HOSPITAL MANEGEMENT SYSTEM

**Project Overview**  
This Hospital Management System is developed using Java with JDBC for database operations, complemented by Swagger UI for API documentation and interactive testing. It manages hospital workflows such as patient and doctor management, appointment scheduling, and billing.

**Technology Stack**

* Java (Core JDBC with Spring Boot REST APIs)
* MySQL
* Swagger UI for REST API documentation and testing
* Eclipse IDE

**Project Structure**

hospital/

├── src/main/java/

│ └── com/example/hospital/

│ ├── HospitalApplication.java # Main Spring Boot application

│ ├── controller/

│ │ ├── DoctorController.java # Handles doctor-related HTTP requests

│ │ └── PatientController.java # Handles patient-related HTTP requests

│ ├── entity/

│ │ ├── Doctor.java # Doctor entity representing doctor table

│ │ └── Patient.java # Patient entity representing patient table

│ ├── repository/

│ │ ├── DoctorRepository.java # JDBC CRUD operations for doctor

│ │ └── PatientRepository.java # JDBC CRUD operations for patient

│ └── service/

│ ├── PatientService.java # Service interface for patient logic

│ └── PatientServiceImpl.java # Implementation of patient service

├── src/main/resources/

│ ├── static/ # Static resources if any (CSS, JS)

│ ├── templates/ # HTML templates if any

│ └── application.properties # Configurations (e.g., DB connection)

├── src/test/java/ # Unit and integration tests

├── pom.xml # Maven configuration file

└── HELP.md # Project documentation

**System Architecture**

* Controller: REST API endpoints exposed for hospital operations.
* Entity: Models matching database tables.
* Repository: Implements CRUD operations using JDBC.
* Service: Business logic separated from controller.
* Main Application: Bootstraps Spring Boot app.

**Swagger UI Integration**  
Swagger UI automatically documents all REST endpoints and provides an interactive interface for testing them without requiring a separate frontend. The Swagger UI typically runs at /swagger-ui.html or /swagger-ui/index.html and allows developers to submit requests, view responses, and understand the API structure visually.

**Core Modules & Features**

| **Module** | **Purpose** |
| --- | --- |
| Patient Management | Patient registration, updates, retrieval |
| Doctor Management | Doctor profile management, scheduling |
| Appointment Scheduling | Managing bookings and appointments |
| Billing System | Payment processing and invoice generation |

**Database Design**  
The database schema consists of tables for:

* Patient: Patient details and medical history.
* Doctor: Doctor details and specialties.
* Appointment: Scheduling information linking patients and doctors.

**Workflow Example**

* Register patients and doctors via API endpoints.
* Schedule appointments through REST APIs.
* Use Swagger UI to view, test, and validate these operations interactively.
* Retrieve patient, doctor, and appointment data through GET requests documented in Swagger.

**Output Interface Benefits**

* Simplifies API testing and debugging.
* Provides clear documentation of each API method and its input/output.
* Enhances collaboration by bridging backend and frontend development with API transparency.

**Security & Future Enhancements**

* Can include authentication and role-based access control using Spring Security.
* Expand modules (inventory, pharmacy).
* Improve error handling and validations

