CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

PROJECT REPORT

ON

Digital Healthcare Administration System

Program Name: BCA

Subject Name/Code: JAVA LAB

(22CAP-352)

Submitted by:

Name: Adil Khan

UID: 22BCA10017
Section:22BCA-10A

Submitted to:

Name: Suman Acharya

Designation: Asst.prof

ABSTRACT

This project introduces a **Digital Healthcare Administration System** developed using **Java** for the front-end logic and **MySQL** as the database backend. The primary aim is to streamline essential hospital functions such as **patient record management**, **doctor information access**, and **appointment scheduling**. The system employs a console-based interface for user interaction and utilizes **prepared SQL statements** to ensure data security and integrity.

Built using **Object-Oriented Programming (OOP)** principles, the design promotes **modularity**, **scalability**, and ease of maintenance. The core structure comprises distinct classes dedicated to managing patients, doctors, and appointments, each with clearly defined responsibilities. The database is structured with **primary keys** and **validation checks** to eliminate redundancy and preserve data consistency.

The system enhances operational efficiency by enabling hospital staff to perform routine tasks with greater accuracy and reduced manual effort. It lays a robust foundation for future enhancements, including the addition of a **graphical user interface (GUI)**, **prescription tracking**, and **billing functionality**, moving towards a comprehensive digital healthcare management platform.

Introduction

In today's digital age, **healthcare systems** are increasingly adopting technology to streamline their operations and improve service quality. One of the critical components in hospital administration is the efficient management of **patient information**, **doctor profiles**, and **appointment scheduling**. Traditional manual methods often result in **errors**, **data loss**, and **delays**, ultimately affecting the quality of care provided to patients.

To address these challenges, a **Digital Healthcare Administration System** has been developed using **Java** for core programming and **MySQL** for backend data storage. This project aims to offer a simple yet robust solution that automates essential hospital tasks through a **console-based interface**, ensuring both ease of use and reliable performance.

Key features of the system include:

- Adding and managing detailed patient records
- Viewing doctor information and availability
- Booking appointments efficiently

The system architecture is based on **Object-Oriented Programming (OOP)** to promote modularity, scalability, and ease of maintenance. Integration with MySQL is achieved using **JDBC (Java Database Connectivity)**, ensuring real-time and secure communication between the application and the database.

Designed with future enhancements in mind, this system can be extended to support advanced features such as **electronic medical records**, **billing integration**, and **automated reporting**. Overall, this project contributes to creating a more **efficient**, **organized**, and **digital healthcare environment**.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Technique

The development of the **Digital Healthcare Administration System** combines **object-oriented programming** principles with **relational database management** to deliver a well-structured, maintainable, and efficient application. The key techniques and tools employed in this project are outlined below:

1. Object-Oriented Programming (OOP) in Java
The core logic of the system is developed using **Java**, leveraging fundamental OOP principles to ensure clarity, modularity, and scalability:

- **Classes and Objects**: The system is divided into distinct classes such as Patient, Doctor, and HospitalManagementSystem, each encapsulating specific functionalities.
- **Encapsulation**: Data and associated methods are logically grouped within each class to enhance **code security** and **readability**.
- **Modularity**: Independent class design allows for easier debugging, testing, and future enhancements.

2. Database Connectivity Using JDBC
Communication between the Java application and the **MySQL database** is handled through **Java Database Connectivity (JDBC)**:

- **Prepared Statements** are implemented to prevent **SQL injection** and optimize database performance.
- **SQL Queries** (e.g., INSERT, SELECT, COUNT) perform core data operations efficiently.
- Data is organized in relational **tables** such as patients, doctors, and appointments to maintain a normalized structure.

3. User Input Handling
The system accepts input from users via a **console interface** using Java's **Scanner class**, facilitating interactive data entry:

- Inputs include **patient details** (name, age, gender), **appointment date**, and **ID verification**.
- Input validation ensures accuracy and prevents incomplete or incorrect data entry.

4. Validation and Availability Check
To ensure the accuracy and integrity of appointment bookings, the system performs:

- **ID Validation**: Confirms whether both **Patient ID** and **Doctor ID** exist in the database.
- **Availability Check**: Queries the appointments table to verify the doctor's availability on the selected date.

These techniques collectively contribute to a **reliable**, **secure**, and **user-friendly** hospital management solution, with scope for future enhancements such as GUI implementation and advanced modules.

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("Digital Healthcare 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.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 **Digital Healthcare Administration 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.

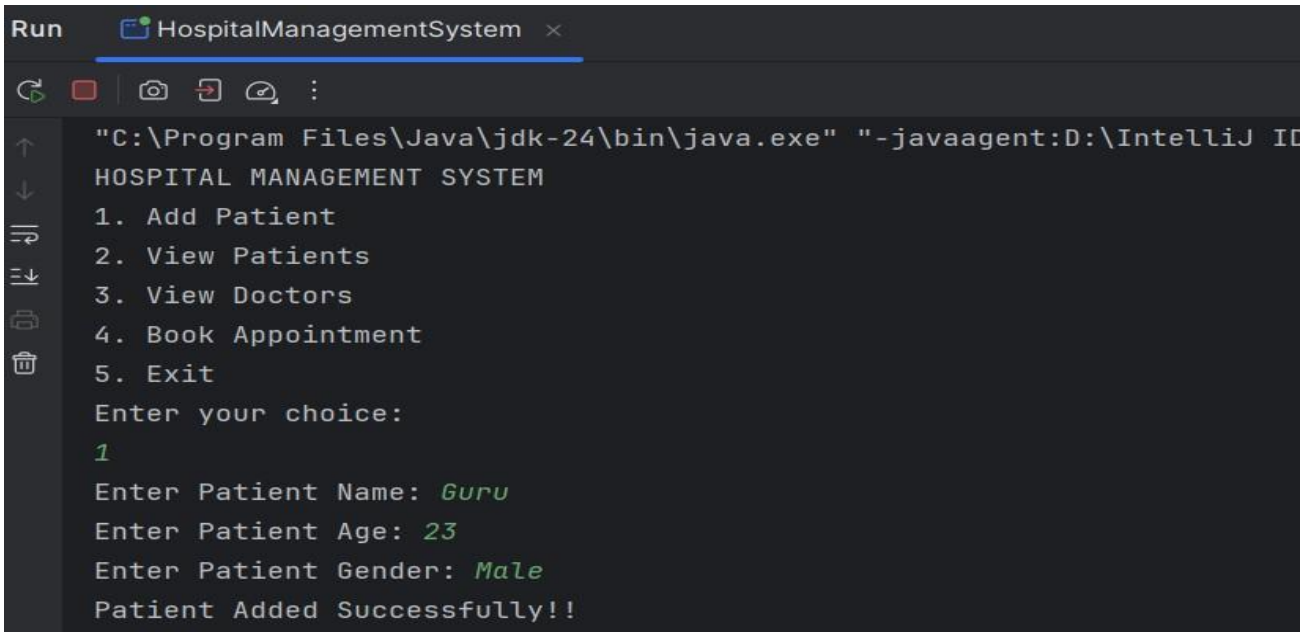
Functionality Testing

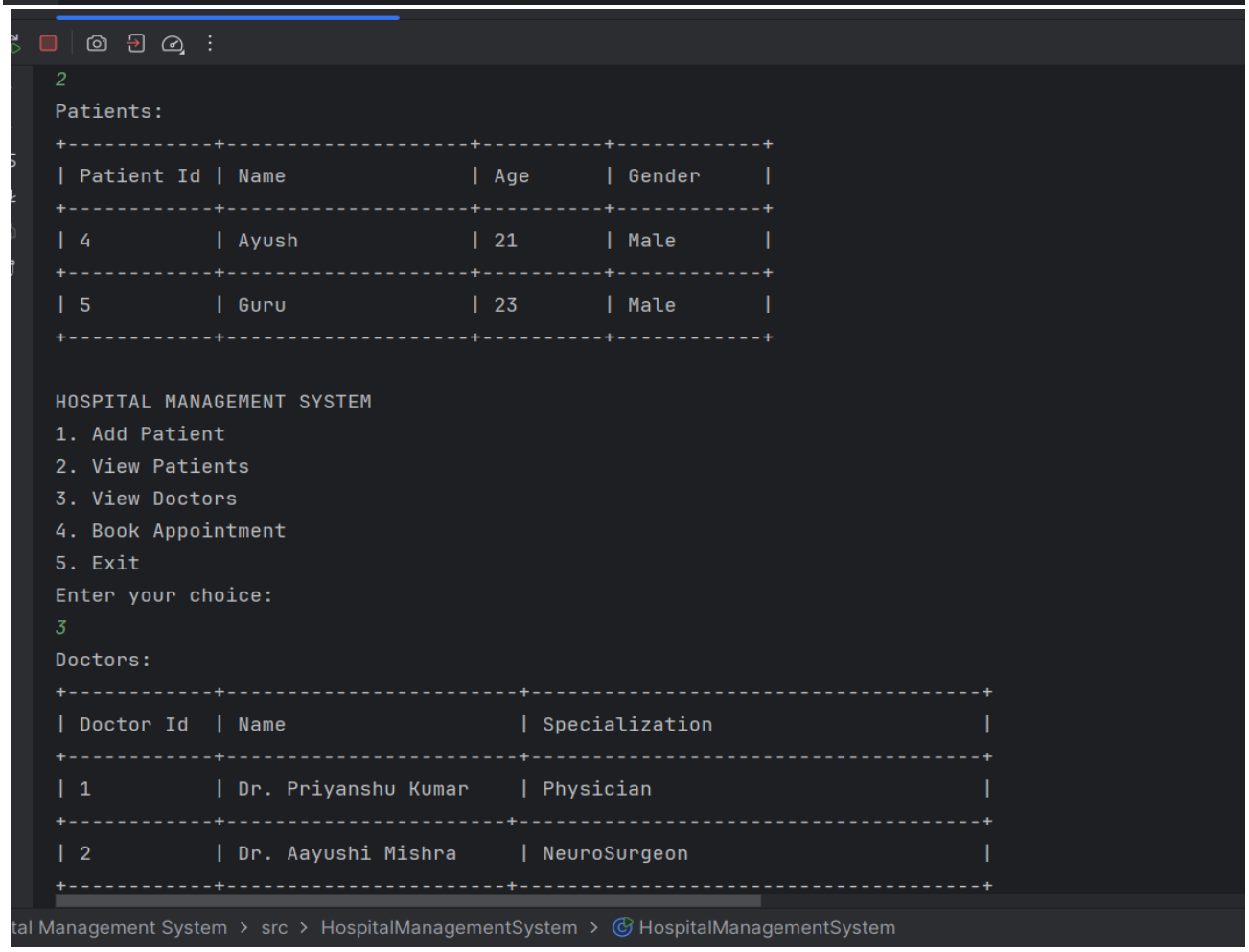
The system was tested across multiple functionalities to ensure proper performance and accurate data handling. The following core features were verified:

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

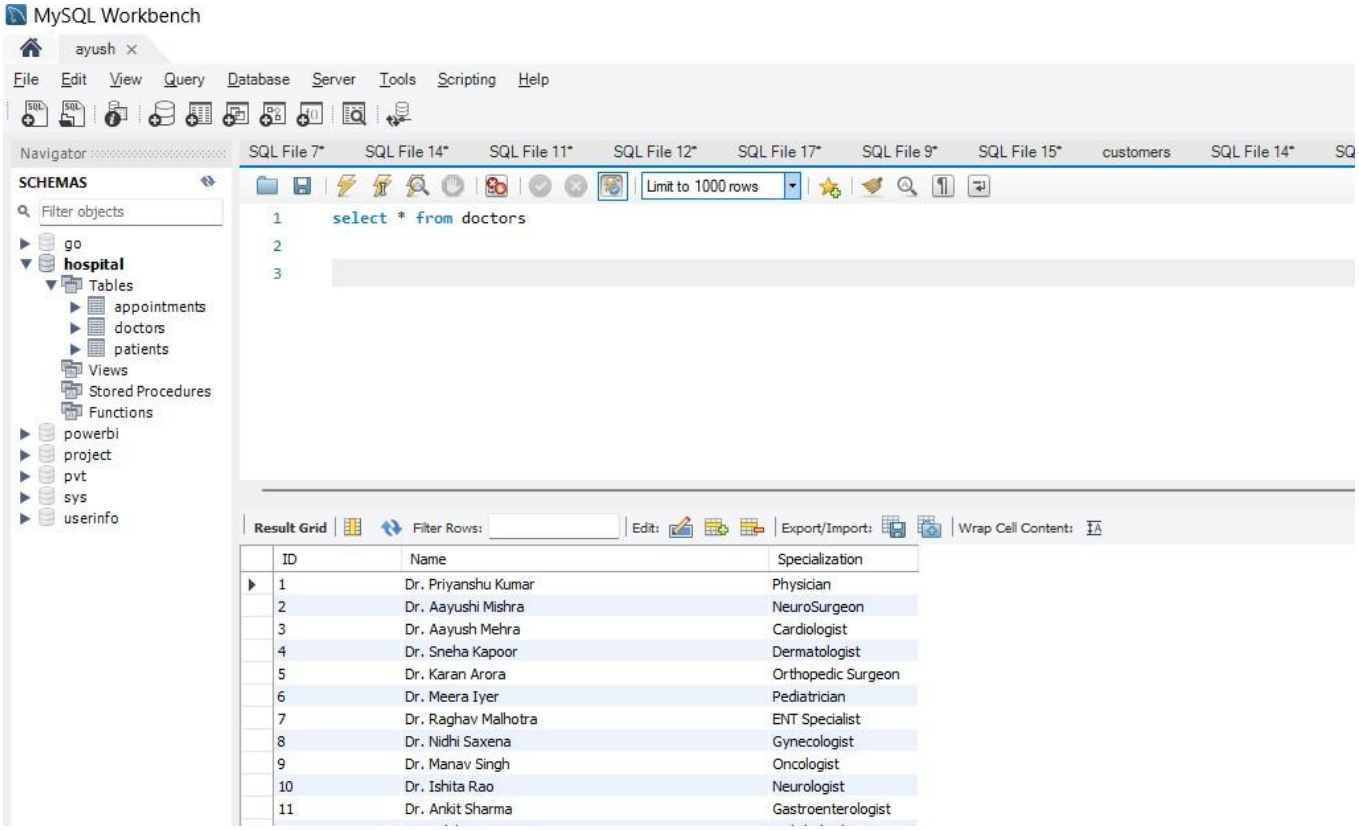
```
"C:\Program Files\Java\jdk-24\bin\java.exe" ...
DIGITAL HEALTHCARE ADMINISTRATION SYSTEM
1. Add Patient
2. View Patients
3. View Doctors
4. Book Appointment
5. Exit
Enter your choice: 1
Enter Patient Name: Riya
Enter Patient Age: 27
Enter Patient Gender: Female
Patient Added Successfully!!
```

```
Patients:
+-----------+--------+-----+--------+
| PatientId | Name   | Age | Gender |
+-----------+--------+-----+--------+
|    6      | Aman   | 30  | Male   |
|    7      | Riya   | 27  | Female |
+-----------+--------+-----+--------+

DIGITAL HEALTHCARE ADMINISTRATION SYSTEM
1. Add Patient
2. View Patients
3. View Doctors
4. Book Appointment
5. Exit
Enter your choice: 3

Doctors:
+-----------+----------------------+--------------------+
| Doctor Id | Name                 | Specialization     |
+-----------+----------------------+--------------------+
|     3     | Dr. Neha Sharma      | Cardiologist       |
|     4     | Dr. Kunal Verma      | Pediatrician       |
+-----------+----------------------+--------------------+
```


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: 7
Enter Doctor Id: 4
Enter appointment date (YYYY-MM-DD): 2025-06-15
Appointment Booked!
```



CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

SUMMARY

The **Digital Healthcare Administration System** project was developed to modernize and simplify critical hospital operations such as **patient registration**, **doctor management**, and **appointment scheduling**. Built as a **console-based application** using **Java (Swing)** for the interface and **MySQL** for backend data storage, the system showcases key concepts of **software engineering** and **database-driven application development**.

During the development phase, **modular programming** principles were strictly followed. Major functionalities—including patient management, doctor handling, and appointment scheduling—were organized into separate classes, ensuring clear structure and maintainability. Secure database interactions were achieved using **PreparedStatement**, effectively protecting the system against **SQL injection** and maintaining **data integrity**.

The project has delivered positive outcomes in terms of **functionality**, **user efficiency**, and **reliability**, enabling hospital staff to perform daily tasks more accurately and with reduced manual errors. Leveraging a **relational database** design, the system ensures efficient storage, retrieval, and analysis of patient and appointment data.

Future enhancements could include the development of a full **graphical user interface (GUI)**, incorporation of **user authentication**, integration of **medical history tracking**, and **billing modules**. Additionally, deploying the system on a **web** or **cloud platform** would significantly improve its accessibility and scalability, transforming it into a more comprehensive hospital management solution.