G. H. RAISONI COLLEGE OF ENGINEERING
& MANAGEMENT, WAGHOLI, PUNE
(An Empowered Autonomous Institute Affiliated to SPPU, Pune)

Department of Information Technology

TAE-2

Subject: Java Programming

Mini Project Report On

"Hotel Management System"

Academic Year 2024-25

Semester-V

G. H. RAISONI COLLEGE OF ENGINEERING & MANAGEMENT, WAGHOLI, PUNE
(An Empowered Autonomous Institute Affiliated to SPPU, Pune)

Department of Information Technology

Academic Year 2024-25

Semester-V

Guided By:-Mrs. Kiran Patil

Submitted by:

1. TYITA32 Bhakti Chavan
2. TYITA60 Krishna Makwana
3. TYITA68 Aau Somwanshi
4. TYITA71 Kalpesh Khairnar
5. TYITA73 Vaishnavi Chakor

GH raisoni COLLEGE
Engineering and Management Pune

CERTIFICATE

This is to certify that **"**Hotel Booking System**"** embodies the original work done by **during Ms.Bhakti Chavan, Mr.Krishna Makwana, Ms.Aau Somwanshi, Mr.Kalpesh Khairnar and Ms.Vaishnavi Chakor** this project submission as a partial fulfilment of the requirement for the Mini Project in subject Java Programming of TY IT, V Semester, of Pune University during the academic year 2024-2025.

Date: 04-09-2024

Place: Pune

Project Guide

(Mrs.Kiran Patil)

Head of Department

(Dr. Poonam Gupta)

ACKNOWLEDGEMENT

We would like to express our sincere thanks to Mrs.Kiran Patil under whose valuable guidance and light of knowledge, we could complete this project.
We take this opportunity to thank all the staff members of Department Of Information Technology Engineering for their help whenever required. Finally we express sincere thanks to all those who have helped us directly or indirectlyin many ways in completion of this project work and I would like to extend my Deep appreciation to all my group members, without their support and Coordination we would not have been able to complete this Project.

INDEX

1. ABSTRACT

The **Hotel Booking System** is a robust software application designed to streamline the process of booking rooms in hostels. It offers a centralized platform for managing reservations, room availability, guest check-ins and check-outs, and payment processing. Developed using Java and powered by a user-friendly interface created with Java Swing, this application simplifies hostel management. It operates in both online and offline modes, ensuring flexibility and reliability in a variety of hostel settings. The system is built to handle multiple bookings simultaneously, reducing administrative workload while enhancing the guest experience.

2. INTRODUCTION

Hostel management involves numerous tasks such as tracking room availability, managing guest records, and processing payments. Handling these tasks manually can lead to inefficiencies and errors, especially during peak seasons. The **Hotel Booking System** addresses these challenges by providing a digital solution that automates room booking, tracks room occupancy, and processes payments securely.

The system allows hostel managers to view real-time room availability, accept reservations, generate booking confirmations, and handle cancellations. By automating these tasks, the system ensures a smooth operational flow, freeing up staff to focus on improving customer service. Additionally, the application is designed to work offline, which is crucial for hostels in remote areas with intermittent internet access.

- **Purpose:** To design a user-friendly application for converting currency values.
- **Scope:** The application will support multiple currencies and provide accurate conversion based on predefined exchange rates.

3. Objective

- **Ease of Use:** To develop a user-friendly graphical user interface (GUI) that makes it easy for hostel staff to manage bookings and track room availability without requiring extensive technical knowledge.
- **Room Availability Management:** To enable real-time tracking of room availability, preventing double bookings and ensuring optimal resource utilization.
- **Automated Booking Process:** To automate the process of room allocation, booking confirmation, and guest check-in/check-out, thereby reducing the need for manual intervention.
- **Payment Handling:** To incorporate a secure payment processing system that allows for seamless billing, invoicing, and payment collection.
- **Offline Functionality:** To provide offline support, allowing the system to function in areas with limited or no internet connectivity.
- **Reporting and Analytics:** To generate detailed reports on bookings, room utilization, and financials, helping hostel managers make informed business decision.

4.Executive Summary

The **Hotel Booking System** is designed to make the booking process quick and efficient for both hotel administrators and guests. By using Java Swing for the graphical user interface, the system ensures that users can navigate through the booking and room management process with ease.

Highlights:

- **User-Friendly Interface:** A clean and intuitive interface that allows hostel staff to check room availability, manage guest bookings, and process payments seamlessly.
- **Automated Room Management:** The system automatically updates room availability and tracks guest check-ins/check-outs in real-time.
- **Offline Support:** The application can function offline, ensuring uninterrupted service in remote locations or during internet outages.
- **Cross-Platform Compatibility:** The system works on major desktop platforms, including Windows, macOS, and Linux, making it accessible across different environments.
- **Enhanced Security:** Secure handling of sensitive guest information and payment details.

5.About Hotel Booking System

The **Hotel Booking System** is a specialized application built to handle the unique challenges of managing a hostel. It automates the process of room assignment, booking confirmations, and payment processing, helping hostel staff handle multiple tasks efficiently.

Strengths:

1. **Real-Time Room Availability:** The system keeps track of room occupancy and availability in real time, preventing overbooking.
2. **Guest Management:** The system stores and manages guest information, including personal details, stay duration, and payment history, in an organized manner.
3. **Automated Processes:** Key operations such as booking confirmation, check-ins, and check-outs are automated, reducing the workload on staff.
4. **Offline Operation:** The application can operate without an internet connection, syncing data when the connection is restored.
5. **User-Friendly Design:** The system is designed to be easy to use for both technical and non-technical staff members.

Limitations:

1. **Static Pricing Model:** The system may need additional configurations for dynamic pricing based on peak seasons or room demand.
2. **Limited to Desktop:** The current version is desktop-based, and does not have mobile or web interfaces.

Data Flow Diagram:

External Entities:

- User: Provides input and receives output.

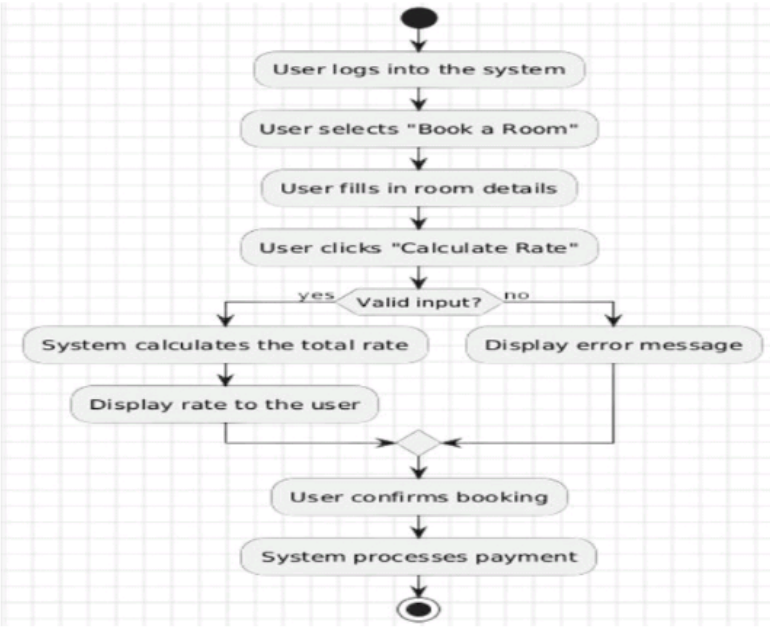
Process:

- Hotel Booking System.

Data Flows:

The arrows indicate the flow of data between components, such as:

- Guests providing booking information.
- Room management checking for room availability.
- The payment system processing payment details.
- The system confirming the booking and updating the database.



6.Features of Hotel Booking system

Room Availability Tracking: The system displays available rooms in real time, allowing staff to
quickly check which rooms are vacant or occupied.

Guest Booking Management: Hotel staff can easily input guest details, choose a

room, and

confirm bookings through the system.

Automated Confirmation: The system generates a unique booking ID and sends a

confirmation

message to the guest.

 Payment Processing: The system handles billing, allowing guests to pay for their stay an invoices.

 Error Handling: The system validates all inputs, ensuring data accuracy and preventing ir bookings.

 Offline Support: The system works in offline mode, ensuring that bookings can still be pr without internet access.

 Report Generation: Hotel managers can generate reports on bookings, occupancy rates, revenue, providing valuable insights into hostel operations.

 Cross-Platform Compatibility: The system is compatible with Windows, macOS, and Lin operating systems.

7.Tools and Technologies

1. **Programming Language:**
   - **Java:** The core programming language used to develop the application. Java's object-oriented nature and cross-platform compatibility make it ideal for building a desktop-based system.

2. **Development Framework:**
   - **Java Swing:** Used for creating the graphical user interface (GUI). Swing components like buttons, text fields, and tables are used to build an intuitive interface.

3. **Database:**
   - **SQLite/MySQL:** The system uses a relational database to store guest details, booking information, and room availability data.

4. **IDE (Integrated Development Environment):**
   - **IntelliJ IDEA/Eclipse:** These IDEs are used for developing and debugging the application, offering robust tools for Java development.

5. **Version Control:**
   - **Git:** GitHub or Bitbucket is used to manage code versions, ensuring that updates and bug fixes are efficiently tracked and implemented.

8. Implementation: -

Use Case Diagram:

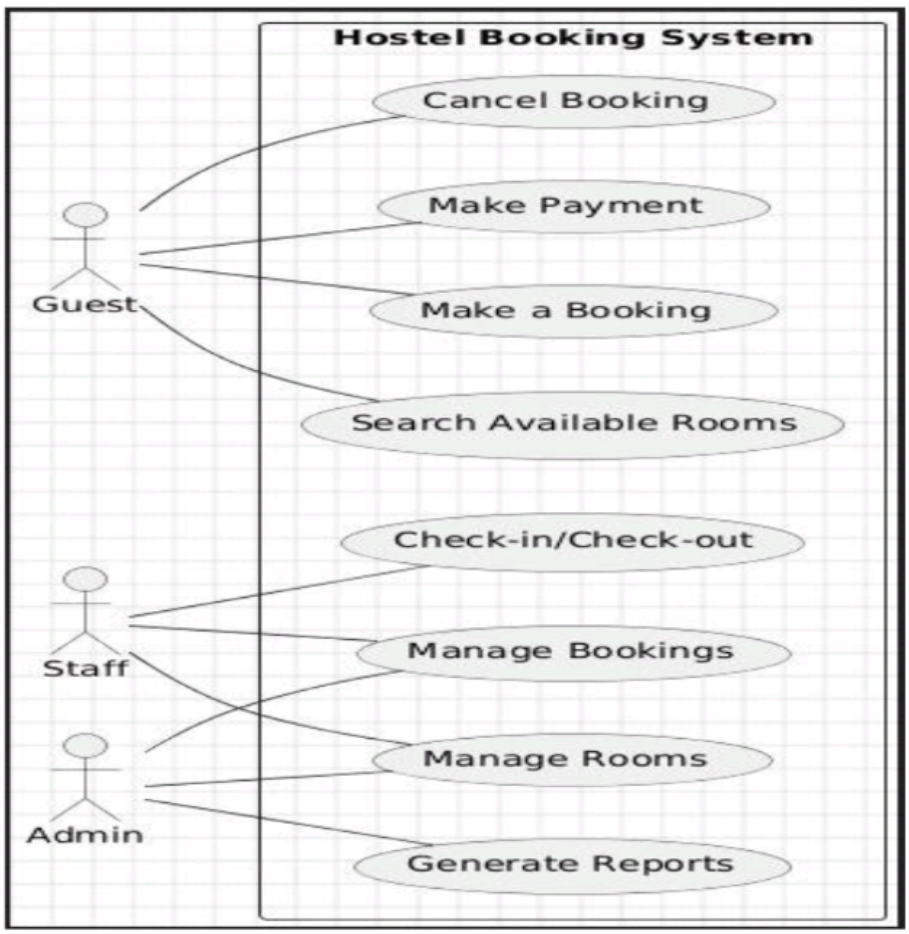
Actors:

1. User: The person interacting with the Currency Converter application.

Use Cases:

- **Search Available Rooms (U1)**: The guest searches for rooms available on selected dates.
- **Make a Booking (U2)**: The guest books a room.
- **Make Payment (U3)**: The guest makes a payment for their booking.
- **Cancel Booking (U4)**: The guest can cancel their booking if needed.
- **Check-in/Check-out (U5)**: Staff handles the check-in and check-out process for the guests.
- **Manage Bookings (U6)**: Both staff and admin can manage bookings (update or cancel reservations).
- **Manage Rooms (U7)**: Both staff and admin can update room availability, add new rooms, or modify room details.
- **Generate Reports (U8)**: Admin generates reports on bookings, room usage, and financial data.

Diagram:



Implemented Class:

1. Class: HotelBookingGUI

Purpose: Provides the graphical user interface for the hotel booking application.

- **Attributes:**
  - dateField: A JTextField for the user to input check-in and check-out dates.
  - roomTypeBox: A JComboBox<String> for selecting the desired room type (e.g., single, double,
  - dormitory).
  - guestDetailsField: A JTextField for inputting guest information (name, contact, number of guests).
    bookingResultLabel: A JLabel to display booking confirmation or error messages.
  - bookingManager: An instance of **BookingManager** that handles the booking logic.
- **Constructor:**
  - Initializes the GUI components and sets up the layout.
  - Configures the action listener for the "Book Now" button.
- **Methods:**
  - bookRoom(): Handles the room booking logic by invoking the **book()** method of the **BookingManager** class. Updates the bookingResultLabel with the booking confirmation or displays an error message if the input is invalid or rooms are unavailable.

2. Class: BookingManager

Purpose: Handles the booking logic, checking room availability and confirming bookings.

- **Attributes:**
  - roomManager: An instance of **RoomManager** to manage room availability.
  - paymentSystem: An instance of **PaymentSystem** to handle payment processing.
- **Constructor:**
  - Initializes the **BookingManager** with the necessary dependencies such as **RoomManager** and **PaymentSystem**.
- **Methods:**
  - book(): Takes guest details, room type, and dates, checks for room availability via **RoomManager**. If available, processes payment through **PaymentSystem** and returns booking confirmation. If unavailable or payment fails, returns an error message.

3. Class: RoomManager

Purpose: Manages room availability and allocation within the system.

- **Attributes:**
  - roomDatabase: A Map<String, List<Room>> representing room types and their availability status.
- **Constructor:**
  - Initializes the **RoomManager** with a list of rooms and their statuses.
- **Methods:**
  - checkAvailability(): Checks for available rooms based on the selected room type and dates.
  - updateRoomStatus(): Updates the room's status after a booking or cancellation.

4. Class: PaymentSystem

Purpose: Handles the payment processing for room bookings.

- **Attributes:**
  - paymentGateway: An instance of **PaymentGateway** to handle transactions.
- **Constructor:**
  - Initializes the **PaymentSystem** with a payment gateway instance.
- **Methods:**
  - processPayment(): Processes the payment based on guest details and booking cost, returns success or failure.

5. Class: PaymentGateway

Purpose: Interacts with external payment services to process transactions.

- **Attributes:**
  - gatewayAPI: A string representing the API endpoint of the payment service.
- **Constructor:**
  - Initializes the **PaymentGateway** with required API credentials.
- **Methods:**
  - initiateTransaction(): Sends payment details to the external service and returns the transaction result (success or failure).

6. Class: Guest

Purpose: Represents the guest making the booking.

- **Attributes:**
  - name: The guest's name.
  - contactDetails: The guest's contact information.
  - numberOfGuests: The number of guests for the booking.
- **Constructor:**
  - Initializes the guest details with the provided information.
- **Methods:**
  - getGuestDetails(): Returns the guest's information required for booking confirmation.

Interaction Diagram

User Interaction:

- The guest inputs check-in and check-out dates, selects the desired room type, and provides guest details in the **HotelBookingGUI**.

Action Trigger:

- The guest clicks the "Book Now" button, which triggers the **bookRoom()** method in **HotelBookingGUI**.

Room Availability Check:

- The **bookRoom()** method calls the **checkRoomAvailability()** method in **RoomManager**, passing the selected dates and room type.

Booking Logic:

- If a room is available, the **RoomManager** reserves the room and updates the room status. The booking details are sent to the **BookingManager** to confirm the reservation.

Payment Processing:

- The **BookingManager** triggers the **processPayment()** method in **PaymentSystem**, handling payment details and verifying the transaction.

Result Display:

- Once payment is successful, the booking confirmation is returned to the **HotelBookingGUI**, where the booking details and confirmation number are displayed to the guest. If payment fails, an error message is shown.

Code:

1. Log In Page:

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class HotelSignupPage {

public static void main(String[] args) {


    JFrame frame = new JFrame("Hotel Management System -S ignup");

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    frame.setSize(600, 400);

    JPanel backgroundPanel = new JPanel() {

      @Override

      protected void paintComponent(Graphics g) {
```

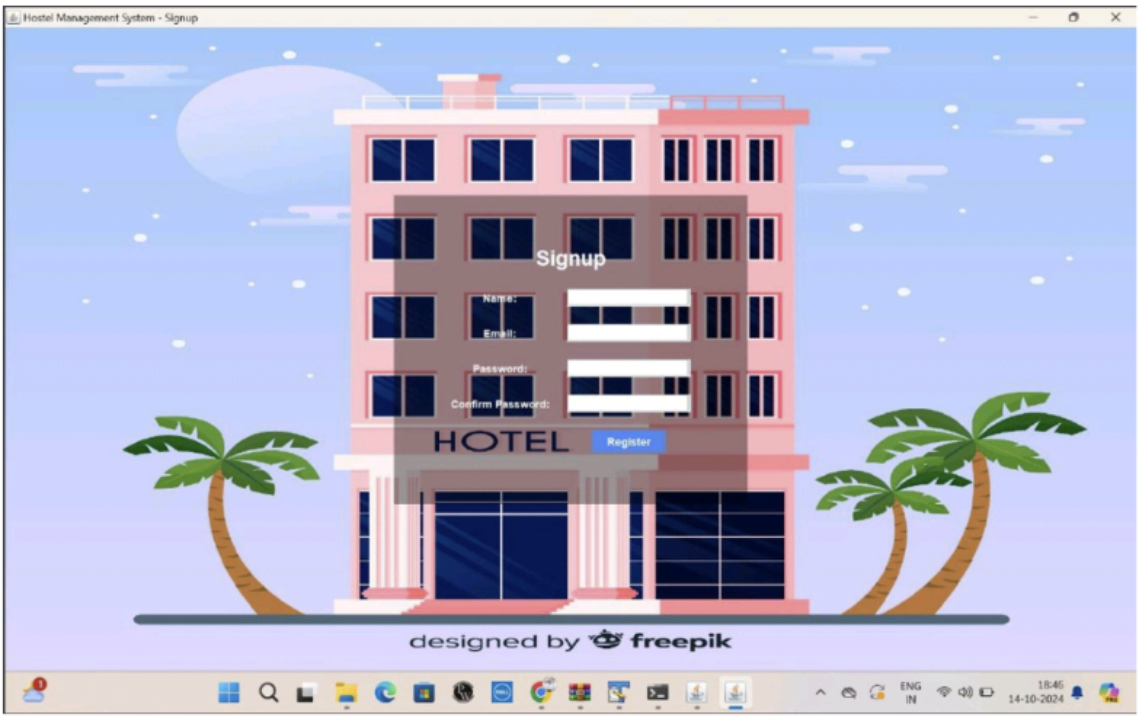
```
        super.paintComponent(g);
        ImageIcon backgroundIma ge = new ImageIcon("hotel_background.jpg");
        g.drawImage(backgroundImage.getImage(), 0, 0, getWidth(), getHeight(), this);

    }
};
backgroundPanel.setLayout(null); // Set null layout for absolute positioning
JLabel signupLabel = new JLabel("Signup");
signupLabel.setFont(new Font("Arial", Font.BOLD, 24));
signupLabel.setForeground(Color.WHITE);
signupLabel.setBounds(250, 80, 100, 30);
backgroundPanel.add(signupLabel);
JLabel nameLabel = new JLabel("Name:");
nameLabel.setForeground(Color.WHITE);
nameLabel.setBounds(200, 130, 100, 25);
backgroundPanel.add(nameLabel);
JTextField nameField = new JTextField();
nameField.setBounds(300, 130, 150, 25);
backgroundPanel.add(nameField);
 JLabel emailLabel = new JLabel("Email:");
emailLabel.setForeground(Color.WHITE);
emailLabel.setBounds(200, 170, 100, 25);
backgroundPanel.add(emailLabel);
JTextField emailField = new JTextField();
emailField.setBounds(300, 170, 150, 25);
backgroundPanel.add(emailField);
JLabel passwordLabel = new JLabel("Password:");
passwordLabel.setForeground(Color.WHITE);
```

```
passwordLabel.setBounds(200, 210, 100, 25);
backgroundPanel.add(passwordLabel);
JPasswordField passwordField = new JPasswordField();
passwordField.setBounds(300, 210, 150, 25);
backgroundPanel.add(passwordField);
JLabel confirmPasswordLabel = new JLabel("Confirm Password:");
confirmPasswordLabel.setForeground(Color.WHITE);
confirmPasswordLabel.setBounds(160, 250, 130, 25);
backgroundPanel.add(confirmPasswordLabel);
JPasswordField confirmPasswordField = new JPasswordField();
confirmPasswordField.setBounds(300, 250, 150, 25);
backgroundPanel.add(confirmPasswordField);
JButton registerButton = new JButton("Register");
registerButton.setBounds(250, 300, 100, 30);
backgroundPanel.add(registerButton);
registerButton.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      String name = nameField.getText();
      String email = emailField.getText();
      String password = new String(passwordField.getPassword());
      String confirmPassword = new String(confirmPasswordField.getPassword());
   if (password.equals(confirmPassword)) {
         JOptionPane.showMessageDialog(frame, "Registration Successful!");
      } else {
         JOptionPane.showMessageDialog(frame, "Passwords do not match!", "Error",
JOptionPane.ERROR_MESSAGE);
      }
```

```
        }
    });
    frame.add(backgroundPanel);
    frame.setVisible(true);
}
}
```

Result:



2. Booking Page :

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

public class BookRoomPage {

    public static void main(String[] args) {
        JFrame frame = new JFrame("Hotel Management System -B ook a Room");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 500);
```

```
JPanel backgroundPanel = new JPanel() {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);


        ImageIcon backgroundImage = new ImageIcon("room_background.jpg");
    } g.drawImage(backgroundImage.getImage(), 0, 0, getWidth(), getHeight(), this);
};
backgroundPanel.setLayout(null); // Set null layout for absolute positioning
JLabel titleLabel = new JLabel("Book a Room");
titleLabel.setFont(new Font("Arial", Font.BOLD, 28));
titleLabel.setForeground(Color.WHITE);
titleLabel.setBounds(320, 50, 200, 30);
backgroundPanel.add(titleLabel);
JLabel roomLabel = new JLabel("Select Room:");
roomLabel.setForeground(Color.WHITE);
roomLabel.setBounds(250, 120, 100, 25);
backgroundPanel.add(roomLabel);

String[] rooms = {"Room 101 - Single", "Room 102 - Double", "Room 103 - Deluxe"};
JComboBox<String> roomComboBox = new JComboBox<>(rooms);
roomComboBox.setBounds(350, 120, 150, 25);
backgroundPanel.add(roomComboBox);
JLabel peopleLabel = new JLabel("Number of People:");
peopleLabel.setForeground(Color.WHITE);
peopleLabel.setBounds(230, 160, 120, 25);
backgroundPanel.add(peopleLabel);
JTextField peopleField = new JTextField("1");
peopleField.setBounds(350, 160, 50, 25);
backgroundPanel.add(peopleField);
JButton calculateRateButton = new JButton("Calculate Rate");
calculateRateButton.setBounds(280, 200, 120, 30);
backgroundPanel.add(calculateRateButton);
JLabel totalLabel = new JLabel("Total: $");
totalLabel.setForeground(Color.WHITE);
totalLabel.setBounds(420, 200, 60, 25);
backgroundPanel.add(totalLabel);
JLabel totalAmountLabel = new JLabel("0");
totalAmountLabel.setForeground(Color.WHITE);
totalAmountLabel.setBounds(480, 200, 100, 25);
```

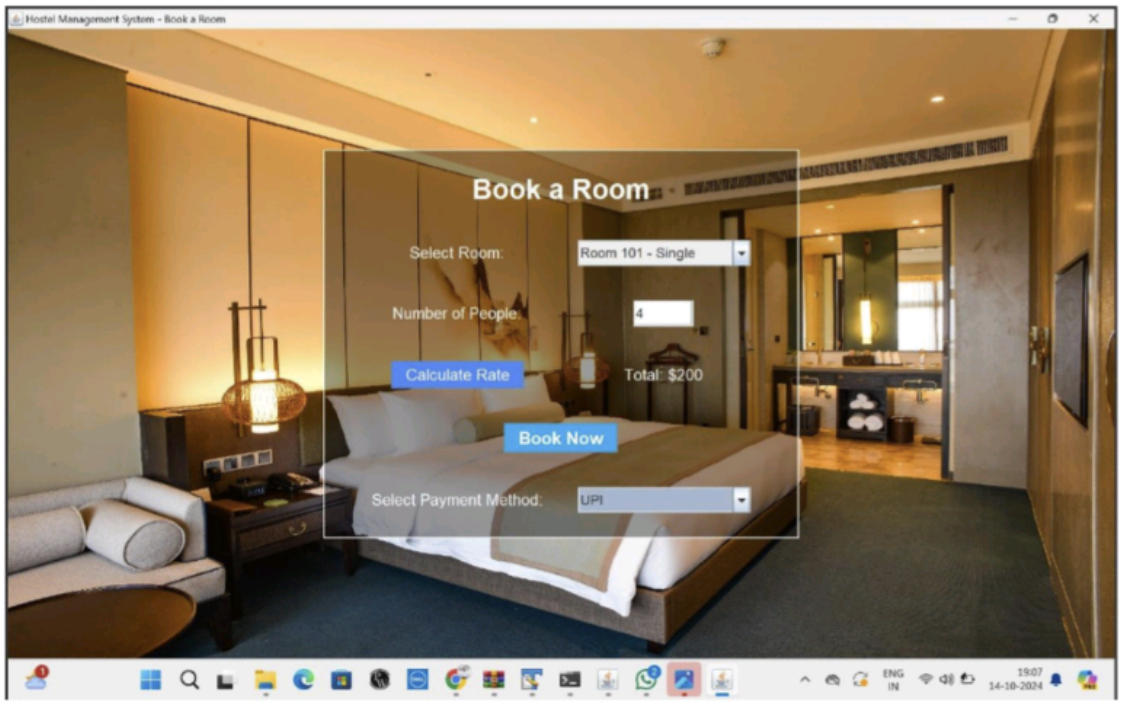
```
        backgroundPanel.add(totalAmountLabel);
        JLabel paymentLabel = new JLabel("Select Payment Method:");
        paymentLabel.setForeground(Color.WHITE);
        paymentLabel.setBounds(180, 240, 150, 25);
        backgroundPanel.add(paymentLabel);
        String[] paymentMethods = {"UPI", "Credit Card", "Debit Card", "Net Banking"};
        JComboBox<String> paymentComboBox = new JComboBox<>(paymentMethods);
        paymentComboBox.setBounds(350, 240, 150, 25);
        backgroundPanel.add(paymentComboBox);
        JButton bookNowButton = new JButton("Book Now");
        bookNowButton.setBounds(350, 280, 100, 30);
        backgroundPanel.add(bookNowButton);
        calculateRateButton.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                String selectedRoom = (String) roomComboBox.getSelectedItem();
                int numberOfPeople = Integer.parseInt(peopleField.getText());
                int rate = 0;
                if (selectedRoom != null) {
                    if (selectedRoom.contains("Single")) {
                        rate = 50;
                    } else if (selectedRoom.contains("Double")) {
                        rate = 100;
                    } else if (selectedRoom.contains("Deluxe")) {
                        rate = 200;
                    }
                }

                int totalAmount = rate * numberOfPeople;
                totalAmountLabel.setText(String.valueOf(totalAmount));
            }
        });
        bookNowButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(frame, "Room Booked Successfully!\nTotal Amount: $" +
totalAmountLabel.getText());
            }
        });
        frame.add(backgroundPanel);
        frame.setVisible(true);
```

```
        }
    }
```

Result:



9. Conclusion:

The Currency Converter application is designed to provide a straightforward and efficient solution for converting amounts between different currencies. By leveraging Java Swing for the graphical user interface and a set of predefined exchange rates, the application ensures reliable and fast conversions without the need for real-time internet access.

10. References:

- https://docs.oracle.com/javase/tutorial/
- https://www.javatpoint.com/java-swing
- https://code.visualstudio.com/docs
- https://code.visualstudio.com/docs/java/java-debugging
- https://docs.oracle.com/javase/tutorial/essential/exceptions/

GitHub Link :-

KiranPatil024/CURRENCY-CONVGUI: Java Programming Currency Converter GUI Project (github.com)