Hospital Management System

# Abstract

The Hospital Management System is a full-stack Spring Boot application designed to manage hospital operations including patient records, physician assignments, appointments, departments, procedures, and certifications. It uses RESTful APIs and a MySQL database to ensure scalable and secure data handling.

# Objectives

- Streamline hospital workflows such as patient registration, appointment scheduling, and staff management.

- Maintain accurate and secure medical records.

- Provide modular and scalable architecture for future enhancements.

# Technologies Used

- Java 21

- Spring Boot Framework

- MySQL Database

- JPA (Java Persistence API)

- Maven for dependency management

- JUnit and Mockito for testing

# System Architecture

- Model Layer: Defines entities like Patient, Physician, Nurse, Department, Appointment, Procedure, TrainedIn, AffiliatedWith.

- Repository Layer: Interfaces with the database using Spring Data JPA.

- Service Layer: Contains business logic for each entity.

- Controller Layer: Exposes RESTful endpoints for client interaction.

# Database Design

- Patient: SSN, Name, Address, Phone, InsuranceID, PCP

- Physician: EmployeeID, Name, Position, SSN

- Nurse: EmployeeID, Name, Position, Registered, SSN

- Department: DepartmentID, Name, Head

- Appointment: AppointmentID, Patient, PrepNurse, Physician, Starto, Endo, ExaminationRoom

- Procedure: Code, Name, Cost

- TrainedIn: PhysicianID, ProcedureID, CertificationDate, CertificationExpires

- AffiliatedWith: PhysicianID, DepartmentID, PrimaryAffiliation

# API Endpoints

- /api/patient [POST, GET, PUT] - Manage patient records

- /api/physician [POST, GET, PUT] - Manage physician details

- /api/nurse [POST, GET, PUT] - Manage nurse details

- /api/department [POST, GET, PUT] - Manage departments

- /api/appointment [POST, GET, PUT] - Manage appointments

- /api/procedure [POST, GET, PUT] - Manage procedures

- /api/trained\_in [POST, GET, PUT] - Manage certifications

- /api/affiliated\_with [POST, GET] - Manage affiliations

# Implementation Details

- Entities are annotated with @Entity and mapped to database tables.

- Controllers delegate requests to services which handle business logic.

- Repositories extend JpaRepository for CRUD operations.

- JUnit and Mockito are used for unit testing service layers.

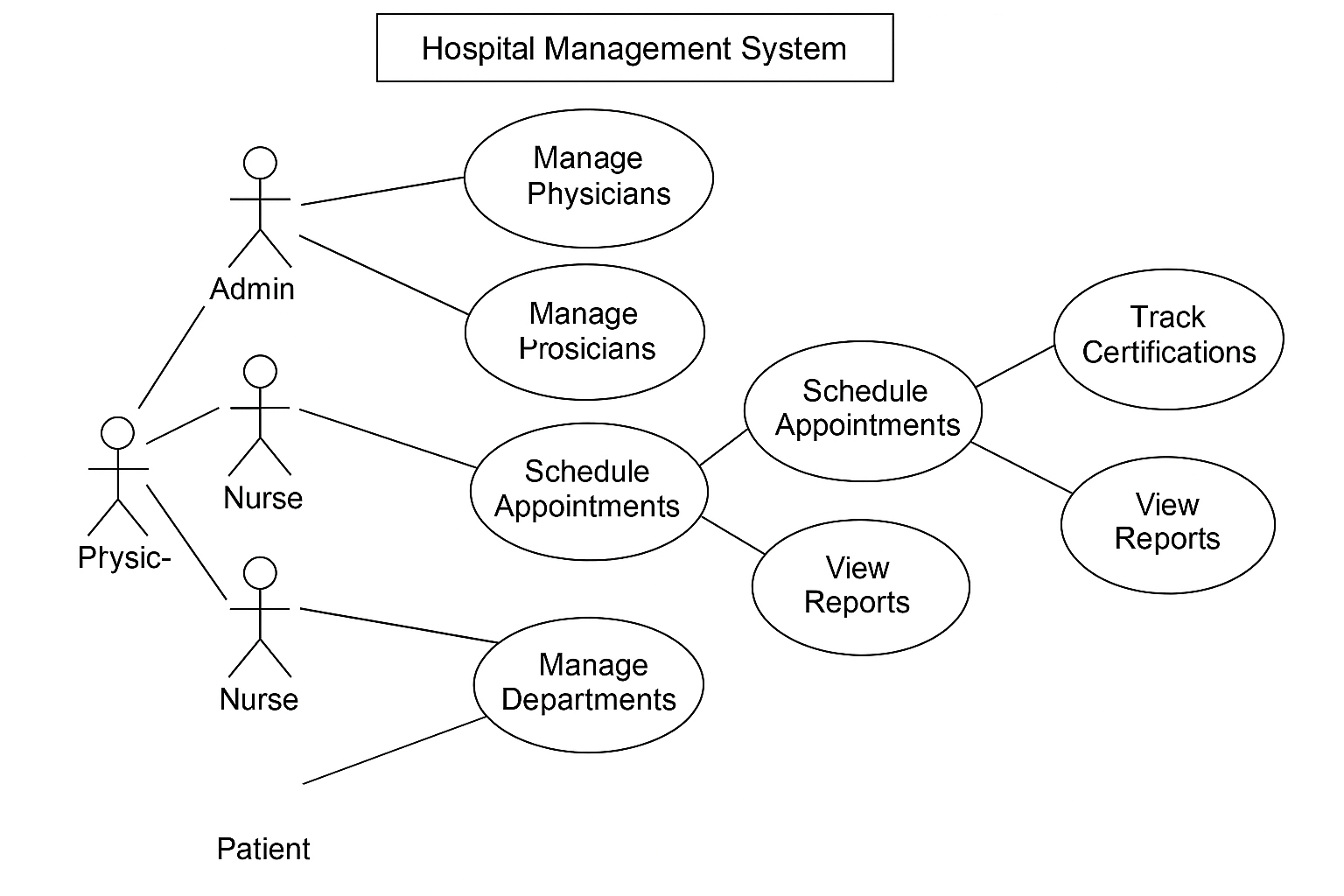
# Challenges Faced

- Designing normalized schema for complex relationships.

- Managing embedded IDs and composite keys.

- Ensuring test coverage with meaningful assertions.

# Use Case Diagram:



# Class Diagram: A diagram of a hospital management system AI-generated content may be incorrect.

# Outcome as a Learner

This project enhanced my understanding of full-stack development, RESTful API design, and relational database modeling. It also strengthened my skills in writing unit tests and debugging service logic.

# Conclusion

The Hospital Management System is a robust and scalable solution for managing healthcare operations. Future improvements could include user authentication, role-based access control, and integration with external health services.