Healthcare Management System - Developer Guide

This document provides technical information for developers working on the Healthcare Management System.

Table of Contents

1.Project Structure

The project follows standard Spring Boot conventions with a Maven-based build system:

```
healthcare/
 ├── src/
|    ├── main/
|    |    ├── java/
|    |    |    └── com/
|    |    |        └── healthcare/
|    |    |            ├── config/          Configuration classes
|    |    |            ├── controller/      MVC controllers
|    |    |            ├── model/           JPA entities
|    |    |            ├── repository/      Data repositories
|    |    |            ├── service/         Business logic
|    |    |            └── HealthcareApplication.java  Main application class
|    |    └── resources/
|    |        ├── static/              Static resources (CSS, JS)
|    |        ├── templates/           Thymeleaf templates
```

```
|   |       └── application.properties  Configuration properties
|   └── test/                      Test classes
├── pom.xml                          Maven configuration
└── README.md
```

2. Technology Stack

- Java: Version 21
- Spring Boot: Version 3.4.4
- Spring Security: User authentication and authorization
- Spring Data JPA: Data access layer
- Thymeleaf: Server-side Java template engine
- Bootstrap: Frontend framework
- MySQL: Database
- Maven: Build and dependency management

3. Setting Up Development Environment

- Prerequisites

  - JDK 21
  - Maven 3.6+
  - MySQL 8.0+
  - Your favorite IDE (IntelliJ IDEA, Eclipse, VS Code)

4. Setup Steps

1. Clone the repository

```
git clone https://github.com/yourusername/healthcare-management.git
```

2. Configure database:
Create a MySQL database and update `src/main/resources/application.properties` with your database credentials:

```
properties
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/healthcare
spring.datasource.username=your_username
spring.datasource.password=your_password
spring.jpa.hibernate.ddl-auto=update
```

5. Access the application :- at http://localhost:8080

- Key Components

- Models

Located in `com.healthcare.model`:

- User : Base user entity with authentication details
- Patient : Medical information for patients
- Doctor : Professional information for doctors
- Appointment : Appointment details between patients and doctors
- Medication : Medication prescriptions

6. Repositories

Located in `com.healthcare.repository`:

- UserRepository : User data operations
- PatientRepository : Patient data operations
- DoctorRepository : Doctor data operations
- AppointmentRepository : Appointment data operations
- MedicationRepository : Medication data operations

7. Services

Located in `com.healthcare.service`:

- UserService : User management operations
- PatientService : Patient-specific operations
- DoctorService : Doctor-specific operations
- AppointmentService : Appointment scheduling and management

- MedicationService : Medication and prescription management

8. Controllers

Located in `com.healthcare.controller`:

- AuthController : Authentication operations
- ProfileController : User profile management
- AppointmentController : Appointment management
- MedicationController : Medication management
- DoctorController : Doctor-specific operations
- HomeController : Dashboard and home page
- ErrorController : Error handling

9.Database Schema

- Key Tables

- users : Basic user information and authentication
- patients : Medical information for patients
- doctors : Professional information for doctors
- appointments : Scheduled appointments between patients and doctors
- medications : Prescribed medications

10. Relationships

- One-to-One: User to Patient/Doctor
- One-to-Many: Patient to Appointments/Medications
- One-to-Many: Doctor to Appointments/Medications

11. Authentication and Authorization

The system uses Spring Security for authentication and authorization:

- Authentication : Form-based authentication with email and password
- Authorization : Role-based access control (PATIENT, DOCTOR, ADMIN)

- Security Configuration : Located in `com.healthcare.config.SecurityConfig`

12. Key Security Features

- Customized login page
- Password encoding with BCrypt
- Role-based access restrictions
- CSRF protection
- Session management

13. Common Development Tasks

- Adding a New Entity

1. Create a new entity class in the `model` package with JPA annotations
2. Create a repository interface in the `repository` package
3. Create a service class in the `service` package
4. Create a controller in the `controller` package if needed
5. Create or update Thymeleaf templates for the UI

14.Implementing a New Feature

1. Design the data model and update entities as needed
2. Implement repository methods for data access
3. Implement service methods for business logic
4. Create or update controllers for handling HTTP requests
5. Create or update Thymeleaf templates for the UI

15.Modifying Existing Feature

1. Locate the relevant entities, repositories, services, and controllers
2. Make necessary changes to the code
3. Update the corresponding templates
4. Test the changes thoroughly

16.Testing

- Unit Testing

- Use JUnit and Mockito for unit tests
- Test services with mocked repositories
- Test controllers with MockMvc

17. Integration Testing

- Use `@SpringBootTest` for integration testing
- Test the full request-response cycle
- Use an H2 in-memory database for testing

18.Deployment

- Production Configuration

  Update `application-prod.properties` with production settings:

  - properties
    database configuration
    spring.datasource.url=jdbc:mysql://prod-db-server:3306/healthcare
    spring.datasource.username=prod_user
    spring.datasource.password=prod_password

  - Logging
    logging.level.root=WARN
    logging.level.com.healthcare=INFO

  - Server configuration
    server.port=8080