CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

PROJECT REPORT

ON

Hospital Management System

Program Name: BCA

Subject Name/Code: JAVA LAB

(22CAP-352)

Submitted by:	Submitted to:
Name: Jishanu Pandey	**Name: Suman Acharya**
UID:22BCA10020	**Designation: Asst.prof**
Section:22BCA-10A	

ABSTRACT

This report outlines the development , a foundational Hospital Administration System created using Java and MySQL. The system aims to simplify routine hospital tasks such as managing patient data, reviewing doctor information, and scheduling appointments.
Featuring a console-based interface, itensures secure data transactions through the use of prepared SQL statements, providing a modular, scalable, and efficient digital management solution.

The application adheres to Object-Oriented Programming (OOP) principles, allowing easy maintenance and future scalability. The project is organized into distinct classes managing patients, doctors, and appointment scheduling, each with specific responsibilities. The relational database is structured with primary keys and validation mechanisms to maintain data integrity and minimize redundancy.

This system facilitates quick and accurate task execution for hospital staff, reducing paperwork and improving data handling. Future enhancements could include a graphical user interface (GUI), prescription modules, and integrated billing systems, paving the way toward a fully operational hospital management suite.

INTRODUCTION

In today's digital era, healthcare institutions increasingly depend on technology to streamline their operational workflow. Accurate management of patient records, doctor schedules, and appointment logistics is crucial for delivering high-quality care. Manual processes are often error-prone and inefficient, leading to compromised healthcare services.

To address these challenges, **it** was developed, utilizing Java as the programming language and MySQL as the database backend. The system offers a lightweight and efficient platform for automating essential hospital operations through a simple console-based interface.

Key functionalities include:

- Registering new patients with detailed profiles
- Viewing comprehensive patient and doctor records
- Scheduling appointments while ensuring doctor availability

The system integrates Object-Oriented Programming (OOP) for clean, maintainable code and connects to the database using Java Database Connectivity (JDBC), ensuring seamless real-time data management.
It is designed with future expansions in mind, including features like electronic medical records, billing systems, and data reporting, ultimately aiming to enhance operational efficiency and patient care outcomes.

DEVELOPMENT TECHNIQUES

The construction incorporates advanced programming and database management techniques, as outlined below:

1. Object-Oriented Programming (OOP) in Java

- **Classes and Objects:** Dedicated classes such as Patient, Doctor, and HospitalManagementSystem handle specific operations.
- **Encapsulation:** Logical grouping of data and methods enhances readability and security.
- **Modularity:** Independent class functionality ensures easy maintenance and scalability.

2. Database Integration Using JDBC

- **Prepared Statements:** Protects against SQL injection and optimizes database queries.
- **SQL Operations:** Core operations like INSERT, SELECT, and COUNT are used for database interaction.
- **Relational Database Structure:** Patients, doctors, and appointments are organized into separate tables.

3. User Interaction

- **Console-Based Input:** Utilizes the Scanner class for user input.
- **Input Parameters:** Includes patient demographics, appointment dates, and ID validations.

4. Validation and Scheduling Logic

- **Data Validation:** Ensures that only valid Patient and Doctor IDs are used.
- **Availability Checks:** Confirms doctor availability before confirming appointments.

PROGRAM CODE

```
java

CopyEdit

package HospitalManagementSystem;

import java.sql.*;

import java.util.Scanner;


public class HospitalManagementSystem {

    private static final String url = "jdbc:mysql://localhost:3306/hospital";

    private static final String username = "root";

    private static final String password = "Jishanu@123";


    public static void main(String[] args) {

        try {

            Class.forName("com.mysql.cj.jdbc.Driver");

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        }


        Scanner scanner = new Scanner(System.in);

        try (Connection connection = DriverManager.getConnection(url, username, password)) {

            Patient patient = new Patient(connection, scanner);

            Doctor doctor = new Doctor(connection);


            while (true) {

                System.out.println("=== MEDI-TRACK: HOSPITAL ADMINISTRATION SYSTEM
===");

                System.out.println("1. Add Patient");

                System.out.println("2. View Patients");
                System.out.println("3. View Doctors");

                System.out.println("4. Book Appointment");

                System.out.println("5. Exit");

                System.out.print("Enter your choice: ");

                int choice = scanner.nextInt();


                switch (choice) {

                    case 1:

                        patient.addPatient();

                        System.out.println();

                        break;

                    case 2:

                        patient.viewPatients();

                        System.out.println();

                        break;

                    case 3:

                        doctor.viewDoctors();

                        System.out.println();

                        break;

                    case 4:

                        bookAppointment(patient, doctor, connection, scanner);

                        System.out.println();

                        break;

                    case 5:

                        System.out.println("Thank you for using MediTrack!");

                        return;

                    default:

                        System.out.println("Invalid choice! Please try again.");

                        break;

                }

            }
```

```
} catch (SQLException e) {
            e.printStackTrace();
        }
    }


    public static void bookAppointment(Patient patient, Doctor doctor, Connection connection,
Scanner scanner) {
        System.out.print("Enter Patient ID: ");
        int patientId = scanner.nextInt();
        System.out.print("Enter Doctor ID: ");
        int doctorId = scanner.nextInt();
        System.out.print("Enter Appointment Date (YYYY-MM-DD): ");
        String appointmentDate = scanner.next();


        if (patient.getPatientById(patientId) && doctor.getDoctorById(doctorId)) {
            if (checkDoctorAvailability(doctorId, appointmentDate, connection)) {
                String appointmentQuery = "INSERT INTO appointments(patients_id, doctors_id,
Appointment_date) VALUES(?, ?, ?)";
                try (PreparedStatement preparedStatement =
connection.prepareStatement(appointmentQuery)) {
                    preparedStatement.setInt(1, patientId);
                    preparedStatement.setInt(2, doctorId);
                    preparedStatement.setString(3, appointmentDate);


                    int rowsAffected = preparedStatement.executeUpdate();
                    if (rowsAffected > 0) {
                        System.out.println("Appointment successfully booked!");
                    } else {
                        System.out.println("Failed to book appointment.");
                    }
                } catch (SQLException e) {
```

```
                e.printStackTrace();
            }
        } else {
            System.out.println("Doctor unavailable on the selected date.");
        }
    } else {
        System.out.println("Invalid Doctor ID or Patient ID.");
    }
}


public static boolean checkDoctorAvailability(int doctorId, String appointmentDate, Connection
connection) {
    String query = "SELECT * FROM appointments WHERE doctors_id = ? AND
Appointment_date = ?";
    try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {
        preparedStatement.setInt(1, doctorId);
        preparedStatement.setString(2, appointmentDate);
        ResultSet resultSet = preparedStatement.executeQuery();
        return !resultSet.next();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
}
```

RESULTS AND ANALYSIS

The system was successfully built and tested, providing an intuitive, console-based interface with a robust backend. Testing focused on core functionalities, validating database interaction, and handling exceptions effectively.

Functionality	**Status**	**Remarks**
Add Patient	Working ✅	Patient data successfully stored
View Patients	Working ✅	Displays patient information cleanly
View Doctors	Working ✅	Lists doctors and their specialties
Book Appointment	Working ✅	Validates and schedules appointments
Doctor Availability Check	Working ✅	Prevents duplicate bookings on same date
Error and Exception Handling	Working ✅	Proper SQL and user input error management

SUMMARY

The development of **A Basic Hospital Administration System** has significantly streamlined common hospital operations including patient registration, doctor management, and appointment scheduling. Built with Java and backed by MySQL, this project exemplifies modular software development and secure database management practices.

Key highlights include modularity through separate classes, secure database operations using PreparedStatement, and real-time validation of user input and doctor availability.
The system not only reduces manual errors but also improves data management and retrieval efficiency.

Future Enhancements:

- Development of a GUI-based version using JavaFX or Swing
- Integration of billing, prescription, and reporting modules
- Implementation of role-based access and authentication
- Cloud deployment for scalability and remote access