HOSPITAL APPOINTMENT MANAGEMENT

1. INTRODUCTION

The Hospital Appointment Management System is a simple, menu-driven, command-line application developed using Core Java, JDBC, and MySQL. The main goal of this project is to computerize the process of booking and managing hospital appointments in an efficient and organized manner.

In traditional hospital systems, appointment booking is often done manually, which can lead to long waiting times, scheduling conflicts, and poor record management. This project addresses these issues by providing a digital platform where appointment details are stored securely in a database and can be accessed whenever required.

The application runs in the Eclipse IDE and interacts with a MySQL database using JDBC. It follows a layered architecture, making the code modular, readable, and easy to maintain.

2. OBJECTIVES OF THE PROJECT

The objectives of the Hospital Appointment Management System are:

- To allow patients to book appointments easily through a simple menu-driven interface
- To store appointment details permanently in a database
- To enable administrators to view all appointment records
- To search appointments based on doctor name
- To cancel appointments when required
- To reduce manual effort and paperwork
- To provide better organization and management of hospital appointment data.

3. SCOPE OF THE PROJECT

This project focuses on basic hospital appointment management. It is suitable for small clinics or hospitals that require a simple appointment-tracking system.

- The scope of this system includes:
- Booking patient appointments
- Viewing appointment details

- Searching appointments by doctor
- Cancelling appointments

The project does not include advanced features such as online payments, patient login authentication, or doctor scheduling dashboards. However, these features can be added in future enhancements.

4. TECHNOLOGY STACK

The following technologies are used in this project:

4.1 Core Java

Java is used to build the application logic, menu handling, and class structure.

4.2 JDBC (Java Database Connectivity)

JDBC is used to connect the Java application with the MySQL database and perform database operations such as insert, select, and delete.

4.3 MySQL

MySQL is used as the backend database to store appointment records permanently.

4.4 Eclipse IDE

Eclipse is used to write, compile, and run the Java program.

5. SYSTEM ARCHITECTURE

The Hospital Appointment Management System follows a layered architecture, which improves code clarity and maintainability.

Layers used:

1. DTO (Data Transfer Object)

2. DAO (Data Access Object)

3. Service Layer

4. Main Class (User Interface)

6. MODULE DESCRIPTION

6.1 AppointmentDTO (Data Transfer Object)

The AppointmentDTO class is used to transfer appointment data between different layers of the application. It contains variables such as patient name, doctor name, department, appointment date, and appointment time.

This class only holds data and does not contain any business logic.

6.2 AppointmentDAO Interface

The DAO interface defines the database operations required by the system. It acts as a contract for database access methods such as:

- Add appointment
- View appointments
- Search appointments by doctor
- Cancel appointment

6.3 AppointmentDAOImpl

The AppointmentDAOImpl class implements the AppointmentDAO interface. It contains actual JDBC code to interact with the MySQL database.

This class is responsible for:

- Establishing database connection
- Executing SQL queries
- Handling database results

6.4 AppointmentService

The AppointmentService class acts as a bridge between the main class and the DAO layer. It contains business logic and calls DAO methods as required.

This layer ensures separation between user input handling and database operations.

6.5 HospitalMain (Main Class)

The HospitalMain class contains the main() method. It provides a menu-driven command-line interface where users can:

- Book an appointment
- View all appointments
- Search appointments by doctor
- Cancel an appointment
- Exit the application

7. DATABASE DESIGN

Database Name:

hospital_db

Table Name:

appointments

Table Structure:

Column Name	Data Type
id	INT (Primary Key, Auto Increment)
patient_name	VARCHAR
doctor_name	VARCHAR
department	VARCHAR
appointment_date	DATE
appointment_time	TIME

The database stores all appointment details entered from the Java application. Any operation performed in Eclipse reflects directly in the MySQL database.

8. WORKING OF THE SYSTEM

1. The user runs the program in Eclipse.

2. A menu is displayed on the console.

3. The patient enters appointment details such as name, doctor, date, and time.

4. The data is sent to the service layer.

5. The service layer calls DAO methods.

6. The DAO layer inserts the data into MySQL using JDBC.

7. Administrators can view, search, or cancel appointments.

8. All changes are reflected in the database.

9. ADMINISTRATIVE ROLE

- Administrators are responsible for:
- Viewing all booked appointments
- Searching appointments by doctor
- Cancelling appointments if the doctor is unavailable

If an appointment is cancelled, the patient must book another appointment with a different time or doctor.

10. ADVANTAGES OF THE SYSTEM

- Simple and easy to use
- Reduces manual work
- Centralized database storage
- Faster appointment management
- Easy to maintain due to layered architecture

11. LIMITATIONS OF THE SYSTEM

- No automatic doctor availability check
- No patient notification system
- No login authentication
- Console-based interface only

12. FUTURE ENHANCEMENTS

- The system can be enhanced by:
- Adding doctor availability scheduling
- Sending confirmation or cancellation notifications
- Creating a graphical user interface (GUI)
- Adding patient and admin login modules
- Implementing online appointment booking

13. CONCLUSION

The Hospital Appointment Management System is a simple and efficient solution for managing hospital appointments using Java and MySQL. It demonstrates the practical use of JDBC connectivity and layered architecture. This project is suitable for academic purposes and provides a strong foundation for building more advanced hospital management systems in the future.