Project: IntelliCare – AI-Driven Outpatient Management System

1. Introduction

This document outlines the Low-Level Design (LLD) for **IntelliCare**, an AI-powered Outpatient Management System designed to streamline outpatient workflows in multi-specialty hospitals and clinics. The system facilitates patient registration, appointment scheduling, consultation tracking, prescription management, and analytics for operational efficiency.

Supports backend development using **Java (Spring Boot)** and **.NET (ASP.NET Core)**.

2. Module Overview

2.1 Patient Registration & Profile Management Module

- Captures patient demographics, medical history, and insurance details.
- Enables profile updates and secure access to health records.

2.2 Appointment & Queue Management Module

- Manages doctor schedules, patient appointments, and real-time queue tracking.
- Sends automated reminders and rescheduling options.

2.3 Consultation & Prescription Module

- Records doctor notes, diagnoses, and prescriptions.
- Integrates with pharmacy systems for e-prescriptions.

2.4 Billing & Insurance Claims Module

- Generates invoices, tracks payments, and manages insurance claims.
- Supports multiple payment gateways and claim status tracking.

2.5 Analytics & Operational Reporting Module

- Provides insights into patient flow, doctor utilization, and revenue.
- Generates dashboards for hospital administrators.

3. Architecture Overview

3.1 Architectural Style

- **Frontend**: Angular or React for responsive UI.
- **Backend**: RESTful APIs for modular service interaction.
- **Database**: Relational (PostgreSQL/MySQL) for structured patient and billing data.

3.2 Component Interaction

- Frontend communicates with backend via REST APIs.
- Backend interacts with database and external pharmacy/insurance APIs.

4. Module-Wise Design

4.1 Patient Registration & Profile Management Module

4.1.1 Features

- Add/update/view patient profiles.
- Upload documents (ID proof, insurance).
- Secure access via OTP/email verification.

4.1.2 Data Flow

- Frontend captures patient data → Backend validates and stores → Database persists.

4.1.3 Entities

- **PatientProfile**
  - PatientID
  - Name
  - DOB
  - ContactInfo
  - InsuranceDetails
  - MedicalHistory

4.2 Appointment & Queue Management Module

4.2.1 Features

- Book/reschedule/cancel appointments.
- Real-time queue tracking and doctor availability.

4.2.2 Entities

- **Appointment**
  - AppointmentID
  - PatientID
  - DoctorID
  - DateTime
  - Status

4.3 Consultation & Prescription Module

4.3.1 Features

- Record diagnosis, treatment plan, and prescriptions.
- Generate e-prescriptions and share with pharmacy.

4.3.2 Entities

- **Consultation**
  - ConsultationID
  - PatientID
  - DoctorID
  - Notes
  - Prescription

4.4 Billing & Insurance Claims Module

4.4.1 Features

- Generate bills, apply insurance, track payments.
- Claim submission and approval tracking.

4.4.2 Entities

- **Invoice**
  - InvoiceID
  - PatientID
  - Amount

    - InsuranceProvider
    - Status

4.5 Analytics & Operational Reporting Module

4.5.1 Features

- Reports on patient visits, revenue, doctor utilization.
- Predictive analytics for peak hours and resource planning.

4.5.2 Entities

- **Report**
    - ReportID
    - Type (Revenue/Utilization/PatientFlow)
    - GeneratedDate
    - Metrics

5. Deployment Strategy

5.1 Local Deployment

- Developer machines for unit testing and validation.

5.2 Staging and Production Environments

- Cloud deployment (AWS/Azure) with auto-scaling and load balancing.

6. Database Design

6.1 Tables and Relationships

- PatientProfile: PK = PatientID
- Appointment: FK = PatientID, DoctorID
- Consultation: FK = PatientID, DoctorID
- Invoice: FK = PatientID
- Report: PK = ReportID

7. User Interface Design

7.1 Wireframes

- Dashboard: Patient stats, appointment queue, revenue summary.
- Appointment Scheduler: Calendar view with doctor slots.
- Consultation Notes: Editable form with prescription generator.

8. Non-Functional Requirements

8.1 Performance

- Support 2,000 concurrent users with <2s response time.

8.2 Usability

- Mobile-friendly UI for doctors and patients.
- Tooltips and help guide for non-tech users.

8.3 Security

- Role-based access (Admin, Doctor, Receptionist).
- Data encryption (AES-256) and secure API tokens.

8.4 Scalability

- Multi-clinic support with centralized data access.

9. Assumptions and Constraints

9.1 Assumptions

- Patients will use mobile/web portals for appointments.
- Doctors will enter consultation notes digitally.

9.2 Constraints

- Initial rollout for outpatient departments only.
- Integration with pharmacy and insurance APIs in phase 2.