🏥 Project Documentation: Pulse HMS

"The Operating System for Modern Healthcare"

📄 Page 1: Executive Summary & Architecture

1.1 Project Overview

Pulse HMS is an enterprise-grade, **Microservices-based Hospital Management System**. Unlike traditional monolithic hospital software, Pulse HMS is designed as a distributed ecosystem where each department (Clinical, Administrative, Finance, Logistics) operates as an independent service.

The system connects **Patients, Doctors, Ambulance Drivers, Lab Technicians, and Administrators** into a single, real-time digital workflow. It eliminates manual paperwork by automating appointments, room allotment, billing aggregation, and emergency dispatch.

1.2 High-Level Architecture

The system is built on a **Cloud-Native Architecture** ensuring scalability, fault tolerance, and separation of concerns.

- **Backend:** Java Spring Boot (11+ Microservices).
- **Frontend:** React 19 (Vite) + TypeScript + Redux Toolkit.
- **Database Pattern:** Database-per-service (MySQL) to ensure loose coupling.
- **Communication Layer:**
  - **Synchronous (REST/Feign):** For critical data retrieval (e.g., Billing fetching Room charges).
  - **Asynchronous (Apache Kafka):** For decoupling high-volume events (e.g., Sending Emails, SMS).
  - **Real-Time (WebSockets):** For live alerts (e.g., "Ambulance Assigned").

1.3 Technology Stack

Layer	Technology Used	Purpose
Backend Framework	Java 17, Spring Boot 3.x	Core Microservice logic.

Layer	Technology Used	Purpose
Frontend Framework	React 19, Vite, TypeScript	High-performance SPA.
State Management	Redux Toolkit	Managing Auth, User Profile, and UI states.
UI Components	Mantine UI, PrimeReact, Tailwind	Responsive, accessible, and modern UI.
Data Visualization	**Apache ECharts**	Professional Gradient Area charts, Donut charts.
Database	MySQL 8.0	Relational data persistence.
Caching	**Redis**	Caching Profiles, Inventory, and OTPs for speed.
Message Broker	**Apache Kafka**	Event-driven notifications.
Security	Spring Security + JWT	Stateless Authentication.
Mapping	OpenStreetMap + Leaflet	Zero-cost GPS tracking & Geocoding.
Media Storage	Cloudinary	Offloading images/PDFs to the cloud.

📄 Page 2: Backend Ecosystem (The Core)

The backend is divided into domain-specific microservices.

2.1 Core Identity Services

- **GatewayMS (The Gatekeeper):**
  - Acts as the single entry point (Port 9000).
  - Routes requests to internal services (e.g., `/user/**` $\to$ `UserMS`).
  - **Security Filter:** Validates JWT Tokens before routing.
- **UserMS (Identity Provider):**
  - Handles Registration & Login.
  - **2-Step Verification:** Generates OTPs, stores them in **Redis** (TTL 5 mins), and verifies via Email.
  - Issues JWT Tokens containing Role & UserID.
- **ProfileMS (The Directory):**
  - Stores extended details for Doctors, Patients, Drivers, and Technicians.
  - **Caching:** Uses Redis (`@Cacheable`) to serve profile data instantly without hitting MySQL.

2.2 Clinical & Operations Services

- **AppointmentMS (The Scheduler):**
  - **Core Logic:** Manages Doctor availability and booking slots.
  - **Room Allotment Module:** An embedded module handling Hospital Beds.
    - *Admission:* Checks bed availability $\to$ Locks Bed $\to$ Creates Allotment.
    - *Discharge:* Calculates rent duration $\to$ Frees Bed $\to$ Exposes charges to Billing.
  - **Analytics:** Aggregates data for the Doctor Dashboard (Today's Schedule, Pending Count).
- **LabMS (Diagnostics):**
  - **Master Data:** Admins define tests (Name, Category, Price).
  - **Workflow:** Request Created $\to$ Sample Collected $\to$ Result Uploaded $\to$ Completed.
  - **Billing Integration:** Exposes unbilled completed tests to BillingMS.
- **PharmacyMS (Inventory):**
  - **Stock Management:** Tracks batch numbers and expiry dates.
  - **Logic:** Prevents selling expired medicine. Deducts stock automatically upon sale.
  - **Analytics:** Queries for "Low Stock" (<10 units) and "Expiring Soon" (<30 days).

2.3 Emergency & Logistics

- **AmbulanceMS (Logistics):**
  - **Uber-like Logic:** Patients request rides; System assigns available drivers.
  - **Route Calculation:** Uses **OpenRouteService** to calculate distance and estimated time.
  - **Driver Dashboard API:** Allows drivers to toggle status (Online/Offline) and manage trip states (Accepted $\to$ On The Way $\to$ Completed).
- **BloodBankMS:**
  - Manages Blood Units (Groups A+, B-, etc.) and Donor Registry.
  - **Aggregation:** Provides stock level counts for the Admin Dashboard visualizations.

2.4 Finance & Support

- **BillingMS (The Aggregator):**
  - It does **not** generate data itself. It acts as a collector.
  - **Process:** When "Generate Bill" is clicked, it calls `AppointmentMS` (Consultation + Room Rent), `PharmacyMS`, `LabMS`, `AmbulanceMS`, and `BloodBankMS`.
  - Sums up all unbilled items and creates a **PatientBill**.
- **PaymentMS:**
  - Integrates with **Razorpay**.
  - Verifies payment signatures and updates Bill Status to `PAID`.
- **NotificationMS:**
  - Listens to Kafka topics (`APPOINTMENT_BOOKED`, `LAB_RESULT_READY`).
  - Sends Emails via SMTP.

📄 Page 3: Frontend Architecture & Dashboard Design

The Frontend is a highly interactive **Single Page Application (SPA)** designed for specific user roles.

3.1 Visual Analytics Strategy (ECharts)

We replaced standard tables with **Apache ECharts** for a premium "Command Center" feel.

- **Wavy KPI Cards:** Custom SVG wave animations inside statistic cards (e.g., Total Revenue).
- **Gradient Area Charts:** Used for "Hospital Survey" to show patient footfall trends with a smooth, glowing gradient fill.
- **Donut Charts:** Interactive charts for "Appointment Status" and "Room Occupancy" with dynamic center text that updates on hover.
- **Bar Charts:** Horizontal bars for "Blood Bank Stock" and vertical bars for "Revenue Breakdown".

3.2 Admin Dashboard (The Control Tower)

- **Top Section:** 4 Wavy KPI Cards (Appointments, Lab Ops, New Patients, Earnings).
- **Middle Section:**
  - *Hospital Survey:* A 7-day trend line graph.
  - *Appointment Distribution:* A Donut chart showing Scheduled vs. Completed vs. Cancelled.
- **Bottom Section:**
  - *Room Analytics:* A specific card showing Occupied/Available beds.
  - *Inventory Alerts:* A list highlighting Low Stock medicines.

3.3 Patient Dashboard (Personal Health Hub)

- **Design Philosophy:** "Status & Action".
- **Live Status Cards:**
  - *Ambulance:* If active, shows "🚑 On the Way". If idle, shows "Book Now".
  - *Bill:* Shows "Total Due" in Red if unpaid.
- **Medical History:**
  - *Timeline:* Vertical timeline of past doctor visits.
  - *Lab Reports:* List of recent tests with status badges (Pending/Completed).
  - *Digital Boarding Pass:* A card displaying current Room Admission details (Room No, Doctor, Date).

3.4 Doctor Dashboard (Daily Planner)

- **Focus:** Workflow Efficiency.

- **Schedule Timeline:** A vertical list of today's patients time-slot wise.
- **Action Buttons:** "Start Consultation" button appears next to the current/next patient.
- **Progress Ring:** A visual ring showing the percentage of today's appointments completed.

📄 Page 4: Specialized Modules & Workflows

4.1 The Room Allotment System (Logic Deep Dive)

Implemented within `AppointmentMS` to save resources, but logically distinct.

1. **Inventory Creation:** Admins create Room Types (e.g., ICU, Private) and define `PricePerDay`. Beds are auto-generated (e.g., `ICU-101-A`).
2. **Admission (Booking):**
   - Admin selects a Patient and Room Type.
   - System finds the first `AVAILABLE` bed.
   - Status updates: Bed $\to$ `OCCUPIED`, Allotment $\to$ `RESERVED`.
3. **Discharge:**
   - System calculates `Days Stayed` = `DischargeDate` - `AdmitDate`.
   - Calculates `Total Cost` = `Days` * `PricePerDay`.
   - Status updates: Bed $\to$ `AVAILABLE`, Allotment $\to$ `DISCHARGED`.
   - **Billing:** This cost is exposed via an API for BillingMS to collect.

4.2 The Lab Technician Console

A dedicated dashboard for Lab staff to manage the test lifecycle.

1. **Queue Management:** A Tabbed Interface.
   - *Tab 1 (Pending):* Requests created by Doctors. Action: **"Collect Sample"**.
   - *Tab 2 (In Progress):* Samples collected. Action: **"Upload Result"**.
2. **Result Upload:** A Modal to enter the test result (e.g., "Hemoglobin: 12g/dL") and attach a Report URL.
3. **Automation:** Upon upload, the status changes to `COMPLETED`, and a Kafka event triggers an email to the patient.

4.3 The Mobile-First Driver Console

Designed for use on smartphones inside an ambulance.

1. **Toggle Switch:** Big switch to go **"Online"** or **"Offline"**.
2. **Trip Card:** When a ride is assigned, a card appears with:

- Patient Name & Phone.
- **Map View:** Interactive Leaflet map showing Pickup & Drop points.
- **Slide to Action:** Buttons to update status: "Start Trip" $\to$ "Complete Trip".

📄 Page 5: Security, Performance & Scalability

5.1 Security Implementation

- **JWT (JSON Web Token):** Stateless authentication. The Gateway verifies the signature before letting any request pass.
- **CORS (Cross-Origin Resource Sharing):** Configured in Gateway to allow requests only from the React Frontend (`localhost:5173`).
- **Password Hashing:** BCrypt used for storing passwords in `UserMS` .

5.2 Performance Optimization

- **Redis Caching:**
  - *Profile Lookup:* Profiles are cached. Subsequent requests take <5ms.
  - *Inventory:* Medicine list pages are cached to reduce DB load.
  - *Cache Eviction:* When a profile is updated or medicine stock changes, the cache is automatically invalidated (`@CacheEvict`) to ensure data consistency.
- **Pagination:** All "Get All" APIs (Doctors, Medicines, Appointments) implement server-side pagination (`Pageable`) to handle thousands of records efficiently.

5.3 Resilience & Scalability

- **Netflix Eureka:** Service Discovery allows services to find each other dynamically without hardcoded URLs.
- **Feign Client Fallbacks:** If `ProfileMS` is down, `AppointmentMS` can still list appointments (showing "Unknown Name" instead of crashing).
- **Docker Ready:** Each service is self-contained and can be containerized for deployment on Kubernetes.