# High Level Design Document

## Introduction

This High Level Design (HLD) document outlines the architecture and core components of **Helios - Secure Healthcare Appointment System**. Helios is a robust, secure platform for managing healthcare appointments, featuring RESTful APIs, JWT authentication, role-based access control, advanced error handling, asynchronous notifications, and Dockerized deployment. The system includes a React-based frontend for both patients and doctors.

---

## 1. System Architecture Overview

**Architecture Description:**
Helios follows a modular, layered architecture with clear separation between frontend, backend, and data storage. The system is containerized using Docker for portability and scalability.

| Module/Component | Role/Responsibility |
|---|---|
| React Frontend | Patient/Doctor dashboards, appointment management UI, communicates via REST APIs |
| Spring Boot Backend | Core business logic, RESTful API endpoints, authentication, authorization, error handling |
| MySQL Database | Persistent storage for users, appointments, notifications, and audit logs |
| Notification Service | Asynchronous background processing for email/SMS notifications |
| Docker | Containerization and deployment orchestration |

---

## 2. Component Interactions

| Interaction Step | Description |
|---|---|
| 1. User accesses frontend | Patient/Doctor logs in or registers via React UI |
| 2. Frontend calls backend REST API | API requests sent to Spring Boot backend (e.g., login, book appointment) |
| 3. Backend authenticates & authorizes | JWT-based authentication and role-based access control enforced |
| 4. Backend interacts with database | CRUD operations via Spring Data JPA to MySQL |
| 5. Backend triggers notifications | Appointment events trigger async notification jobs |
| 6. Notification service sends updates | Emails/SMS sent to users asynchronously |

---

## 3. Data Flow Overview

| Data Flow | Source | Destination | Description |
|---|---|---|---|
| User credentials/auth requests | Frontend | Backend | Login/registration, JWT token issuance |
| Appointment data | Frontend/Backend | Backend/Database | Booking, updating, or canceling appointments |
| Notification events | Backend | Notification Svc | Async job queue for notifications |
| Notification delivery | Notification Svc | Users | Email/SMS sent to patients/doctors |
| Role/permission checks | Backend | Database | Enforce access control on API endpoints |

---

## 4. Technology Stack

| Layer/Component | Technology/Framework |
|---|---|
| Frontend | React, JavaScript, HTML/CSS |
| Backend | Java, Spring Boot, Spring Data JPA, Spring Security |
| Database | MySQL |
| Notifications | Java (Async), Email/SMS API |
| Build/Dependency | Maven |
| Containerization | Docker |

---

## 5. Scalability, Reliability & Security

- **Scalability:**
  - Stateless backend enables horizontal scaling via Docker containers.
  - Database can be clustered or replicated for high availability.
  - Asynchronous notification processing decouples user actions from background jobs.
- **Reliability:**
  - Advanced error handling and logging for traceability.
  - Health checks and container orchestration for fault tolerance.
- **Security:**
  - JWT-based authentication and role-based access control.
  - Secure password storage, input validation, and HTTPS enforcement.
  - Audit logging for sensitive operations.

---

**End of Document**