# HOTEL MANAGEMENT SYSTEM

**A MINI PROJECT REPORT**

**Submitted by**

| | |
|---|---|
| **JYOTHI SAKTHI H** | **220701113** |
| **M AKASH** | **220701502** |

In partial fulfillment for the award of the degree of

BACHELOR OF

ENGINEERING IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE

(AUTONOMOUS) THANDALAM

CHENNAI-602105

2023 - 24

## BONAFIDE CERTIFICATE

Certified that this project report "**HOTEL MANAGEMENT SYSTEM**" is the Bonafide work of **"JYOTHI SAKTHI H (220701113),M AKASH (220701502)"** who carried out the project work under my supervision.

**Submitted for the Practical Examination held on _______________________**

| | |
|---|---|
| **SIGNATURE** | **SIGNATURE** |
| **Dr.R.SABITHA** | **Dr G DHARANI DEVI** |
| **Professor and II Year Academic Head** | **Assistant Professor (SG),** |
| **Computer Science and Engineering,** | **Computer Science and Engineering,** |
| **Rajalakshmi Engineering College (Autonomous),** | **Rajalakshmi Engineering College, (Autonomous),** |
| **Thandalam, Chennai - 602 105** | **Thandalam, Chennai - 602 105** |

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

## ABSTRACT

The hospitality industry continuously seeks innovative solutions to enhance the efficiency and accuracy of room reservations and customer service. This project presents an advanced hotel booking management system designed to streamline the booking process, manage customer data securely, and ensure seamless interactions between hotel staff and guests. The system addresses common challenges such as manual data entry errors, booking conflicts, and inefficient customer service by providing a user-friendly interface and robust backend architecture. It incorporates data validation techniques to maintain data integrity and leverages SQL for database management. By automating the booking process, the system aims to improve overall operational efficiency, reduce administrative workload, and enhance the guest experience. This report provides a comprehensive overview of the system's objectives, technological framework, detailed design, implementation, and evaluation, demonstrating its effectiveness in transforming hotel management practices.

**Keywords:** Hotel management system, room reservation, customer data, automation, data validation, SQL database, user interface, operational efficiency, data integrity.

## ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M. THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S. N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department **Dr. P. KUMAR M.E, Ph.D.,** for being ever supporting force during our project work.

We also extend our sincere and hearty thanks to our internal guide, **Dr G DHARANI DEVI** for her valuable guidance and motivation during the completionof this project.

Our sincere thanks to our family members, friends and other staff members of Computer Science & Engineering.

1. **JYOTHI SAKTHI H**
2. **M AKASH**

## TABLE OF CONTENTS

# CHAPTER 1
# INTRODUCTION

## 1.1 INTRODUCTION

Hotel Management System (HMS) is a pivotal solution within the hospitality industry, aimed at enhancing operational efficiency and guest satisfaction. By streamlining reservation management, guest services, and administrative tasks, HMS ensures a seamless experience for both guests and staff.

This system offers a comprehensive solution to challenges faced in managing room bookings, handling inquiries, and maintaining accurate records. Through automation and integration, it simplifies complex tasks such as room allocation, billing, and inventory management.

HMS empowers hotel administrators to make informed decisions, maximize revenue, and improve overall performance. Its user-friendly interface, robust features, and scalability make it indispensable for hotels seeking to elevate guest experiences and stay competitive in the dynamic hospitality landscape.

## 1.2 OBJECTIVES

The development of this hotel booking management system is driven by several key objectives:

- **Automation of the Reservation Process:** To reduce the manual effort involved in booking rooms and managing customer data.
- **Enhancement of Data Security:** To ensure that customer information is securely stored and handled, protecting it from unauthorized access.
- **Reduction of Errors:** To minimize the risk of errors in data entry and booking management, enhancing overall data integrity.
- **User-Friendly Interface**: To provide an intuitive and easy-to-use interface guests, improving the user experience.
- **Scalability:** To ensure the system can handle increasing volumes of data and users as the hotel grows.

## 1.3 MODULES

This Hotel Management System is composed of several interconnected modules, each designed to handle specific aspects of hotel management:

- **Customer Registration Module:** Captures and validates personal details of customers, ensuring all necessary information is collected.
- **Room Booking Module:** Manages the allocation of rooms, booking dates, and customer preferences, ensuring efficient use of hotel resources.
- **Data Validation Module**: Checks that all mandatory fields are filled correctly and validates the format of data such as phone numbers and Aadhar numbers.
- **Database Management Module:** Handles the storage, retrieval, and management of booking and customer data, ensuring data integrity and security.
- **User Interface Module**: Provides a graphical user interface for hotel staff to interact with the system, making the booking process straightforward and efficient.

# CHAPTER 2
# SURVEY OF TECHNOLOGY

## 2.1 SOFTWARE DESCRIPTION

The hotel booking management system relies on a sophisticated stack of software technologies to power its development, deployment, and ongoing operation. This section provides an in-depth overview of the software components that form the backbone of the system.

## 2.2 LANGUAGES

### 2.2.1 SQL

As the primary language for database management, SQL enables the creation, querying, and manipulation of relational databases, ensuring data integrity and consistency throughout the hotel booking process.

### 2.2.2 JAVA

Java serves as the backbone for the system's backend server components and user interface development, offering platform independence, robustness, and a vast ecosystem of libraries and tools for rapid application development.

# CHAPTER 3
# REQUIREMENTS AND ANALYSIS

## 3.1 REQUIREMENT SPECIFICATION

1. **User Management:**
   - User authentication with different roles (admin, receptionist, guest).
   - Admins can manage user accounts.
2. **Booking System**:
   - Users can submit booking details (name, email, phone, etc.).
   - Validation for mandatory fields.
   - Admins can view, edit, and cancel bookings.
3. **Room Management**:
   - System maintains room availability.
   - Users can view available rooms.
   - Admins manage room details (type, capacity, price).

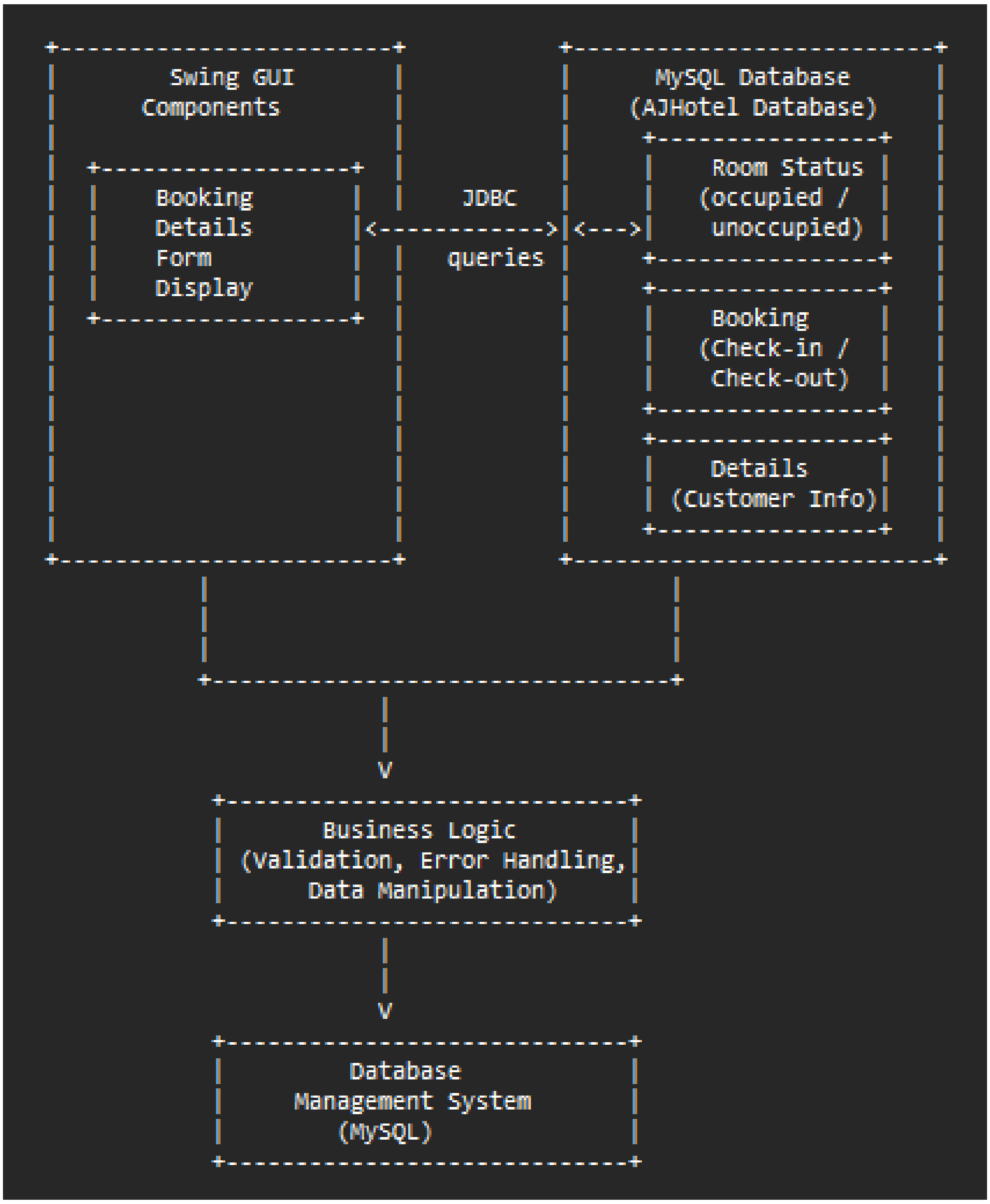
## 3.2 HARDWARE REQUIREMENTS

- A standard PC or server with at least 4GB of RAM and a multi-core processor.
- Adequate storage for database

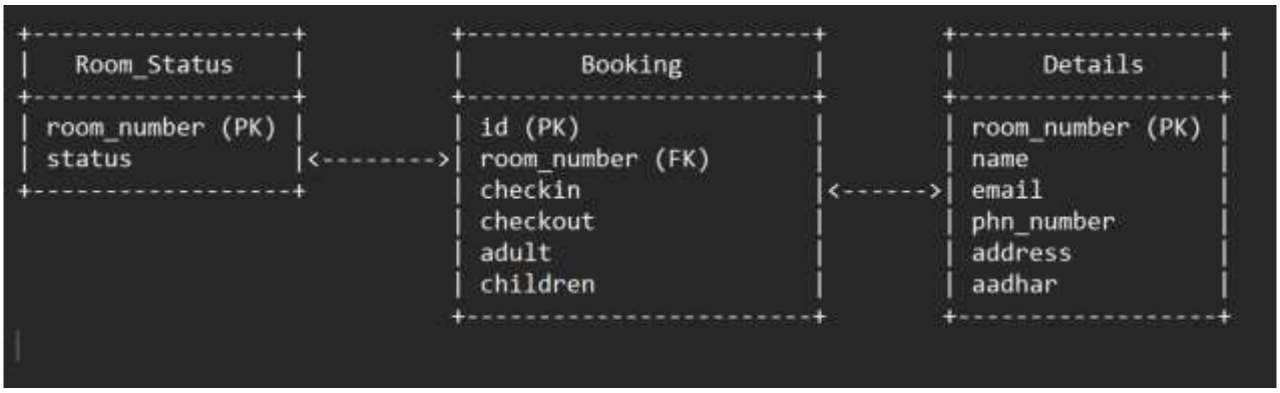
## 3.3 SOFTWARE REQUIREMENTS

- Java Development Kit (JDK)
- MySQL or another RDBMS
- Integrated Development Environment (IDE) like Eclipse or Visual Studio Code

## 3.4 ARCHITECTURE DIAGRAM


Swing GUI Components
Booking Details Form Display
JDBC
queries
MySQL Database (AJHotel Database)
Room Status (occupied / unoccupied)
Booking (Check-in / Check-out)
Details (Customer Info)
Business Logic (Validation, Error Handling, Data Manipulation)
Database Management System (MySQL)

## 3.5 ENTITY-RELATIONSHIP DIAGRAM



```
+-----------------+         +-----------------------+        +-----------------+
|   Room_Status   |         |        Booking        |        |     Details     |
+-----------------+         +-----------------------+        +-----------------+
| room_number (PK) |        | id (PK)               |        | room_number (PK) |
| status          |<-------->| room_number (FK)      |        | name            |
+-----------------+         | checkin               |<------>| email           |
                            | checkout              |        | phn_number      |
                            | adult                 |        | address         |
                            | children              |        | aadhar          |
                            +-----------------------+        +-----------------+
```

## 3.6 NORMALIZATION

The database is normalized to the third normal form (3NF) to eliminate redundancy and ensure data integrity. This process involves organizing the tables and their relationships to minimize data duplication.

# CHAPTER 4
# PROGRAM CODE

```
import javax.swing.*;
import javax.swing.border.AbstractBorder;
import javax.swing.text.BadLocationException;
import javax.swing.text.MaskFormatter;
import javax.swing.text.PlainDocument;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Area;
import java.awt.geom.Rectangle2D;
import java.awt.geom.RoundRectangle2D;
import java.io.File;
import java.io.IOException;
import java.sql.*;
import javax.imageio.ImageIO;
import javax.swing.text.AttributeSet;
import java.time.LocalDate;

public class AJHotel extends JFrame {

    private JLabel titleLabel;
    private Timer timer;
    private int delay = 100;
    private int pauseDuration = 2000;
    private int currentIndex = 0;
    private String text = "AJ HOTEL MANAGEMENT";
    private BackgroundPanel backgroundPanel;
    private Connection connection;
    private CustomDatePicker checkInDatePicker;
    private CustomDatePicker checkOutDatePicker;
    public JTextField adultField;
    public JTextField childrenField;
    public AJHotel() {
        setTitle("AJ Hotel");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new CardLayout());
        initializeDatabase();
        initializeUI();
    }

    private void initializeDatabase() {
        try {
            connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/AJHotel",
"root", "Jklmn@123");
            createBookingTable();
        } catch (SQLException e) {
            handleSQLException("Database Connection Error", e);
        }
    }

    private void createBookingTable() {
        try (Statement statement = connection.createStatement()) {
            String sql = "CREATE TABLE IF NOT EXISTS booking (" +
             "id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, "
+
             "room_number INT, " +
             "checkin DATE, " +
             "adult INT, " +
             "children INT" +
             ")";
            statement.executeUpdate(sql);
            System.out.println("Table 'booking' created successfully!");
        } catch (SQLException e) {
            handleSQLException("Error creating 'booking' table", e);
        }
    }
    private void initializeUI() {
        titleLabel = new JLabel("", SwingConstants.CENTER);
        titleLabel.setFont(new Font("Bauhaus 93", Font.BOLD, 100));
        JPanel mainPanel = new JPanel(new BorderLayout());
        mainPanel.add(titleLabel, BorderLayout.CENTER);
        add(mainPanel, "mainPanel");
        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
        setSize(screenSize.width, screenSize.height);
        setLocationRelativeTo(null);
        timer = new Timer(delay, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (currentIndex < text.length()) {
                    titleLabel.setText(text.substring(0, currentIndex + 1));
                    currentIndex++;
                } else {
                    timer.stop();
                    Timer pauseTimer = new Timer(pauseDuration, new
ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            titleLabel.setText("");
                            showLoginScreen();
                        }
                    });
                    pauseTimer.setRepeats(false);
                    pauseTimer.start();
                }
            }
        });
        timer.start();
    }

    private void handleSQLException(String message, SQLException e)
{
        System.err.println(message + ": " + e.getMessage());
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "An error occurred: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }

    private static class NumericDocument extends PlainDocument {
    @Override
    public void insertString(int offs, String str, AttributeSet a) throws
BadLocationException {
        if (str == null) {
            return;
        }
        try {
            Integer.parseInt(str);
            super.insertString(offs, str, a);
        } catch (NumberFormatException e) {
            }
    }
}

    private void showLoginScreen() {
        backgroundPanel = new
BackgroundPanel("C:\\Users\\jyoth\\OneDrive\\Desktop\\DBMS_Projec
t\\bg_image.jpg");
        backgroundPanel.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.gridx = 0;
        gbc.gridy = 0;

        JPanel loginPanel = new JPanel(new GridBagLayout());
        loginPanel.setOpaque(false);

        JLabel userLabel = new JLabel("Username:");
        userLabel.setFont(new Font("Arial", Font.PLAIN, 25));
        userLabel.setForeground(Color.BLACK);
        loginPanel.add(userLabel, gbc);
        gbc.gridx = 1;
        JTextField userField = new TranslucentTextField(15);
        userField.setFont(new Font("Arial", Font.PLAIN, 25));
        loginPanel.add(userField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 1;
        JLabel passLabel = new JLabel("Password:");
        passLabel.setFont(new Font("Arial", Font.PLAIN, 25));
        passLabel.setForeground(Color.BLACK);
        loginPanel.add(passLabel, gbc);

        gbc.gridx = 1;
        JPasswordField passField = new TranslucentPasswordField(15);
        passField.setFont(new Font("Arial", Font.PLAIN, 25));
        loginPanel.add(passField, gbc);

        gbc.gridx = 1;
        gbc.gridy = 2;
        JButton loginButton = new JButton("Login");
        loginButton.setFont(new Font("Arial", Font.BOLD, 18));
        loginPanel.add(loginButton, gbc);

        loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = userField.getText();
                String password = new String(passField.getPassword());
                if (("AKASH".equals(username) && "aj".equals(password)) ||
("JYOTHI".equals(username) && "aj".equals(password))) {
                    showMainScreen();
                } else {
                    JOptionPane.showMessageDialog(AJHotel.this, "Invalid
username or password", "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        TranslucentPanel translucentPanel = new TranslucentPanel();
        translucentPanel.setLayout(new GridBagLayout());
        GridBagConstraints tGbc = new GridBagConstraints();
        tGbc.insets = new Insets(10, 10, 10, 10);
        tGbc.gridx = 0;
        tGbc.gridy = 0;
        translucentPanel.add(loginPanel, tGbc);
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets(50, 30, 50, 0);
        backgroundPanel.add(translucentPanel, gbc);

        add(backgroundPanel, "loginPanel");
        CardLayout cl = (CardLayout) getContentPane().getLayout();
        cl.show(getContentPane(), "loginPanel");
    }

    private void showMainScreen() {
        backgroundPanel.removeAll();
        backgroundPanel.revalidate();
        backgroundPanel.repaint();

        TranslucentPanel mainPanel = new TranslucentPanel();
        mainPanel.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JPanel firstRowPanel = new JPanel(new
FlowLayout(FlowLayout.CENTER, 20, 0));
        firstRowPanel.setOpaque(false);

        JLabel checkInLabel = new JLabel("Check In");
        checkInLabel.setFont(new Font("Arial", Font.BOLD, 20));
        firstRowPanel.add(checkInLabel);

        JLabel checkOutLabel = new JLabel("Check Out");
        checkOutLabel.setFont(new Font("Arial", Font.BOLD, 20));
        firstRowPanel.add(checkOutLabel);

        checkInDatePicker = new CustomDatePicker();
        firstRowPanel.add(checkInDatePicker);

        checkOutDatePicker = new CustomDatePicker();
        firstRowPanel.add(checkOutDatePicker);

        String[] options = { "Select","Solo Traveller (1 Traveller, 1
Room)", "Couple/Pair (2 Travellers, 1 Room)", "Family Traveller",
"Group Traveller" };
        JComboBox<String> optionsDropdown = new
JComboBox<>(options);
        optionsDropdown.setFont(new Font("Arial", Font.PLAIN, 18));
        firstRowPanel.add(optionsDropdown);

        gbc.gridx = 0;
        gbc.gridy = 0;
        mainPanel.add(firstRowPanel, gbc);

        JPanel secondRowPanel = new JPanel(new
FlowLayout(FlowLayout.CENTER, 20, 0));
        secondRowPanel.setOpaque(false);

        JLabel doubleBedroomLabel = new JLabel("Double Bedrooms:");
        JTextField doubleBedroomField = new JTextField(2);
        doubleBedroomField.setEditable(false);

        JLabel singleBedroomLabel = new JLabel("Single Bedrooms:");
        JTextField singleBedroomField = new JTextField(2);
        singleBedroomField.setEditable(false);

        JTextField roomField = new
```

```
JFormattedTextField(createFormatter("##"));
        JLabel roomLabel = new JLabel("Rooms:");
        roomField.setEditable(false);
        roomField.setFont(new Font("Arial", Font.PLAIN, 20));
        roomField.setColumns(3);
        roomField.setDocument(new NumericDocument());
        secondRowPanel.add(roomField);

        adultField = new JFormattedTextField(createFormatter("##"));
        adultField.setFont(new Font("Arial", Font.PLAIN, 20));
        adultField.setColumns(3);
        adultField.setDocument(new NumericDocument());
        secondRowPanel.add(adultField);
        JLabel adultLabel = new JLabel("Adult:");
        adultField.setEditable(false);

        childrenField = new JFormattedTextField(createFormatter("##"));
        childrenField.setFont(new Font("Arial", Font.PLAIN, 20));
        childrenField.setColumns(3);
        childrenField.setDocument(new NumericDocument());
        secondRowPanel.add(childrenField);
        JLabel childrenLabel = new JLabel("Children:");
        childrenField.setEditable(false);

        secondRowPanel.add(doubleBedroomLabel);
        secondRowPanel.add(doubleBedroomField);
        secondRowPanel.add(singleBedroomLabel);
        secondRowPanel.add(singleBedroomField);
        secondRowPanel.add(roomLabel);
        secondRowPanel.add(roomField);
        secondRowPanel.add(adultLabel);
        secondRowPanel.add(adultField);
        secondRowPanel.add(childrenLabel);
        secondRowPanel.add(childrenField);

        gbc.gridx = 0;
        gbc.gridy = 1;
        mainPanel.add(secondRowPanel, gbc);

        JButton bookButton = new JButton("Book");
        bookButton.setFont(new Font("Arial", Font.BOLD, 25));
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.anchor = GridBagConstraints.CENTER;
        mainPanel.add(bookButton, gbc);

        JButton bookingButton = new JButton("Booking");
        bookingButton.setFont(new Font("Arial", Font.BOLD, 25));
        bookingButton.setPreferredSize(bookButton.getPreferredSize());
        gbc.gridx = 0;
        gbc.gridy = 3;
        gbc.anchor = GridBagConstraints.CENTER;
        mainPanel.add(bookingButton, gbc);

        JButton checkoutButton = new JButton("Checkout");
        checkoutButton.setFont(new Font("Arial", Font.BOLD, 25));
        checkoutButton.setPreferredSize(bookButton.getPreferredSize());
        gbc.gridx = 0;
        gbc.gridy = 4;
        gbc.anchor = GridBagConstraints.CENTER;
        mainPanel.add(checkoutButton, gbc);

          optionsDropdown.addActionListener(new ActionListener() {
          @Override
           public void actionPerformed(ActionEvent e) {
              String selectedOption = (String)
    optionsDropdown.getSelectedItem();

              switch (selectedOption) {
                  case "Solo Traveller (1 Traveller, 1 Room)":
                      singleBedroomField.setText("1");
                      doubleBedroomField.setText("0");
                      adultField.setText("1");
                      childrenField.setText("0");
                      roomField.setText("1");

                      singleBedroomField.setEditable(false);
                      doubleBedroomField.setEditable(false);
                      adultField.setEditable(false);
                      childrenField.setEditable(false);
                      break;

                  case "Couple/Pair (2 Travellers, 1 Room)":
                      singleBedroomField.setText("1");
                      doubleBedroomField.setText("0");
                      adultField.setText("2");
                      childrenField.setText("0");
                      roomField.setText("1");

                      singleBedroomField.setEditable(false);
                      doubleBedroomField.setEditable(false);
                      adultField.setEditable(false);
                      childrenField.setEditable(false);
                      break;

                  case "Family Traveller":
                      singleBedroomField.setText("");
                      doubleBedroomField.setText("");
                      adultField.setText("");
                      childrenField.setText("");
                      roomField.setText("");

                      singleBedroomField.setEditable(true);
                      doubleBedroomField.setEditable(true);
                      adultField.setEditable(true);
                      childrenField.setEditable(true);
                      break;

                  case "Group Traveller":
                      singleBedroomField.setText("");
                      doubleBedroomField.setText("");
```

```
                adultField.setText("");
                childrenField.setText("");
                roomField.setText("");

                singleBedroomField.setEditable(true);
                doubleBedroomField.setEditable(true);
                adultField.setEditable(true);
                childrenField.setEditable(false);
                break;
        }
    }
});

bookingButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        displayBookingDetails();
    }
});

bookButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        displayBookingForm();
    }
});

checkoutButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        displayCheckoutForm();
    }
});

gbc.insets = new Insets(50, 30, 50, 0);
backgroundPanel.add(mainPanel, gbc);
```

```
        CardLayout cl = (CardLayout) getContentPane().getLayout();
        cl.show(getContentPane(), "loginPanel");
    }

    private void displayCheckoutForm() {
        JPanel checkoutPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;

         JLabel roomNumberLabel = new JLabel("Room Number:");
        roomNumberLabel.setFont(new Font("Arial", Font.BOLD, 20));
        gbc.gridx = 0;
        gbc.gridy = 0;
        checkoutPanel.add(roomNumberLabel, gbc);

        JTextField roomNumberField = new JTextField(5);
        roomNumberField.setFont(new Font("Arial", Font.PLAIN, 20));
        gbc.gridx = 1;
        checkoutPanel.add(roomNumberField, gbc);

         JTextArea detailsArea = new JTextArea(10, 30);
        detailsArea.setFont(new Font("Arial", Font.PLAIN, 18));
        detailsArea.setEditable(false);
        JScrollPane detailsScrollPane = new JScrollPane(detailsArea);
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridwidth = 2;
        checkoutPanel.add(detailsScrollPane, gbc);

         JButton confirmCheckoutButton = new JButton("Confirm
Checkout");
        confirmCheckoutButton.setFont(new Font("Arial", Font.BOLD,
25));
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 2;
        gbc.anchor = GridBagConstraints.SOUTHEAST;
        checkoutPanel.add(confirmCheckoutButton, gbc);

         roomNumberField.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String roomNumber = roomNumberField.getText();
                String details = fetchDetailsForRoom(roomNumber);
                detailsArea.setText(details);
            }
        });

         confirmCheckoutButton.addActionListener(new ActionListener()
{
            @Override
            public void actionPerformed(ActionEvent e) {
                String roomNumber = roomNumberField.getText();
                confirmCheckout(roomNumber);
            }
        });

        backgroundPanel.removeAll();
        backgroundPanel.add(checkoutPanel);
        backgroundPanel.revalidate();
        backgroundPanel.repaint();

        CardLayout cl = (CardLayout) getContentPane().getLayout();
        cl.show(getContentPane(), "checkoutPanel");
    }

    private String fetchDetailsForRoom(String roomNumber) {
        String details = "";
        Connection conn = null;
        PreparedStatement detailsStmt = null;
        PreparedStatement bookingStmt = null;
        ResultSet rs = null;
        try {
             Class.forName("com.mysql.cj.jdbc.Driver");
             String url = "jdbc:mysql://localhost:3306/AJHotel";
            String user = "root";
            String password = "Jklmn@123";
            conn = DriverManager.getConnection(url, user, password);

            String detailsQuery = "SELECT * FROM details WHERE
room_number = ?";
            detailsStmt = conn.prepareStatement(detailsQuery);
            detailsStmt.setString(1, roomNumber);
            rs = detailsStmt.executeQuery();

            if (rs.next()) {
                String name = rs.getString("name");
                String email = rs.getString("email");
                String phoneNumber = rs.getString("phn_number");
                String address = rs.getString("address");
                String aadhar = rs.getString("aadhar");

                details = "Room Number: " + roomNumber + "\n"
                        + "Name: " + name + "\n"
                        + "Email: " + email + "\n"
                        + "Phone Number: " + phoneNumber + "\n"
                        + "Address: " + address + "\n"
                        + "Aadhar: " + aadhar + "\n";
            }
                rs.close();
            String bookingQuery = "SELECT * FROM booking WHERE
room_number = ?";
            bookingStmt = conn.prepareStatement(bookingQuery);
            bookingStmt.setString(1, roomNumber);
            rs = bookingStmt.executeQuery();

            if (rs.next()) {
                String checkin = rs.getString("checkin");
                int adult = rs.getInt("adult");
            int children = rs.getInt("children");
               details += "Check-in: " + checkin + "\n"
                  + "Adults: " + adult + "\n"
                  + "Children: " + children + "\n";
         }
      } catch (SQLException e) {
         e.printStackTrace();
         details = "SQL Error fetching details for room number: " +
roomNumber + " - " + e.getMessage();
      } catch (ClassNotFoundException e) {
         e.printStackTrace();
         details = "JDBC Driver not found. Ensure the MySQL JDBC
Driver is included in your project.";
      } finally {
            try {
            if (rs != null) rs.close();
            if (detailsStmt != null) detailsStmt.close();
            if (bookingStmt != null) bookingStmt.close();
            if (conn != null) conn.close();
         } catch (SQLException se) {
            se.printStackTrace();
         }
      }
      return details;
   }

    private void confirmCheckout(String roomNumber) {
      Connection conn = null;
      PreparedStatement insertStmt = null;
      PreparedStatement deleteBookingStmt = null;
      PreparedStatement deleteDetailsStmt = null;
      PreparedStatement updateRoomStatusStmt = null;
      try {
         conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/AJHotel",
"root", "Jklmn@123");
         String insertQuery = "INSERT INTO check_out (room_number,
name, email, phn_number, checkin, checkout, adult, children, address,
aadhar) " +
            "SELECT b.room_number, d.name, d.email, d.phn_number,
b.checkin, CURDATE(), b.adult, b.children, d.address, d.aadhar " +
            "FROM booking b JOIN details d ON b.room_number =
d.room_number " +
            "WHERE b.room_number = ?";
insertStmt = conn.prepareStatement(insertQuery);
insertStmt.setString(1, roomNumber);
insertStmt.executeUpdate();
String updateRoomStatusQuery = "UPDATE room_status SET status =
'unoccupied' WHERE room_number = ?";
updateRoomStatusStmt =
conn.prepareStatement(updateRoomStatusQuery);
updateRoomStatusStmt.setString(1, roomNumber);
updateRoomStatusStmt.executeUpdate();

            String deleteDetailsQuery = "DELETE FROM details WHERE
room_number = ?";
            deleteDetailsStmt = conn.prepareStatement(deleteDetailsQuery);
            deleteDetailsStmt.setString(1, roomNumber);
            deleteDetailsStmt.executeUpdate();

            String deleteBookingQuery = "DELETE FROM booking
WHERE room_number = ?";
            deleteBookingStmt =
conn.prepareStatement(deleteBookingQuery);
            deleteBookingStmt.setString(1, roomNumber);
            deleteBookingStmt.executeUpdate();

            JOptionPane.showMessageDialog(this, "Checked out room
number: " + roomNumber);
            showMainScreen();
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error confirming
checkout: " + e.getMessage());
```

```
        } finally {
            try {
                if (insertStmt != null) insertStmt.close();
                if (deleteBookingStmt != null) deleteBookingStmt.close();
                if (deleteDetailsStmt != null) deleteDetailsStmt.close();
                if (conn != null) conn.close();
            } catch (SQLException se) {
                se.printStackTrace();
            }
        }
    }

    private void displayBookingDetails() {
        JPanel bookingDetailsPanel = new JPanel();
        bookingDetailsPanel.setLayout(new BorderLayout());
        JTextArea bookingDetailsTextArea = new JTextArea();
        bookingDetailsTextArea.setEditable(false);
        bookingDetailsTextArea.setFont(new Font("Arial", Font.PLAIN,
16));

        try {
            String sql = "SELECT b.id, b.room_number, b.checkin, b.adult,
b.children, " +
                        "c.name, c.phn_number, c.aadhar, c.email " +
                        "FROM booking b JOIN details c ON b.room_number =
c.room_number";
            PreparedStatement statement =
connection.prepareStatement(sql);
            ResultSet resultSet = statement.executeQuery();
            StringBuilder bookingDetails = new StringBuilder();
            while (resultSet.next()) {
                int id = resultSet.getInt("id");
                int roomNumber = resultSet.getInt("room_number");
                Date checkin = resultSet.getDate("checkin");
                int adult = resultSet.getInt("adult");
                int children = resultSet.getInt("children");
                String name = resultSet.getString("name");
                String phoneNumber = resultSet.getString("phn_number");
                String aadharNumber = resultSet.getString("aadhar");
                String email = resultSet.getString("email");

                bookingDetails.append("Booking ID: 
").append(id).append("\n");
                bookingDetails.append("Room Number: 
").append(roomNumber).append("\n");
                bookingDetails.append("Name: 
").append(name).append("\n");
                bookingDetails.append("Check-in Date: 
").append(checkin).append("\n");
                bookingDetails.append("Phone Number: 
").append(phoneNumber).append("\n");
                bookingDetails.append("Aadhar Number: 
").append(aadharNumber).append("\n");
                bookingDetails.append("Number of Adults: 
").append(adult).append("\n");
                bookingDetails.append("Number of Children: 
").append(children).append("\n");
                bookingDetails.append("Email: 
").append(email).append("\n\n");
            }

            bookingDetailsTextArea.setText(bookingDetails.toString());
            statement.close();
            resultSet.close();
        } catch (SQLException ex) {
            handleSQLException("Error retrieving booking details from the 
database", ex);
        }
        JScrollPane scrollPane = new 
JScrollPane(bookingDetailsTextArea);
        bookingDetailsPanel.add(scrollPane, BorderLayout.CENTER);
        JDialog dialog = new JDialog();
        dialog.setTitle("Booking Details");
        dialog.setModal(true);
        dialog.setContentPane(bookingDetailsPanel);
        dialog.pack();

        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
        int x = (screenSize.width - dialog.getWidth()) / 2;
        int y = (screenSize.height - dialog.getHeight()) / 2;
        dialog.setLocation(x, y);
        dialog.setVisible(true);
    }

    private void displayBookingForm() {
        backgroundPanel.removeAll();
        backgroundPanel.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;
        TranslucentPanel formPanel = new TranslucentPanel();
        formPanel.setLayout(new GridBagLayout());
        GridBagConstraints formGbc = new GridBagConstraints();
        formGbc.insets = new Insets(10, 10, 10, 10);
        formGbc.fill = GridBagConstraints.HORIZONTAL;

        JButton backButton = new JButton("Back");
        backButton.setFont(new Font("Arial", Font.PLAIN, 14));
        backButton.setPreferredSize(new Dimension(80, 30));
        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                showMainScreen();
            }
        });
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.NORTHWEST;
        backgroundPanel.add(backButton, gbc);
```

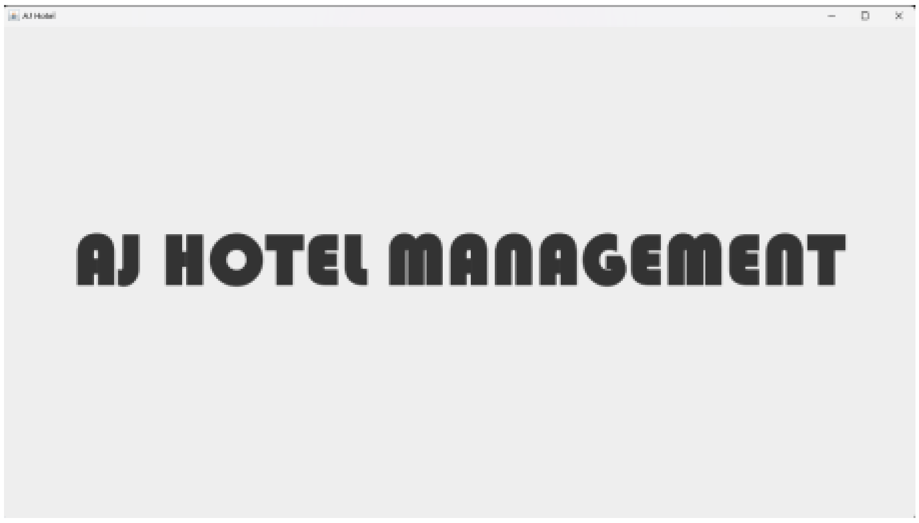
```
JLabel bookingLabel = new JLabel("Booking Details");
bookingLabel.setFont(new Font("Arial", Font.BOLD, 30));
formGbc.gridx = 0;
formGbc.gridy = 1;
formGbc.gridwidth = 2;
formPanel.add(bookingLabel, formGbc);
formGbc.gridwidth = 1;

JLabel nameLabel = new JLabel("Name:");
nameLabel.setFont(new Font("Arial", Font.PLAIN, 20));
formGbc.gridx = 0;
formGbc.gridy = 2;
formPanel.add(nameLabel, formGbc);

JTextField nameField = new JTextField(20);
nameField.setFont(new Font("Arial", Font.PLAIN, 20));
formGbc.gridx = 1;
formPanel.add(nameField, formGbc);

JLabel emailLabel = new JLabel("Email:");
emailLabel.setFont(new Font("Arial", Font.PLAIN, 20));
formGbc.gridx = 0;
formGbc.gridy = 3;
formPanel.add(emailLabel, formGbc);

JTextField emailField = new JTextField(20);
emailField.setFont(new Font("Arial", Font.PLAIN, 20));
formGbc.gridx = 1;
formPanel.add(emailField, formGbc);

JLabel phoneLabel = new JLabel("Phone Number:");
phoneLabel.setFont(new Font("Arial", Font.PLAIN, 20));
formGbc.gridx = 0;
formGbc.gridy = 4;
formPanel.add(phoneLabel, formGbc);

JFormattedTextField phoneField = new
```

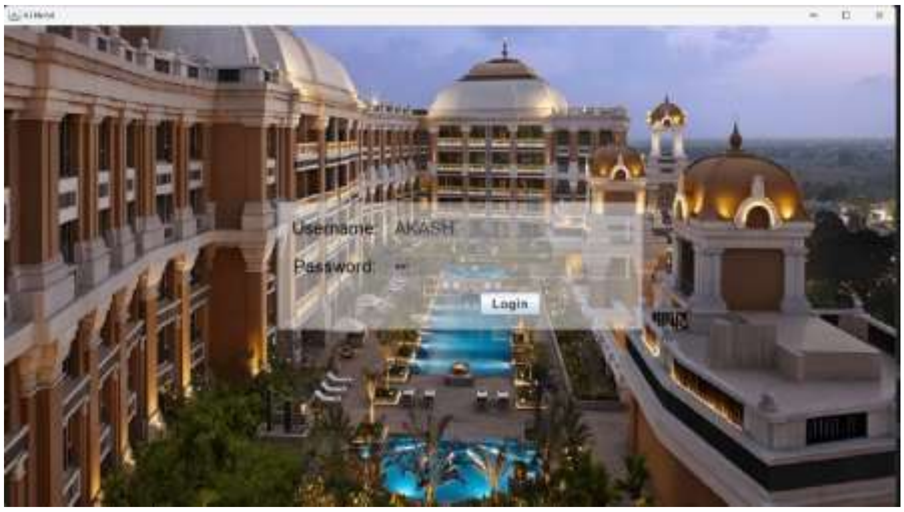
```
JFormattedTextField(createFormatter("##########"));
        phoneField.setFont(new Font("Arial", Font.PLAIN, 20));
        phoneField.setColumns(20);
        formGbc.gridx = 1;
        formPanel.add(phoneField, formGbc);

        JLabel addressLabel = new JLabel("Address:");
        addressLabel.setFont(new Font("Arial", Font.PLAIN, 20));
        formGbc.gridx = 0;
        formGbc.gridy = 5;
        formPanel.add(addressLabel, formGbc);

        JTextField addressField = new JTextField(20);
        addressField.setFont(new Font("Arial", Font.PLAIN, 20));
        formGbc.gridx = 1;
        formPanel.add(addressField, formGbc);

        JLabel aadharLabel = new JLabel("Aadhar Number:");
        aadharLabel.setFont(new Font("Arial", Font.PLAIN, 20));
        formGbc.gridx = 0;
        formGbc.gridy = 6;
        formPanel.add(aadharLabel, formGbc);

        JFormattedTextField aadharField = new
JFormattedTextField(createFormatter("############"));
        aadharField.setFont(new Font("Arial", Font.PLAIN, 20));
        aadharField.setColumns(20);
        formGbc.gridx = 1;
        formPanel.add(aadharField, formGbc);

        JLabel roomNumberLabel = new JLabel("Room Number:");
        roomNumberLabel.setFont(new Font("Arial", Font.PLAIN, 20));
        formGbc.gridx = 0;
        formGbc.gridy = 7;
        formPanel.add(roomNumberLabel, formGbc);

        JTextField roomNumberField = new JTextField(20);
        roomNumberField.setFont(new Font("Arial", Font.PLAIN, 20));
        formGbc.gridx = 1;
        formPanel.add(roomNumberField, formGbc);

        JButton submitButton = new JButton("Submit");
        submitButton.setFont(new Font("Arial", Font.BOLD, 20));
        formGbc.gridx = 0;
        formGbc.gridy = 8;
        formGbc.gridwidth = 2;
        formPanel.add(submitButton, formGbc);

        submitButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (nameField.getText().trim().isEmpty() ||
            emailField.getText().trim().isEmpty() ||
            phoneField.getText().trim().isEmpty() ||
            addressField.getText().trim().isEmpty() ||
            aadharField.getText().trim().isEmpty() ||
            roomNumberField.getText().trim().isEmpty()) {
            JOptionPane.showMessageDialog(AJHotel.this, "All fields are
mandatory.", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (phoneField.getText().length() != 10) {
            JOptionPane.showMessageDialog(AJHotel.this, "Phone number
must be exactly 10 digits.", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (aadharField.getText().length() != 12) {
            JOptionPane.showMessageDialog(AJHotel.this, "Aadhar number
must be exactly 12 digits.", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        String name = nameField.getText();
        String email = emailField.getText();
        long phoneNumber = Long.parseLong(phoneField.getText());
        String address = addressField.getText();
        long aadhar = Long.parseLong(aadharField.getText());
        int roomNumber = Integer.parseInt(roomNumberField.getText());
        LocalDate today = LocalDate.now();

        java.sql.Date checkin = Date.valueOf(today);
        int adult = Integer.parseInt(adultField.getText());
        int children = Integer.parseInt(childrenField.getText());

        try {
            connection.setAutoCommit(false);
            String bookingSql = "INSERT INTO booking (room_number,
checkin, adult, children) VALUES (?, ?, ?, ?)";
            String updateRoomStatusSql = "UPDATE room_status SET
status = 'occupied' WHERE room_number = ?";
            try (PreparedStatement bookingStmt =
connection.prepareStatement(bookingSql);
            PreparedStatement updateStmt =
connection.prepareStatement(updateRoomStatusSql)) {

    bookingStmt.setInt(1, roomNumber);
    bookingStmt.setDate(2, checkin);
    bookingStmt.setInt(3, adult);
    bookingStmt.setInt(4, children);
    bookingStmt.executeUpdate();
    updateStmt.setInt(1, roomNumber);
    updateStmt.executeUpdate();
}

            String detailsSql = "INSERT INTO details (room_number, name,
email, phn_number, address, aadhar) VALUES (?, ?, ?, ?, ?, ?)";
            try (PreparedStatement detailsStmt =
connection.prepareStatement(detailsSql)) {
                detailsStmt.setInt(1, roomNumber);
                detailsStmt.setString(2, name);
            detailsStmt.setString(3, email);
            detailsStmt.setLong(4, phoneNumber);
            detailsStmt.setString(5, address);
            detailsStmt.setLong(6, aadhar);
            detailsStmt.executeUpdate();
         }
         connection.commit();
         JOptionPane.showMessageDialog(AJHotel.this, "Booking details
submitted successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
         showMainScreen();
      } catch (SQLException ex) {
         try {
            connection.rollback();
         } catch (SQLException rollbackEx) {
            handleSQLException("Error rolling back transaction",
rollbackEx);
         }
         handleSQLException("Error inserting booking details into the
database", ex);
      } finally {
         try {
            connection.setAutoCommit(true);
         } catch (SQLException autoCommitEx) {
            handleSQLException("Error setting auto-commit to true",
autoCommitEx);
         }
      }
   }
});
      gbc.gridx = 0;
      gbc.gridy = 1;
      gbc.gridwidth = 2;
      backgroundPanel.add(formPanel, gbc);

      backgroundPanel.revalidate();
      backgroundPanel.repaint();
```

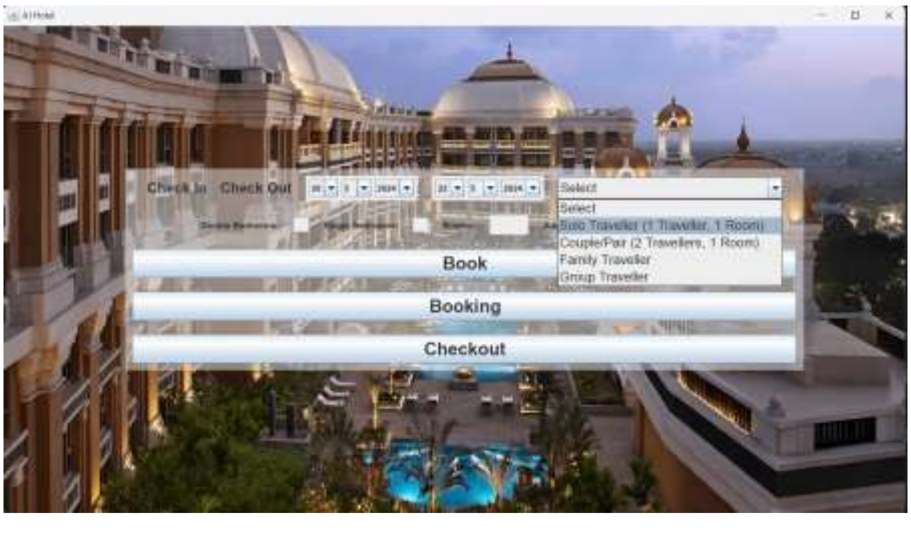
```
    }

    protected MaskFormatter createFormatter(String s) {
        MaskFormatter formatter = null;
        try {
            formatter = new MaskFormatter(s);
            formatter.setPlaceholderCharacter(' ');
        } catch (java.text.ParseException exc) {
            System.err.println("formatter is bad: " + exc.getMessage());
        }
        return formatter;
    }

    private static class BackgroundPanel extends JPanel {
        private Image backgroundImage;
        public BackgroundPanel(String imagePath) {
            try {
                backgroundImage = ImageIO.read(new File(imagePath));
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            if (backgroundImage != null) {
                g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(),
this);
            }
        }
    }

    private static class TranslucentPanel extends JPanel {
        public TranslucentPanel() {
            setOpaque(false);
        }
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            Graphics2D g2d = (Graphics2D) g.create();

g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_
OVER, 0.5f));
            g2d.setColor(getBackground());
            g2d.fillRect(0, 0, getWidth(), getHeight());
            g2d.dispose();
        }
    }

    private static class TranslucentTextField extends JTextField {
        public TranslucentTextField(int columns) {
            super(columns);
            setOpaque(false);
        }

        @Override
        protected void paintComponent(Graphics g) {
            if (!isOpaque() && getBorder() instanceof
RoundedCornerBorder) {
                Graphics2D g2 = (Graphics2D) g.create();
                g2.setColor(getBackground());
                g2.fill(((RoundedCornerBorder)
getBorder()).getBorderShape(0, 0, getWidth() - 1, getHeight() - 1));
                g2.dispose();
            }
            super.paintComponent(g);
        }
    }

    private static class TranslucentPasswordField extends JPasswordField
{
        public TranslucentPasswordField(int columns) {
            super(columns);
            setOpaque(false);
        }

        @Override
        protected void paintComponent(Graphics g) {
            if (!isOpaque() && getBorder() instanceof
RoundedCornerBorder) {
                Graphics2D g2 = (Graphics2D) g.create();
                g2.setColor(getBackground());
                g2.fill(((RoundedCornerBorder)
getBorder()).getBorderShape(0, 0, getWidth() - 1, getHeight() - 1));
                g2.dispose();
            }
            super.paintComponent(g);
        }
    }

    private static class RoundedCornerBorder extends AbstractBorder {
        private final Color ALPHA_ZERO = new Color(0x0, true);

        @Override
        public void paintBorder(Component c, Graphics g, int x, int y, int
width, int height) {
            Graphics2D g2 = (Graphics2D) g.create();
            g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
            Shape border = getBorderShape(x, y, width - 1, height - 1);
            g2.setPaint(ALPHA_ZERO);
            Container parent = c.getParent();
            if (parent != null) {
                g2.setPaint(parent.getBackground());
                Area corner = new Area(new Rectangle2D.Double(x, y, width,
height));
                corner.subtract(new Area(border));
                g2.fill(corner);
            }
            g2.setPaint(c.getForeground());
            g2.draw(border);
            g2.dispose();
        }

        public Shape getBorderShape(int x, int y, int w, int h) {
            int r = h;
            return new RoundRectangle2D.Double(x, y, w, h, r, r);
        }

        @Override
        public Insets getBorderInsets(Component c) {
            return new Insets(4, 8, 4, 8);
        }

        @Override
        public Insets getBorderInsets(Component c, Insets insets) {
            insets.set(4, 8, 4, 8);
            return insets;
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new AJHotel().setVisible(true);
            }
        });
    }
}
```

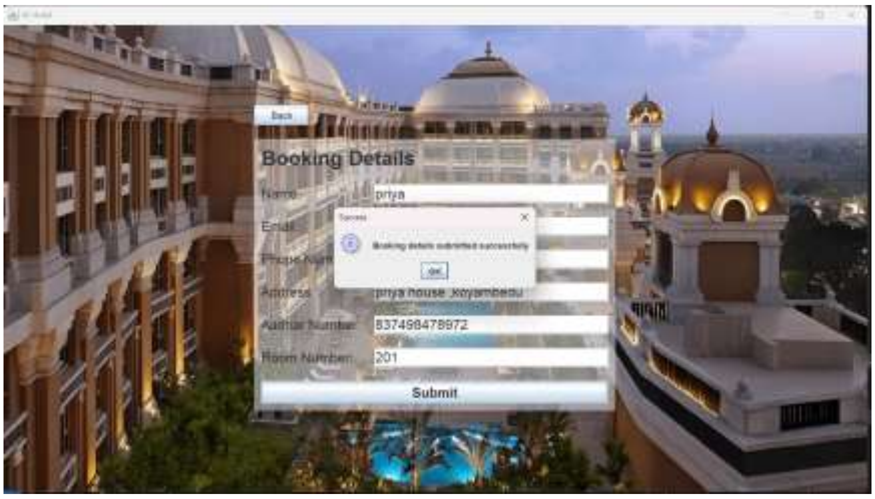
# CHAPTER 5
# RESULT AND CONCLUSIONS

## 5.1 SCREENSHOTS









Booking Details

Name priya

Email

Phone Number

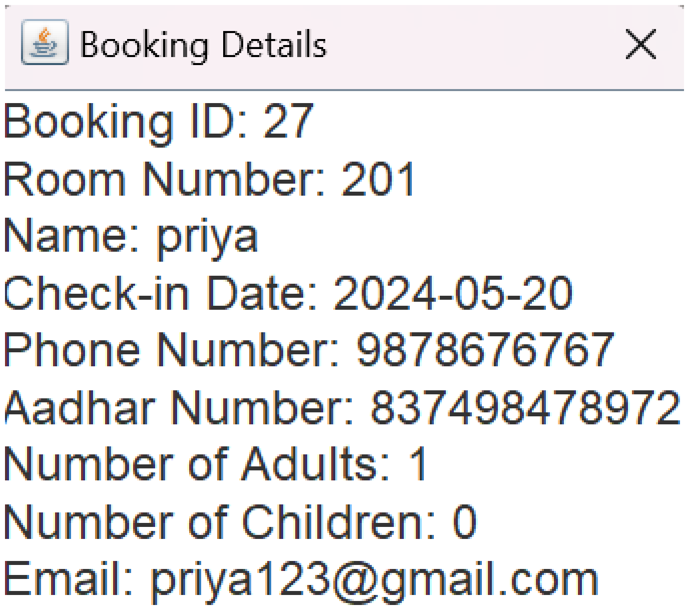
Address priya house ,koyambedu

Aadhar Number 837498478972

Room Number 201

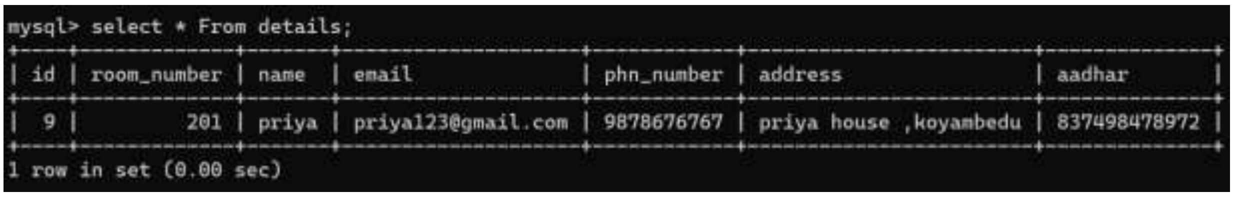
Success

Booking details submitted successfully

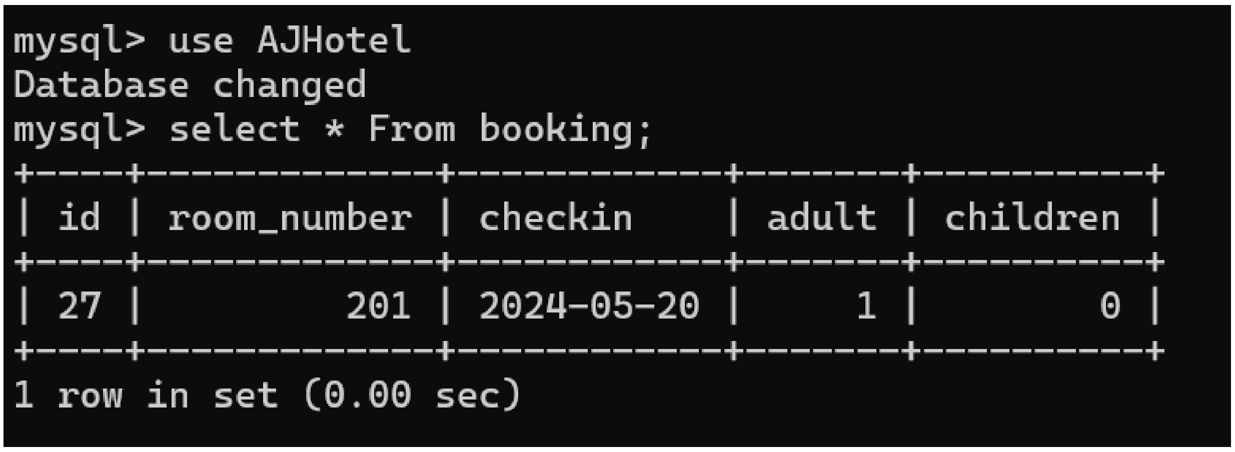


Booking Details

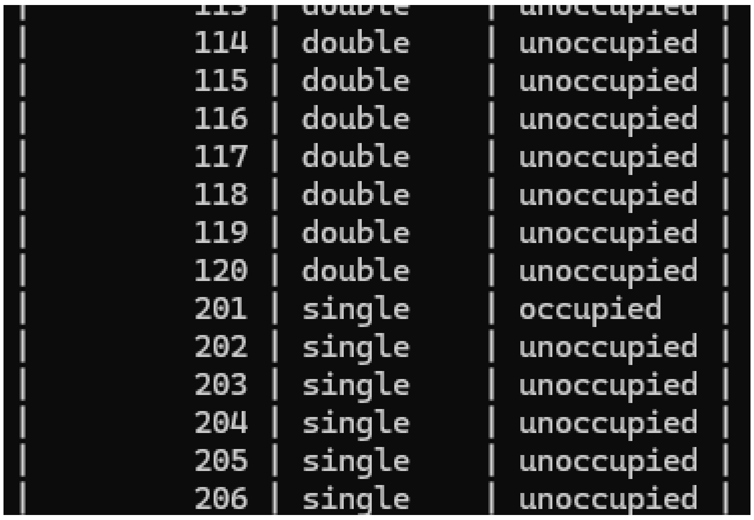
Booking ID: 27
Room Number: 201
Name: priya
Check-in Date: 2024-05-20
Phone Number: 9878676767
Aadhar Number: 837498478972
Number of Adults: 1
Number of Children: 0
Email: priya123@gmail.com



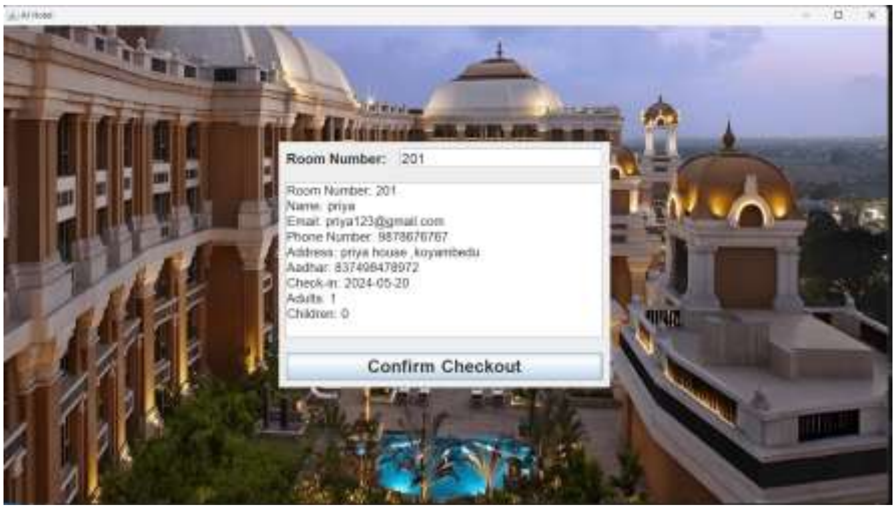
```
mysql> select * From details;
+----+-------------+-------+--------------------+------------+-------------------------+--------------+
| id | room_number | name  | email              | phn_number | address                 | aadhar       |
+----+-------------+-------+--------------------+------------+-------------------------+--------------+
|  9 |         201 | priya | priya123@gmail.com | 9878676767 | priya house ,koyambedu  | 837498478972 |
+----+-------------+-------+--------------------+------------+-------------------------+--------------+
1 row in set (0.00 sec)
```



```
mysql> use AJHotel
Database changed
mysql> select * From booking;
+----+-------------+------------+-------+----------+
| id | room_number | checkin    | adult | children |
+----+-------------+------------+-------+----------+
| 27 |         201 | 2024-05-20 |     1 |        0 |
+----+-------------+------------+-------+----------+
1 row in set (0.00 sec)
```

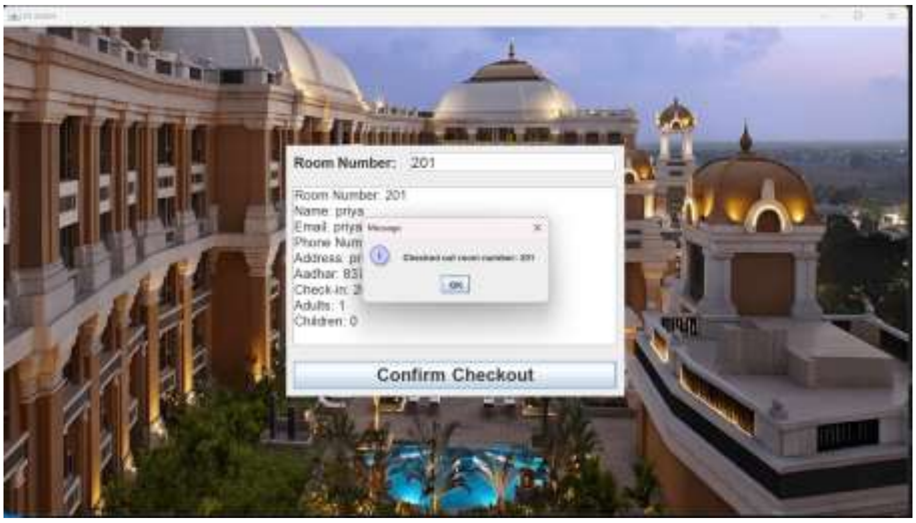


```
|          114 | double     | unoccupied |
|          115 | double     | unoccupied |
|          116 | double     | unoccupied |
|          117 | double     | unoccupied |
|          118 | double     | unoccupied |
|          119 | double     | unoccupied |
|          120 | double     | unoccupied |
|          201 | single     | occupied   |
|          202 | single     | unoccupied |
|          203 | single     | unoccupied |
|          204 | single     | unoccupied |
|          205 | single     | unoccupied |
|          206 | single     | unoccupied |
```



Room Number: 201

Room Number: 201
Name: priya
Email: priya123@gmail.com
Phone Number: 9878676767
Address: priya house ,koyambedu
Aadhar: 837498478972
Check-in: 2024-05-20
Adults: 1
Children: 0

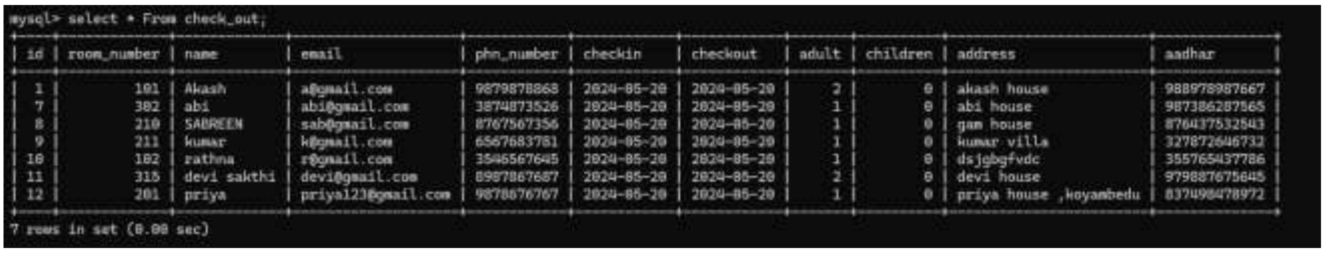


Room Number: 201

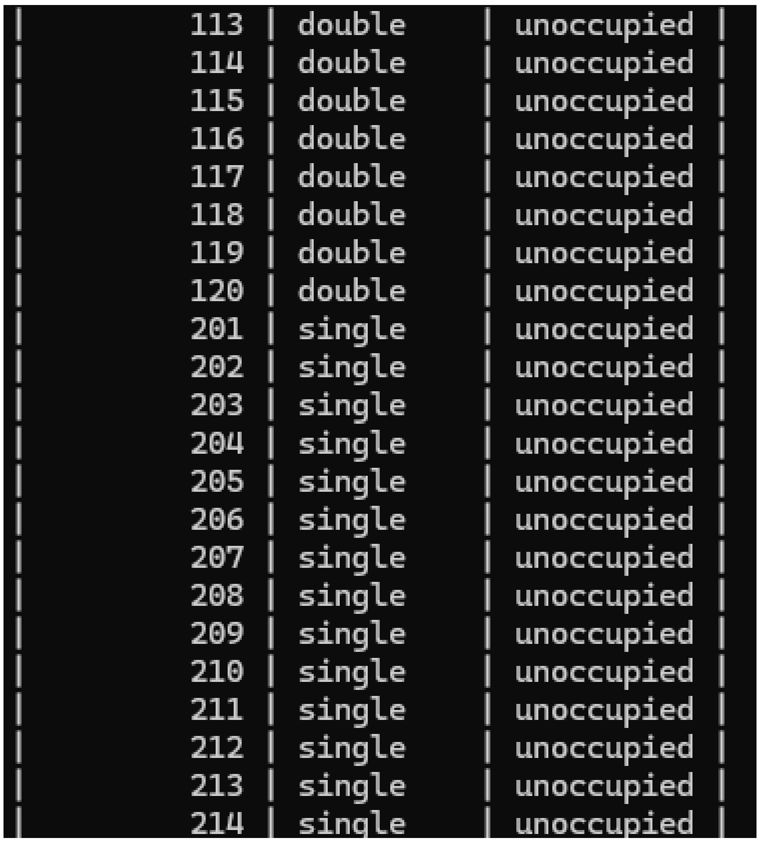
Room Number: 201
Name: priya
Email: priya
Phone Num
Address: pr
Aadhar: 83
Check-in: 2
Adults: 1
Children: 0

Message

Checked out room number: 201



```
mysql> select * From check_out;
+----+-------------+-------------+--------------------+------------+------------+------------+-------+----------+------------------------+--------------+
| id | room_number | name        | email              | phn_number | checkin    | checkout   | adult | children | address                | aadhar       |
+----+-------------+-------------+--------------------+------------+------------+------------+-------+----------+------------------------+--------------+
|  1 |         101 | Akash       | a@gmail.com        | 9879878868 | 2024-05-20 | 2024-05-20 |     2 |        0 | akash house            | 988978987667 |
|  7 |         302 | abi         | abi@gmail.com      | 3874873526 | 2024-05-20 | 2024-05-20 |     1 |        0 | abi house              | 987386287565 |
|  8 |         210 | SABREEN     | sab@gmail.com      | 8767567356 | 2024-05-20 | 2024-05-20 |     1 |        0 | gam house              | 876437532543 |
|  9 |         211 | kumar       | k@gmail.com        | 6567683781 | 2024-05-20 | 2024-05-20 |     1 |        0 | kumar villa            | 327872646732 |
| 10 |         102 | rathna      | r@gmail.com        | 3546567645 | 2024-05-20 | 2024-05-20 |     1 |        0 | dsjgbgfvdc             | 355765437786 |
| 11 |         315 | devi sakthi | devi@gmail.com     | 8987867687 | 2024-05-20 | 2024-05-20 |     2 |        0 | devi house             | 979887675645 |
| 12 |         201 | priya       | priya123@gmail.com | 9878676767 | 2024-05-20 | 2024-05-20 |     1 |        0 | priya house ,koyambedu | 837498478972 |
+----+-------------+-------------+--------------------+------------+------------+------------+-------+----------+------------------------+--------------+
7 rows in set (0.00 sec)
```



```
|         113 | double    | unoccupied |
|         114 | double    | unoccupied |
|         115 | double    | unoccupied |
|         116 | double    | unoccupied |
|         117 | double    | unoccupied |
|         118 | double    | unoccupied |
|         119 | double    | unoccupied |
|         120 | double    | unoccupied |
|         201 | single    | unoccupied |
|         202 | single    | unoccupied |
|         203 | single    | unoccupied |
|         204 | single    | unoccupied |
|         205 | single    | unoccupied |
|         206 | single    | unoccupied |
|         207 | single    | unoccupied |
|         208 | single    | unoccupied |
|         209 | single    | unoccupied |
|         210 | single    | unoccupied |
|         211 | single    | unoccupied |
|         212 | single    | unoccupied |
|         213 | single    | unoccupied |
|         214 | single    | unoccupied |
```

# CHAPTER 6
# CONCLUSION

Our Hotel Management System represents a significant advancement in streamlining hotel operations and enhancing customer service standards. Through meticulous design, implementation, and testing, the system now offers robust functionality and user-friendly features. Critical functionalities such as room booking and checkout processes are seamlessly integrated, ensuring smooth user experiences and accurate data management. Moving forward, opportunities for improvement include enhancing security through user authentication systems and integrating online payment gateways for guest convenience. In summary, our Hotel Management System effectively meets current needs, and ongoing refinement will ensure its continued relevance and effectiveness in meeting the evolving demands of the hospitality sector.

# CHAPTER 7
# REFERENCES

- Java Documentation: https://docs.oracle.com/javase/8/docs/
- MySQL Documentation: https://dev.mysql.com/doc/
- Swing Tutorial: https://docs.oracle.com/javase/tutorial/uiswing/