Project Title: Hospital Management System (HMS)

1. Overview

The **Hospital Management System (HMS)** is a web-based application that streamlines administrative and patient care operations in a hospital. Designed for scalability and ease of use, the system is modular and follows the **MVC architecture**, making it compatible with both **Java (Spring MVC)** and **.NET (ASP.NET Core MVC)** frameworks.

The core modules include:

1. **Patient Management** – Manages patient records and appointments.
2. **Doctor Management** – Handles doctor profiles and availability.
3. **Appointment Scheduling** – Enables patients to book appointments with doctors.
4. **Billing and Payments** – Manages patient billing and payments.
5. **User Management** – Handles authentication, authorization, and user roles.

2. Assumptions

1. The application will be deployed locally during development using a relational database (e.g., MySQL or SQL Server).
2. Role-based authentication will secure sensitive information.
3. ORM frameworks (Hibernate for Java or Entity Framework for .NET) will handle database interactions.
4. No containerization will be used for local deployment.

3. Module-Level Design

3.1 Patient Management Module

Purpose: Manages patient details and their medical history.

- **Controller**:
    - PatientController
        - addPatient(patientData)
        - updatePatient(patientId, patientData)
        - getPatientDetails(patientId)
        - deletePatient(patientId)

- **Service**:
    - PatientService
        - Validate patient data.
        - Interact with the database for CRUD operations.
- **Model**:
    - **Entity**: Patient
        - Attributes:
            - patientId (PK)
            - name (VARCHAR)
            - dateOfBirth (DATE)
            - gender (VARCHAR)
            - contactNumber (VARCHAR)
            - address (VARCHAR)
            - medicalHistory (TEXT)

3.2 Doctor Management Module

Purpose: Manages doctor profiles and their availability schedules.

- **Controller**:
    - DoctorController
        - addDoctor(doctorData)
        - updateDoctor(doctorId, doctorData)
        - getDoctorDetails(doctorId)
- **Service**:
    - DoctorService
        - Manage doctor profiles and availability.
- **Model**:
    - **Entity**: Doctor
        - Attributes:
            - doctorId (PK)
            - name (VARCHAR)

        - specialization (VARCHAR)
        - contactNumber (VARCHAR)
        - availabilitySchedule (TEXT)

3.3 Appointment Scheduling Module

Purpose: Facilitates scheduling of appointments between patients and doctors.

- **Controller**:
    - AppointmentController
        - scheduleAppointment(appointmentData)
        - getAppointmentDetails(appointmentId)
        - cancelAppointment(appointmentId)
- **Service**:
    - AppointmentService
        - Validate and schedule appointments.
- **Model**:
    - **Entity**: Appointment
        - Attributes:
            - appointmentId (PK)
            - patientId (FK)
            - doctorId (FK)
            - appointmentDate (DATE)
            - timeSlot (VARCHAR)
            - status (ENUM: CONFIRMED, CANCELLED)

3.4 Billing and Payments Module

Purpose: Handles billing and payment processing for patients.

- **Controller**:
    - BillingController
        - generateBill(billData)
        - getBillDetails(billId)
        - processPayment(billId, paymentData)

- **Service**:
    - BillingService
        - Generate bills and validate payments.
- **Model**:
    - **Entity**: Bill
        - Attributes:
            - billId (PK)
            - patientId (FK)
            - totalAmount (DECIMAL)
            - paymentStatus (ENUM: PAID, UNPAID)
            - billDate (DATE)

3.5 User Management Module

Purpose: Manages user authentication and roles for secure access.

- **Controller**:
    - UserController
        - registerUser(userData)
        - loginUser(username, password)
        - getUserProfile(userId)
- **Service**:
    - UserService
        - Manage user credentials and roles.
- **Model**:
    - **Entity**: User
        - Attributes:
            - userId (PK)
            - username (VARCHAR)
            - password (VARCHAR, Encrypted)
            - role (ENUM: ADMIN, PATIENT, DOCTOR)

4. Database Schema

4.1 Table Definitions

1. **Patient Table**

```
CREATE TABLE Patient (
  patientId INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(100),
  dateOfBirth DATE,
  gender VARCHAR(10),
  contactNumber VARCHAR(15),
  address VARCHAR(255),
  medicalHistory TEXT
);
```

2. **Doctor Table**

```
CREATE TABLE Doctor (
  doctorId INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(100),
  specialization VARCHAR(100),
  contactNumber VARCHAR(15),
  availabilitySchedule TEXT
);
```

3. **Appointment Table**

```
CREATE TABLE Appointment (

  appointmentId INT PRIMARY KEY AUTO_INCREMENT,
  patientId INT,
  doctorId INT,
  appointmentDate DATE,
  timeSlot VARCHAR(20),
  status ENUM('CONFIRMED', 'CANCELLED'),
  FOREIGN KEY (patientId) REFERENCES Patient(patientId),
  FOREIGN KEY (doctorId) REFERENCES Doctor(doctorId)
);
```

4. **Bill Table**

```
CREATE TABLE Bill (
  billId INT PRIMARY KEY AUTO_INCREMENT,
  patientId INT,
  totalAmount DECIMAL(10, 2),
  paymentStatus ENUM('PAID', 'UNPAID'),
  billDate DATE,
```

```
    FOREIGN KEY (patientId) REFERENCES Patient(patientId)
);
```

5. **User Table**

```
CREATE TABLE User (
    userId INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE,
    password VARCHAR(255),
    role ENUM('ADMIN', 'PATIENT', 'DOCTOR')
);
```

5. Local Deployment Details

1. **Environment Setup:**
    - Install JDK 17 or .NET SDK 7.0.
    - Install MySQL or SQL Server.
    - Use an application server (Tomcat for Java, Kestrel for .NET).
2. **Deployment Steps:**
    - Clone the repository.
    - Configure the database connection string in application.properties (Java) or appsettings.json (.NET).
    - Run the provided SQL scripts to initialize the database schema.
    - Build and start the application locally.

6. Conclusion

This document outlines the low-level design for the **Hospital Management System**, ensuring modularity, scalability, and compatibility with **Spring MVC** and **ASP.NET Core MVC** frameworks.