K.RAMAKRISHNAN
COLLEGE OF TECHNOLOGY
An Autonomous Institution
Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015 & ISO 14001:2015 Certified Institution, Accredited with 'A+' grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.

A Project Report

on

HOTEL MANAGEMENT SYSTEM

Submitted in partial fulfillment of requirements for the award of the course

of

EGB1201 – JAVA PROGRAMMING

Under the guidance of

Mrs. Hema R., M.E.,

Assistant Professor / Information Technology

Submitted By

GAYATHRI.M(2303811710622027)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(Autonomous)

TRICHY - 621112

DECEMBER 2024

K.RAMAKRISHNAN
COLLEGE OF TECHNOLOGY
An Autonomous Institution
Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015 & ISO 14001:2015 Certified Institution, Accredited with 'A+' grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(Autonomous Institution affiliated to Anna University, Chennai)

TRICHY - 621112

BONAFIDE CERTIFICATE

Certified that this project report on **“HOTEL MANAGEMENT SYSTEM”** is the Bonafide work of **GAYATHRI.M(2303811710622027)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

Signature	Signature
Mrs. HEMA R., M.E.,	**Dr. SYEDAKBAR S., M.E.,Ph.D.,**
SUPERVISOR,	**HEAD OF THE DEPARTMENT,**
Department of Information Technology,	Department of ECE,
K. Ramakrishnan College of Technology,	K. Ramakrishnan College of Technology,
Trichy - 621112	Trichy - 621112

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

VISION OF THE DEPARTMENT

To create innovative and socially responsible Electronics and Communication Engineers with design skills and research focus to meet Societal and Industrial needs.

MISSION OF THE DEPARTMENT

- M1: To provide high quality education and professional ethics to students through enhanced learning environment
- M2: To impart a creative environment towards centre of excellence in department with design skill and exposure for research.
- M3: To nurture required employable skills of students to satisfy the industry and social needs with ethical and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

- PEO1: Core Knowledge Development: Graduates will have enhanced engineering skills in the field of electronics, communication and interdisciplinary areas to serve the society with global standards.

- PEO2: Professional development: Graduates will apply the technical knowledge for continuous up gradation of their professional skills to become an inimitable employee, researcher or entrepreneur.

- PEO3: Analytical Thinking: Graduates will have analytic and thinking skills to provide the innovative solutions for industry and societal requirements.

PROGRAM OUTCOMES

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- PSO1: To analyse, design and develop solutions by applying foundational concepts of electronics and communication engineering.
- PSO2: To apply design principles and best practices for developing quality products for scientific and business applications.

ABSTRACT

The Hotel Management System (HMS) is a software application designed to streamline the operations and management of hotel services. It is an integrated system that automates various functions within a hotel, including reservations, room management, billing, guest check-in/check-out, and inventory control. The goal of the system is to improve operational efficiency, enhance guest experiences, and optimize resource utilization.The HMS allows hotel staff to manage bookings and room availability in real-time, process payments securely, and generate reports for financial and operational analysis. Guests can make online reservations, check room availability, and view services offered by the hotel. The system also supports customer relationship management (CRM) by tracking guest preferences, handling requests, and providing personalized services.This system provides a user-friendly interface for hotel staff, reduces manual work, and minimizes human errors. It ensures smooth operations, quicker service delivery, and enhanced guest satisfaction. The system is designed to be scalable and adaptable, suitable for hotels of different sizes and types, from small boutique hotels to large luxury resorts. Furthermore, it integrates with other hotel management tools such as accounting software, inventory management, and employee scheduling, providing a holistic approach to managing the hotel’s operations efficiently. The implementation of this Hotel Management System results in better resource allocation, improved customer service, and increased profitability for hotel businesses. The Hotel Management System (HMS) is an automated solution developed to optimize and manage the day-to-day operations of a hotel or resort. In the hospitality industry, efficient management is crucial for ensuring guest satisfaction, improving operational workflows, and maximizing revenue. The HMS provides a comprehensive platform that addresses various key aspects of hotel management, including reservations, billing, room allocation, front desk services, housekeeping, customer feedback, and inventory management.The **Hotel Management System (HMS)** is a comprehensive software solution designed to automate and optimize the operations of a hotel.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
The Hotel Management System (HMS) is a software application designed to automate and streamline the operations of a hotel, including front desk management, reservations, billing, and customer service. It helps hotel management staff efficiently manage the booking process, track room availability, and maintain guest records.	**PO 1** **PO 2** **PO 3** **PO 4** **PO 5** **PO 7** **PO 11**	**PSO 1** **PSO 2** **PSO 3**

Note: 1- Low, 2-Medium, 3- High

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective of a Hotel Management System (HMS) is to streamline and automate hotel operations, improving efficiency and enhancing the overall guest experience. By integrating key functions such as reservations, room management, check-in/check-out, billing, and housekeeping, the system reduces manual tasks and minimizes errors, leading to smoother operations. Additionally, HMS aims to optimize resource management, ensuring that rooms, staff, and inventory are efficiently allocated. It also helps in maximizing revenue through dynamic pricing, real-time booking management, and by providing detailed financial reports for informed decision-making. Furthermore, the system enhances communication across departments, ensuring quick response times to guest requests. Through effective customer relationship management (CRM), the HMS tailors services to guest preferences, fostering loyalty and improving customer satisfaction. By providing secure handling of guest data and integrating with other hotel management tools, the system supports long-term scalability and adaptability, allowing hotels to grow without compromising service quality.

1.2 Overview

A Hotel Management System (HMS) is an integrated software solution designed to streamline and automate various operations within a hotel or hospitality business. It serves as a comprehensive tool for managing all aspects of hotel operations, from room reservations and guest check-in/check-out to billing, housekeeping, and inventory management. The system helps to simplify daily tasks, improve operational efficiency, and enhance guest experiences. HMS is a combination of several modules that work together to ensure smooth functioning. It includes functionalities for reservation management, which allows guests to book rooms online or in-person, with real-time availability and automatic updates to avoid overbookings. The system also handles guest check-ins and check-outs, ensuring that the process is quick, secure, and accurate. Billing and accounting modules automatically generate invoices, manage payments, and track financial transactions, reducing manual errors and ensuring transparency.

1.3 Java Programming Concepts

The development of a Hotel Management System (HMS) involves various programming concepts and methodologies that enable it to function efficiently and integrate various modules seamlessly. Below are key programming concepts commonly used in the design and development of an HMS

➢ Object-Oriented Programming (OOP)

The use of Object-Oriented Programming (OOP) is fundamental in building a robust and modular Hotel Management System. OOP principles such as **Encapsulation**, **Inheritance**, **Polymorphism**, and **Abstraction** help to structure the system efficiently. The HMS can be divided into objects representing various entities like **Rooms**, **Guests**, **Employees**, **Reservations**, and **Invoices**.

➢ Database Management and SQL

A critical aspect of an HMS is managing large amounts of data efficiently. Relational Database Management Systems (RDBMS) such as **MySQL**, **PostgreSQL**, or **SQLite** are used to store and retrieve hotel data. Structured Query Language (SQL) is used for querying and managing this data

➢ Data Structures

Data structures like **arrays**, **lists**, **queues**, **stacks**, and **hash tables** are used to store and manage data within the system

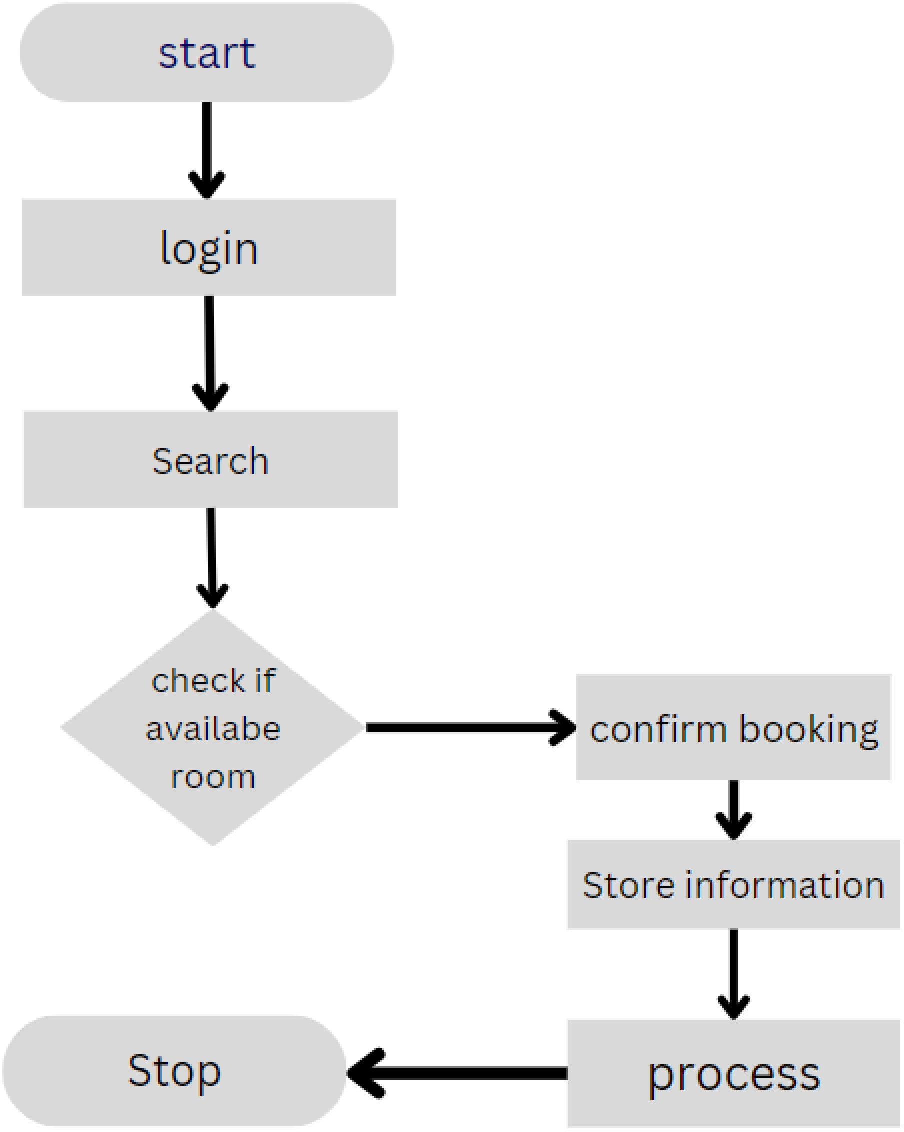
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the development of a Hotel Management System (HMS) involves creating a comprehensive, automated platform that will streamline and improve various operations of a hotel, from guest bookings to check-out and reporting. The system will be designed to enhance the guest experience, optimize resource allocation, and provide real-time data for management decision-making. Below is a detailed outline of the proposed work for the Hotel Management system. A **Hotel Management System (HMS)** is a comprehensive software solution designed to automate and streamline the day-to-day operations of a hotel. Its primary objective is to improve efficiency, enhance guest experiences, and optimize resource management. The system includes modules for **reservations**, where guests can book rooms, check availability, and manage their stay, along with **front desk operations** for seamless check-ins/outs and billing. It also covers **room management**, tracking room status, availability, and pricing, while integrating **housekeeping**, inventory control, and maintenance management. The **billing system** generates invoices, processes payments, and applies discounts or offers, ensuring accurate financial tracking. A key feature is **reporting and analytics**, offering real-time insights into occupancy rates, revenue, and staff performance. The system can integrate with **customer relationship management (CRM)** tools to handle guest feedback, loyalty programs, and personalized offer.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 Module 1

Reservation Management Module

- **Purpose**: Manages guest bookings, including room availability, reservations, cancellations, and modifications.
- **Key Features**:
  - Real-time room availability and booking.
  - Online reservation integration.
  - Reservation confirmation and modification.
  - Managing group bookings.

3.2 Module 2

Front Desk Management Module

- **Purpose**: Handles guest check-ins and check-outs, room assignments, and guest information.
- **Key Features**:
  - Guest check-in/check-out process.
  - Room allocation and management.
  - Display of room availability and status.
  - Guest profile management.
  - Handling guest requests and complaints.

3.3 Module 3

Billing and Payment Module

- **Purpose**: Manages invoicing, billing, payments, and financial transactions.
- **Key Features**:

- Generation of invoices for rooms and additional services.
- Payment processing (credit cards, online payments, cash).
- Handling taxes, discounts, and refunds.
- Managing deposits and outstanding balances.
- Generating financial reports (revenue, payments).

3.4 Module 4

Room Management Module

- **Purpose**: Tracks and manages room status, availability, and maintenance.
- **Key Features**:
    - Room status updates (clean, dirty, under maintenance).
    - Managing room types and rates.
    - Housekeeping scheduling and room inspection.
    - Tracking room occupancy and availability in real-time.
    - Assigning rooms to guests based on availability and preferences.

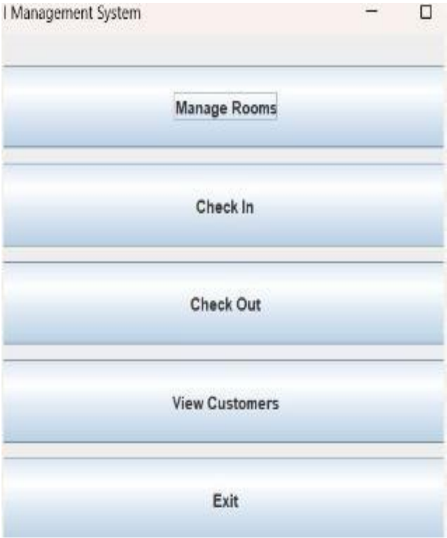
3.5 Module 5

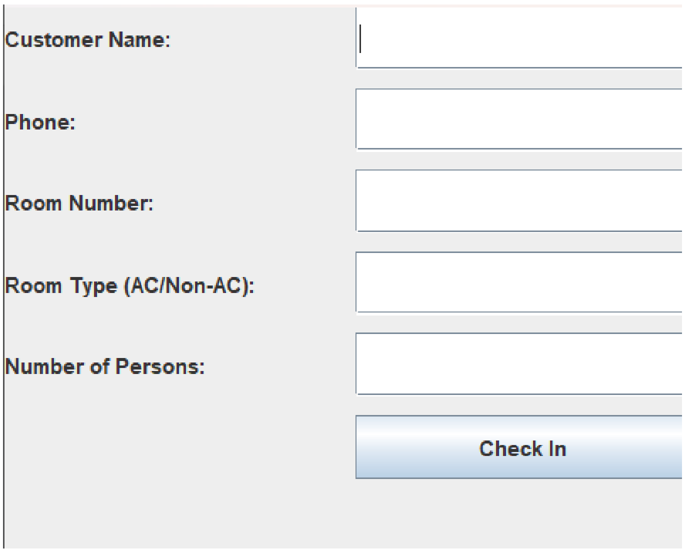
Staff and Workforce Management Module

- **Purpose**: Manages hotel staff schedules, performance, and payroll.
- **Key Features**:
    - Managing staff attendance, shifts, and time-offs.
    - Tracking staff performance and task assignments.
    - Payroll management and salary calculation.
    - Generating reports on workforce productivity.

CHAPTER 4

RESULTS AND DISCUSSION



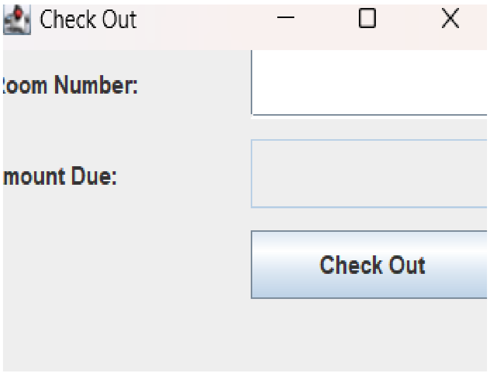

Customer Name:

Phone:

Room Number:

Room Type (AC/Non-AC):

Number of Persons:



:oom Number:

mount Due:

CHAPTER 5

CONCLUSION

The implementation of the Hotel Management System has brought about significant improvements in operational efficiency, customer satisfaction, and revenue management. Despite the challenges faced during the integration and initial training phase, the long-term benefits far outweigh the initial hurdles. By automating key processes, enhancing communication between departments, and offering personalized services to guests, the system has proven to be a valuable asset for the hotel. With continuous optimization and future technological advancements, the HMS will continue to drive growth, customer loyalty, and operational excellence. A **Hotel Management System (HMS)** built using **Java** can provide an efficient, scalable, and user-friendly solution for managing hotel operations. By using Java, developers can take advantage of its object-oriented features, platform independence, and vast array of libraries, making it a great choice for developing a hotel management system that can handle multiple functionalities.a Java-based hotel management system offers a strong foundation for managing various aspects of hotel operations. It provides significant advantages in terms of flexibility, scalability, and ease of maintenance. With continued development, this system can evolve to meet the growing needs of the hospitality industry.

REFERENCES

1. "Head First Java" by Kathy Sierra and Bert Bates
2. "Java: The Complete Reference" by Herbert Schildt
3. TutorialsPoint Java Tutorial
4. GeeksforGeeks Java Programming
5. JDBC Tutorial
6. Hotel Management System GitHub Repository
7. John R. Walker:Introduction to Hospitality Management
8. David K. Hayes:Revenue Management for the Hospitality Industry
9. Michael J. O’Fallon & Denney G. Rutherford:Hotel Management and Operations
10. Robert A. A. L. McCool: Strategic Hotel Management

APPENDIX

(Coding)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.HashMap;

public class HotelManagementSystem {
    // Data Storage
    private static HashMap<Integer, Room> rooms = new HashMap<>();
    private static HashMap<Integer, Customer> customers = new HashMap<>();
    private static int bookingIdCounter = 1;

    public static void main(String[] args) {
        // Initialize GUI
        JFrame frame = new JFrame("Hotel Management System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 400);

        // Main Menu
        JPanel menuPanel = new JPanel();
        menuPanel.setLayout(new GridLayout(5, 1, 10, 10));
        menuPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        JButton btnManageRooms = new JButton("Manage Rooms");
        JButton btnCheckIn = new JButton("Check In");
        JButton btnCheckOut = new JButton("Check Out");
        JButton btnViewCustomers = new JButton("View Customers");
        JButton btnExit = new JButton("Exit");

        menuPanel.add(btnManageRooms);
        menuPanel.add(btnCheckIn);
        menuPanel.add(btnCheckOut);
        menuPanel.add(btnViewCustomers);
        menuPanel.add(btnExit);

        frame.add(menuPanel);
        frame.setVisible(true);

        // Button Listeners
        btnManageRooms.addActionListener(e -> openManageRooms(frame));
        btnCheckIn.addActionListener(e -> openCheckIn(frame));
        btnCheckOut.addActionListener(e -> openCheckOut(frame));
        btnViewCustomers.addActionListener(e -> viewCustomers(frame));
        btnExit.addActionListener(e -> System.exit(0));
    }

    // Manage Rooms
    private static void openManageRooms(JFrame parentFrame) {
        JFrame frame = new JFrame("Manage Rooms");
        frame.setSize(400, 300);
        frame.setLayout(new BorderLayout());

        JPanel topPanel = new JPanel(new FlowLayout());
        JLabel lblRoomNumber = new JLabel("Room Number:");
        JTextField txtRoomNumber = new JTextField(10);
        JLabel lblRoomType = new JLabel("Room Type (AC/Non-AC):");
        JTextField txtRoomType = new JTextField(10);
        JButton btnAddRoom = new JButton("Add Room");
        topPanel.add(lblRoomNumber);
        topPanel.add(txtRoomNumber);
        topPanel.add(lblRoomType);
        topPanel.add(txtRoomType);
        topPanel.add(btnAddRoom);

        DefaultListModel<String> roomListModel = new DefaultListModel<>();
        JList<String> roomList = new JList<>(roomListModel);
        rooms.forEach((k, v) -> roomListModel.addElement("Room " + k + " - " + (v.isOccupied() ? "Occupied" : "Available") + " - " + (v.isAc() ? "AC" : "Non-AC")));
        JScrollPane scrollPane = new JScrollPane(roomList);

        btnAddRoom.addActionListener(e -> {
            try {
                int roomNumber = Integer.parseInt(txtRoomNumber.getText());
                String roomType = txtRoomType.getText().toUpperCase();

                if (rooms.containsKey(roomNumber)) {
                    JOptionPane.showMessageDialog(frame, "Room already exists!", "Error", JOptionPane.ERROR_MESSAGE);
                } else if (!roomType.equals("AC") && !roomType.equals("NON-AC")) {
                    JOptionPane.showMessageDialog(frame, "Invalid room type! Please enter AC or Non-AC.", "Error", JOptionPane.ERROR_MESSAGE);
                } else {
                    Room room = new Room(roomNumber, roomType.equals("AC"));
                    rooms.put(roomNumber, room);
                    roomListModel.addElement("Room " + roomNumber + " - Available - " + (room.isAc() ? "AC" : "Non-AC"));
                    txtRoomNumber.setText("");
                    txtRoomType.setText("");
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, "Invalid room number!", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        });

        frame.add(topPanel, BorderLayout.NORTH);
        frame.add(scrollPane, BorderLayout.CENTER);
        frame.setVisible(true);
    }

    // Check-In
    private static void openCheckIn(JFrame parentFrame) {
        JFrame frame = new JFrame("Check In");
        frame.setSize(400, 350);  // Increased size for number of persons
        frame.setLayout(new GridLayout(7, 2, 10, 10));  // Increased grid to add a new row for number
of persons
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        JLabel lblName = new JLabel("Customer Name:");
        JTextField txtName = new JTextField();
        JLabel lblPhone = new JLabel("Phone:");
        JTextField txtPhone = new JTextField();
        JLabel lblRoomNumber = new JLabel("Room Number:");
        JTextField txtRoomNumber = new JTextField();
        JLabel lblRoomType = new JLabel("Room Type (AC/Non-AC):");
        JTextField txtRoomType = new JTextField();
        JLabel lblNumPersons = new JLabel("Number of Persons:");
        JTextField txtNumPersons = new JTextField();  // New field for number of persons
        JButton btnCheckIn = new JButton("Check In");

        frame.add(lblName);
        frame.add(txtName);
        frame.add(lblPhone);
        frame.add(txtPhone);
        frame.add(lblRoomNumber);
        frame.add(txtRoomNumber);
        frame.add(lblRoomType);
        frame.add(txtRoomType);
        frame.add(lblNumPersons);  // Add label for number of persons
        frame.add(txtNumPersons);  // Add text field for number of persons
        frame.add(new JLabel());
        frame.add(btnCheckIn);

        btnCheckIn.addActionListener(e -> {
            String name = txtName.getText();
            String phone = txtPhone.getText();
            int roomNumber;
            String roomType = txtRoomType.getText().toUpperCase();
            int numPersons;

            try {
                roomNumber = Integer.parseInt(txtRoomNumber.getText());
                numPersons = Integer.parseInt(txtNumPersons.getText());  // Parse the number of persons
                if (numPersons <= 0) {
                    JOptionPane.showMessageDialog(frame, "Number of persons must be greater than 0!",
"Error", JOptionPane.ERROR_MESSAGE);
                    return;
                }
                if (!rooms.containsKey(roomNumber)) {
                    JOptionPane.showMessageDialog(frame, "Room does not exist!", "Error", JOptionPane.ERROR_MESSAGE);
                    return;
                }

                Room room = rooms.get(roomNumber);
                if (room.isOccupied()) {
                    JOptionPane.showMessageDialog(frame, "Room is already occupied!", "Error", JOptionPane.ERROR_MESSAGE);
                    return;
                }

                // Check if room type matches
                if (!roomType.equals("AC") && !roomType.equals("NON-AC")) {
                    JOptionPane.showMessageDialog(frame, "Invalid room type! Please enter AC or Non-AC.", "Error", JOptionPane.ERROR_MESSAGE);
                    return;
                }

                if ((roomType.equals("AC") && !room.isAc()) || (roomType.equals("NON-AC") && room.isAc())) {
                    JOptionPane.showMessageDialog(frame, "Room type does not match!", "Error", JOptionPane.ERROR_MESSAGE);
                    return;
                }

                Customer customer = new Customer(bookingIdCounter++, name, phone, roomNumber, numPersons);  // Create customer with numPersons
                customers.put(customer.getId(), customer);
```

```
                room.setOccupied(true);
                JOptionPane.showMessageDialog(frame, "Check-in successful!");
                txtName.setText("");
                txtPhone.setText("");
                txtRoomNumber.setText("");
                txtRoomType.setText("");
                txtNumPersons.setText("");  // Clear the number of persons field
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, "Invalid input! Please enter valid details.",
"Error", JOptionPane.ERROR_MESSAGE);
            }
        });

        frame.setVisible(true);
    }

    // Check-Out with Payment
    private static void openCheckOut(JFrame parentFrame) {
        JFrame frame = new JFrame("Check Out");
        frame.setSize(300, 200);
        frame.setLayout(new GridLayout(4, 2, 10, 10));

        JLabel lblRoomNumber = new JLabel("Room Number:");
        JTextField txtRoomNumber = new JTextField();
        JButton btnCheckOut = new JButton("Check Out");
        JLabel lblAmount = new JLabel("Amount Due: ");
        JTextField txtAmountDue = new JTextField();
        txtAmountDue.setEditable(false);

        frame.add(lblRoomNumber);
        frame.add(txtRoomNumber);
        frame.add(lblAmount);
        frame.add(txtAmountDue);
        frame.add(new JLabel());
        frame.add(btnCheckOut);

        btnCheckOut.addActionListener(e -> {
            try {
                int roomNumber = Integer.parseInt(txtRoomNumber.getText());
                if (!rooms.containsKey(roomNumber)) {
                    JOptionPane.showMessageDialog(frame, "Room does not exist!", "Error",
JOptionPane.ERROR_MESSAGE);
                    return;
                }
                Room room = rooms.get(roomNumber);
                if (!room.isOccupied()) {
                    JOptionPane.showMessageDialog(frame, "Room is not occupied!", "Error",
JOptionPane.ERROR_MESSAGE);
                    return;
                }

                Customer customer = findCustomerByRoomNumber(roomNumber);
                if (customer != null) {
                    // Show amount due
                    txtAmountDue.setText("$" + room.getRoomPrice());

                    // Prompt for payment
                    String paymentMethod = JOptionPane.showInputDialog(frame, "Enter payment
method (Cash/Card):");
                    if (paymentMethod != null) {
                        JOptionPane.showMessageDialog(frame, "Check-out successful! Thank you for
your payment.");
```

```
                        room.setOccupied(false);
                        customer.setPaid(true);
                    }
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, "Invalid room number!", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        });

        frame.setVisible(true);
    }

    // View Customers
    private static void viewCustomers(JFrame parentFrame) {
        JFrame frame = new JFrame("View Customers");
        frame.setSize(400, 300);

        DefaultListModel<String> customerListModel = new DefaultListModel<>();
        JList<String> customerList = new JList<>(customerListModel);
        customers.forEach((k, v) -> customerListModel.addElement("ID: " + k + ", Name: " +
v.getName() + ", Room: " + v.getRoomNumber() + ", Persons: " + v.getNumPersons()));  // Display
number of persons

        JScrollPane scrollPane = new JScrollPane(customerList);

        frame.add(scrollPane);
        frame.setVisible(true);
    }

    // Helper methods for Room and Customer
```

```
static class Room {
    private int roomNumber;
    private boolean isAc;
    private boolean isOccupied;

    public Room(int roomNumber, boolean isAc) {
        this.roomNumber = roomNumber;
        this.isAc = isAc;
        this.isOccupied = false;
    }

    public int getRoomNumber() {
        return roomNumber;
    }

    public boolean isAc() {
        return isAc;
    }

    public boolean isOccupied() {
        return isOccupied;
    }

    public void setOccupied(boolean occupied) {
        isOccupied = occupied;
    }

    public double getRoomPrice() {
        return isAc ? 200.0 : 150.0;
    }
}
```

```
static class Customer {
    private int id;
    private String name;
    private String phone;
    private int roomNumber;
    private boolean isPaid;
    private int numPersons;  // Added field for the number of persons

    public Customer(int id, String name, String phone, int roomNumber, int numPersons) {
        this.id = id;
        this.name = name;
        this.phone = phone;
        this.roomNumber = roomNumber;
        this.isPaid = false;
        this.numPersons = numPersons;  // Initialize the number of persons
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getPhone() {
        return phone;
    }

    public int getRoomNumber() {
            return roomNumber;
        }

        public boolean isPaid() {
            return isPaid;
        }

        public void setPaid(boolean isPaid) {
            this.isPaid = isPaid;
        }

        public int getNumPersons() {
            return numPersons;  // Getter for the number of persons
        }

        public void setNumPersons(int numPersons) {
            this.numPersons = numPersons;  // Setter for the number of persons
        }
    }

    // Find a customer by their room number
    private static Customer findCustomerByRoomNumber(int roomNumber) {
        for (Customer customer : customers.values()) {
            if (customer.getRoomNumber() == roomNumber) {
                return customer;
            }
        }
        return null;
    }
}
```