Project: Patient Wellness Monitoring System

1. Introduction

This document outlines the Low-Level Design (LLD) for a **Patient Wellness Monitoring System**, which assists healthcare providers in monitoring patient progress, tracking wellness plan adherence, and offering personalized recommendations.

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

- Facilitates registration of new patients and management of their health profiles.
- Includes functionality for updating medical history and contact information.

2.2 Wellness Plan Management Module

- Allows doctors to create and assign personalized wellness plans for patients.
- Includes templates for common wellness strategies.

2.3 Progress Tracking Module

- Monitors patient adherence to assigned wellness plans.
- Displays progress charts and completion rates.

2.4 Notification and Alerts Module

- Sends reminders to patients for medication, exercises, and follow-up visits.
- Alerts healthcare providers about missed activities or irregularities.

2.5 Reporting and Analytics Module

- Generates detailed health reports and analytics for wellness outcomes.
- Provides insights for improving patient health plans.

3. Architecture Overview

3.1 Architectural Style

- **Frontend**: Angular or React for intuitive user interfaces.

- **Backend**: REST API-based architecture for business logic and data management.
- **Database**: Relational Database (MySQL/PostgreSQL/SQL Server) for secure data storage.

3.2 Component Interaction

- The frontend interacts with the backend through REST APIs for patient management and monitoring.
- The backend interfaces with the database to handle patient data and wellness plans.

4. Module-Wise Design

4.1 Patient Registration and Profile Management Module

4.1.1 Features

- Register new patients with basic details and medical history.
- Update patient profiles as needed.

4.1.2 Data Flow

- Patients or staff input details through the frontend.
- Backend processes the data and updates the database.

4.1.3 Entities

- **PatientProfile**
  - PatientID
  - Name
  - Age
  - ContactDetails
  - MedicalHistory

4.2 Wellness Plan Management Module

4.2.1 Features

- Create custom wellness plans with predefined templates.
- Assign plans to patients and update them as needed.

4.2.2 Entities

- **WellnessPlan**
  - PlanID
  - PlanName
  - Activities
  - AssignedTo

4.3 Progress Tracking Module

4.3.1 Features

- Track daily activities completed by the patient.
- Provide progress dashboards for patients and doctors.

4.3.2 Entities

- **Progress**
  - ProgressID
  - PatientID
  - CompletedActivities

- PlanID

4.4 Notification and Alerts Module

4.4.1 Features

- Send automatic reminders for tasks and appointments.
- Notify doctors of missed or irregular patient activities.

4.4.2 Entities

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

4.5 Reporting and Analytics Module

4.5.1 Features

- Generate weekly or monthly health reports.
- Provide analytics to improve wellness outcomes.

4.5.2 Entities

- **Report**
  - ReportID
  - PatientID
  - Summary
  - Date

5. Deployment Strategy

5.1 Local Deployment

- Both frontend and backend deployed on local machines for development and initial testing.

5.2 Testing Environments

- Staging environments configured using containerized setups for consistency.

6. Database Design

6.1 Tables and Relationships

- **PatientProfile**: Primary Key: PatientID.
- **WellnessPlan**: Primary Key: PlanID.
- **Progress**: Foreign Key: PlanID, PatientID.
- **Notification**: Foreign Key: PatientID.
- **Report**: Foreign Key: PatientID.

7. User Interface Design

7.1 Wireframes

- **Dashboard**: Displays wellness plan adherence and notifications.
- **Patient Profile**: Displays patient details and medical history.
- **Progress Tracker**: Visual representation of completed activities.

8. Non-Functional Requirements

8.1 Performance

- System should handle updates for 500 patients concurrently.

8.2 Usability

- Designed for ease of use by both patients and healthcare professionals.

8.3 Security

- Role-based access control to ensure data privacy.

8.4 Scalability

- Capable of supporting multiple healthcare facilities.

9. Assumptions and Constraints

9.1 Assumptions

- Patients have access to mobile or web interfaces.

9.2 Constraints

- Limited to a single healthcare facility during the initial phase.