Product Requirements & Specification Document

Project Name

Helios - Secure Healthcare Appointment System

Overview

Helios is a secure, scalable healthcare appointment management platform designed for modern clinics and hospitals. It features RESTful APIs, JWT-based 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. Objectives

- Enable secure, efficient appointment scheduling and management for patients and doctors.
- Ensure robust authentication, authorization, and data protection.
- Provide real-time and asynchronous notifications.
- Support scalable, containerized deployment.

---

2. Stakeholders

Role	Responsibilities
Patients	Book, view, and manage appointments
Doctors	Manage schedules, view appointments
Admins	System configuration, user management
Developers	Build, deploy, and maintain the system

---

3. Functional Requirements

ID	Requirement
FR1	User registration and login (patients, doctors, admins)
FR2	JWT-based authentication for all API endpoints
FR3	Role-based access control (RBAC)
FR4	Patients: book, view, cancel appointments
FR5	Doctors: view, approve, reject, and manage appointments
FR6	Admins: manage users, view system logs
FR7	RESTful API for all core operations

FR8	Asynchronous email/SMS notifications for appointment events
FR9	Advanced error handling with meaningful responses
FR10	React frontend: patient and doctor dashboards

---

4. Non-Functional Requirements

ID	Requirement
NFR1	High security (encryption, secure storage)
NFR2	Scalability (Dockerized, stateless services)
NFR3	High availability and reliability
NFR4	Responsive UI (React, mobile-friendly)
NFR5	Compliance with healthcare data standards

---

5. System Architecture

```
[React Frontend] <-> [Spring Boot REST API] <-> [MySQL DB]
                                  |
                        [Async Notification Service]
                                  |
                              [Docker]
```

- **Frontend:** React (patient/doctor dashboards)
- **Backend:** Java, Spring Boot, Spring Data JPA
- **Database:** MySQL
- **Notifications:** Asynchronous (email/SMS)
- **Deployment:** Docker containers, orchestrated via Docker Compose

---

6. API Specifications

Endpoint	Method	Auth	Description
/api/auth/register	POST	Public	Register user
/api/auth/login	POST	Public	Login, returns JWT
/api/appointments	GET	JWT	List appointments (role-based)
/api/appointments	POST	JWT	Book appointment (patient)
/api/appointments/{id}	PUT	JWT	Update appointment (doctor/admin)
/api/appointments/{id}	DELETE	JWT	Cancel appointment (patient/admin)
/api/users	GET	Admin	List users
/api/notifications	POST	JWT	Send notification (async)

7. Data Model (Simplified)

Entity	Fields
User	id, name, email, password, role (PATIENT/DOCTOR/ADMIN), status
Appointment	id, patient_id, doctor_id, datetime, status, notes
Notification	id, user_id, type (email/SMS), content, status, timestamp

---

8. Security

- **Authentication:** JWT tokens, password hashing (BCrypt)
- **Authorization:** RBAC enforced at API level
- **Data Protection:** HTTPS, encrypted sensitive fields
- **Error Handling:** No sensitive info in error messages

---

9. Background Processing

- **Notifications:** Asynchronous via message queue or thread pool
- **Failure Handling:** Retry logic, dead-letter queue for failed notifications

---

10. Deployment

Component	Containerized	Notes
Backend API	Yes	Spring Boot, Maven
Frontend	Yes	React, Nginx
Database	Yes	MySQL official image
Notification	Yes	Optional microservice

- **Orchestration:** Docker Compose
- **Environment Variables:** For secrets/configuration

---

11. UI/UX Requirements

- **Theme:** Futuristic, clean, startup-inspired
- **Dashboards:** Separate for patients and doctors
- **Accessibility:** WCAG 2.1 compliance
- **Responsiveness:** Mobile and desktop support

---

12. Acceptance Criteria

- All core features implemented and tested

- Secure authentication and RBAC enforced
- Asynchronous notifications operational
- Dockerized deployment scripts provided
- Responsive, accessible frontend

---

13. Out of Scope

- Payment processing
- Telemedicine/video calls
- Third-party EHR integration

---

14. Milestones

Milestone	Description
M1: API & Auth	Core API, JWT, RBAC
M2: Appointment Logic	Booking, management, notifications
M3: Frontend Dashboards	React UI for all roles
M4: Dockerization	Containerized deployment
M5: Testing & Launch	QA, bugfixes, production release

End of Document