Object-Oriented Programming

Project Report

Btech 4th Semester

2024-2025

Project Title: Hospital Management System

Submitted To: Dr. Deepak Kumar Sharma

Submitted By:

1. Ayush Gupta
   Sap Id: 500125537
   Roll No: R2142231692
2. Anshika Tomar
   SapId: 500119658
   Roll No: R2142230736
3. Shashank Dimri
   SapId: 500124397
   Roll No: R2142231682
4. Kanakpreet Kaur
   SapId: 500127119
   Roll No: R2142232117

Abstract

A Java application named the Hospital Management System was created to mimic and automate fundamental administrative tasks in a hospital environment. In contrast to conventional systems that are heavily dependent on sophisticated databases, this system uses JDBC (Java Database Connectivity) to communicate with a MySQL database, enabling effective storage, retrieval, and management of patient, employee, room allocation, and hospital resource data.

The project takes advantage of Object-Oriented Programming (OOP) concepts to simulate hospital parts and operations in the real world, with particular attention to handling patient admission, discharge, and room booking. The system includes features like searching for available rooms, showing patient information, discharging patients, and handling employee information—supporting seamless operations from check-in to check-out for patients. With JDBC handling the database operations, the system maintains data consistency and up-to-date information in real time, making it a strong hospital administration solution.

The core elements of the system are classes like PatientDischarge, SearchRoom, and Employee that contain the critical logic for handling patient life cycle and hospital staff. The system also has facilitating features for room availability tracking and billing to ensure smooth integration of hospital activities. It also contains moduluar pieces for employee management, such as their contact details, designations, and salaries.

In summary, the project successfully illustrates the use of Java programming, database management, and OOP to simulate and automate actual hospital management operations. It is a great academic exercise that illustrates the capability of software systems to make hospital operations better, more efficient, and able to support patient care in health care settings.

Acknowledgement

We would like to express our sincere gratitude to Dr. Deepak Kumar Sharma for his invaluable guidance, encouragement, and continuous support throughout the development of this project. His expert insights, patience, and dedication to teaching have played a crucial role in shaping our understanding and approach towards building this application.

Under his mentorship, we were able to gain hands-on experience in Java programming, file handling techniques, and object-oriented design. His feedback helped us improve the structure, logic, and efficiency of our code, making this project a strong learning experience as well as a practical implementation.

We also extend our heartfelt thanks to the entire faculty of our department for creating an academically rich and supportive environment. Their inspiration and mentorship have been vital to our growth.

Finally, we would like to acknowledge the teamwork and dedication of every member involved in this project. It is through shared effort, late-night discussions, and mutual support that we were able to complete this work successfully.

Team Members:

Ayush Gupta

Anshika Tomar

Shashank Dimri

Kanakpreet Kaur

Reason for choosing of the Healthcare domain:

The healthcare sector was selected for this project because it has an urgent need for effective administrative and patient management systems. Through the creation of a Hospital Management System based on JDBC for database connection and Swing for the user interface, this project seeks to solve typical problems in healthcare environments, including patient tracking, room management, and employee records. Building on these technologies guarantees a real-time, dependable solution that remains user-friendly for hospital personnel. The emphasis is on building an efficient, database-oriented system that has the capability to streamline hospital operations and deliver improved patient care quality.

Main Functionalities and Highlights of the code:-

- Patient Management
  - ➢ Add new patients with essential details like name, disease, room assignment, and contact information.
  - ➢ View complete patient records, including room status and discharge updates.
  - ➢ Ensure data integrity by handling incomplete or invalid patient information with robust exception handling.
- Discharge Process
  - ➢ Allows the discharge of patients by updating room availability and removing patient records.
  - ➢ Automatically updates room status to ‘Available’ when a patient is discharged.
  - ➢ Error handling ensures smooth discharge process, even if patient data is missing or incorrect.
- Employee Management
  - ➢ Manage employee records including ID, name, designation, department, and salary.
  - ➢ Display all employee information in an organized table format for easy review.
  - ➢ Handles database-related exceptions to ensure smooth employee data retrieval.
- Room Management
  - ➢ View and search room availability by type and status (available or occupied).
  - ➢ Alerts are triggered if a room assignment is attempted for an already occupied room.
  - ➢ Exception handling ensures that errors like missing room details are gracefully managed.

- Database Connectivity with JDBC
  - ➢ Uses JDBC for connecting to the MySQL database, ensuring real-time interaction with patient, employee, and room data.
  - ➢ Executes SQL queries for adding, updating, and retrieving records, while handling SQL exceptions.
- Graphical User Interface with Swing
  - ➢ The Swing framework is used for building a user-friendly interface, enabling easy navigation between modules like patient management, discharge, and employee records.
  - ➢ Tables and combo boxes are used for intuitive data display and interaction, improving usability.
- Data Storage
  - ➢ All critical data (patients, employees, rooms) is stored in a MySQL database, ensuring persistence and easy retrieval.
  - ➢ Data integrity is maintained with appropriate exception handling to prevent data loss or corruption during database interactions.
- Error Handling
  - ➢ Comprehensive exception handling is implemented throughout the program to address issues like database connection failures, invalid data inputs, and incorrect room assignments.
  - ➢ Ensures smooth user experience by preventing crashes and providing meaningful error messages.
- Billing System
  - ➢ Update billing details based on the patient's stay, including room charges, treatment, and additional services.
  - ➢ Ensure accurate and up-to-date financial records for each patient, with automated billing adjustments upon discharge.
  - ➢ Proper exception handling for billing calculations and data retrieval to prevent discrepancies and errors.
- Room Occupancy Tracking
  - ➢ Track room occupancy in real-time, ensuring no double booking or conflict when assigning rooms to patients.
  - ➢ Display room availability status dynamically, allowing hospital staff to make quick decisions on room assignments.
  - ➢ Exception handling ensures that room assignment is prevented if the room is already occupied or data is incomplete.

Complete Source Code of the Project

The following section includes the full implementation of the Hospital Management System. The project is divided into multiple classes, each responsible for specific functionalities, following Object-Oriented Programming (OOP) principles.

Class 1: Login (*Handles user authentication*)

```
import javax.swing.;
 import java.awt.;
 import java.awt.event.*;
public class Login extends JFrame {
private JTextField usernameField = new JTextField();
 private JPasswordField passwordField = new JPasswordField();
private JButton loginButton = new JButton("Login");
private JButton cancelButton = new JButton("Cancel");
private static final String CORRECT_USERNAME = "Ayush12";
private static final String CORRECT_PASSWORD = "12345";

public Login() {
    setTitle("Login Page");
    setSize(400, 250);
    setLayout(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    getContentPane().setBackground(Color.LIGHT_GRAY);
    setLocationRelativeTo(null); // for centering the window in middle

    JLabel usernameLabel = new JLabel("Username:");
    usernameLabel.setBounds(40, 20, 100, 30);
    usernameLabel.setFont(new Font("Times New Roman", Font.BOLD, 16));
    add(usernameLabel);

    JLabel passwordLabel = new JLabel("Password:");
    passwordLabel.setBounds(40, 70, 100, 30);
    passwordLabel.setFont(new Font("Times New Roman", Font.BOLD, 16));
    add(passwordLabel);

    usernameField.setBounds(150, 20, 150, 30);
    usernameField.setFont(new Font("Times New Roman", Font.PLAIN, 15));
```

```
    usernameField.setBackground(new Color(255, 179, 0));
    add(usernameField);

    passwordField.setBounds(150, 70, 150, 30);
    passwordField.setFont(new Font("Times New Roman", Font.PLAIN, 15));
    passwordField.setBackground(Color.ORANGE);
    add(passwordField);

    loginButton.setBounds(50, 140, 100, 40);
    loginButton.setFont(new Font("Times New Roman", Font.BOLD, 15));
    add(loginButton);

    cancelButton.setBounds(200, 140, 100, 40);
    cancelButton.setFont(new Font("Times New Roman", Font.BOLD, 15));
    add(cancelButton);

    loginButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            handleLogin();
        }
    });

    cancelButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            usernameField.setText("");
            passwordField.setText("");
        }
    });

    setVisible(true);
}

private void handleLogin() {
    try {
        String enteredUsername = usernameField.getText().trim();
        String enteredPassword = new String(passwordField.getPassword()).trim();
```

```
        if (enteredUsername.isEmpty() || enteredPassword.isEmpty()) {
            throw new IllegalArgumentException("Please fill in both fields.");
        }

        if (enteredUsername.equals(CORRECT_USERNAME) &&
enteredPassword.equals(CORRECT_PASSWORD)) {
            JOptionPane.showMessageDialog(this, "Login successful!");
            dispose();
            new HomeScreen();
        } else {
            throw new Exception("Incorrect username or password. Please try again.");
        }
    } catch (IllegalArgumentException ex) {
        JOptionPane.showMessageDialog(this, "Please check your input.", "Input Error",
JOptionPane.PLAIN_MESSAGE);
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, "Login failed. Try again.");
    }
}
public static void main(String[] args) {
    new Login();
}
}
```

Class 2: Home Screen (*Central Navigation Hub*)

```
import javax.swing.;
 import java.awt.;
import java.awt.event.*;
public class HomeScreen extends JFrame { public HomeScreen() { setTitle("Hospital
Management System - HomeScreen"); setSize(600, 400);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); setLocationRelativeTo(null); // to Center
the window setLayout(new GridLayout(3, 3, 10, 10));
    JButton addPatientButton = new JButton("Add New Patients");
    JButton roomButton = new JButton("Room");
    JButton departmentButton = new JButton("Department");
    JButton employeeButton = new JButton("All Employee Info");
    JButton patientInfoButton = new JButton("Patient Info");
```

```
    JButton dischargeButton = new JButton("Patient Discharge");
    JButton updateDetailsButton = new JButton("Update Patient Details");
    JButton searchRoomButton = new JButton("Search Room");

    JButton[] buttons = {addPatientButton, roomButton, departmentButton, employeeButton,
patientInfoButton,
            dischargeButton, updateDetailsButton, searchRoomButton};
    for (JButton button : buttons) {
        button.setFont(new Font("Times New Roman", Font.BOLD, 14));
        add(button);
    }

    addPatientButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new AddPatient();
        }
    });

    roomButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new Room();
        }
    });

    departmentButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new Department();
        }
    });

    employeeButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new Employee();
        }
    });

    patientInfoButton.addActionListener(new ActionListener() {
```

```
        public void actionPerformed(ActionEvent e) {
            new PatientInfo();
        }
    });

    dischargeButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new PatientDischarge();
        }
    });

    updateDetailsButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new UpdatePatientDetail();
        }
    });

    searchRoomButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new SearchRoom();
        }
    });
    setVisible(true);
}

public static void main(String[] args) {
    new HomeScreen();
}

}
```

Class 3: AddPatient (*Manages adding new patient records*)

```
import javax.swing.*;
import java.awt.*;
 import java.awt.event.*;
 import java.sql.*;
import java.time.LocalDate;
 import java.time.LocalTime;
 import java.time.format.DateTimeFormatter;
public class AddPatient extends JFrame {
private JComboBox<String> idTypeComboBox = new JComboBox<>(new String[]{"Aadhar
Card", "Voter ID", "PAN ID"});
private JTextField idNumberField = new JTextField();
private JTextField nameField = new JTextField();
private JComboBox<String> genderComboBox = new JComboBox<>(new String[]{"Male",
"Female"});
private JTextField diseaseField = new JTextField();
private JLabel timeLabel = new JLabel(getCurrentTime());
private JLabel dateLabel = new JLabel(getCurrentDate());
private JTextField depositField = new JTextField();
private JComboBox<String> roomComboBox = new JComboBox<>();
private JButton submitButton = new JButton("Submit");
private JButton cancelButton = new JButton("Cancel");

private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public AddPatient() {

    setTitle("Add New Patient");
    setSize(450, 450);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout());

    JPanel formPanel = new JPanel(new GridLayout(9, 2, 10, 10));
    formPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
```

```
formPanel.add(new JLabel("ID Type:"));
formPanel.add(idTypeComboBox);
formPanel.add(new JLabel("ID Number:"));
formPanel.add(idNumberField);
formPanel.add(new JLabel("Name:"));
formPanel.add(nameField);
formPanel.add(new JLabel("Gender:"));
formPanel.add(genderComboBox);
formPanel.add(new JLabel("Disease:"));
formPanel.add(diseaseField);
formPanel.add(new JLabel("Time:"));
formPanel.add(timeLabel);
formPanel.add(new JLabel("Admission Date:"));
formPanel.add(dateLabel);
formPanel.add(new JLabel("Deposit Amount:"));
formPanel.add(depositField);
formPanel.add(new JLabel("Select Room:"));
formPanel.add(roomComboBox);

JPanel buttonPanel = new JPanel();
buttonPanel.add(submitButton);
buttonPanel.add(cancelButton);

submitButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        handleSubmit();
    }
});

cancelButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        dispose();
    }
});

add(formPanel, BorderLayout.CENTER);
```

```
    add(buttonPanel, BorderLayout.SOUTH);

    loadAvailableRooms();
    setVisible(true);
}

private static String getCurrentTime() {
    return LocalTime.now().format(DateTimeFormatter.ofPattern("hh:mm a"));
}

private static String getCurrentDate() {
    return LocalDate.now().toString();
}

private void handleSubmit() {
    String idType = (String) idTypeComboBox.getSelectedItem();
    String idNumber = idNumberField.getText().trim();
    String name = nameField.getText().trim();
    String gender = (String) genderComboBox.getSelectedItem();
    String disease = diseaseField.getText().trim();
    String deposit = depositField.getText().trim();
    String time = timeLabel.getText();
    String date = dateLabel.getText();
    String selectedRoom = (String) roomComboBox.getSelectedItem();

    if (idNumber.isEmpty() || name.isEmpty() || disease.isEmpty() || deposit.isEmpty() ||
selectedRoom == null) {
        JOptionPane.showMessageDialog(this, "Please fill all fields.");
        return;
    }

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
        if (isRoomOccupied(con, selectedRoom)) {
            JOptionPane.showMessageDialog(this, "This room is already assigned!", "Room Error",
JOptionPane.WARNING_MESSAGE);
            return;
```

```
        }

        String sql = "INSERT INTO patient_info (ID, Number, Name, Gender, Patient_Disease,
Room_Number, Time, Deposit, admission_date) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
        try (PreparedStatement ps = con.prepareStatement(sql)) {
            ps.setString(1, idType);
            ps.setString(2, idNumber);
            ps.setString(3, name);
            ps.setString(4, gender);
            ps.setString(5, disease);
            ps.setString(6, selectedRoom);
            ps.setString(7, time);
            ps.setString(8, deposit);
            ps.setString(9, date);
            ps.executeUpdate();
        }

        updateRoomStatus(con, selectedRoom);
        JOptionPane.showMessageDialog(this, "Patient Added Successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);
        dispose();

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void loadAvailableRooms() {
    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         Statement stmt = con.createStatement();
         ResultSet rs = stmt.executeQuery("SELECT room_no FROM Room WHERE Availablility
= 'Available'")) {

        boolean hasRooms = false;
        while (rs.next()) {
```

```
            roomComboBox.addItem(rs.getString("room_no"));
            hasRooms = true;
        }

        if (!hasRooms) {
            roomComboBox.addItem("No Available Rooms");
            roomComboBox.setEnabled(false);
        }

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private boolean isRoomOccupied(Connection con, String roomNumber) throws SQLException {
    String sql = "SELECT Availablility FROM Room WHERE room_no = ?";
    try (PreparedStatement pst = con.prepareStatement(sql)) {
        pst.setString(1, roomNumber);
        try (ResultSet rs = pst.executeQuery()) {
            return rs.next() && rs.getString("Availablility").equalsIgnoreCase("Occupied");
        }
    }
}

private void updateRoomStatus(Connection con, String roomNumber) throws SQLException {
    String sql = "UPDATE Room SET Availablility = 'Occupied' WHERE room_no = ?";
    try (PreparedStatement pst = con.prepareStatement(sql)) {
        pst.setString(1, roomNumber);
        pst.executeUpdate();
    }
}

public static void main(String[] args) {
    new AddPatient();
} }
```

Class 4: PatientInfo (*Display Patient Record*)

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
 import java.awt.*;
 import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
public class PatientInfo extends JFrame implements Displayable { private JTable patientTable;
private DefaultTableModel tableModel; private JButton backButton;
private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public PatientInfo() {
    setTitle("Patient Information");
    setSize(1000, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout());

    tableModel = new DefaultTableModel();
    tableModel.addColumn("ID Type");
    tableModel.addColumn("ID Number");
    tableModel.addColumn("Name");
    tableModel.addColumn("Gender");
    tableModel.addColumn("Disease");
    tableModel.addColumn("Room Number");
    tableModel.addColumn("Time Admitted");
    tableModel.addColumn("Deposit");
    tableModel.addColumn("Admission Date");

    patientTable = new JTable(tableModel);
    JScrollPane scrollPane = new JScrollPane(patientTable);
    add(scrollPane, BorderLayout.CENTER);

    backButton = new JButton("Back to HomeScreen");
    backButton.setFont(new Font("Arial", Font.BOLD, 14));
```

```
    backButton.setBackground(Color.LIGHT_GRAY);
    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen(); // Open HomeScreen
            dispose(); // Close PatientInfo window
        }
    });

    JPanel buttonPanel = new JPanel();
    buttonPanel.add(backButton);
    add(buttonPanel, BorderLayout.SOUTH);

    displayData();

    setVisible(true);
}

public void displayData() {
    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         Statement stmt = con.createStatement();
         ResultSet rs = stmt.executeQuery("SELECT * FROM patient_info")) {

        while (rs.next()) {
            tableModel.addRow(new Object[]{
                rs.getString("ID"),
                rs.getString("Number"),
                rs.getString("Name"),
                rs.getString("Gender"),
                rs.getString("Patient_Disease"),
                rs.getString("Room_Number"),
                rs.getString("Time"),
                rs.getString("Deposit"),
                rs.getDate("admission_date")
            });
        }
```

```
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

public static void main(String[] args) {
    new PatientInfo();
}

}
```

Class 5: Displayable (Interface for uniform data display)

```
public interface Displayable {
void displayData();
}
```

Class 6: Department (Display Hospital Departments)

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
public class Department extends JFrame implements Displayable { private JTable
departmentTable; private DefaultTableModel tableModel; private JButton backButton;
private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public Department() {
    setTitle("Hospital Departments");
    setSize(700, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null); // Center the window

    tableModel = new DefaultTableModel();
    tableModel.addColumn("Dept ID");
    tableModel.addColumn("Department Name");
```

```
    tableModel.addColumn("Head of Department");
    tableModel.addColumn("Contact");
    tableModel.addColumn("Location");
    tableModel.addColumn("Doctors");
    tableModel.addColumn("Nurses");

    departmentTable = new JTable(tableModel);
    JScrollPane scrollPane = new JScrollPane(departmentTable);
    add(scrollPane, BorderLayout.CENTER);

    // Back Button
    backButton = new JButton("Back to HomeScreen");
    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
        }
    });

    JPanel buttonPanel = new JPanel();
    buttonPanel.add(backButton);
    add(buttonPanel, BorderLayout.SOUTH);

    displayData(); // Calls the interface method

    setVisible(true);
}

public void displayData() {
    String query = "SELECT * FROM department";

    try (
        Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
        PreparedStatement ps = con.prepareStatement(query);
        ResultSet rs = ps.executeQuery()
    ) {
```

```
        while (rs.next()) {
            tableModel.addRow(new Object[]{
                rs.getString("dept_id"),
                rs.getString("dept_name"),
                rs.getString("head_of_department"),
                rs.getString("contact_number"),
                rs.getString("location"),
                rs.getInt("no_of_doctors"),
                rs.getInt("no_of_nurses")
            });
        }

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

public static void main(String[] args) {
    new Department();
}
 }
```

Class 7:Employee (*Display employee detail*)

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
public class Employee extends JFrame { private JTable employeeTable; private
DefaultTableModel tableModel; private JButton backButton;
private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";
```

```
public Employee() {
    setTitle("Employee Information");
    setSize(800, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout());

    tableModel = new DefaultTableModel();
    tableModel.addColumn("Emp ID");
    tableModel.addColumn("Name");
    tableModel.addColumn("Designation");
    tableModel.addColumn("Dept ID");
    tableModel.addColumn("Contact");
    tableModel.addColumn("Salary");

    employeeTable = new JTable(tableModel);
    JScrollPane scrollPane = new JScrollPane(employeeTable);
    add(scrollPane, BorderLayout.CENTER);

    backButton = new JButton("Back to HomeScreen");
    backButton.setFont(new Font("Arial", Font.BOLD, 14));
    backButton.setBackground(Color.LIGHT_GRAY);
    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
        }
    });

    JPanel buttonPanel = new JPanel();
    buttonPanel.add(backButton);
    add(buttonPanel, BorderLayout.SOUTH);

    loadEmployeeData();

    setVisible(true);
}
```

```
private void loadEmployeeData() {
   String query = "SELECT emp_id, name, designation, department_id, contact_number, salary
FROM employee";

   try (
      Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
      PreparedStatement ps = con.prepareStatement(query);
      ResultSet rs = ps.executeQuery()
   ) {
      while (rs.next()) {
         tableModel.addRow(new Object[]{
            rs.getString("emp_id"),
            rs.getString("name"),
            rs.getString("designation"),
            rs.getString("department_id"),
            rs.getString("contact_number"),
            rs.getDouble("salary")
         });
      }

   } catch (SQLException e) {
      JOptionPane.showMessageDialog(this, "Database Connection Failed: " + e.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
   }
}

public static void main(String[] args) {
   new Employee();
}}
```

Class 8: Room (*Display room details*)

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
import java.sql.*;
public class Room extends JFrame implements Displayable { private JTable roomTable; private
DefaultTableModel tableModel; private JButton backButton;
private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public Room() {
    setTitle("Room Information");
    setSize(600, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null); // Center the window

    tableModel = new DefaultTableModel();
    roomTable = new JTable(tableModel);
    tableModel.addColumn("Room No");
    tableModel.addColumn("Availability");
    tableModel.addColumn("Price");
    tableModel.addColumn("Room Type");

    add(roomTable, BorderLayout.CENTER);

    // Back Button
    backButton = new JButton("Back to HomeScreen");
    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
        }
    });

    JPanel buttonPanel = new JPanel();
    buttonPanel.add(backButton);
    add(buttonPanel, BorderLayout.SOUTH);

    displayData(); // Calls the interface method
```

```
    setVisible(true);
}

public void displayData() {
    String query = "SELECT * FROM Room";
    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement ps = con.prepareStatement(query);
         ResultSet rs = ps.executeQuery()) {

        while (rs.next()) {
            tableModel.addRow(new Object[]{
                rs.getString("room_no"),
                rs.getString("Availablility"),
                rs.getString("Price"),
                rs.getString("Room_Type")
            });
        }

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}
public static void main(String[] args) {
    new Room();
} }
```

Class 9: UpdatePatientDetail (*Updating patient Details*)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
public class UpdatePatientDetail extends JFrame { private JComboBox patientNameComboBox;
private JLabel timeLabel, dateLabel, amtPaidLabel, pendingAmtLabel; private JTextField
```

```
totalAmountField, additionalDepositField; private JButton checkButton, updateTotalButton,
makeDepositButton, backButton;
private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public UpdatePatientDetail() {
    setTitle("Update Patient Details");
    setSize(500, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout(10, 10));

    JPanel formPanel = new JPanel(new GridLayout(7, 2, 10, 10));
    formPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));


    formPanel.add(new JLabel("Select Patient:"));
    patientNameComboBox = new JComboBox<>();
    loadPatientNames();
    formPanel.add(patientNameComboBox);

formPanel.add(new JLabel("Check-in Time:")); timeLabel = new JLabel("N/A");
formPanel.add(timeLabel);
formPanel.add(new JLabel("Check-in Date:"));
dateLabel = new JLabel("N/A");
formPanel.add(dateLabel);

    formPanel.add(new JLabel("Amount Paid:"));
    amtPaidLabel = new JLabel("N/A");
    formPanel.add(amtPaidLabel);

    formPanel.add(new JLabel("Enter Total Bill Amount:"));
    totalAmountField = new JTextField();
    formPanel.add(totalAmountField);

    formPanel.add(new JLabel("Remaining Amount:"));
```

```
pendingAmtLabel = new JLabel("N/A");
formPanel.add(pendingAmtLabel);

formPanel.add(new JLabel("Enter Additional Deposit:"));
additionalDepositField = new JTextField();
additionalDepositField.setEnabled(false);
formPanel.add(additionalDepositField);

JPanel buttonPanel = new JPanel(new GridLayout(1, 4, 10, 10));
checkButton = new JButton("Check");
updateTotalButton = new JButton("Update Total Bill");
makeDepositButton = new JButton("Make Additional Deposit");
backButton = new JButton("Back to HomeScreen");

buttonPanel.add(checkButton);
buttonPanel.add(updateTotalButton);
buttonPanel.add(makeDepositButton);
buttonPanel.add(backButton);

add(formPanel, BorderLayout.CENTER);
add(buttonPanel, BorderLayout.SOUTH);

checkButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        loadPatientDetails();
    }
});

updateTotalButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        updateTotalBill();
    }
});

makeDepositButton.setEnabled(false);
makeDepositButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
```

```
            makeAdditionalDeposit();
        }
    });

    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
        }
    });

    setVisible(true);
}

private void loadPatientNames() {
    String query = "SELECT Name FROM patient_info";
    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement pst = con.prepareStatement(query);
         ResultSet rs = pst.executeQuery()) {

        while (rs.next()) {
            patientNameComboBox.addItem(rs.getString("Name"));
        }

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void loadPatientDetails() {
    String selectedName = (String) patientNameComboBox.getSelectedItem();
    if (selectedName == null) return;

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
```

```
        PreparedStatement pst = con.prepareStatement(
                "SELECT Time, admission_date, Deposit, Total_Bill, Pending_Amount FROM
patient_info WHERE Name = ?")) {
        pst.setString(1, selectedName);
        try (ResultSet rs = pst.executeQuery()) {
            if (rs.next()) {
                timeLabel.setText(rs.getString("Time"));
                dateLabel.setText(rs.getString("admission_date"));
                amtPaidLabel.setText(rs.getString("Deposit"));
                totalAmountField.setText(rs.getString("Total_Bill"));
                pendingAmtLabel.setText(rs.getString("Pending_Amount"));

                // Enable deposit field and button after total bill is set
                additionalDepositField.setEnabled(true);
                makeDepositButton.setEnabled(true);
            }
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void updateTotalBill() {
    String selectedName = (String) patientNameComboBox.getSelectedItem();
    if (selectedName == null || totalAmountField.getText().trim().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Enter a valid total bill amount.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement pst = con.prepareStatement(
                 "UPDATE patient_info SET Total_Bill = ?, Pending_Amount = Total_Bill - Deposit
WHERE Name = ?")) {
```

```
        double totalAmount = Double.parseDouble(totalAmountField.getText().trim());
        pst.setDouble(1, totalAmount);
        pst.setString(2, selectedName);
        pst.executeUpdate();
        loadPatientDetails();
        JOptionPane.showMessageDialog(this, "Total Bill & Pending Amount Updated!",
"Success", JOptionPane.INFORMATION_MESSAGE);

    } catch (SQLException | NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void makeAdditionalDeposit() {
    String selectedName = (String) patientNameComboBox.getSelectedItem();
    if (selectedName == null || additionalDepositField.getText().trim().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Enter a valid deposit amount.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement updateDeposit = con.prepareStatement(
                 "UPDATE patient_info SET Deposit = Deposit + ?, Pending_Amount = Total_Bill -
Deposit WHERE Name = ?")) {

        double additionalDeposit = Double.parseDouble(additionalDepositField.getText().trim());
        updateDeposit.setDouble(1, additionalDeposit);
        updateDeposit.setString(2, selectedName);
        updateDeposit.executeUpdate();

        // Refresh display
        loadPatientDetails();
        additionalDepositField.setText("");
        JOptionPane.showMessageDialog(this, "Deposit Updated Successfully!", "Success",
```

```
JOptionPane.INFORMATION_MESSAGE);

   } catch (SQLException | NumberFormatException e) {
      JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
   }
}

public static void main(String[] args) {
   new UpdatePatientDetail();
} }
```

Class 10: SearchRoom (*Search Room based on Availability*)

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
public class SearchRoom extends JFrame implements Displayable { private JComboBox
statusComboBox; private JTable roomTable; private DefaultTableModel tableModel; private
JButton searchButton, backButton;
private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public SearchRoom() {
   setTitle("Search Room Availability");
   setSize(600, 400);
   setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
   setLocationRelativeTo(null);
   setLayout(new BorderLayout(10, 10));

   JPanel panel = new JPanel();
   panel.add(new JLabel("Room Status:"));

   statusComboBox = new JComboBox<>(new String[]{"Available", "Occupied"});
```

```
    panel.add(statusComboBox);
    searchButton = new JButton("Search");
    panel.add(searchButton);

    add(panel, BorderLayout.NORTH);

    tableModel = new DefaultTableModel();
    tableModel.addColumn("Room No");
    tableModel.addColumn("Availability");
    tableModel.addColumn("Price");
    tableModel.addColumn("Room Type");

    roomTable = new JTable(tableModel);
    JScrollPane scrollPane = new JScrollPane(roomTable);
    add(scrollPane, BorderLayout.CENTER);

    JPanel buttonPanel = new JPanel();
    backButton = new JButton("Back to Home");
    buttonPanel.add(backButton);
    add(buttonPanel, BorderLayout.SOUTH);

    searchButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            displayData();
        }
    });

    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
        }
    });

    setVisible(true);
}
public void displayData() {
```

```
   String selectedStatus = (String) statusComboBox.getSelectedItem();
   if (selectedStatus == null) return;

   tableModel.setRowCount(0);

   try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
        PreparedStatement pst = con.prepareStatement("SELECT * FROM Room WHERE
Availablility = ?")) {
       pst.setString(1, selectedStatus);
       try (ResultSet rs = pst.executeQuery()) {
          while (rs.next()) {
             tableModel.addRow(new Object[]{
                   rs.getString("room_no"),
                   rs.getString("Availablility"),
                   rs.getString("Price"),
                   rs.getString("Room_Type")
             });
          }
       }
   } catch (SQLException e) {
       JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
   }
}

public static void main(String[] args) {
   new SearchRoom();
}}
```

Class 11: DischargePatients (*Dischare patients*)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
import java.util.ArrayList;
```

```
public class PatientDischarge extends JFrame { private JComboBox patientIdComboBox; private
JLabel nameLabel = new JLabel("N/A"); private JLabel roomLabel = new JLabel("N/A");
private JButton checkButton = new JButton("Check"); private JButton dischargeButton = new
JButton("Discharge"); private JButton backButton = new JButton("Back to HomeScreen");
private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public PatientDischarge() {
    setTitle("CHECK-OUT");
    setSize(400, 300);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new GridLayout(5, 2, 10, 10));

    add(new JLabel("Select Patient ID:"));
    patientIdComboBox = new JComboBox<>();
    loadPatientIDs();
    add(patientIdComboBox);

    add(new JLabel("Patient Name:"));
    add(nameLabel);

    add(new JLabel("Room Number:"));
    add(roomLabel);

    checkButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            loadPatientDetails();
        }
    });
    add(checkButton);

    dischargeButton.setEnabled(false);
    dischargeButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            dischargePatient();
```

```
        }
    });
    add(dischargeButton);

    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
        }
    });
    add(backButton);

    setVisible(true);
}

private void loadPatientIDs() {
    String query = "SELECT Number FROM patient_info";

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement pst = con.prepareStatement(query);
         ResultSet rs = pst.executeQuery()) {

        ArrayList<String> ids = new ArrayList<>();
        while (rs.next()) {
            ids.add(rs.getString("Number"));
        }

        if (ids.isEmpty()) {
            patientIdComboBox.addItem("No Patients Available");
            patientIdComboBox.setEnabled(false);
        } else {
            for (String id : ids) {
                patientIdComboBox.addItem(id);
            }
        }
```

```
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void loadPatientDetails() {
    String selectedID = (String) patientIdComboBox.getSelectedItem();
    if (selectedID == null || selectedID.equals("No Patients Available")) {
        return;
    }

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement pst = con.prepareStatement("SELECT Name, Room_Number FROM
patient_info WHERE Number = ?")) {
        pst.setString(1, selectedID);
        try (ResultSet rs = pst.executeQuery()) {
            if (rs.next()) {
                nameLabel.setText(rs.getString("Name"));
                roomLabel.setText(rs.getString("Room_Number"));
                dischargeButton.setEnabled(true);
            } else {
                nameLabel.setText("N/A");
                roomLabel.setText("N/A");
                dischargeButton.setEnabled(false);
            }
        }

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void dischargePatient() {
    String selectedID = (String) patientIdComboBox.getSelectedItem();
```

```
    String roomNumber = roomLabel.getText();

    if (selectedID == null || roomNumber.equals("N/A")) {
        JOptionPane.showMessageDialog(this, "Invalid selection.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
        try (PreparedStatement pst = con.prepareStatement("DELETE FROM patient_info WHERE
Number = ?")) {
            pst.setString(1, selectedID);
            pst.executeUpdate();
        }

        try (PreparedStatement pst = con.prepareStatement("UPDATE Room SET Availablility =
'Available' WHERE room_no = ?")) {
            pst.setString(1, roomNumber);
            pst.executeUpdate();
        }

        JOptionPane.showMessageDialog(this, "Patient Discharged Successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);
        dispose();

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

public static void main(String[] args) {
    new PatientDischarge();
}

}
```

Code Snippets with Explanation and Sample Output

1. Login Module

Code:

```
import javax.swing.;
import java.awt.;
import java.awt.event.*;
public class Login extends JFrame { // windows and buttons
private JTextField usernameField = new JTextField(); private JPasswordField passwordField =
new JPasswordField(); private JButton loginButton = new JButton("Login"); private JButton
cancelButton = new JButton("Cancel");
private static final String CORRECT_USERNAME = "Ayush12";
private static final String CORRECT_PASSWORD = "12345";

public Login() {
    setTitle("Login Page");
    setSize(400, 250);
    setLayout(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    getContentPane().setBackground(Color.LIGHT_GRAY);
    setLocationRelativeTo(null); // for centering the window in middle

    JLabel usernameLabel = new JLabel("Username:");
    usernameLabel.setBounds(40, 20, 100, 30);
    usernameLabel.setFont(new Font("Times New Roman", Font.BOLD, 16));
    add(usernameLabel);

    JLabel passwordLabel = new JLabel("Password:");
    passwordLabel.setBounds(40, 70, 100, 30);
    passwordLabel.setFont(new Font("Times New Roman", Font.BOLD, 16));
    add(passwordLabel);

    usernameField.setBounds(150, 20, 150, 30);
```

```
    usernameField.setFont(new Font("Times New Roman", Font.PLAIN, 15));
    usernameField.setBackground(new Color(255, 179, 0));
    add(usernameField);

    passwordField.setBounds(150, 70, 150, 30);
    passwordField.setFont(new Font("Times New Roman", Font.PLAIN, 15));
    passwordField.setBackground(Color.ORANGE);
    add(passwordField);

    loginButton.setBounds(50, 140, 100, 40);
    loginButton.setFont(new Font("Times New Roman", Font.BOLD, 15));
    add(loginButton);

    cancelButton.setBounds(200, 140, 100, 40);
    cancelButton.setFont(new Font("Times New Roman", Font.BOLD, 15));
    add(cancelButton);

    loginButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            handleLogin();
        }
    });

    cancelButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            usernameField.setText("");
            passwordField.setText("");
        }
    });

    setVisible(true);
}

private void handleLogin() {
    try {
        String enteredUsername = usernameField.getText().trim();
        String enteredPassword = new String(passwordField.getPassword()).trim();
```

```
        if (enteredUsername.isEmpty() || enteredPassword.isEmpty()) {
            throw new IllegalArgumentException("Please fill in both fields.");
        }

        if (enteredUsername.equals(CORRECT_USERNAME) &&
enteredPassword.equals(CORRECT_PASSWORD)) {
            JOptionPane.showMessageDialog(this, "Login successful!");
            dispose();
            new HomeScreen();
        } else {
            throw new Exception("Incorrect username or password. Please try again.");
        }
    } catch (IllegalArgumentException ex) {
        JOptionPane.showMessageDialog(this, "Please check your input.", "Input Error",
JOptionPane.PLAIN_MESSAGE);
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, "Login failed. Try again.");
    }
}

public static void main(String[] args) {
    new Login();
} }
```

Concepts Used:

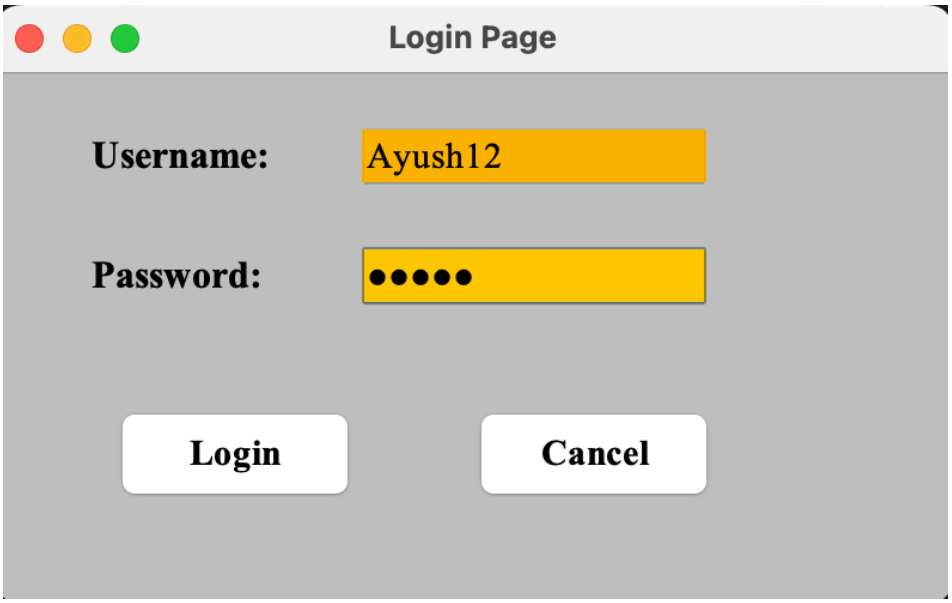
- Classes
- Objects
- Constructors
- this keyword
- Access Modifiers
- Method Overloading
- Exception Handling
- Nested Classes
- GUI (Swing)
- Event Handling
- Event Listeners

- Components and Containers
- Layout Managers
- String Handling
- Input/Output Statements

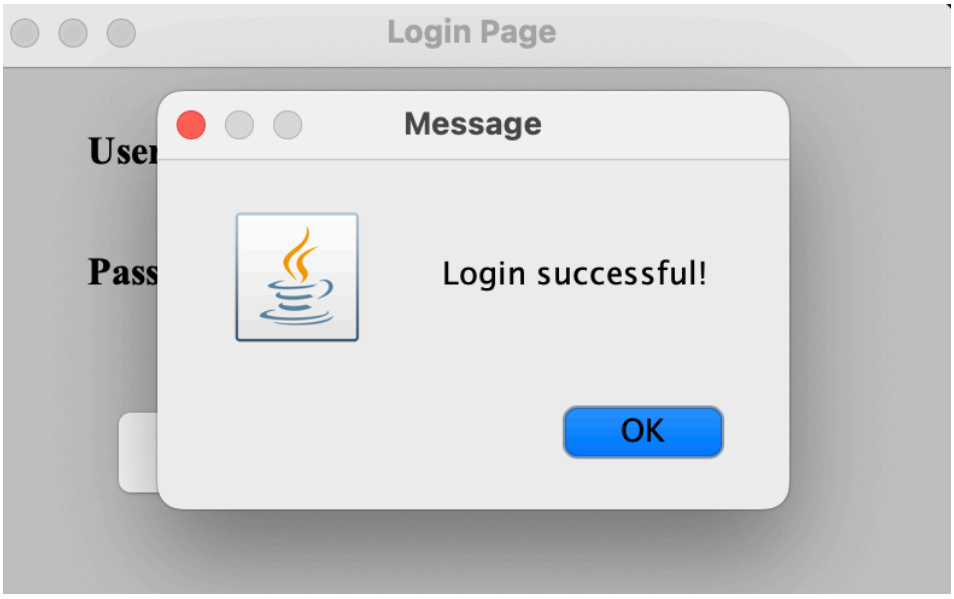
Explanation:

This Java application provides a basic login window via Swing for the GUI. There are two fields for password and username, and two buttons — Login and Cancel. On clicking "Login", the application compares the provided credentials with the actual username (Ayush12) and password (12345). If the credentials match, then a success message is displayedand the window is closed. If they don't match, then an error message is displayed. Clicking "Cancel" will empty both input fields. The code employs fundamental OOP principles, exception handling, and event-driven programming.

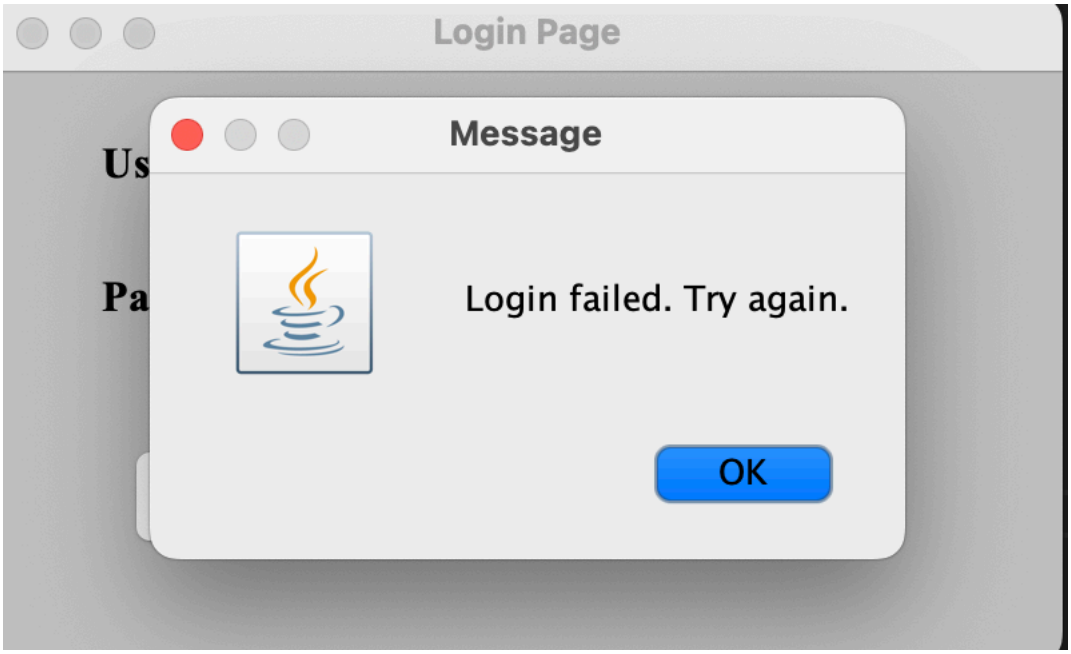
Output:



Output: When Login is successful



Output: When Login Fails



2. HomeScreen Module

Code:

import javax.swing.;
import java.awt.; import java.awt.event.*;
public class HomeScreen extends JFrame { public HomeScreen() { setTitle("Hospital Management System - HomeScreen"); setSize(600, 400); setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); setLocationRelativeTo(null); // to Center the window setLayout(new GridLayout(3, 3, 10, 10));

```
   JButton addPatientButton = new JButton("Add New Patients");
    JButton roomButton = new JButton("Room");
    JButton departmentButton = new JButton("Department");
    JButton employeeButton = new JButton("All Employee Info");
    JButton patientInfoButton = new JButton("Patient Info");
    JButton dischargeButton = new JButton("Patient Discharge");
    JButton updateDetailsButton = new JButton("Update Patient Details");
    JButton searchRoomButton = new JButton("Search Room");

    JButton[] buttons = {addPatientButton, roomButton, departmentButton, employeeButton,
patientInfoButton,
            dischargeButton, updateDetailsButton, searchRoomButton};

    for (JButton button : buttons) {
       button.setFont(new Font("Times New Roman", Font.BOLD, 14));
       add(button);
    }

// after clicking on button the respective constructor is called
addPatientButton.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) { new AddPatient(); } });
   roomButton.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
          new Room();
       }
    });

    departmentButton.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
          new Department();
       }
    });

    employeeButton.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
          new Employee();
       }
```

```
    });

    patientInfoButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new PatientInfo();
        }
    });

    dischargeButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new PatientDischarge();
        }
    });

    updateDetailsButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new UpdatePatientDetail();
        }
    });

    searchRoomButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new SearchRoom();
        }
    });

    setVisible(true);
}

public static void main(String[] args) {
    new HomeScreen();
}}
```

Concepts Used:

- Classes
- Objects
- Constructors

- this keyword (implicitly)
- Access Modifiers
- GUI (Swing)
- Components and Containers
- Layout Managers (GridLayout)
- Event Handling
- Event Listeners
- Inheritance (extends JFrame)
- String Handling (button labels)

Explanation:

This code creates the Home Screen for a Hospital Management System using Java Swing. It shows a window with multiple buttons like “Add New Patients”, “Room”, “Patient Info”, etc. Each button opens a different window (by calling another class) when clicked. The layout is arranged in a grid, and event handling is used to respond to button clicks.

Output:

3. AddPatient Module

Code:

import javax.swing.;
import java.awt.;
import java.awt.event.;

```
import java.sql.;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
public class AddPatient extends JFrame {
private JComboBox<String> idTypeComboBox = new JComboBox<>(new String[]{"Aadhar
Card", "Voter ID", "PAN ID"});
private JTextField idNumberField = new JTextField();
private JTextField nameField = new JTextField();
private JComboBox<String> genderComboBox = new JComboBox<>(new String[]{"Male",
"Female"});
private JTextField diseaseField = new JTextField();
private JLabel timeLabel = new JLabel(getCurrentTime());
private JLabel dateLabel = new JLabel(getCurrentDate());
private JTextField depositField = new JTextField();
private JComboBox<String> roomComboBox = new JComboBox<>();
private JButton submitButton = new JButton("Submit");
private JButton cancelButton = new JButton("Cancel");

private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";


public AddPatient() {
    setTitle("Add New Patient");
    setSize(450, 450);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout());

    JPanel formPanel = new JPanel(new GridLayout(9, 2, 10, 10));
    formPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

    formPanel.add(new JLabel("ID Type:"));  //Left
    formPanel.add(idTypeComboBox);      // Right
    formPanel.add(new JLabel("ID Number:"));
```

```
formPanel.add(idNumberField);
formPanel.add(new JLabel("Name:"));
formPanel.add(nameField);
formPanel.add(new JLabel("Gender:"));
formPanel.add(genderComboBox);
formPanel.add(new JLabel("Disease:"));
formPanel.add(diseaseField);
formPanel.add(new JLabel("Time:"));
formPanel.add(timeLabel);
formPanel.add(new JLabel("Admission Date:"));
formPanel.add(dateLabel);
formPanel.add(new JLabel("Deposit Amount:"));
formPanel.add(depositField);
formPanel.add(new JLabel("Select Room:"));
formPanel.add(roomComboBox);

JPanel buttonPanel = new JPanel();
buttonPanel.add(submitButton);
buttonPanel.add(cancelButton);

submitButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        handleSubmit();
    }
});

cancelButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        dispose();
    }
});

add(formPanel, BorderLayout.CENTER);
add(buttonPanel, BorderLayout.SOUTH);

loadAvailableRooms();
setVisible(true);
```

```
}

private static String getCurrentTime() {
    return LocalTime.now().format(DateTimeFormatter.ofPattern("hh:mm a"));
}

private static String getCurrentDate() {
    return LocalDate.now().toString();
}

private void handleSubmit() {
    String idType = (String) idTypeComboBox.getSelectedItem();
    String idNumber = idNumberField.getText().trim();
    String name = nameField.getText().trim();
    String gender = (String) genderComboBox.getSelectedItem();
    String disease = diseaseField.getText().trim();
    String deposit = depositField.getText().trim();
    String time = timeLabel.getText();
    String date = dateLabel.getText();
    String selectedRoom = (String) roomComboBox.getSelectedItem();

    if (idNumber.isEmpty() || name.isEmpty() || disease.isEmpty() || deposit.isEmpty() ||
selectedRoom == null) {
        JOptionPane.showMessageDialog(this, "Please fill all fields.");
        return;
    }

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
        if (isRoomOccupied(con, selectedRoom)) {
            JOptionPane.showMessageDialog(this, "This room is already assigned!", "Room Error",
JOptionPane.WARNING_MESSAGE);
            return;
        }

        String sql = "INSERT INTO patient_info (ID, Number, Name, Gender, Patient_Disease,
Room_Number, Time, Deposit, admission_date) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
```

```
        try (PreparedStatement ps = con.prepareStatement(sql)) {
            ps.setString(1, idType);
            ps.setString(2, idNumber);
            ps.setString(3, name);
            ps.setString(4, gender);
            ps.setString(5, disease);
            ps.setString(6, selectedRoom);
            ps.setString(7, time);
            ps.setString(8, deposit);
            ps.setString(9, date);
            ps.executeUpdate();
        }

        updateRoomStatus(con, selectedRoom);
        JOptionPane.showMessageDialog(this, "Patient Added Successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);
        dispose();

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void loadAvailableRooms() {
    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         Statement stmt = con.createStatement();
         ResultSet rs = stmt.executeQuery("SELECT room_no FROM Room WHERE Availablility
= 'Available'")) {

        boolean hasRooms = false;
        while (rs.next()) {
            roomComboBox.addItem(rs.getString("room_no"));
            hasRooms = true;
        }
```

```
        if (!hasRooms) {
            roomComboBox.addItem("No Available Rooms");
            roomComboBox.setEnabled(false);
        }

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private boolean isRoomOccupied(Connection con, String roomNumber) throws SQLException {
    String sql = "SELECT Availablility FROM Room WHERE room_no = ?";
    try (PreparedStatement pst = con.prepareStatement(sql)) {
        pst.setString(1, roomNumber);
        try (ResultSet rs = pst.executeQuery()) {
            return rs.next() && rs.getString("Availablility").equalsIgnoreCase("Occupied");
        }
    }
}

private void updateRoomStatus(Connection con, String roomNumber) throws SQLException {
    String sql = "UPDATE Room SET Availablility = 'Occupied' WHERE room_no = ?";
    try (PreparedStatement pst = con.prepareStatement(sql)) {
        pst.setString(1, roomNumber);
        pst.executeUpdate();
    }
}

public static void main(String[] args) {
    new AddPatient();
} }
```

Concepts Used:

- Swing
- Event Handling
- JDBC

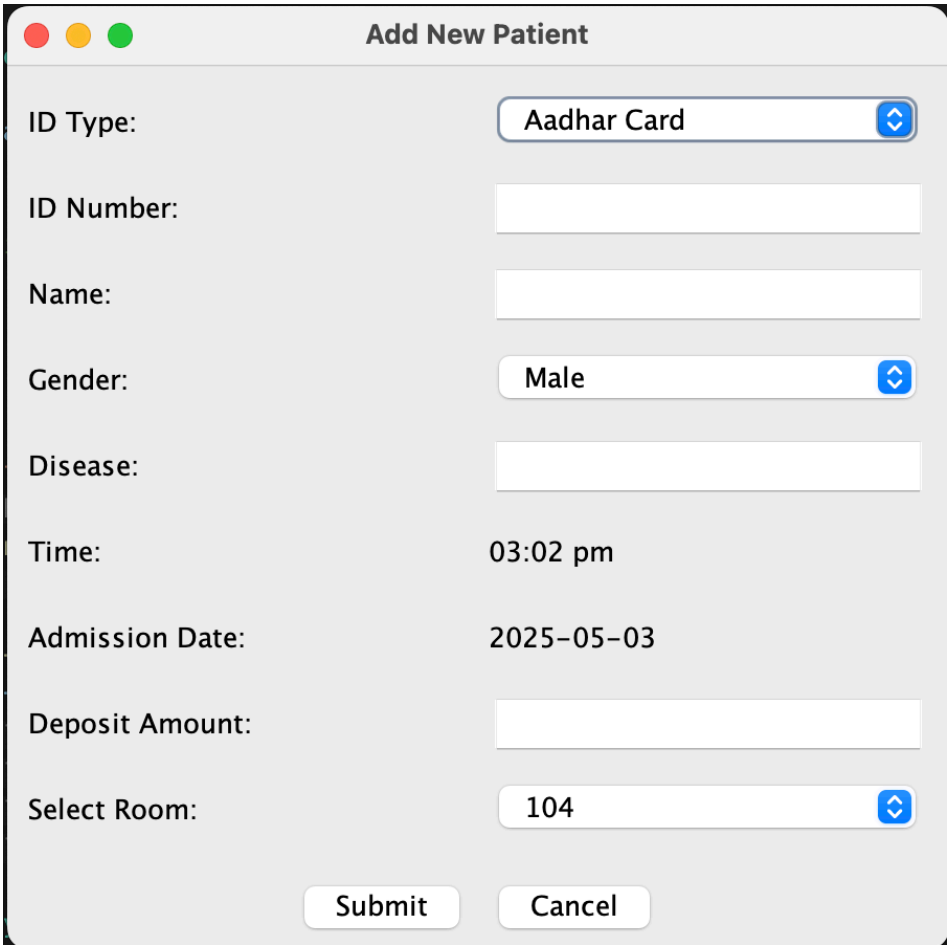
- Layout Managers
- LocalDate & LocalTime
- Exception Handling
- Form Validation
- Database Query Execution

Explanation:

This code generates a form in Java Swing to insert a new patient into the hospital system. It gathers patient information such as ID, name, disease, gender, room, deposit, and current date & time.

On clicking Submit, it validates if all the fields are entered and if the room chosen is available. It then adds the patient's information into the database and changes the room status to "Occupied". On clicking Cancel, the window closes.

Output:



4. Room Module

Code:

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
public class Room extends JFrame implements Displayable { private JTable roomTable; private
DefaultTableModel tableModel; private JButton backButton;

private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";
public Room() {
    setTitle("Room Information");
    setSize(600, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null); // Center the window

    tableModel = new DefaultTableModel();
    roomTable = new JTable(tableModel);
    tableModel.addColumn("Room No");
    tableModel.addColumn("Availability");
    tableModel.addColumn("Price");
    tableModel.addColumn("Room Type");

    add(roomTable, BorderLayout.CENTER);

    backButton = new JButton("Back to HomeScreen");
    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
        }
    });

    JPanel buttonPanel = new JPanel();
    buttonPanel.add(backButton);
```

```
    add(buttonPanel, BorderLayout.SOUTH);

    displayData(); // Calls the interface method

    setVisible(true);
}

public void displayData() {
    String query = "SELECT * FROM Room";
    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement ps = con.prepareStatement(query);
         ResultSet rs = ps.executeQuery()) {

        while (rs.next()) {
            tableModel.addRow(new Object[]{
                rs.getString("room_no"),
                rs.getString("Availablility"),
                rs.getString("Price"),
                rs.getString("Room_Type")
            });
        }

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}
public static void main(String[] args) {
    new Room();
} }
```

Concepts Used:

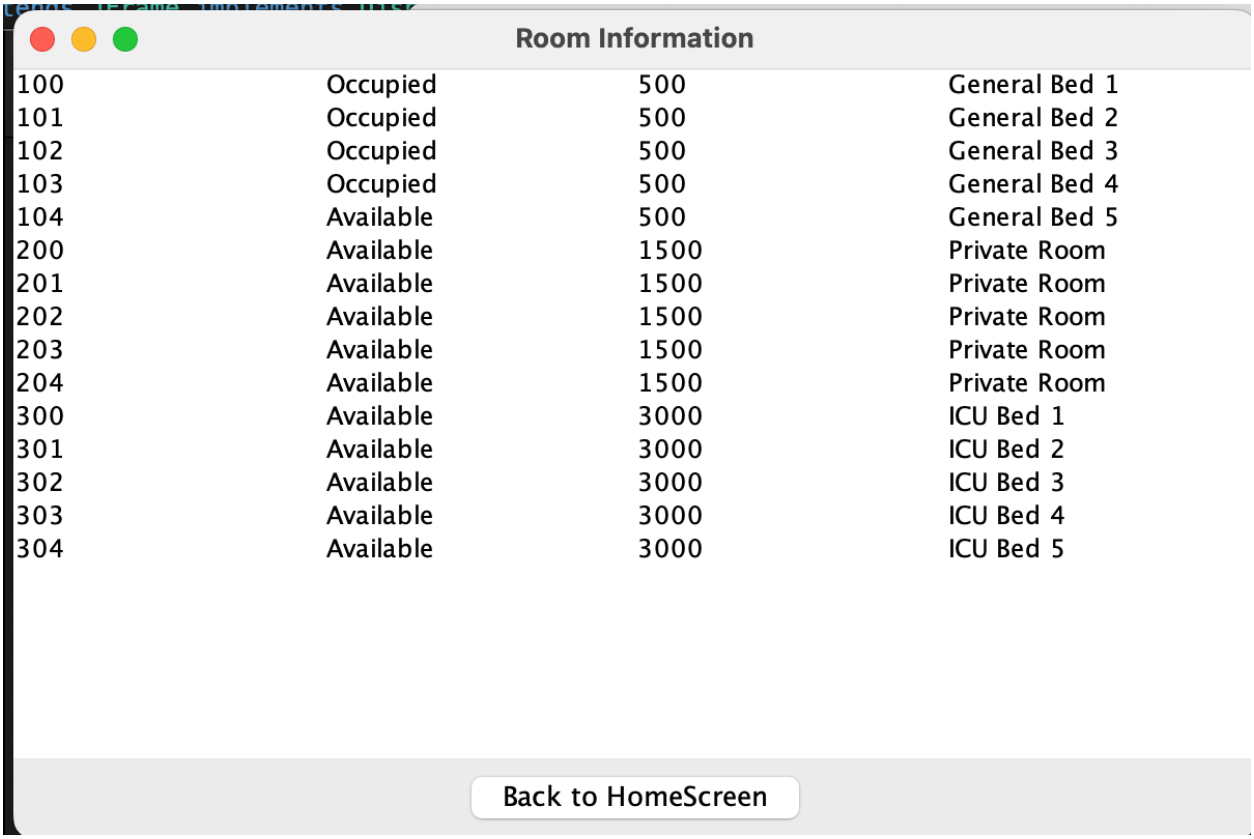
- Swing
- JDBC
- Event Handling
- DefaultTableModel

- PreparedStatement
- ResultSet
- Database Query Execution
- Interface Implementation

Explanation:

This code generates a window showing room details from a hospital management system with Swing as the user interface. It connects to a MySQL database through JDBC, fetches room details (such as room number, availability, price, and type), and shows them in a JTable. The data is inserted into the table dynamically through DefaultTableModel. There is also a back button to go back to the home screen. Event handling is used for button actions, and PreparedStatement is used to safely execute the SQL query.

Output:



5. Department Module

Code:

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
```

```
import java.awt.;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
public class Department extends JFrame implements Displayable { private JTable
departmentTable; private DefaultTableModel tableModel; private JButton backButton;
private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public Department() {
    setTitle("Hospital Departments");
    setSize(700, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null); // Center the window

    tableModel = new DefaultTableModel();
    tableModel.addColumn("Dept ID");
    tableModel.addColumn("Department Name");
    tableModel.addColumn("Head of Department");
    tableModel.addColumn("Contact");
    tableModel.addColumn("Location");
    tableModel.addColumn("Doctors");
    tableModel.addColumn("Nurses");

    departmentTable = new JTable(tableModel);
    JScrollPane scrollPane = new JScrollPane(departmentTable);
    add(scrollPane, BorderLayout.CENTER);

    // Back Button
    backButton = new JButton("Back to HomeScreen");
    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
        }
    });
```

```
    JPanel buttonPanel = new JPanel();
    buttonPanel.add(backButton);
    add(buttonPanel, BorderLayout.SOUTH);

    displayData();
    setVisible(true);
}

public void displayData() {
    String query = "SELECT * FROM department";

    try (
        Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
        PreparedStatement ps = con.prepareStatement(query);
        ResultSet rs = ps.executeQuery()
    ) {

        while (rs.next()) {
            tableModel.addRow(new Object[]{
                rs.getString("dept_id"),
                rs.getString("dept_name"),
                rs.getString("head_of_department"),
                rs.getString("contact_number"),
                rs.getString("location"),
                rs.getInt("no_of_doctors"),
                rs.getInt("no_of_nurses")
            });
        }

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}
```

```
public static void main(String[] args) {
    new Department();
} }
```

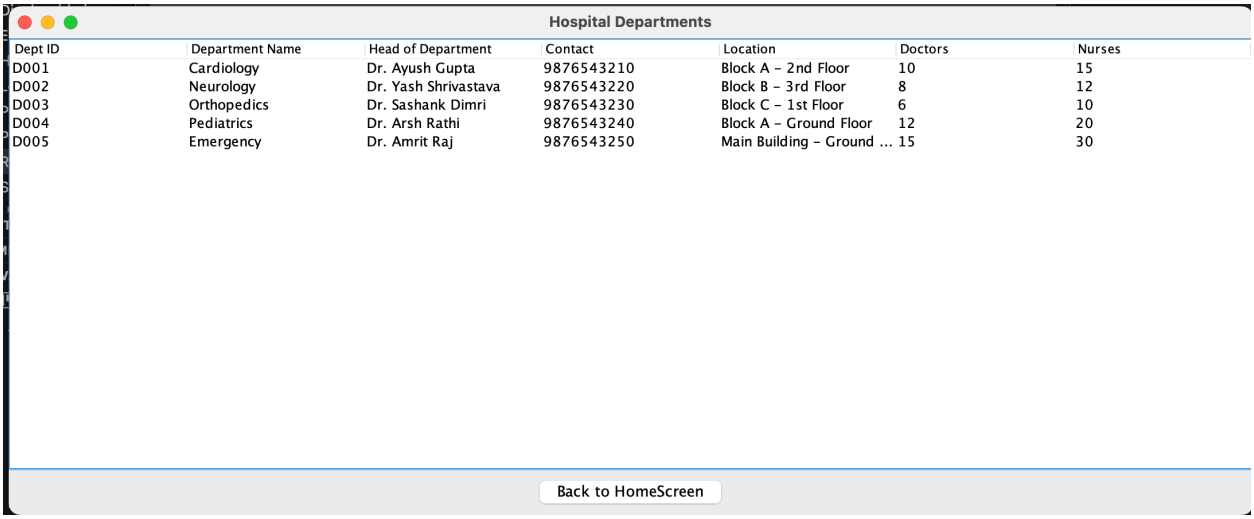
Concepts Used:

- Swing
- JDBC
- DefaultTableModel
- Event Handling
- PreparedStatement
- Inheritance

Explanation:

This code creates a window using Swing to display hospital department details in a JTable. It connects to a MySQL database using JDBC, retrieves department data (such as department ID, name, head, contact, location, and number of doctors and nurses), and displays it in the table using DefaultTableModel. Event handling is implemented for the back button, allowing the user to return to the home screen. PreparedStatement is used to execute the SQL query safely.

Output:



6. Displayable Module

Code:

```
public interface Displayable {
    void displayData();
}
```

Concepts Used:

- Interface

Explanation:

An interface in Java defines a contract that a class must follow. In this case, the Displayable interface declares a method displayData() that any class implementing it must define. This ensures that the classes using this interface will have a displayData() method to show or handle data, promoting consistency across different classes.

7. PatientInfo Module

Code:

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
public class PatientInfo extends JFrame implements Displayable { private JTable patientTable;
private DefaultTableModel tableModel; private JButton backButton;
private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public PatientInfo() {
    setTitle("Patient Information");
    setSize(1000, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout());

    tableModel = new DefaultTableModel();
    tableModel.addColumn("ID Type");
    tableModel.addColumn("ID Number");
```

```
    tableModel.addColumn("Name");
    tableModel.addColumn("Gender");
    tableModel.addColumn("Disease");
    tableModel.addColumn("Room Number");
    tableModel.addColumn("Time Admitted");
    tableModel.addColumn("Deposit");
    tableModel.addColumn("Admission Date");

    patientTable = new JTable(tableModel);
    JScrollPane scrollPane = new JScrollPane(patientTable);
    add(scrollPane, BorderLayout.CENTER);

    backButton = new JButton("Back to HomeScreen");
    backButton.setFont(new Font("Arial", Font.BOLD, 14));
    backButton.setBackground(Color.LIGHT_GRAY);
    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen(); // Open HomeScreen
            dispose(); // Close PatientInfo window
        }
    });

    JPanel buttonPanel = new JPanel();
    buttonPanel.add(backButton);
    add(buttonPanel, BorderLayout.SOUTH);

    displayData();

    setVisible(true);
}

public void displayData() {
    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         Statement stmt = con.createStatement();
         ResultSet rs = stmt.executeQuery("SELECT * FROM patient_info")) {
```

```
        while (rs.next()) {
            tableModel.addRow(new Object[]{
                rs.getString("ID"),
                rs.getString("Number"),
                rs.getString("Name"),
                rs.getString("Gender"),
                rs.getString("Patient_Disease"),
                rs.getString("Room_Number"),
                rs.getString("Time"),
                rs.getString("Deposit"),
                rs.getDate("admission_date")
            });
        }

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

public static void main(String[] args) {
    new PatientInfo();
}

}
```

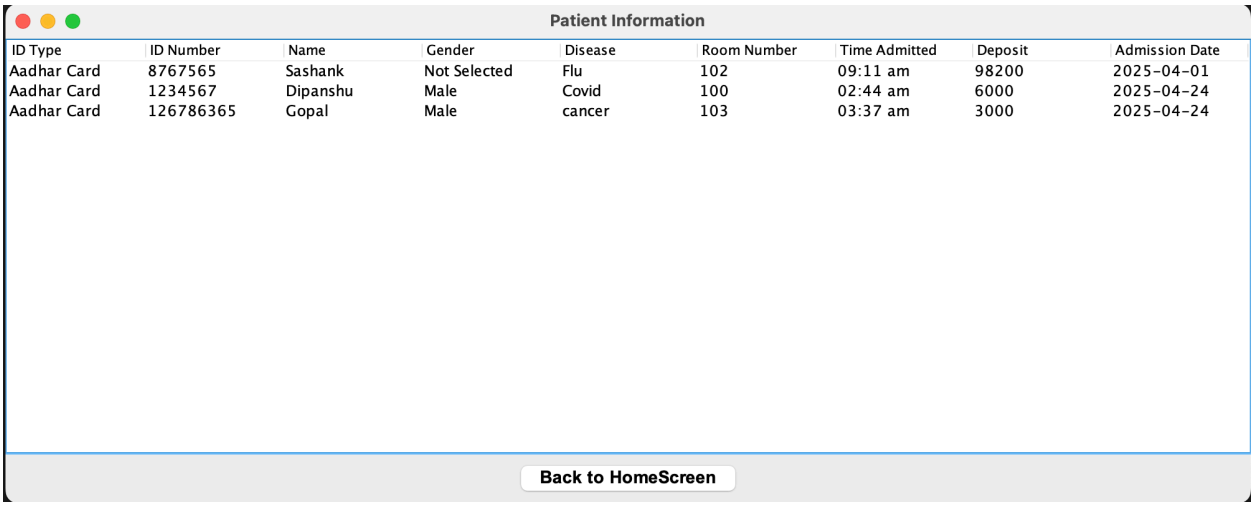
Concepts Used:

- JTable (Swing)
- DefaultTableModel (Swing)
- JButton (Swing)
- ActionListener (Event Handling)
- JPanel (Swing)
- Database Connection (JDBC)
- ResultSet (JDBC)
- SQL SELECT Query (JDBC)
- Exception Handling
- Interface Implementation (Displayable)

Explanation:

The PatientInfo class employs a combination of JDBC and Swing components to show patient information from a MySQL database in a graphical frontend. The class inherits from JFrame to generate a window and implements the Displayable interface to call the displayData() method to get and display the information. The JTable is employed to display the data in a table form, utilizing DefaultTableModel to dynamically manage the structure of the table by adding rows and columns. A JButton is included to enable the user to go back to the home screen, with an ActionListener to process the button click event. The class makes a connection to the MySQL database through JDBC, performs a SELECT query to get patient data, and fills the JTable with data from the ResultSet. Exception handling is provided to handle SQL errors gracefully and give feedback to the user if an error occurs. In summary, this solution integrates database connectivity, GUI elements, and event-driven programming to provide an interactive interface for viewing patient data.

Output:



8. Employee Module

Code:

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
public class Employee extends JFrame { private JTable employeeTable; private
DefaultTableModel tableModel; private JButton backButton;
```

```
private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public Employee() {
    setTitle("Employee Information");
    setSize(800, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout());

    tableModel = new DefaultTableModel();
    tableModel.addColumn("Emp ID");
    tableModel.addColumn("Name");
    tableModel.addColumn("Designation");
    tableModel.addColumn("Dept ID");
    tableModel.addColumn("Contact");
    tableModel.addColumn("Salary");

    employeeTable = new JTable(tableModel);
    JScrollPane scrollPane = new JScrollPane(employeeTable);
    add(scrollPane, BorderLayout.CENTER);

    backButton = new JButton("Back to HomeScreen");
    backButton.setFont(new Font("Arial", Font.BOLD, 14));
    backButton.setBackground(Color.LIGHT_GRAY);
    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
        }
    });

    JPanel buttonPanel = new JPanel();
    buttonPanel.add(backButton);
    add(buttonPanel, BorderLayout.SOUTH);
```

```
    loadEmployeeData();
    setVisible(true);
}

private void loadEmployeeData() {
        String query = "SELECT emp_id, name, designation, department_id, contact_number,
salary FROM employee";

        try (
            Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
            PreparedStatement ps = con.prepareStatement(query);
            ResultSet rs = ps.executeQuery()
        ) {
            while (rs.next()) {
                tableModel.addRow(new Object[]{
                    rs.getString("emp_id"),
                    rs.getString("name"),
                    rs.getString("designation"),
                    rs.getString("department_id"),
                    rs.getString("contact_number"),
                    rs.getDouble("salary")
                });
            }

        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Database Connection Failed: " + e.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
        }
    }

public static void main(String[] args) {
    new Employee();
}}
```

Concepts Used:

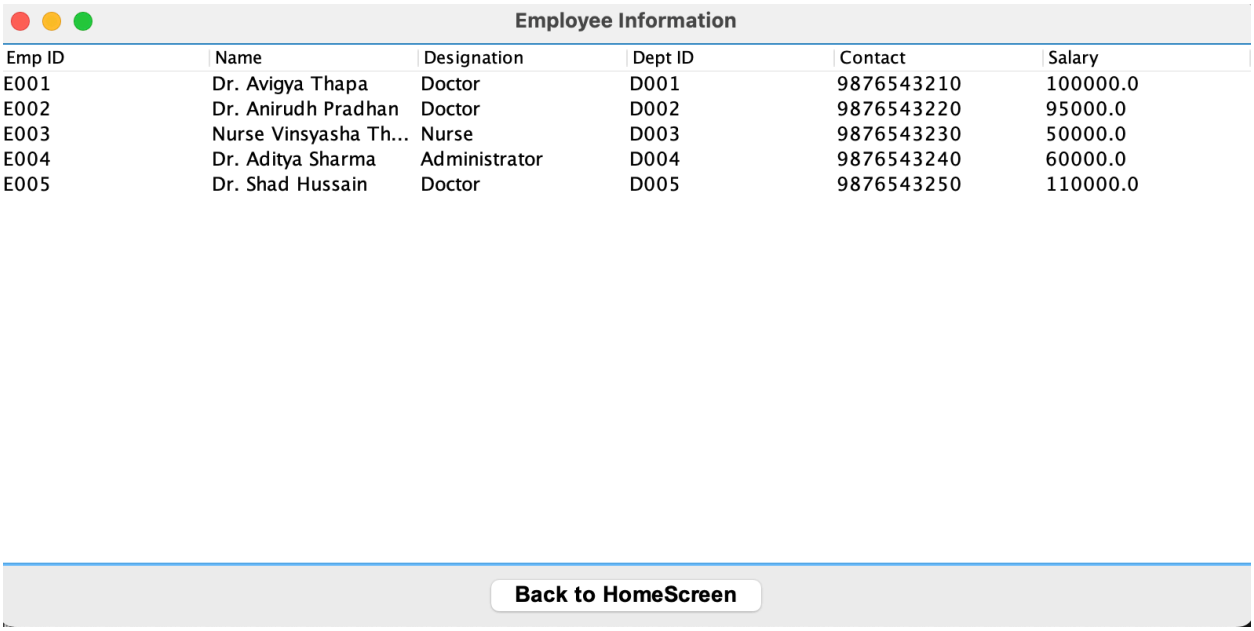
- Swing (GUI Framework)

- Event-Driven Programming
- JDBC (Java Database Connectivity)
- SQL (Structured Query Language)
- DefaultTableModel
- PreparedStatement
- ResultSet

Explanation:
In the Employee class, a graphical user interface (GUI) is created to display employee information from a MySQL database. This class extends JFrame, providing a window for the application. The JTable and DefaultTableModel are used to create a table structure that dynamically displays employee data, including columns like Employee ID, Name, Designation, Department ID, Contact, and Salary. The JScrollPane ensures that the table is scrollable, which is useful when there are many records. A JButton is also included to allow the user to navigate back to the home screen, and the action is handled using an ActionListener.

The class connects to a MySQL database using JDBC, with the database URL, username, and password stored as constants. In the loadEmployeeData() method, a PreparedStatement is used to execute a SELECT query, which retrieves employee data from the database. Each row from the ResultSet is added to the table. Error handling with SQLExceptionensures that if the database connection fails or an issue arises during data retrieval, a message is displayed to the user. This combination of Swing components, JDBC, and event-driven programming makes it easy for users to view employee details interactively within the application.

Output:



9. PatientDischarge Module

Code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
import java.util.ArrayList;

public class PatientDischarge extends JFrame {
    private JComboBox<String> patientIdComboBox;
    private JLabel nameLabel = new JLabel("N/A");
    private JLabel roomLabel = new JLabel("N/A");
    private JButton checkButton = new JButton("Check");
    private JButton dischargeButton = new JButton("Discharge");
    private JButton backButton = new JButton("Back to HomeScreen");

    private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "";
```

```
public PatientDischarge() {
    setTitle("CHECK-OUT");
    setSize(400, 300);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new GridLayout(5, 2, 10, 10));

    add(new JLabel("Select Patient ID:"));
    patientIdComboBox = new JComboBox<>();
    loadPatientIDs();
    add(patientIdComboBox);

    add(new JLabel("Patient Name:"));
    add(nameLabel);

    add(new JLabel("Room Number:"));
    add(roomLabel);

    checkButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            loadPatientDetails();
        }
    });
    add(checkButton);

    dischargeButton.setEnabled(false);
    dischargeButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            dischargePatient();
        }
    });
    add(dischargeButton);

    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
```

```
            }
        });
        add(backButton);

        setVisible(true);
    }

    private void loadPatientIDs() {
        String query = "SELECT Number FROM patient_info";

        try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
             PreparedStatement pst = con.prepareStatement(query);
             ResultSet rs = pst.executeQuery()) {

            ArrayList<String> ids = new ArrayList<>();
            while (rs.next()) {
                ids.add(rs.getString("Number"));
            }

            if (ids.isEmpty()) {
                patientIdComboBox.addItem("No Patients Available");
                patientIdComboBox.setEnabled(false);
            } else {
                for (String id : ids) {
                    patientIdComboBox.addItem(id);
                }
            }

        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }

    private void loadPatientDetails() {
        String selectedID = (String) patientIdComboBox.getSelectedItem();
```

```
        if (selectedID == null || selectedID.equals("No Patients Available")) {
            return;
        }

        try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
             PreparedStatement pst = con.prepareStatement("SELECT Name, Room_Number FROM
patient_info WHERE Number = ?")) {
            pst.setString(1, selectedID);
            try (ResultSet rs = pst.executeQuery()) {
                if (rs.next()) {
                    nameLabel.setText(rs.getString("Name"));
                    roomLabel.setText(rs.getString("Room_Number"));
                    dischargeButton.setEnabled(true);
                } else {
                    nameLabel.setText("N/A");
                    roomLabel.setText("N/A");
                    dischargeButton.setEnabled(false);
                }
            }

        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }

    private void dischargePatient() {
        String selectedID = (String) patientIdComboBox.getSelectedItem();
        String roomNumber = roomLabel.getText();

        if (selectedID == null || roomNumber.equals("N/A")) {
            JOptionPane.showMessageDialog(this, "Invalid selection.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }
```

```
      try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
          try (PreparedStatement pst = con.prepareStatement("DELETE FROM patient_info
WHERE Number = ?")) {
              pst.setString(1, selectedID);
              pst.executeUpdate();
          }

          try (PreparedStatement pst = con.prepareStatement("UPDATE Room SET Availablility =
'Available' WHERE room_no = ?")) {
              pst.setString(1, roomNumber);
              pst.executeUpdate();
          }

          JOptionPane.showMessageDialog(this, "Patient Discharged Successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);
          dispose();

      } catch (SQLException e) {
          JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
      }
   }

   public static void main(String[] args) {
      new PatientDischarge(); } }
```

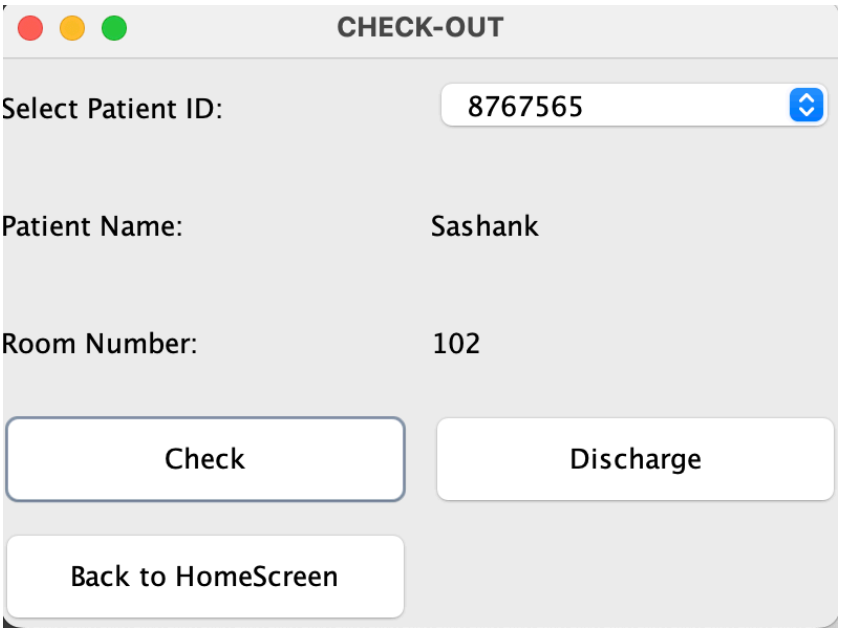
Concepts Used:

- Java Swing
- Event Handling
- JDBC
- SQL Queries
- ComboBox
- Layout Management
- ArrayList
- Exception Handling
- Conditional Statements
- Prepared Statements

Explanation:

This program uses Java Swing to create a graphical user interface (GUI) that allows users to discharge patients from a hospital system. The layout is managed using GridLayout, and components like JLabel, JComboBox, and JButton are used to display and interact with data. Event handling is implemented through ActionListener to respond to button clicks. The program connects to a MySQL database using JDBC, retrieves patient IDs and details using SQL queries, and interacts securely with the database using PreparedStatement to prevent SQL injection. It utilizes ArrayList to store and manage the list of patient IDs temporarily. Conditional statements are used to check for valid input and control the program flow. Exception handling ensures that any errors during database operations are caught and reported to the user in a friendly way.

Output:



10. UpdatePatientDetails Module

Code:

import javax.swing.;
import java.awt.;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
public class UpdatePatientDetail extends JFrame { private JComboBox patientNameComboBox; private JLabel timeLabel, dateLabel, amtPaidLabel, pendingAmtLabel; private JTextField totalAmountField, additionalDepositField; private JButton checkButton, updateTotalButton, makeDepositButton, backButton;

```
private static final String DB_URL = "jdbc:mysql://localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public UpdatePatientDetail() {
    setTitle("Update Patient Details");
    setSize(500, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout(10, 10));

    JPanel formPanel = new JPanel(new GridLayout(7, 2, 10, 10));
    formPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));



    formPanel.add(new JLabel("Select Patient:"));
    patientNameComboBox = new JComboBox<>();
    loadPatientNames();
    formPanel.add(patientNameComboBox);

formPanel.add(new JLabel("Check-in Time:")); timeLabel = new JLabel("N/A");
formPanel.add(timeLabel);
formPanel.add(new JLabel("Check-in Date:"));
dateLabel = new JLabel("N/A");
formPanel.add(dateLabel);

    formPanel.add(new JLabel("Amount Paid:"));
    amtPaidLabel = new JLabel("N/A");
    formPanel.add(amtPaidLabel);

    formPanel.add(new JLabel("Enter Total Bill Amount:"));
    totalAmountField = new JTextField();
    formPanel.add(totalAmountField);

    formPanel.add(new JLabel("Remaining Amount:"));
    pendingAmtLabel = new JLabel("N/A");
    formPanel.add(pendingAmtLabel);
```

```
formPanel.add(new JLabel("Enter Additional Deposit:"));
additionalDepositField = new JTextField();
additionalDepositField.setEnabled(false);
formPanel.add(additionalDepositField);

JPanel buttonPanel = new JPanel(new GridLayout(1, 4, 10, 10));
checkButton = new JButton("Check");
updateTotalButton = new JButton("Update Total Bill");
makeDepositButton = new JButton("Make Additional Deposit");
backButton = new JButton("Back to HomeScreen");

buttonPanel.add(checkButton);
buttonPanel.add(updateTotalButton);
buttonPanel.add(makeDepositButton);
buttonPanel.add(backButton);

add(formPanel, BorderLayout.CENTER);
add(buttonPanel, BorderLayout.SOUTH);

checkButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        loadPatientDetails();
    }
});

updateTotalButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        updateTotalBill();
    }
});

makeDepositButton.setEnabled(false);
makeDepositButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        makeAdditionalDeposit();
    }
```

```
    });

    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
        }
    });

    setVisible(true);
}

private void loadPatientNames() {
    String query = "SELECT Name FROM patient_info";
    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement pst = con.prepareStatement(query);
         ResultSet rs = pst.executeQuery()) {

        while (rs.next()) {
            patientNameComboBox.addItem(rs.getString("Name"));
        }

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void loadPatientDetails() {
    String selectedName = (String) patientNameComboBox.getSelectedItem();
    if (selectedName == null) return;

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement pst = con.prepareStatement(
                 "SELECT Time, admission_date, Deposit, Total_Bill, Pending_Amount FROM
```

```
patient_info WHERE Name = ?")) {
        pst.setString(1, selectedName);
        try (ResultSet rs = pst.executeQuery()) {
            if (rs.next()) {
                timeLabel.setText(rs.getString("Time"));
                dateLabel.setText(rs.getString("admission_date"));
                amtPaidLabel.setText(rs.getString("Deposit"));
                totalAmountField.setText(rs.getString("Total_Bill"));
                pendingAmtLabel.setText(rs.getString("Pending_Amount"));

                // Enable deposit field and button after total bill is set
                additionalDepositField.setEnabled(true);
                makeDepositButton.setEnabled(true);
            }
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void updateTotalBill() {
    String selectedName = (String) patientNameComboBox.getSelectedItem();
    if (selectedName == null || totalAmountField.getText().trim().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Enter a valid total bill amount.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement pst = con.prepareStatement(
                 "UPDATE patient_info SET Total_Bill = ?, Pending_Amount = Total_Bill - Deposit
WHERE Name = ?")) {

        double totalAmount = Double.parseDouble(totalAmountField.getText().trim());
        pst.setDouble(1, totalAmount);
```

```
        pst.setString(2, selectedName);
        pst.executeUpdate();

        loadPatientDetails();
        JOptionPane.showMessageDialog(this, "Total Bill & Pending Amount Updated!",
"Success", JOptionPane.INFORMATION_MESSAGE);

    } catch (SQLException | NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void makeAdditionalDeposit() {
    String selectedName = (String) patientNameComboBox.getSelectedItem();
    if (selectedName == null || additionalDepositField.getText().trim().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Enter a valid deposit amount.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement updateDeposit = con.prepareStatement(
                 "UPDATE patient_info SET Deposit = Deposit + ?, Pending_Amount = Total_Bill -
Deposit WHERE Name = ?")) {

        double additionalDeposit = Double.parseDouble(additionalDepositField.getText().trim());
        updateDeposit.setDouble(1, additionalDeposit);
        updateDeposit.setString(2, selectedName);
        updateDeposit.executeUpdate();

        loadPatientDetails();
        additionalDepositField.setText("");
        JOptionPane.showMessageDialog(this, "Deposit Updated Successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);
```

```
    } catch (SQLException | NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

public static void main(String[] args) {
    new UpdatePatientDetail();
} }
```

Concepts used:

- Java Swing
- JFrame, JPanel, JLabel, JComboBox, JTextField, JButton
- Layout Managers (BorderLayout, GridLayout)
- Event Handling (ActionListener)
- JDBC (Java Database Connectivity)
- SQL Queries (SELECT, UPDATE)
- PreparedStatement
- ResultSet
- Exception Handling (try-catch)
- Input Validation
- String and Number Parsing (Double.parseDouble)
- Dynamic GUI Updates
- Calling Methods from Event Handlers
- Use of Constants (static final)

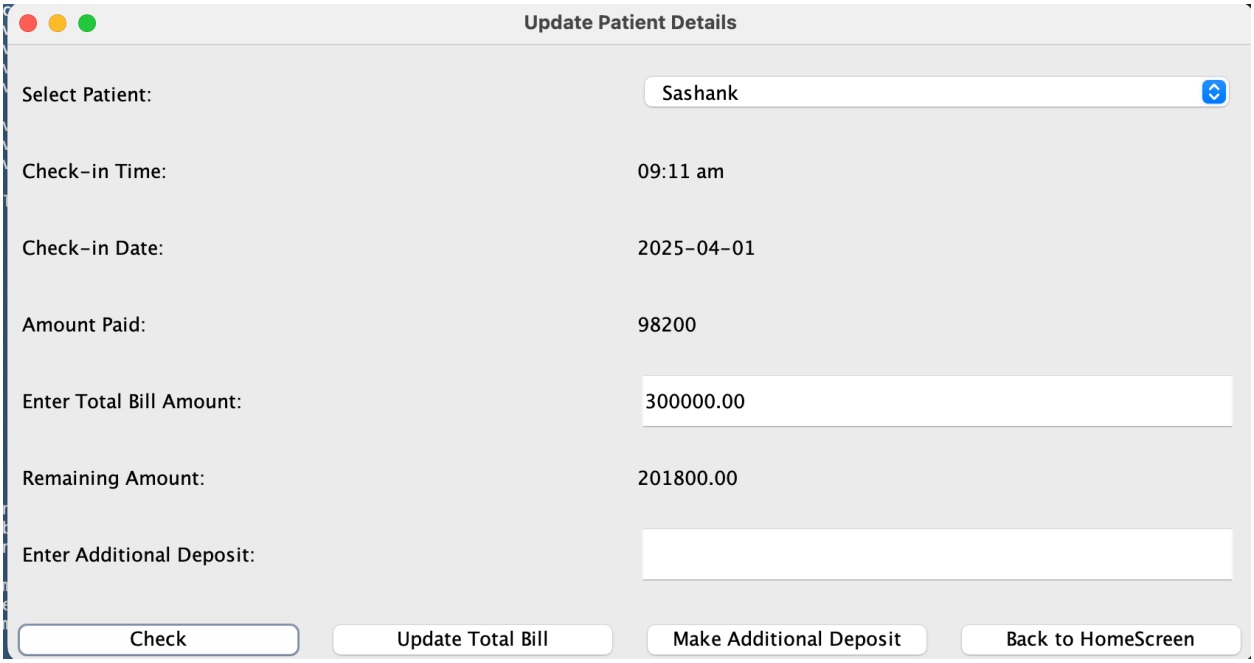
Explanation:

This Java code utilizes a number of important concepts to create a hospital management module that modifies patient billing information. The graphical user interface (GUI) is created with Java Swing, with elements like JFrame for the main window, JPanel for layout organization, and UI components like JLabel, JComboBox, JTextField, and JButton to show data and communicate with the user. The organization is maintained through BorderLayout and GridLayout, which organize the interface in a well-structured and neat way.

User actions like button clicks are processed through event handling through the ActionListener interface so that the application can dynamically respond to user inputs like verifying patient information, computing the total bill, or placing further deposits. Backend connection is made possible using JDBC (Java Database Connectivity) to enable the program to interact with a

MySQL database. SQL queries are executed securely and efficiently using the PreparedStatement, while parameterized statements are supported for preventing SQL injection. The data retrieved from the database is stored and accessed by the ResultSet object. SQL queries like SELECT and UPDATE are utilized to retrieve and update patient records, such as fields like total bill, deposit amount, and pending dues. Combined, these ideas enable a smooth and interactive patient billing update system.

Output:



11. SearchRoom Module

Code:

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
public class SearchRoom extends JFrame implements Displayable { private JComboBox
statusComboBox; private JTable roomTable; private DefaultTableModel tableModel; private
JButton searchButton, backButton;private static final String DB_URL = "jdbc:mysql://
```

```
localhost:3306/hospital_Management";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public SearchRoom() {
    setTitle("Search Room Availability");
    setSize(600, 400);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout(10, 10));



    JPanel panel = new JPanel();
    panel.add(new JLabel("Room Status:"));

    statusComboBox = new JComboBox<>(new String[]{"Available", "Occupied"});
    panel.add(statusComboBox);

    searchButton = new JButton("Search");
    panel.add(searchButton);

    add(panel, BorderLayout.NORTH);

    tableModel = new DefaultTableModel();
    tableModel.addColumn("Room No");
    tableModel.addColumn("Availability");
    tableModel.addColumn("Price");
    tableModel.addColumn("Room Type");

    roomTable = new JTable(tableModel);
    JScrollPane scrollPane = new JScrollPane(roomTable);
    add(scrollPane, BorderLayout.CENTER);

    JPanel buttonPanel = new JPanel();
    backButton = new JButton("Back to Home");
    buttonPanel.add(backButton);
```

```
    add(buttonPanel, BorderLayout.SOUTH);

    searchButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            displayData();
        }
    });

    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new HomeScreen();
            dispose();
        }
    });

    setVisible(true);
}
public void displayData() {
    String selectedStatus = (String) statusComboBox.getSelectedItem();
    if (selectedStatus == null) return;

    tableModel.setRowCount(0);

    try (Connection con = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         PreparedStatement pst = con.prepareStatement("SELECT * FROM Room WHERE
Availablility = ?")) {
        pst.setString(1, selectedStatus);
        try (ResultSet rs = pst.executeQuery()) {
            while (rs.next()) {
                tableModel.addRow(new Object[]{
                        rs.getString("room_no"),
                        rs.getString("Availablility"),
                        rs.getString("Price"),
                        rs.getString("Room_Type")
                });
            }
```

```
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

public static void main(String[] args) {
    new SearchRoom();
}
 }
```

Concepts Used:

- Java Swing
- JFrame, JPanel, JLabel, JButton, JComboBox, JTable
- Layout Managers
- Event Handling
- JDBC
- PreparedStatement and ResultSet
- Exception Handling
- Database Connection
- DefaultTableModel
- Modular Programming
- Encapsulation
- Polymorphism

Explanation:

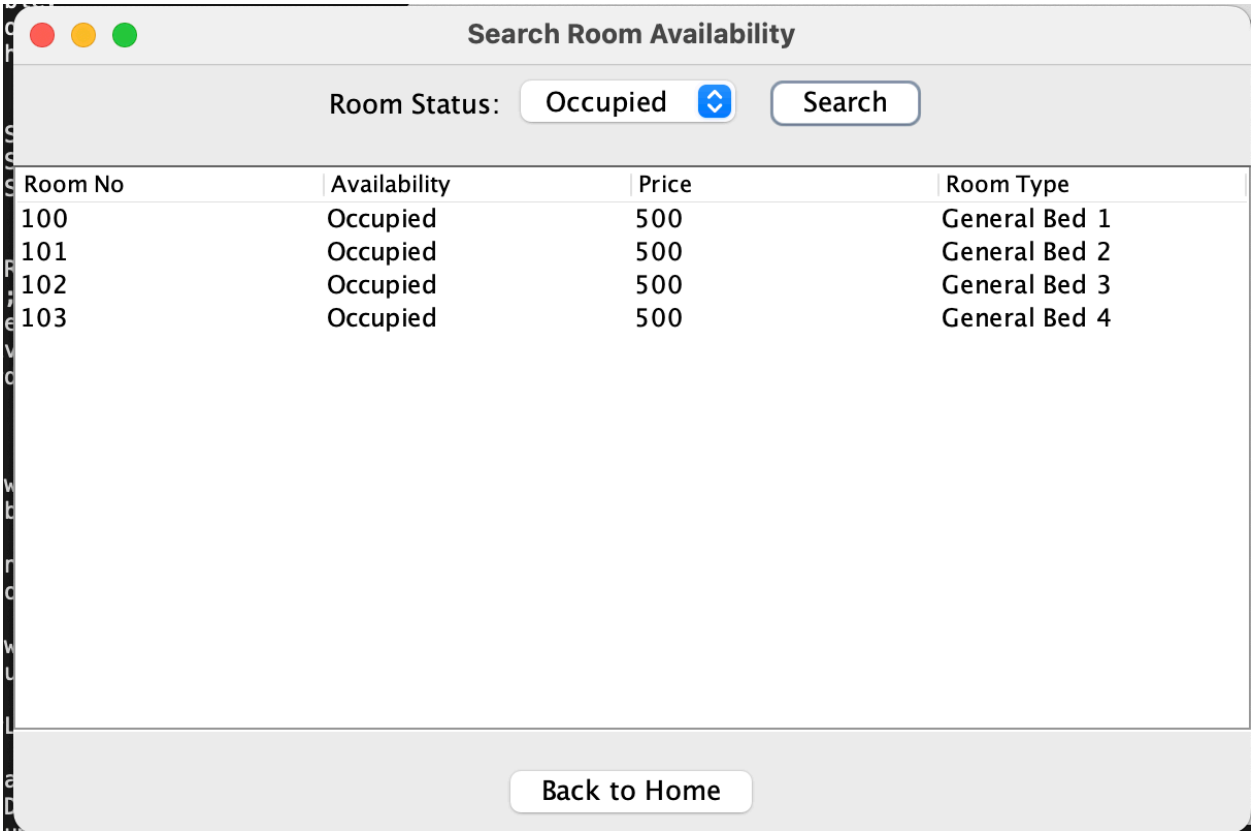
This Java program, SearchRoom, is part of a hospital management system that allows users to check the availability of rooms based on their status—either “Available” or “Occupied.” It utilizes Java Swing to create a graphical user interface (GUI), consisting of components such as JFrame, JPanel, JLabel, JComboBox, JButton, and JTable. The combo box lets users select the desired room status, and the table displays the results of the search.

When the Search button is clicked, an ActionListener triggers the displayData() method. This method establishes a connection to a MySQL database using JDBC (Java Database Connectivity). A PreparedStatement is used to execute a parameterized SELECT query that fetches records from the Room table where the Availability matches the user’s selection.

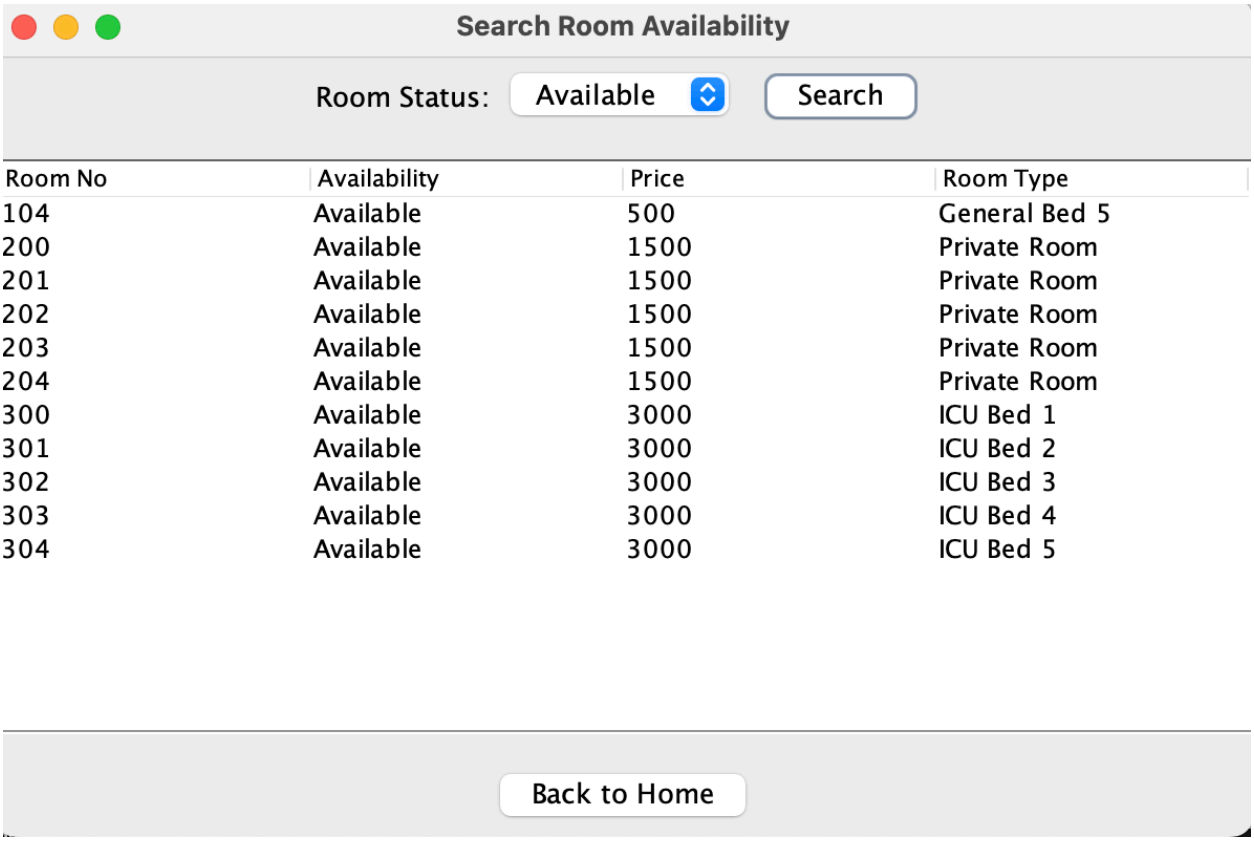
The results are then populated into the JTable using a DefaultTableModel, which dynamically updates the displayed data. A Back to Home button is also provided, which navigates the user back to the home screen.

This program demonstrates important concepts in Java such as GUI development, event handling, and secure database access using JDBC, making it an effective and user-friendly module for real-time room status tracking in a hospital environment.

Output: (Occupied)



Output: (Available)



Conclusion:

In this project, a Hospital Management System was developed to automate and simplify hospital operations such as managing patient records, checking room availability, and handling billing tasks. The application was built using Java Swing for creating an interactive and user-friendly interface, while JDBC was used to connect the system to a MySQL database for secure and efficient data management. The system includes core features like managing patient information, searching for rooms based on their availability status (Available/Occupied), and updating billing details such as total charges, paid amount, and pending balance. These functionalities help hospital staff manage day-to-day tasks more effectively.

Important programming concepts of object-oriented programming (OOP) — including encapsulation, modular design, and code reusability — were applied throughout the development. JDBC's use of prepared statements and result sets ensured safe and reliable database interactions.In conclusion, the project effectively meets the essential needs of a hospital management system, offering a user-friendly and secure solution for staff operations. It also lays a strong foundation for future enhancements, such as user authentication, analytics, and integration with larger healthcare systems.