Project: Hospital Appointment and Management System

1. Introduction

This document outlines the Low-Level Design (LLD) for a **Hospital Appointment and Management System**, which provides an efficient way to schedule patient appointments, track medical histories, and manage doctor schedules.

This design supports both **Java (Spring Boot)** and **.NET (ASP.NET Core)** frameworks for backend development.

2. Module Overview

2.1 Patient Registration and Profile Management Module

- Registers patients with personal and medical information.
- Allows patients to update their profiles and medical history.

2.2 Appointment Scheduling Module

- Enables patients to book appointments with doctors based on their availability.
- Includes functionality for rescheduling and canceling appointments.

2.3 Doctor Schedule Management Module

- Manages the availability and schedule of doctors.
- Allows doctors to update their availability for patient appointments.

2.4 Medical History Management Module

- Tracks patient medical history and stores records of treatments, medications, and diagnoses.
- Provides access to medical data for doctors and authorized staff.

2.5 Notification and Reminder Module

- Sends appointment reminders to patients.
- Notifies doctors and patients about upcoming appointments, cancellations, or changes.

3. Architecture Overview

3.1 Architectural Style

- **Frontend**: Angular or React for dynamic user interfaces.
- **Backend**: REST API-based architecture for data handling and business logic.
- **Database**: Relational Database (MySQL/PostgreSQL/SQL Server) for structured data storage.

3.2 Component Interaction

- The frontend interacts with the backend through REST APIs for scheduling, medical records, and profile management.
- The backend connects to the database for storing and retrieving patient and doctor data.

4. Module-Wise Design

4.1 Patient Registration and Profile Management Module

4.1.1 Features

- Allows the registration of new patients.
- Enables patients to update their personal and medical details.

4.1.2 Data Flow

- Patients input their details through the frontend.
- The backend processes and stores this information in the database.

4.1.3 Entities

- **PatientProfile**
  - PatientID
  - Name
  - DateOfBirth
  - MedicalHistory
  - ContactDetails

4.2 Appointment Scheduling Module

4.2.1 Features

- Enables patients to book appointments with available doctors.
- Provides an option to reschedule or cancel existing appointments.

4.2.2 Entities

- **Appointment**
  - AppointmentID
  - PatientID
  - DoctorID
  - AppointmentDate
  - Status (Scheduled, Cancelled, Completed)

4.3 Doctor Schedule Management Module

4.3.1 Features

- Allows doctors to update their availability and set time slots for patient appointments.
- Enables doctors to view and manage their schedules.

4.3.2 Entities

- **DoctorSchedule**
  - DoctorID
  - AvailableTimeSlots
  - DoctorID

4.4 Medical History Management Module

4.4.1 Features

- Tracks and stores patient medical records, treatments, diagnoses, and medications.
- Provides doctors access to historical patient data.

4.4.2 Entities

- **MedicalHistory**
  - HistoryID
  - PatientID
  - Diagnosis
  - Treatment
  - DateOfVisit

4.5 Notification and Reminder Module

4.5.1 Features

- Sends automated reminders to patients about upcoming appointments.
- Notifies patients and doctors about appointment changes or cancellations.

4.5.2 Entities

- **Notification**
  - NotificationID
  - PatientID
  - Message
  - Timestamp

5. Deployment Strategy

5.1 Local Deployment

- The system is initially deployed in a local environment for development and testing.

5.2 Testing Environments

- Use containerized environments for consistency across testing and staging setups.

6. Database Design

6.1 Tables and Relationships

- **PatientProfile**: Primary Key: PatientID.
- **Appointment**: Foreign Key: PatientID, DoctorID.
- **DoctorSchedule**: Primary Key: DoctorID, AvailableTimeSlots.
- **MedicalHistory**: Foreign Key: PatientID.

- **Notification**: Foreign Key: PatientID.

7. User Interface Design

7.1 Wireframes

- **Dashboard**: Shows upcoming appointments and reminders.
- **Patient Profile**: Displays patient information and medical history.
- **Doctor Schedule**: Allows doctors to view and manage their schedules.

8. Non-Functional Requirements

8.1 Performance

- The system should support simultaneous appointment booking for up to 1,000 users.

8.2 Usability

- Designed to be intuitive for both healthcare providers and patients.

8.3 Security

- Secure login and role-based access to ensure patient confidentiality.

8.4 Scalability

- System should scale to accommodate a growing number of healthcare facilities and users.

9. Assumptions and Constraints

9.1 Assumptions

- Patients and doctors have access to the internet or mobile devices for using the system.

9.2 Constraints

- Initial deployment is limited to a single healthcare facility.