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:

Name: Ayush Kumar Chaudhary

UID:22BCA10216
Section:22BCA-10A

Submitted to:

Name: Suman Acharya

Designation: Asst.prof

ABSTRACT

This project presents a **Hospital Management System** developed using **Java** and **MySQL**, aimed at simplifying hospital operations such as managing patient records, viewing doctor details, and booking appointments. The system features a **console-based interface** for user interaction and ensures data integrity through the use of **prepared SQL statements**. The objective is to provide a **user-friendly, modular, and efficient** way to handle hospital operations digitally.

The system is designed using **Object-Oriented Programming (OOP)** principles to ensure scalability and maintainability. Core components of the system include classes for handling patients, doctors, and appointment logic, each encapsulated for focused responsibilities. The database backend is optimized with primary keys and validation checks to prevent data redundancy and maintain consistency.

The HMS enables hospital staff to perform essential tasks quickly and accurately, reducing paperwork and improving record-keeping. With potential for future upgrades such as graphical user interface (GUI), prescription management, and billing modules, this project serves as a solid foundation for a fully integrated hospital management solution.

Introduction

Healthcare institutions are increasingly relying on technology to manage day-to-day operations efficiently. One of the critical needs in a hospital setup is a reliable and organized system to manage patient data, doctor information, and appointments. Manual handling of these tasks often leads to errors, delays, and data loss, which can directly impact the quality of patient care.

To address these challenges, the **Hospital Management System (HMS)** was developed using **Java** as the programming language and **MySQL** as the backend database. This project aims to offer a lightweight yet powerful solution that automates common hospital administrative tasks through a simple, console-based interface.

The system allows users to:

- **Add new patients** with relevant details
- **View patient and doctor records**
- **Book appointments** by checking doctor availability

It follows the principles of **Object-Oriented Programming (OOP)** for clean and maintainable code and uses **JDBC** for database connectivity. This ensures seamless interaction between the Java application and the MySQL database, enabling real-time data storage and retrieval.

The project is designed with future scalability in mind and can be extended to include features like electronic medical records, billing, and reporting modules. The overall goal is to create a digital framework that not only improves hospital workflow but also contributes to better healthcare outcomes.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Technique

The development of the **Hospital Management System (HMS)** involves the integration of **object-oriented programming** with **relational database management** to provide a structured and efficient application. Below are the key techniques and tools used in this project:

1. Object-Oriented Programming (OOP) in Java

The system is built using **Java**, utilizing OOP concepts such as:

- **Classes and Objects:** Separate classes like Patient, Doctor, and HospitalManagementSystem encapsulate specific responsibilities.
- **Encapsulation:** Data and methods are grouped logically, improving code readability and security.
- **Modularity:** Each class functions independently, promoting better maintenance and scalability.

2. Database Connectivity Using JDBC

The system connects to a **MySQL** database using **Java Database Connectivity (JDBC)**. Key components:

- **Prepared Statements:** Used to avoid SQL injection and ensure efficient execution.
- **SQL Queries:** Used to perform operations such as INSERT, SELECT, and COUNT.
- **Database Tables:** Tables like patients, doctors, and appointments store the data in a relational structure.

3. User Input Handling

The application interacts with users via the **Java Console (Scanner class)**, allowing inputs like:

- Patient name, age, gender
- Appointment date
- Doctor and patient ID validation

4. Validation and Availability Check

Before booking an appointment, the system:

- Validates if both the **Patient ID** and **Doctor ID** exist.
- Checks the **doctor's availability** for a specific date by querying the appointment table.

PROGRAMMING CODE

```
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 = "Ayush@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("HOSPITAL MANAGEMENT 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.println("Enter your choice: ");
                int choice = scanner.nextInt();

                switch(choice){
                    case 1:
                        // Add Patient
                        patient.addPatient();
                        System.out.println();
                        break;
                    case 2:
                        // View Patient
                        patient.viewPatients();
```

```
                System.out.println();
                break;
            case 3:
                // View Doctors
                doctor.viewDoctors();
                System.out.println();
                break;
            case 4:
                // Book Appointment
                bookAppointment(patient, doctor, connection, scanner);
                System.out.println();
                break;
            case 5:
                System.out.println("THANK YOU! FOR USING HOSPITAL MANAGEMENT SYSTEM!!");
                return;
            default:
                System.out.println("Enter valid choice!!!");
                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 Booked!");
                }else{
                   System.out.println("Failed to Book Appointment!");
                }
             } catch (SQLException e) {
                e.printStackTrace();
             }
          } else {
             System.out.println("Doctor not available on this date!!");
          }
       } else {
          System.out.println("Either doctor or patient doesn't exist!!!");
       }
    }


    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();  // true if no appointment found
       } catch (SQLException e) {
          e.printStackTrace();
       }
       return false;
    }


}
```

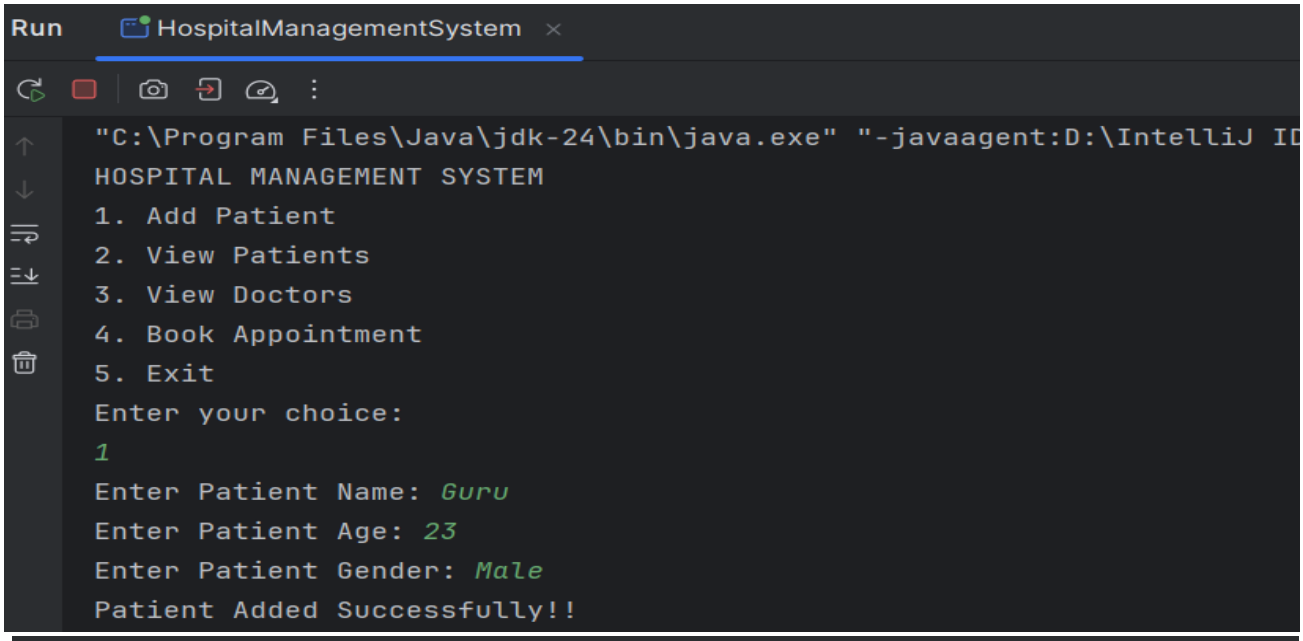
Result and Analysis

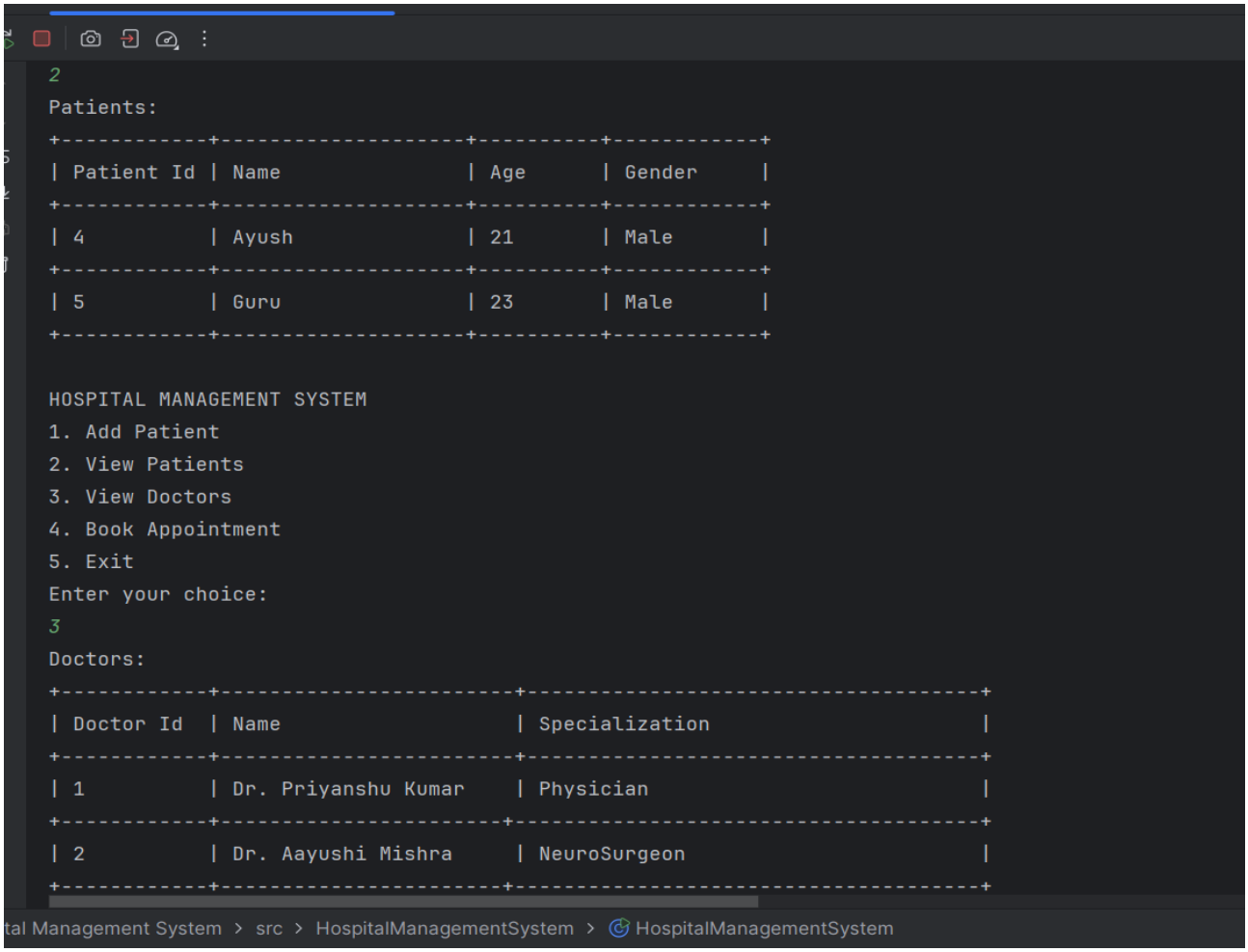
The **Hospital Management System** was successfully developed using Java with a Swing-based console interface and MySQL as the backend database. The application focuses on essential hospital operations such as adding/viewing patients and doctors, and booking appointments. The system was thoroughly tested for functionality, data integrity, and performance.

Functionality Testing

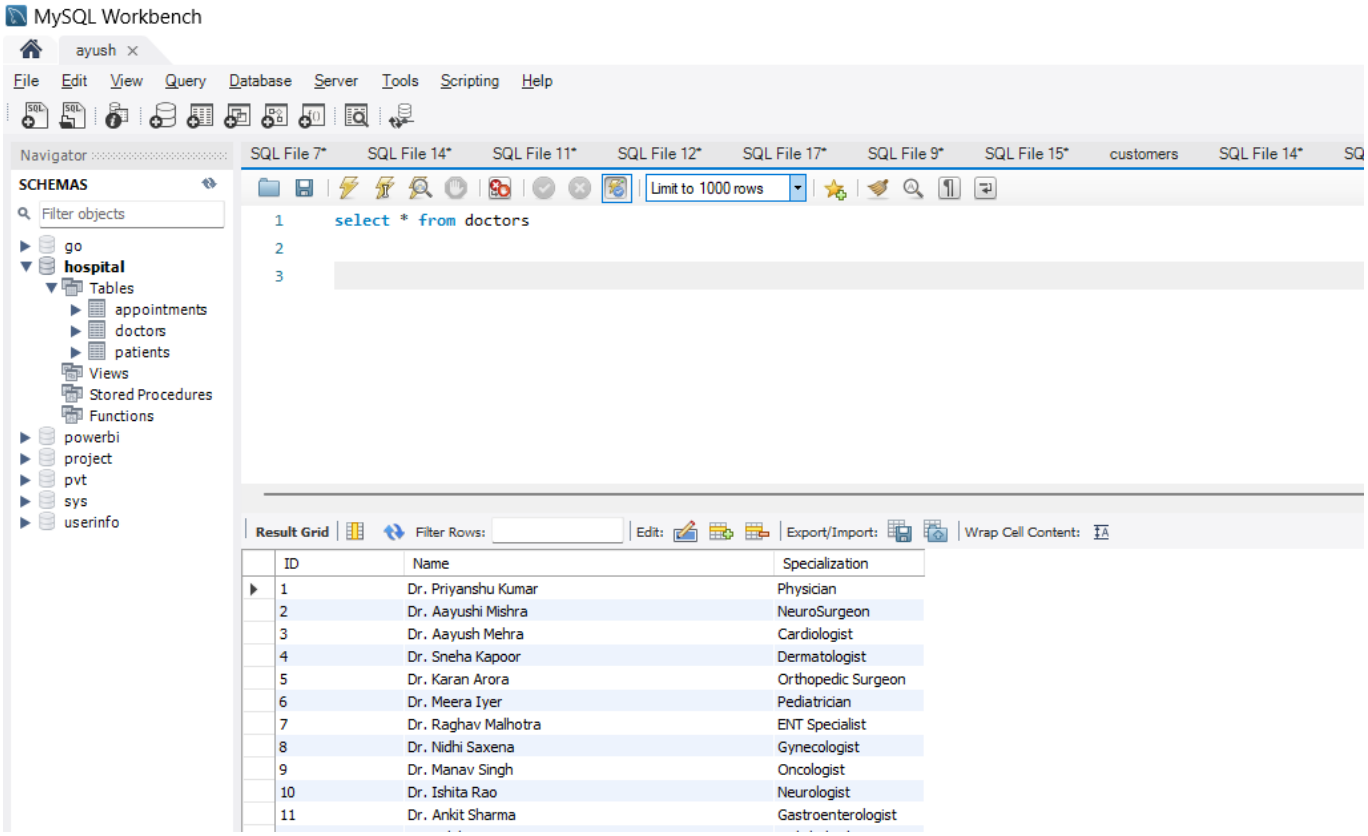
The system was tested across multiple functionalities to ensure proper performance and accurate data handling. The following core features were verified:

Functionality	Status	Remarks
Add Patient	Working ☑	Patient details inserted successfully
View Patients	Working ☑	Displays all patients in a formatted table
View Doctors	Working ☑	Lists all registered doctors and specialties
Book Appointment	Working ☑	Books appointment after validation
Doctor Availability	Working ☑	Prevents duplicate bookings for same date
Error Handling	Working ☑	SQL exceptions and input errors are managed


Run
HospitalManagementSystem
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:D:\IntelliJ ID
HOSPITAL MANAGEMENT SYSTEM
1. Add Patient
2. View Patients
3. View Doctors
4. Book Appointment
5. Exit
Enter your choice:
1
Enter Patient Name: Guru
Enter Patient Age: 23
Enter Patient Gender: Male
Patient Added Successfully!!


2
Patients:
Patient Id	Name	Age	Gender
4	Ayush	21	Male
5	Guru	23	Male
HOSPITAL MANAGEMENT SYSTEM
1. Add Patient
2. View Patients
3. View Doctors
4. Book Appointment
5. Exit
Enter your choice:
3
Doctors:
Doctor Id	Name	Specialization
1	Dr. Priyanshu Kumar	Physician
2	Dr. Aayushi Mishra	NeuroSurgeon
tal Management System > src > HospitalManagementSystem > HospitalManagementSystem

```
4
Enter Patient Id: 5
Enter Doctor Id: 52
Enter appointment date (YYYY-MM-DD): 2025-05-20
Appointment Booked!
```



ID	Name	Specialization
1	Dr. Priyanshu Kumar	Physician
2	Dr. Aayushi Mishra	NeuroSurgeon
3	Dr. Aayush Mehra	Cardiologist
4	Dr. Sneha Kapoor	Dermatologist
5	Dr. Karan Arora	Orthopedic Surgeon
6	Dr. Meera Iyer	Pediatrician
7	Dr. Raghav Malhotra	ENT Specialist
8	Dr. Nidhi Saxena	Gynecologist
9	Dr. Manav Singh	Oncologist
10	Dr. Ishita Rao	Neurologist
11	Dr. Ankit Sharma	Gastroenterologist

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

SUMMARY

The **Hospital Management System** project was developed to streamline and simplify hospital operations such as patient registration, doctor management, and appointment scheduling. This console-based application, built using **Java (Swing)** for the interface and **MySQL** as the backend database, effectively demonstrates core principles of software engineering and database-driven application development.

Throughout the development process, modular programming practices were followed. Each major functionality—adding/viewing patients, managing doctors, and handling appointments—was encapsulated in separate classes. Database interactions were securely managed using PreparedStatement to prevent SQL injection and ensure data integrity.

The project has shown positive results in terms of functionality, efficiency, and user experience. It allows hospital staff to manage daily activities smoothly without manual errors, thereby increasing productivity and accuracy. The use of a relational database makes it easier to store, retrieve, and analyze patient and appointment data reliably.

In the future, the system can be further enhanced by integrating a graphical user interface (GUI), user authentication, medical history modules, and billing features. It can also be deployed on a web or cloud platform to improve accessibility and scalability.