The HMS will manage all hospital operations, including:

- Patient registration, management, and history tracking.
- Scheduling and management appointments.
- Prescription management and history recording.

- Billing and report generation.
- User roles and permissions for doctors, admin staff, and nurses.
- Real-time search and filtering for patients, appointments, and reports.

The system will be accessible through a web-based user interface, designed for hospital staff such as doctors, nurses, and administrative personnel.

1.3. Objectives

- Automate the management of patient records and hospital operations.
- Improve efficiency and reduce manual effort in hospital administrative tasks.
- Provide real-time data access for hospital staff.
- Ensure secure access and privacy for patient data.

1.4. Definitions, Acronyms, and Abbreviations

- **HMS:** Hospital Management System.
- **ER Diagram:** Entity-Relationship Diagram.
- **UI:** User Interface.

2. System Overview

- The HMS is designed as an event-driven microservice architecture with the following modules:

1. **Patient Management Module**
2. **Appointment Scheduling Module**
3. **Prescription and Medication Management Module**
4. **Billing and Reporting Module**
5. **User and Access Control Module**

- Each module operates independently but shares data through secure APIs. The system will be built using a Single Page Application (SPA) framework for responsive UI design.

3. Functional Requirements

3.1 User Roles

1. **Admin:** Manages users, system configurations, and hospital settings.
2. **Doctor:** Manages patient appointments, prescriptions, and medical reports.
3. **Nurse:** Assists in patient care, manages appointments, and maintains patient records.

3.2 Key Features

1. **Patient Management**
    - Register, update, and delete patient records.
    - Search for patients using various parameters (e.g., name, ID, status).

2. **Appointment Management**
    - Schedule, update, and cancel appointments.
    - View appointment details by doctor, date, or patient.
3. **Prescription Management**
    - Create and manage prescriptions linked to patient records.
    - Track prescription history.
4. **Billing and Reports**
    - Generate and view bills by patient.
    - Generate medical reports and other hospital records.
5. **User Authentication and Authorization**
    - Secure login for different roles (admin, doctor, nurse).
    - Role-based access to system modules and data.

4. Non-Functional Requirements

1) **Performance:** The system should support up to 100 concurrent users and respond to user interactions within 2 seconds.
2) **Security:** All patient data must be encrypted and stored securely. User authentication will include role-based access controls.
3) **Scalability:** The system architecture should support scaling to accommodate larger datasets and additional hospital branches.
4) **Usability:** The system UI will be intuitive, with minimal training required for hospital staff.
5) **Maintainability:** Code must be modular and well-documented to support future updates and maintenance.

5. Diagram and Use Case Diagram

5.1 ER Diagram

- The ER Diagram defines the entities and their relationships:
- **Patient**: Stores patient data such as ID, name, contact, and medical information.
- **Appointment**: Links patients with doctors, storing appointment details and status.
- **Procedure**: Records of medical procedures performed, linking to patient records.
- **Medical Report**: Details lab tests and other medical records associated with patients.

5.2 Use Case Diagram

The use case diagram illustrates the main interactions between users (admin, doctors, nurses) and the system. Key use cases include:

- Add, update, and search patients.
- Manage appointments.
- Issue prescriptions and generate bills.
- Access reports and data specific to user roles.

6. System Design

6.1 Architecture

- The HMS follows an event-driven microservice architecture. Each module (patient management, appointments, prescriptions, etc.) is an independent microservice communicating through APIs. The front-end SPA framework ensures responsive and interactive UI components.

6.2 Database Design

- The system uses a relational database with entities such as:
    - Patients
    - Appointments
    - Prescriptions
    - Medical Procedures
    - Reports

Refer to the ER Diagram for detailed relationships and entity attributes.

6.3 API Endpoints

- The system uses RESTful APIs for interaction between the front-end and back-end. Key API endpoints include:
  - **/api/patients**: Manage patient data.
  - **/api/appointments**: Handle appointment scheduling and updates.
  - **/api/prescriptions**: Create and retrieve prescriptions.
  - **/api/reports**: Generate and access medical reports.

7. Implementation and Testing

7.1 Development Tools

- **Programming Language**: JavaScript/TypeScript for the front-end; Java/Python for the back end.
- **Frameworks**: React/Vue for SPA; Spring Boot/Flask for microservices.
- **Database**: MySQL/PostgreSQL for data storage.

7.2 Testing

- **Unit Testing**: Tests for individual modules (patient registration, appointment management, etc.).
- **Integration Testing**: Tests for data flow between modules (e.g., prescriptions linked to patients).
- **User Acceptance Testing (UAT)**: Testing by hospital staff to validate system functionality.