

# Mini Project

- **Project Title: Hospital Management System**

**Course Name:** Fundamentals of Java Programming

**Submitted By:** Utkarsha Wagajkar & Megha Kinikar

**Guided By:** Prof. N.D Chaudhari

**Academic Year:** 2025–2026

- **Acknowledgement**

We would like to express our heartfelt gratitude to Prof. Nikita Chaudhari-Shinde for her invaluable guidance, encouragement, and constant support throughout the development of this project. Her expertise and insightful feedback greatly contributed to the successful completion of our work.

- **Table of Contents**

1. Introduction
2. Objectives
3. Literature Review / Background
4. System Requirements
5. Project Design (UML Diagrams)
6. Implementation
7. Output & Screenshots
8. Testing and Results
9. Conclusion & Future Scope
10. References

## **1. Introduction**

The project titled Hospital Management System is designed to simulate the operations of a real-world hospital. It allows users to add patient records, view doctor details, view patient information, and book appointments. The system demonstrates Object-Oriented Programming (OOP) concepts in Java and uses file handling for storing and retrieving data efficiently.

## **2. Objectives**

- To develop a Java-based system for managing hospital operations efficiently.
- To implement OOP concepts such as inheritance, encapsulation, and polymorphism.

- To demonstrate file handling for data storage and retrieval.
- To provide functionality for adding patients, viewing doctors/patients, and booking appointments.
- To apply modular programming principles for maintainability and scalability.

### **3. Literature Review / Background**

Hospital Management Systems are crucial for automating patient data handling, appointment scheduling, and doctor management. Modern systems integrate electronic records and streamline hospital workflows. This project provides a simplified, console-based Java implementation that focuses on applying OOP and file handling concepts while managing key hospital functions.

### **4. System Requirements**

Hardware:

Processor: Intel i3 or higher

RAM: Minimum 4 GB

Hard Disk: 500 MB free space

Software:

Operating System: Windows / Linux / macOS

JDK Version: 17 or higher

IDE: IntelliJ IDEA

### **5. Project Design (UML Diagrams)**

Use Case Diagram – Depicts interactions between admin, doctor, and patient.

Class Diagram – Shows relationships among main classes like Patient, Doctor, and Appointment.

Sequence Diagram – Describes the flow of operations such as adding a patient or booking an appointment.

### **6. Implementation**

#### **1] Patient.java**

```
package HospitalManagementSystem;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;
public class Patient {
private Connection connection;
```

```

private Scanner scanner;
public Patient(Connection connection,Scanner scanner){
this.connection = connection;
this.scanner = scanner;
}
public void addPatient(){
System.out.println("Enter Patient Name: ")
String name=scanner.next();
System.out.println("Enter Patient Age: ");
int age=scanner.nextInt();
System.out.println("Enter Patient Gender: ");
String gender=scanner.next()
try {
String query = "INSERT INTO patients(name,age,gender) VALUES(?,?,?)";
PreparedStatement preparedStatement = connection.prepareStatement(query);
preparedStatement.setString(1,name);
preparedStatement.setInt(2,age);
preparedStatement.setString(3,gender);
int affectedRows = preparedStatement.executeUpdate();
if(affectedRows>0){
System.out.println("Patient Added Successfully!!");
}
else{
System.out.println("Failed to add Patient");
}
} catch (SQLException e){
e.printStackTrace();
}
}

public void viewPatient() {
String query = "select * from patients";
try {
PreparedStatement preparedStatement = connection.prepareStatement(query);
ResultSet resultSet = preparedStatement.executeQuery();
System.out.println("Patients:");
System.out.println("+-----+-----+-----+-----+");
System.out.println("| Patient id | Name
| Age
| Gender  |");
System.out.println("+-----+-----+-----+-----+");
while (resultSet.next()) {
int id = resultSet.getInt("id");
String name = resultSet.getString("name");

```

```

int age = resultSet.getInt("age");
String gender = resultSet.getString("gender");
System.out.printf("|%-12s|%-20s|%-12s|%-11s|\n",id,name,age,gender);
System.out.println("+-----+-----+-----+-----+");
}
} catch (SQLException e) {
e.printStackTrace();
}
}
public boolean getPatientById(int id){
String query = "SELECT * FROM patients WHERE id = ?";
try{
PreparedStatement preparedStatement = connection.prepareStatement(query);
preparedStatement.setInt(1,id);
ResultSet resultSet = preparedStatement.executeQuery();
if(resultSet.next()){
return true;
}else {
return false;
}
} catch (SQLException e){
e.printStackTrace();
}
return false;
}
}

```

## 2] Doctor.java

```

package HospitalManagementSystem;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;
public class Doctor {
private Connection connection;
public Doctor(Connection connection){
this.connection = connection;
}
public void viewDoctors() {
String query = "select * from doctors";
try {
PreparedStatement preparedStatement = connection.prepareStatement(query);

```

```

ResultSet resultSet = preparedStatement.executeQuery();
System.out.println("Doctor:");
System.out.println("+-----+-----+-----+");
System.out.println("| Doctor id | Name
| Specialization
|");
System.out.println("+-----+-----+-----+");
while (resultSet.next()) {
int id = resultSet.getInt("id");
String name = resultSet.getString("name");
String specialization = resultSet.getString("specialization");
System.out.printf("|%-12s|%-21s|%-21s|\n",id,name,specialization);
System.out.println("+-----+-----+-----+");
}
} catch (SQLException e) {
e.printStackTrace();
}
}

public boolean getDoctorById(int id){
String query = "SELECT * FROM doctors WHERE id = ?";
try{
PreparedStatement preparedStatement = connection.prepareStatement(query);
preparedStatement.setInt(1,id);
ResultSet resultSet = preparedStatement.executeQuery();
if(resultSet.next()){
return true;
}else {
return false;
}
} catch (SQLException e){
e.printStackTrace();
}
return false;
}
}

```

### 3] HospitalManagementSystem.java

```

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 = "Uvw@2023";
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 Patient");
            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.viewPatient();
                    System.out.println();
                    break;
                case 3:
                    //view Doctor
                    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 visiting!");
return;
default:
System.out.println("Enter valid choice!!");
}
}
} catch (SQLException e) {
e.printStackTrace();
}
}

public static void bookAppointment(Patient patient, Doctor doctor, Connection connection,
Scanner scanner) {
System.out.println("Enter Patient id:");
int patientId = scanner.nextInt();
System.out.println("Enter Doctor Id:");
int doctorId = scanner.nextInt();
System.out.println("Enter appointment date(YYYY-MM-DD):");
String appointmentDate = scanner.next();
if (patient.getPatientById(patientId) && doctor.getDoctorById(doctorId)) {
if (chechDoctorAvailability(doctorId, appointmentDate,connection)) {
String appointmentQuery = "INSERT INTO
appointments(patient_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 rowAffected = preparedStatement.executeUpdate();
if (rowAffected > 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 chechDoctorAvailability(int doctorId, String appointmentdate,
Connection connection) {
String query = "SELECT COUNT(*) 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();
if (resultSet.next()) {
int count = resultSet.getInt(1);
if (count == 0) {
return true;
} else {
return false;
}
}
} catch (SQLException e) {
e.printStackTrace();
}
return false;
}
}

```

## 7. Output & Screenshots-

```
HospitalManagementSystem x
[C:\Program Files\Java\jdk1.8.0_192\bin\java.exe] ...
HOSPITAL MANAGEMENT SYSTEM
1. Add Patient
2. view Patient
3. view Doctors
4. Book Appointment
5. Exit
Enter your choice:
2
Patients:
+-----+-----+-----+-----+
| Patient id | Name           | Age  | Gender |
+-----+-----+-----+-----+
| 1          | utkarsha       | 20   | female |
+-----+-----+-----+-----+
| 2          | Sudha          | 55   | female |
+-----+-----+-----+-----+
| 3          | arati          | 25   | female |
+-----+-----+-----+-----+
| 4          | vaidehi        | 23   | female |
+-----+-----+-----+-----+
```

```
HospitalManagementSystem x
HOSPITAL MANAGEMENT SYSTEM
1. Add Patient
2. view Patient
3. view Doctors
4. Book Appointment
5. Exit
Enter your choice:
3
Doctor:
+-----+-----+-----+
| Doctor id | Name           | Specialization |
+-----+-----+-----+
| 1         | Pratiksha Wagajkar | cardiology     |
+-----+-----+-----+
| 2         | Akanksha Wagajkar  | neurologist    |
+-----+-----+-----+
```

```
HospitalManagementSystem x
HOSPITAL MANAGEMENT SYSTEM
1. Add Patient
2. view Patient
3. view Doctors
4. Book Appointment
5. Exit
Enter your choice:
1
Enter Patient Name:
Megha
Enter Patient Age:
19
Enter Patient Gender:
Female
Patient Added Successfully!!
```

```
HospitalManagementSystem x
HOSPITAL MANAGEMENT SYSTEM
1. Add Patient
2. view Patient
3. view Doctors
4. Book Appointment
5. Exit
Enter your choice:
4
Enter Patient id:
2
Enter Doctor Id:
1
Enter appointment date(YYYY-MM-DD):
2025-10-31
Appointment Booked!
```

```
HospitalManagementSystem x
HOSPITAL MANAGEMENT SYSTEM
1. Add Patient
2. view Patient
3. view Doctors
4. Book Appointment
5. Exit
Enter your choice:
5
Thank you for visiting!

Process finished with exit code 0
```