High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **MedSync - Secure Healthcare Appointment Platform**. MedSync is a full-featured, secure system for healthcare appointment booking and management, supporting user authentication, role-based access, secure file uploads, real-time notifications, and analytics.

1. System Architecture Overview

Architecture Description:
MedSync is a modular, fullstack web application with a React/Next.js frontend, Node.js/Express backend, RESTful APIs, and dual-database integration (PostgreSQL and MongoDB). JWT is used for authentication, and Multer handles secure file uploads. Real-time notifications are delivered via WebSockets.

Main System Modules

Module	Description
Frontend (Next.js)	User interfaces for patients, doctors, and admins; real-time notifications.
API Gateway (Express)	Handles RESTful API requests, authentication, and routing.
Auth Service	JWT-based authentication and role-based authorization.
Appointment Service	Manages booking, scheduling, and appointment data.
File Upload Service	Secure upload and retrieval of medical records (Multer, S3/local).
Notification Service	Real-time notifications (WebSockets).
Admin Dashboard	Analytics, user management, and system monitoring.
PostgreSQL DB	Stores structured data (users, appointments, roles).
MongoDB	Stores unstructured data (medical records, logs).

2. Component Interactions

Interaction	Flow Description
User ↔ Frontend	Users interact via web UI for booking, uploads, and notifications.
Frontend ↔ API Gateway	Frontend sends RESTful requests; receives data and notifications.
API Gateway ↔ Auth Service	Validates JWT tokens and enforces role-based access.
API Gateway ↔ Appointment Service	Handles appointment CRUD operations.
API Gateway ↔ File Upload Service	Manages secure upload/download of medical records.

API Gateway ↔ Notification Service	Triggers and delivers real-time notifications to users.
API Gateway ↔ PostgreSQL/MongoDB	Reads/writes structured and unstructured data as required.
Admin Dashboard ↔ API Gateway	Admins access analytics and user management features.

3. Data Flow Overview

Data Flow	Source	Destination	Purpose
User Registration/Login	Frontend	Auth Service	Authenticate and issue JWT tokens
Appointment Booking/Management	Frontend	Appointment Svc	Create, update, view appointments
Medical Record Upload	Frontend	File Upload Svc	Store/retrieve files in MongoDB/S3
Notifications	Backend	Frontend	Real-time updates via WebSockets
Analytics/User Management	Admin Dashboard	API Gateway	Admin operations and reporting

4. Technology Stack

Layer/Component	Technology/Framework
Frontend	Next.js, React, Tailwind CSS, TypeScript
Backend/API	Node.js, Express, TypeScript
Authentication	JWT
File Uploads	Multer, S3/local storage
Real-time	WebSockets (e.g., Socket.io)
Relational Database	PostgreSQL
NoSQL Database	MongoDB
Analytics/Dashboard	Custom (React/Next.js)

5. Scalability, Reliability & Security

- **Scalability:** Stateless backend enables horizontal scaling; databases can be clustered/sharded as needed.
- **Reliability:** JWT for secure, stateless sessions; input validation and error handling throughout.
- **Security:** Role-based access control, encrypted file storage, HTTPS enforced, secure JWT handling, and audit logging.
- **Data Integrity:** PostgreSQL for transactional data; MongoDB for flexible document storage.

End of Document