Hospital Management System

Abstract

The Hospital Management System (HMS) is a web-based application designed to manage the day-to-day operations of hospitals. The system provides secure login with role-based dashboards for administrators, doctors, receptionists, and patients. It supports the management of patients, doctors, appointments, departments, billing, pharmacy, laboratory, and more using 15 relational database tables. The backend is implemented in Java (Spring Boot) with Spring Security for authentication, Spring Data JPA for database operations, and MySQL as the relational database. The frontend is designed with HTML, CSS, and JavaScript, featuring a user-friendly interface with a hospital-themed background image. The system provides CRUD (Create, Read, Update, Delete) functionality, scheduling, and data integrity, ensuring a streamlined hospital workflow.

Keywords

Hospital Management System, Spring Boot, Java, MySQL, Web Application, Role-Based Access, Healthcare Software

I. Introduction

Hospitals require efficient management systems to ensure proper handling of patients, doctors, appointments, billing, and administrative records. Traditional manual systems often result in errors, data duplication, and inefficiency. This project proposes a Hospital Management System (HMS) that digitizes these operations using a secure and scalable software solution. The application leverages Java Spring Boot for backend logic, MySQL for database storage, and HTML/CSS/JavaScript for the user interface. The solution is designed to provide reliability, efficiency, and accessibility to stakeholders.

II. Objectives

1. To develop a role-based hospital management system for different stakeholders.
2. To provide secure login and authentication for admin, doctor, receptionist, and patient.
3. To manage hospital operations including appointments, billing, pharmacy, laboratory, and room allocation.
4. To ensure data integrity and real-time availability using MySQL.
5. To design a user-friendly interface with responsive UI.

III. System Architecture

A. Tech Stack

- Backend: Java 17, Spring Boot 3.x
- Frontend: HTML, CSS, JavaScript
- Database: MySQL (15 tables)
- Security: Spring Security (form-based login)
- Build Tool: Maven

B. Architecture Diagram

[Frontend (HTML/CSS/JS)] <--> [Spring Boot REST API] <--> [MySQL Database]

IV. Database Design

Main Tables (15 total):
1. patients (patient_id, name, age, gender, phone, email, address)
2. doctors (doctor_id, name, specialization, phone, email, department_id)
3. appointments (appointment_id, patient_id, doctor_id, date, time, status)
4. departments (department_id, name, head_doctor_id)
5. rooms (room_id, room_number, type, status, patient_id)
6. staff (staff_id, name, role, department_id)
7. users (user_id, username, password, role)
8. billing (bill_id, patient_id, amount, date, status)
9. medicines (medicine_id, name, type, stock, price)
10. pharmacy_sales (sale_id, patient_id, medicine_id, quantity, date)
11. lab_tests (test_id, patient_id, test_name, result, date)
12. nurses (nurse_id, name, phone, department_id)
13. receptionists (receptionist_id, name, phone, shift)
14. emergency_cases (case_id, patient_id, doctor_id, description, date)
15. medical_history (history_id, patient_id, description, treatment, date)

V. Modules

- Authentication & Role Management
- Secure login using Spring Security
- Role-based dashboards (admin/doctor/patient/receptionist)

- Patient Management
- CRUD operations for patients
- Medical history records

- Doctor Management
- CRUD operations for doctors
- Assigning to departments

- Appointment Management
- Patient can request appointments
- Doctors/receptionists can approve or reschedule

- Billing & Pharmacy
- Generate patient bills
- Manage medicines and pharmacy sales

- Laboratory
- Record and manage lab tests
- Store results linked to patients

- Room & Emergency Management
- Room allocation to patients
- Emergency case registration

VI. Implementation

- Backend: REST API using Spring Boot controllers and services.
- Database: MySQL schema with 15 interrelated tables, handled via Spring Data JPA.
- Frontend: HTML pages with hospital background, styled using CSS, and dynamic content loaded with JavaScript (AJAX fetch API).
- Security: Spring Security with password encryption and role-based authorization.

VII. Results & Discussion

The system was tested with sample hospital data. It successfully performed operations such as:
- Admin creating doctor profiles
- Patients booking appointments
- Receptionists scheduling appointments and generating bills
- Doctors updating patient medical records
- Pharmacy managing medicines and sales
The system improved hospital workflow efficiency, reduced errors, and ensured real-time access to information.

VIII. Conclusion

The proposed Hospital Management System provides a robust, scalable, and user-friendly solution for managing hospital operations. It ensures role-based access, maintains data integrity, and supports essential hospital workflows. Future enhancements may include integration of cloud hosting, mobile application, and AI-based patient health prediction systems.

References

1. Craig Walls, Spring in Action, 6th Edition, Manning Publications, 2022.
2. MySQL Documentation: https://dev.mysql.com/doc/
3. Spring Boot Reference Guide: https://spring.io/projects/spring-boot