Hospital Management System (HMS)

A Java-based Console Prototype Demonstrating OOP and Database Design

Prepared by

Mohammed Uaheel – KSD24CS115

Hafiz Abdulla – KSD24CS078

Mohammed Shazwan – KSD24CS111

Muhammed Falah - KSD24CS118

# 

# Table of Contents

1. Introduction

2. Project Design & Architecture

3. Implemented Modules & Features

4. Conclusion & Future Scope

# 1. Introduction

Project Title: Hospital Management System (HMS)

Overview: This project presents a robust, Java-based Hospital Management System developed to demonstrate core principles of Object-Oriented Programming (OOP) and software architecture. The system provides a console-based interface for three distinct user roles—Patient, Doctor, and Administrator—each with tailored functionalities. The primary objective is to streamline the appointment management process, from booking and viewing to cancellation, while ensuring data persistence through a lightweight, embedded SQLite database.

Purpose: The HMS prototype was built to simulate a real-world scenario of managing hospital operations. Its purpose is to showcase a functional application with a clean, layered architecture, secure user authentication, and reliable data management. It serves as a proof of concept for a scalable system that could be expanded to include more complex features in the future.

## Key Features

**Secure Authentication & Role-Based Access:** A foundational system for user login and navigation.

**Comprehensive Appointment Management:** A full suite of tools for booking, viewing, and canceling appointments.

**Data Persistence:** All user, profile, and appointment data is stored permanently in a local database.

**Administrative Tools:** Functions for managing doctor accounts and viewing system-wide reports.

# 2. Project Design & Architecture

Architectural Pattern: The project is designed with a layered architecture to promote modularity and separation of concerns. This approach ensures that each component has a single, well-defined responsibility, making the codebase easier to manage, debug, and expand.

## Component Breakdown

**Model Layer:** Contains POJOs (User, Patient, Doctor, Appointment, AvailabilitySlot) that encapsulate data and business logic.

**Database Layer:** DatabaseManager.java acts as the data access layer, performing all CRUD operations using JDBC and Prepared Statements.

**App Layer:** HospitalManagementApp.java handles user input, console UI, and orchestrates data flow between layers.

## Database Schema

|  |  |
| --- | --- |
| Table | Schema |
| USERS | userId (PK), username (UNIQUE), password, role |
| PATIENTS | patientId (PK), name, mobileNumber, email, dateOfBirth, profilePicture, userId (FK) |
| DOCTORS | doctorId (PK), name, specialty, mobileNumber, email, profilePicture, userId (FK) |
| AVAILABILITY\_SLOTS | slotId (PK), doctorId (FK), date, startTime, endTime, isBooked |
| APPOINTMENTS | appointmentId (PK), patientId (FK), doctorId (FK), date, time, status, slotId (FK) |

# 3. Implemented Modules & Features

## Database Persistence Layer (DatabaseManager.java)

Functions: Provides robust methods for creating/managing tables and CRUD operations like saveUser(), savePatient(), saveDoctor().

Implementation Highlights: Uses JDBC API for database interaction, with transactions ensuring data integrity.

## Patient Module

Purpose: To manage patient-centric tasks and interactions.

**Login & Registration:** Allows secure login for existing patients and registration of new accounts.

**Book Appointment:** Patients can view available doctors and slots, then book appointments. The slot is updated as booked.

**View My Appointments:** Patients can see all their scheduled appointments with doctor, date, and time details.

**Cancel My Appointment:** Patients can cancel an appointment, updating its status and freeing up the slot.

**Edit My Profile:** Patients can update personal details like contact information.

## Doctor Module

Purpose: To allow doctors to manage availability and appointments.

**Doctor Check-in:** Simulates doctor's presence and availability.

**Set My Availability:** Doctors add new availability slots, made available to patients.

**View My Appointments:** Doctors can view scheduled appointments including patient details.

**Cancel Appointment:** Doctors can cancel an appointment, updating the slot availability.

**Edit My Profile:** Doctors can update their specialty and contact details.

## Admin Module

Purpose: To provide administrative control over doctors and system data.

**Add Doctor Account:** Admin can create new doctor accounts and profiles.

**Remove Doctor Account:** Admin can delete doctor accounts and profiles.

**View All Appointments Report:** Admin can retrieve and view every appointment in the system.

# 4. Conclusion & Future Scope

Summary of Accomplishments: Working HMS prototype with persistent SQLite database, OOP design, and modular architecture.

Future Work:

**Graphical User Interface (GUI):** Replace console UI with JavaFX/Swing for better UX.

**Web-based Interface:** Develop a browser-accessible version.

**Advanced Scheduling:** Support recurring appointments and automatic slot generation.

**Notifications System:** Add email/SMS alerts for appointments.

**Advanced Reporting:** Generate exportable reports in PDF/Excel formats.