HOSPITAL MANAGEMENT SYSTEM

A MINI-PROJECT REPORT

Submitted by

Aadithya Rajasekaran 240701001

Sachin Sundar 240701619

in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **"HOSPITAL MANAGEMENTSYSTEM"** is the Bonafide work of **"AadithyaRajasekaran, Sachin Sundar"** who carried out the project work under my supervision.

SIGNATURE

Adhithyan S

ASSISTANT PROFESSOR SG

Dept. of Computer Science and Eng,

Rajalakshmi Engineering College

Chennai

This mini project report is submitted for the viva voce examination to be held on _____

INTERNAL EXAMINER EXTERNAL EXAMINER

ABSTRACT

The Hospital Management System is a full-stack Java web application that manages patients, doctors, appointments and user authentication using Servlets, JSP, JDBC and MySQL. The system demonstrates CRUD operations, MVC architecture, reusable components, session-based authentication, JSP-based frontend design, and relational database integration. Features include patient registration, doctor management, appointment booking, login module, secure session handling, and dynamic dashboards. The system closely resembles real-world small-scale hospital workflow and is ideal for academic full-stack Java projects.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M. THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head of The Department **Dr. E.M. MALATHY** and our Deputy Head of The Department **Dr. J. MANORANJINI** for being ever supporting force during our project work.

We also extend our sincere and hearty thanks to our internal guide **Adhithyan S**, for his valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. AADITHYA RAJASEKARAN
2. SACHIN SUNDAR

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

This project "Hospital Management System" is designed to computerize and streamline the management of patient records, doctor information and appointment scheduling using modern web technologies. The application follows the MVC architectural pattern and uses Java Servlets for backend logic, JSP for UI and MySQL for persistent storage.

1.2 SCOPE OF THE WORK

The scope of this Hospital Management System includes the following core modules and functionalities:

• Patient Management Module

Enables secure creation, viewing, updating, and deletion of patient records. Provides structured storage for name, age, gender and enables efficient retrieval for administrative and medical purposes.

• Doctor Management Module

Allows administrators to manage doctor information including specialization, availability, and doctor listings. Supports all CRUD operations ensuring up-to-date resource information.

• Appointment Scheduling Module

Facilitates booking, listing, and canceling appointments by linking patients with doctors through a validated scheduling system. Ensures foreign key integrity between patient and doctor records through MySQL constraints.

• Authentication Module

Implements secure login functionality for users and administrators. Supports session-based access control using HttpSession to prevent unauthorized access to restricted pages.

• Database Integration

Provides stable and efficient interaction with MySQL using JDBC. Handles all backend operations such as retrieval, insertion, updating, and deletion of records through a dedicated DatabaseHelper class.

• Web-Based User Interface

Implements the UI using JSP pages integrated with HTML and CSS. Allows users to interact with the system through intuitive forms and dynamic tables for patient, doctor, and appointment management.

1.3 PROBLEM STATEMENT

Hospitals that rely on traditional manual record-keeping often encounter challenges such as misplaced files, slow retrieval of information, scheduling conflicts, and poor coordination between administrative units. Managing patient records, doctor details, and appointments manually leads to inefficiency, errors, and reduced quality of service.

There is a clear need for a digital solution that automates hospital tasks, maintains accurate records, and supports quick decision-making.

This project addresses the problem by developing a web-based Hospital Management System using Java Servlets, JSP, and MySQL. It streamlines the operations of patient management, doctor management, and appointment scheduling, providing a secure, user-friendly, and easily maintainable software solution that mirrors real-world hospital administration.

1.4 AIM AND OBJECTIVES OF THE PROJECT

Aim:

To design and implement a fully functional Hospital Management System using Java (JSP + Servlets), JDBC, and MySQL that automates patient registration, doctor management, and appointment scheduling through a secure, efficient, web-based platform.

Objectives:

- ☑ To implement a secure login and authentication system for users and administrators
- ☑ To design modules that support patient registration, doctor information management, and appointment booking
- ☑ To develop JSP-based interfaces for dynamic form handling and real-time data display
- ☑ To perform all backend operations (CRUD) using JDBC with MySQL for reliable data storage
- ☑ To create a scalable MVC-based architecture that can be extended with modules like billing, prescriptions, and reports
- ☑ To simulate real-world hospital workflow through seamless integration between user interface and database operations

CHAPTER 2
SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Component	Detailed Specification
Processor	2.0 GHz Dual-Core or higher
RAM	Minimum 4 GB DDR3/DDR4
Storage	200 MB free disk space
Network	Stable Internet or LAN connection

2.2 SOFTWARE SPECIFICATIONS

Component	Detailed Specification
Operating System	Windows / macOS / Linux
Programming Language	Java (JDK 17 or later)
Server	Apache Tomcat 9.0 or later
Backend Framework	Java Servlets + JSP
Database	MySQL Server 8.0+
JDBC Driver	MySQL Connector/J
Development Tools	VS Code / IntelliJ / Eclipse
Database Utility	MySQL Workbench / phpMyAdmin

CHAPTER 3
MODULE DESCRIPTION

The system is structured into four major modules: Authentication Module, Patient Module, Doctor Module, and Appointment Module. Additionally, a Database Utility Layer manages all MySQL interactions.

3.1 PATIENT MANAGEMENT MODULE

- Allows administrators to add, view, edit, and delete patient records
- Stores patient name, age, gender, and unique ID
- Displays patient details dynamically through JSP
- Enables quick retrieval through JDBC-based database queries

3.2 DOCTOR MANAGEMENT MODULE

- Maintains doctor information such as name and specialization
- Supports full CRUD operations via Servlets
- Ensures updated listings for patient appointments
- Integrates with MySQL for persistent data management

3.3 APPOINTMENT SCHEDULING MODULE

- Allows booking appointments by selecting patient and doctor from dropdown lists
- Validates relationships using foreign key constraints
- Stores appointment dates and retrieves schedules in sorted order
- Prevents orphan records by enforcing database constraints
- Displays patient and doctor names instead of numeric IDs using SQL JOIN queries

3.4 AUTHENTICATION MODULE

- Implements secure login using username and password

- Uses HttpSession for access control and session management
- Redirects unauthorized users away from restricted pages
- Allows safe logout using session invalidation

3.5 DATABASE UTILITY MODULE

- DatabaseHelper.java manages all MySQL operations using JDBC
- Provides reusable methods for CRUD operations (patients, doctors, appointments)
- Implements proper exception handling and secure connection usage
- Loads com.mysql.cj.jdbc.Driver and manages connection pooling (optional)

CHAPTER 4

SAMPLE CODING

DB CONNECTION CODE:

```
// Database connection utility class
public class DatabaseHelper {
     private static final String URL =
 "jdbc:mysql://localhost:3306/hospital_db";
    privatestatic final String USER = "root";
    privatestatic final String PASSWORD = "password";

    static {
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
        }catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    publicstatic Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

APPOINTMENT.JAVA CODE:

```
package model;

public class Appointment {
    private int id;
    private int patientId;
    private int doctorId;
    private String appointmentDate;

    // New fields
    private String patientName;
```

```
    private String doctorName;

    public Appointment(int id, int patientId, int doctorId, String
appointmentDate) {
        this.id = id;
        this.patientId = patientId;
        this.doctorId = doctorId;
        this.appointmentDate = appointmentDate;
    }

    //Getters and setters for existing fields
    public int getId() { return id; }
    public int getPatientId() { return patientId; }
    public int getDoctorId() { return doctorId; }
    public String getAppointmentDate() { return appointmentDate; }

    public void setId(int id) { this.id = id; }
    public void setPatientId(int patientId) { this.patientId =
patientId; }
    public void setDoctorId(int doctorId) { this.doctorId = doctorId; }
    public void setAppointmentDate(String appointmentDate) {
        this.appointmentDate = appointmentDate;
    }

    //NEW: patientName & doctorName accessors
    public String getPatientName() { return patientName; }
    public void setPatientName(String patientName) {
        this.patientName = patientName;
    }

    public String getDoctorName() { return doctorName; }
    public void setDoctorName(String doctorName) {
        this.doctorName = doctorName;
    }
}
```

CHAPTER 5
SCREENSHOTS

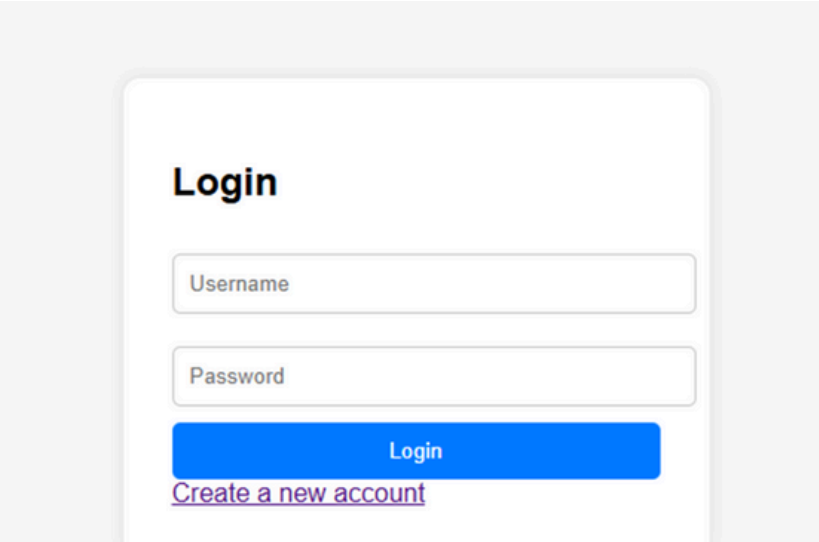

Fig 5.1 LOGIN PAGE

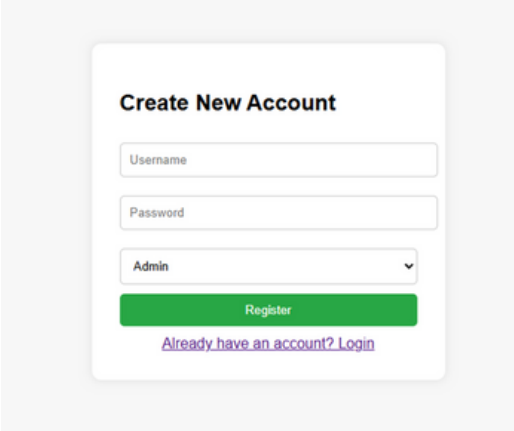

Fig 5.2 CREATE NEW ACCOUNT

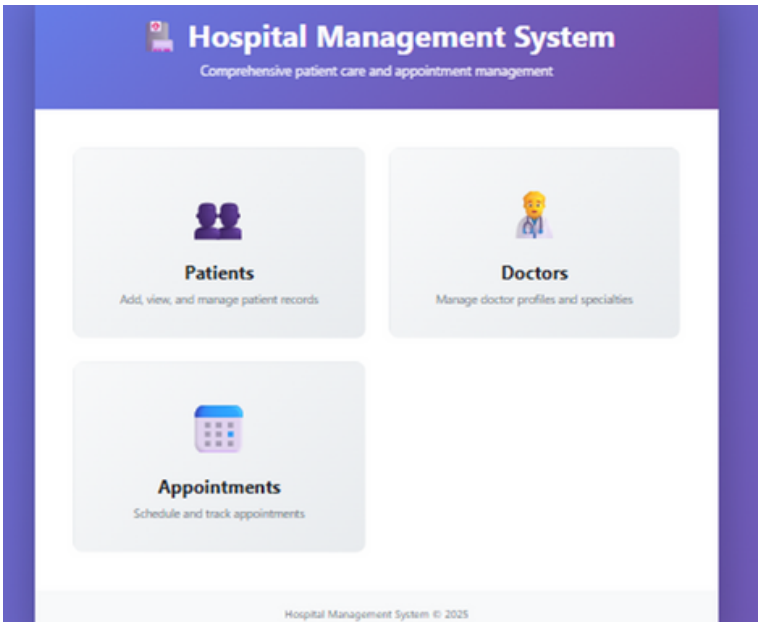

Fig 5.3 DASHBOARD

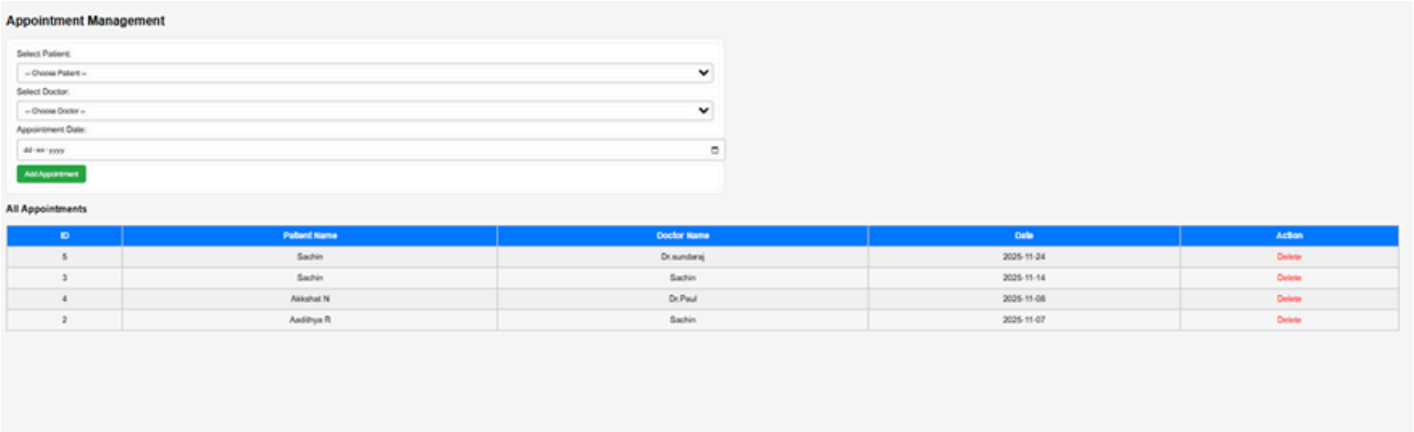

Fig 5.4 APPOINTMENTS

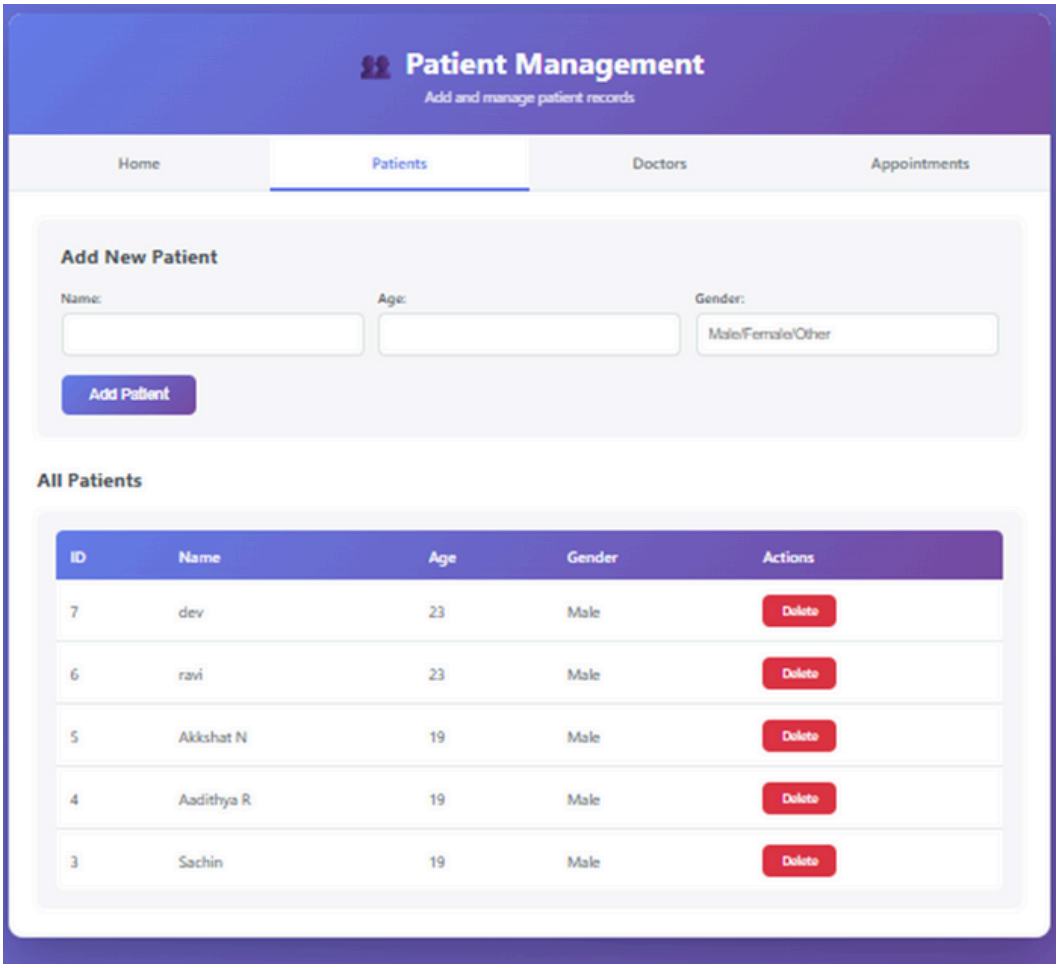

Fig 5.5 PATIENTS

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION

The Hospital Management System developed using Java Servlets, JSP, JDBC, and MySQL successfully fulfills the essential requirements of a small-scale hospital administration environment.

The system provides three key operational modules — Patient Management, Doctor Management, and Appointment Scheduling, along with secure authentication and a structured dashboard interface.

By integrating a JSP-based front-end with a robust MySQL-backed JDBC layer, the project ensures smooth interaction between user actions and backend operations. Dynamic data retrieval, form-based inputs, and CRUD functionalities contribute to accurate record handling and efficient workflow management.

The project demonstrates a clear understanding of web application architecture, session management, MVC pattern, database connectivity, and modular full-stack development. It is highly suitable for academic purposes and serves as a practical foundation for developing real-world healthcare applications.

FUTURE ENHANCEMENTS

To expand the scope and functionality of the current Hospital Management System, the following enhancements are recommended:

1. Billing & Payment Management

Add modules for generating bills, maintaining patient invoices, and processing payments.

2. Electronic Medical Records (EMR)

Introduce features for storing prescriptions, medical history, lab reports, and patient treatment logs.

3. Role-Based Access System

Enhance the authentication module by adding multiple roles such as Admin, Doctor, Nurse, and Receptionist, each with customized access.

4. Email & SMS Notifications

Send appointment reminders, doctor availability alerts, and confirmation messages to patients.

5. Advanced Appointment Scheduling

Implement calendar-based scheduling, time-slot selection, and auto-conflict detection to avoid overlapping appointments.

6. Reporting and Analytics

Provide dashboards for patient statistics, doctor workload, appointment trends, and database insights using SQL aggregation.

7. Cloud Deployment

Migrate the system to cloud-based platforms like AWS, Azure, or Firebase for remote access and improved scalability.

8. Frontend Modernization

Upgrade the JSP-based UI to modern frameworks such as React, Angular, or Bootstrap for a responsive, mobile-friendly interface.

CHAPTER 7

REFERENCES

1. **Oracle Java Servlet Documentation**
   https://docs.oracle.com/javaee/7/api/javax/servlet/package-summary.html

2. **JDBC Tutorial – Oracle**
   https://docs.oracle.com/javase/tutorial/jdbc/

3. **MySQL 8.0 Reference Manual**
   https://dev.mysql.com/doc/refman/8.0/en/

4. **MySQL Connector/J Developer Guide**
   https://dev.mysql.com/doc/connector-j/8.0/en/

5. **Apache Tomcat Documentation**
   https://tomcat.apache.org/tomcat-9.0-doc/