RR Hotels

Chapter 1

Introduction

1.1. **Problem description**: -

A hotel booking management system (**RR Hotels**) is a software application that helps hotels efficiently manage their guest bookings, room availability, and services. The system must provide functionality for handling reservations, tracking customer details, managing rooms, and generating reports. Below is a descriptive breakdown of the core requirements and features:

Key Features:

1. **Customer Management**:
   - Store customer details such as name, address, phone number, and email.
   - Track customer booking history for loyalty programs or personalized services.
2. **Room Management**:
   - Maintain a list of rooms with details like room number, type (e.g., single, double, suite), and availability status (vacant, reserved, occupied).
   - Define pricing for different room types and apply seasonal or promotional rates.
3. **Booking Management**:
   - Allow customers to book rooms by selecting dates, room types, and additional services (e.g., breakfast, spa, etc.).
   - Check room availability for a specified period.
   - Handle overlapping bookings by preventing double-booking of the same room.
   - Support both single and group bookings.
4. **Check-In and Check-Out**:
   - Facilitate smooth check-in/check-out by updating room status.
   - Calculate total charges based on room rate, services used, and duration of stay.
   - Generate invoices and receipts for customers.

5. **Payment and Cancellation**:
    - Track payments made by guests, including deposits or full payments.
    - Implement cancellation policies (e.g., refundable or non-refundable based on cancellation time).
    - Issue refunds for eligible cancellations.
6. **Report Generation**:
    - Generate daily reports for management, including:
        - Total number of rooms booked/available.
        - Earnings for the day, week, or month.
        - Occupancy rate and revenue per available room.
    - Track seasonal trends and guest preferences.
7. **User Authentication and Roles**:
    - Implement role-based access control, where hotel staff (e.g., managers, front desk personnel) have different levels of access.
    - Allow administrators to manage staff accounts and permissions.

8. **Notifications and Alerts**:
    - Send booking confirmations and reminders to customers via email.
    - Notify management of low room availability or maintenance needs.

Challenges to Consider:

- **Concurrency**: Handling multiple users booking rooms simultaneously requires careful management of room availability.
- **Scalability**: As the hotel expands, the system should be able to accommodate more rooms and bookings without significant performance degradation.
- **Data Consistency**: Ensuring that no double-booking occurs due to simultaneous reservations is critical to the system's reliability.

1.2. Aim and Objectives of the Project:

The aim of the **Hotel Booking Management System (RR Hotels)** is to develop a user-friendly, efficient, and scalable software solution using Python that automates the processes involved in managing hotel bookings. This system will handle various aspects such as customer reservations, room availability, payments, and reports, while providing a smooth experience for both hotel staff and guests.

Objective of the project:

1. **Automate Room Booking and Reservation Management:**
    a. To create a seamless process for customers to reserve rooms by checking availability, selecting room types, and making payments.
    b. To maintain real-time room status (vacant, reserved, occupied) and prevent double bookings.
2. **Efficient Customer and Booking Record Management:**
    a. To maintain a centralized database of customer information, booking history, and preferences for future reference.
    b. To ensure that customer data is easily accessible and modifiable by hotel staff when needed.
3. **Facilitate Smooth Check-In/Check-Out Processes:**
    a. To simplify and speed up the check-in/check-out procedure for guests by automating the updating of room status and calculating billing amounts.
    b. To generate accurate invoices and receipts based on room usage and additional services availed by guests.
4. **Payment and Cancellation Handling:**
    a. To securely process customer payments, whether deposits or full amounts, and track financial transactions.
    b. To implement flexible cancellation policies with automated refund calculations based on the hotel's terms.
5. **Room and Pricing Management:**
    a. To allow hotel administrators to easily manage room details such as room type, availability, pricing, and special promotions.
    b. To adjust room rates automatically based on seasonal trends or special events.

6. **Provide Analytical Reports for Decision-Making:**
    a. To generate real-time reports on occupancy rates, revenue, and booking trends to help management make informed business decisions.
    b. To offer daily, weekly, and monthly reports on room usage, customer demographics, and financial performance.
7. **Enhance Customer Experience:**
    a. To send automated booking confirmations and reminders via email or SMS, improving communication with guests.
    b. To offer a streamlined reservation process that enhances customer satisfaction and minimizes errors in bookings.
8. **Role-Based Access Control:**
    a. To implement different access levels for hotel staff, where managers have full control over the system while other employees, such as front desk personnel, have limited access based on their responsibilities.
    b. To ensure secure access to sensitive customer and financial data by using authentication mechanisms.
9. **Scalability and Concurrency Management:**
    a. To build a system capable of handling a growing number of rooms, guests, and bookings efficiently.
    b. To implement concurrency control mechanisms to ensure that multiple users can book rooms simultaneously without conflicts.
10. **Data Security and Privacy:**
    a. To protect customer and financial data by employing encryption and secure protocols in payment processing.
    b. To comply with data privacy regulations and ensure that sensitive information is not exposed to unauthorized users.

Chapter 2

BACKGROUND STUDY

2.1. Software requirements:

1.Customer Management: -

- **Add New Customer**: The system should allow the entry of customer details (name, address, phone, email).
- **Customer Lookup**: Search for existing customers by name, phone number, or email.
- **Customer Booking History**: Maintain customer booking history for future reference.

2.Room Management: -

- **Room Categories**: Define various room types (Single, Double, Suite) and corresponding prices.
- **Room Availability**: Check room availability for specific dates.
- **Room Status**: Track room status (Vacant, Reserved, Occupied).
- **Room Pricing**: Prices should be flexible based on special occasions or promotional offers.

3. Booking Management: -

- **Room Booking**: Handle room bookings by selecting room type, check-in, check-out dates.
- **Booking Confirmation**: Generate booking confirmations with customer details, booking ID, and room information.
- **Check-In/Check-Out**: Mark rooms as occupied when guests check in and vacant when they check out.
- **Booking Modifications**: Ability to modify bookings or cancel reservations as per policies.

4. Payments (Cash/GPay): -

- **Cash Payment**: Record payments made in cash.
- **GPay Payment**: Accept payments via Google Pay and record transaction details.
- **Invoice Generation**: Automatically generate invoices after booking confirmation or during check-out, showing total amount, taxes, and payment details.

5. Reports and Analytics: -

- **Daily Reports**: Generate daily reports on bookings, revenue, and room occupancy.
- **Monthly Reports**: Generate monthly booking and revenue reports.
- **Occupancy Rate**: Track the percentage of occupied rooms over a period of time.
- **Customer Data Analysis**: Analyze customer booking patterns and preferences.

6.Usability: -

- **User Interface**: A simple console-based or basic GUI application using tkinter for interacting with the system.
- **Language**: The entire system will be developed using Python.

7.Performance: -

- The system should handle concurrent bookings without performance bottlenecks.
- The database queries should be optimized for faster results, especially for checking room availability and generating reports.

8.Security: -

- Ensure customer and payment data are stored securely in the database.
- Role-based access control to limit functionality based on user roles (e.g., administrator, front desk).

9.Scalability: -

- As the number of rooms, customers, and bookings increase, the system should scale appropriately without performance issues.

10.Availability: -

- The system should be available to manage bookings and generate reports 24/7.

11.Programming Language: -

- **Python 3.x**: All development will be done using Python.

12.Backend and Logic: -

- **Python Functions and Classes**: Use Python classes to define hotel entities like Customer, Room, Booking, and Payment.
- **File Handling**: Use JSON or CSV files to store data persistently for a small-scale system (if not using a database).
- **Concurrency**: Use the threading module for handling simultaneous booking requests if required.

13.Database: -

- **SQLite (Python Standard Library sqlite3)**:
  - The database will store information related to customers, rooms, bookings, and payments.
  - SQLite will be used since it is a lightweight, serverless, and file-based database that integrates well with Python.

- **Customers**: customer_id, name, address, phone, email.
- **Rooms**: room_id, room_type, price, status (vacant, reserved, occupied).
- **Bookings**: booking_id, customer_id, room_id, check_in, check_out, total_amount, payment_status.
- **Payments**: payment_id, booking_id, payment_method (cash, GPay), amount_paid, transaction_id (for GPay).

14.Frontend (User Interface): -

- **Console Interface**:
    - Use Python's input() and print() statements for basic console interaction, allowing staff to enter customer information, make bookings, and manage rooms.
- **Graphical User Interface (Optional)**:
    - **tkinter**: Python's built-in GUI library for creating a simple desktop application.
        - Basic forms for entering customer details, booking rooms, and generating reports.
        - Buttons for booking actions, check-in, check-out, and payment processing.

15.Reporting and Analytics: -

- **Pandas**:
    - For handling and analysing large datasets related to bookings, rooms, and customers.
    - Use pandas DataFrames to generate reports and summaries, such as occupancy rate, total earnings, and customer booking preferences.
- **Matplotlib/Seaborn** (Optional):
    - For graphical representation of data such as charts showing occupancy rates or revenue trends over time.

16.Payment Integration: -

- **Cash Payments**:
    - Simple manual entry of the payment amount during the booking or check-out process.
- **GPay Integration**:
    - Use the **Google Pay UPI SDK** for handling GPay transactions.
    - Record transaction details such as the UPI transaction ID and store it in the database along with payment status.

17.SQLite (Python Standard Library): -

- No external installation needed. Use the built-in sqlite3 module for database management.

18.Pandas: -

- Install using pip install pandas for handling data analytics and generating reports.

19.Matplotlib/Seaborn (Optional): -

- Install using pip install matplotlib seaborn for generating charts and graphical reports.

20.Tkinter: -

- Tkinter is included with Python, so no external installation is needed if you opt for a GUI.

21.Customer Management Module: -

- Manages adding new customers, searching for existing ones, and viewing booking history.

22.Room Management Module: -

- Handles room availability, status updates (vacant/occupied), and room pricing.

23.Booking Management Module: -

- Handles the process of booking rooms, including selecting check-in and check-out dates, and managing multiple room types.

24.Payment Module: -

- Handles payment processing (both cash and GPay), stores payment data, and generates invoices.

25.Report Generation Module: -

- Uses pandas to create daily, monthly, or custom reports on revenue, bookings, and occupancy rates.

26.Frontend Interface (CLI/GUI): -

- The interface for hotel staff to interact with the system, either through a console or basic GUI using tkinter.

2.2. Key Features of the Project:

1.**Customer Management: -**

- **Add New Customer**: Allows staff to input customer details (name, phone number, email, and address) into the system.
- **Customer Lookup**: Easily search for customers using their name, phone number, or email to retrieve previous bookings.
- **Customer History**: Maintain a booking history for each customer, enabling personalized services or loyalty programs.

2.**Room Management: -**

- **Room Types**: Categorize rooms into various types (e.g., Single, Double, Suite) and assign specific prices for each type.
- **Room Availability**: Track room availability in real-time, displaying which rooms are vacant, reserved, or occupied.
- **Room Status Update**: Automatically update room status (vacant/occupied) during check-in and check-out.
- **Room Pricing Flexibility**: Allow dynamic pricing based on seasons, weekends, or special promotions.

3.**Booking Management: -**

- **Room Reservation**: Let customers book rooms by specifying check-in and check-out dates, room types, and the number of rooms required.
- **Real-Time Availability Check**: Check for room availability during the booking process to avoid double-booking.
- **Booking Modification**: Allow staff to modify or cancel existing bookings as needed.
- **Booking Confirmation**: Generate a confirmation receipt upon successful booking, detailing customer information, booking ID, room details, and dates.

4.**Check-In and Check-Out: -**

- **Check-In**: Update the room’s status to "Occupied" when a customer checks in.
- **Check-Out**: Update the room’s status to "Vacant" when the customer checks out, and calculate the total cost of their stay.

5.**Payment Management: -**

- **Payment Methods**: Support payments via:
    - **Cash**: Manually record cash payments.
    - **GPay (Google Pay)**: Integrate GPay for easy digital payments. Record GPay transaction details like the transaction ID.
- **Invoice Generation**: Automatically generate an invoice upon successful payment, displaying the total amount, room charges, taxes, and payment method.

6.Reports and Analytics: -

- **Daily and Monthly Reports**: Generate reports that provide insights on daily and monthly bookings, revenue, and occupancy rates.
- **Customer Data Analytics**: Track booking patterns, preferred room types, and customer preferences for personalized services.
- **Occupancy Rate Calculation**: Measure how many rooms were occupied over a specified period, aiding in operational and financial planning.
- **Revenue Reports**: Summarize total earnings from bookings and services over specific timeframes.

7.**User Interface: -**

- **Command-Line Interface (CLI)**: Allow hotel staff to interact with the system through simple commands and inputs for managing bookings, payments, and reports.
- **Graphical User Interface (GUI)** (Optional): Use tkinter for a basic desktop app, allowing users to perform tasks like booking, check-ins, check-outs, and reporting via a simple graphical interface.

8.**Data Storage and Database Management: -**

- **SQLite Database**: Store customer, room, booking, and payment data using the built-in sqlite3 module.
- **Efficient Queries**: Optimize database queries for retrieving room availability, customer information, and booking history.

9.**Cancellation and Refunds: -**

- **Booking Cancellations**: Allow customers to cancel bookings, either for free or with a penalty based on the hotel's cancellation policy.
- **Refund Handling**: Process refunds automatically and update the payment record if eligible.

10.**Security and Access Control: -**

- **User Roles**: Assign different levels of access (e.g., admin, front desk) for different staff members to restrict permissions and sensitive data.
- **Data Security**: Ensure sensitive information, such as customer data and payment details, are securely stored in the database.

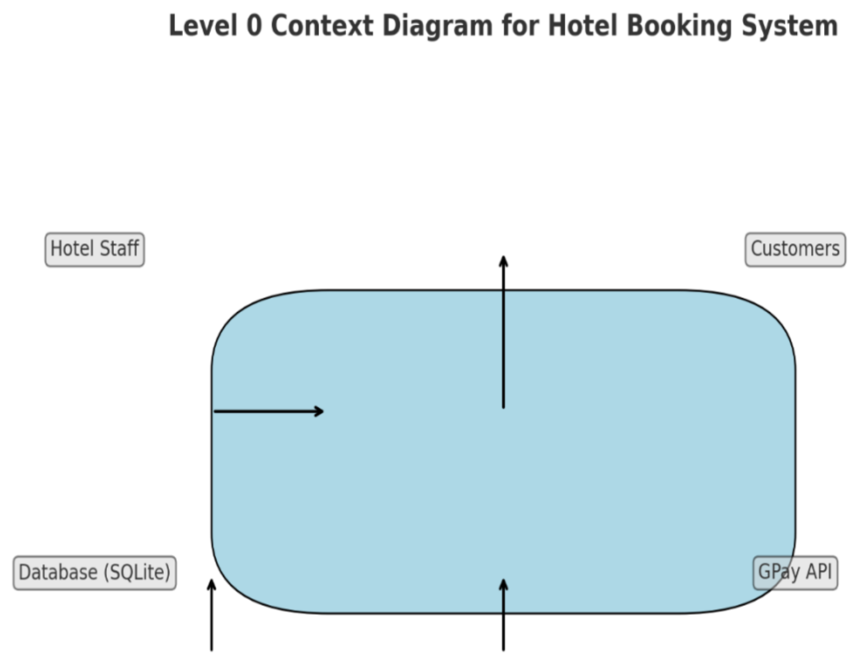
11.**Scalability: -**

- **Multiple Bookings**: Handle multiple, simultaneous bookings without performance bottlenecks.
- **Extendable**: Easily scale the system to add more features, room types, or additional payment methods.

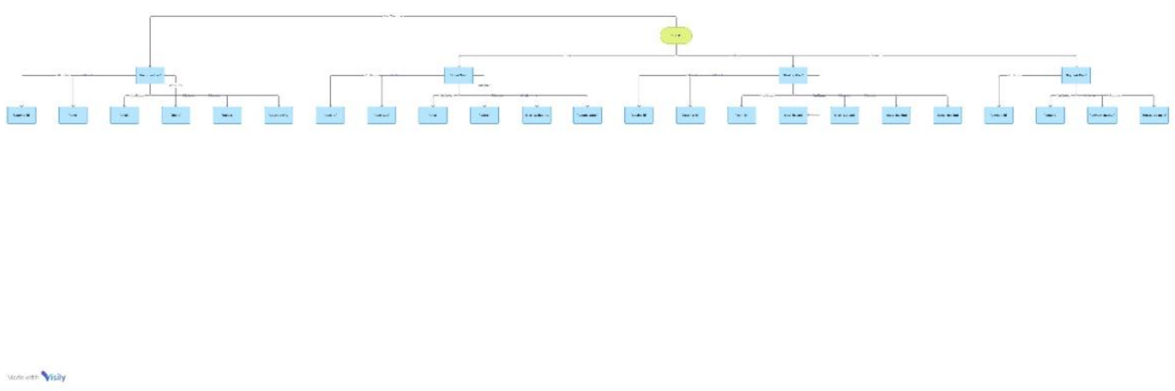
CHAPTER 3

SYSTEM DESIGN

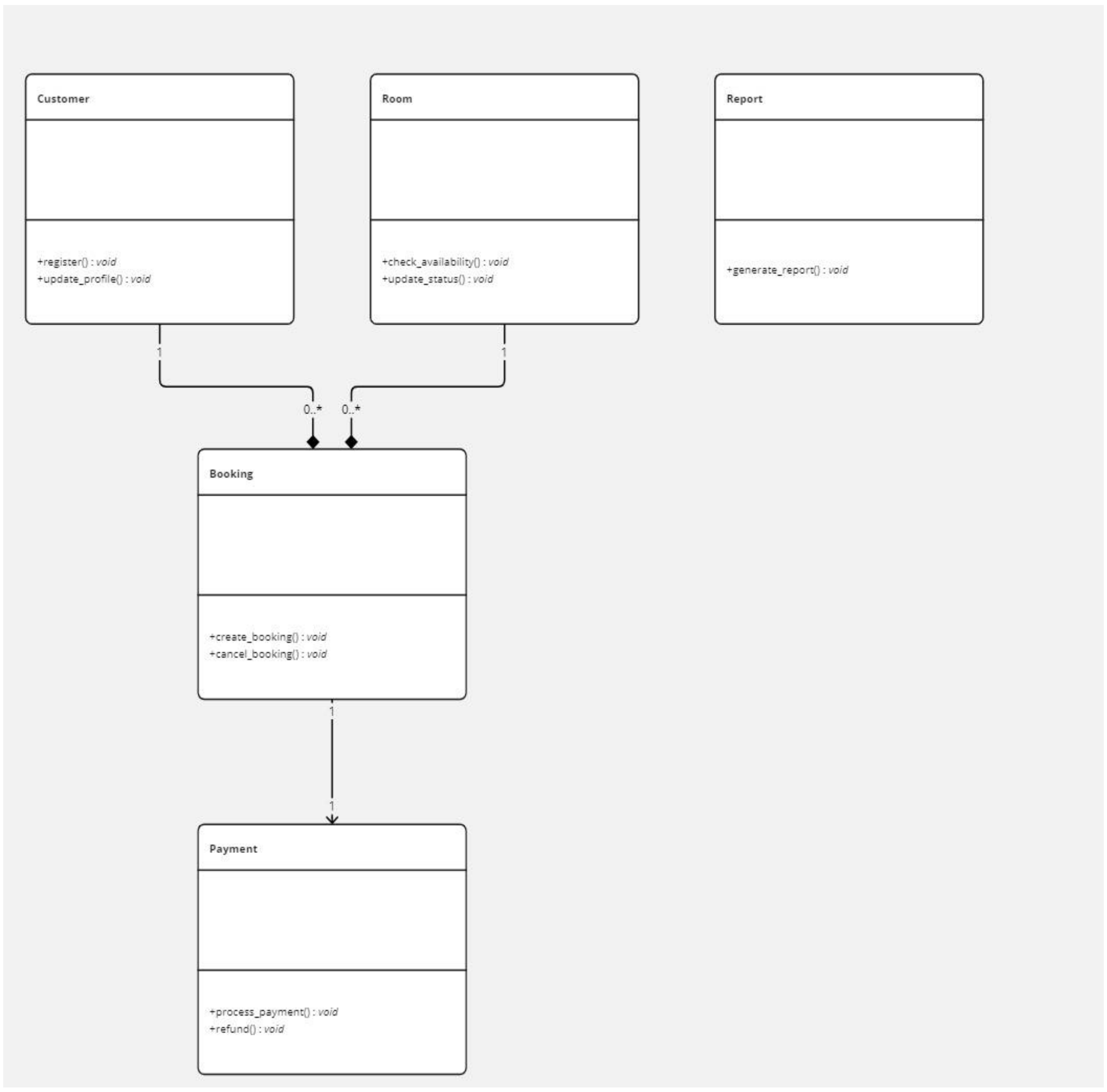
3.1. Data Flow Diagram:



Level 1: High-Level System Flow:



3.2 Entity relationship diagram



CHAPTER 4

MODULE AND DESCRIPTION

1. **Customer Management Module: -**

- **Description**: Manages customer-related functionalities, such as registration, profile updates, and customer details retrieval.
- **Key Functions**:
  - register_customer(): Registers a new customer in the system.
  - update_customer_profile(): Updates customer information.
  - get_customer_details(customer_id): Retrieves details of a customer by ID.

2. **Room Management Module: -**

- **Description**: Handles operations related to room inventory, including availability checks and updates to room status.
- **Key Functions**:
  - add_room(): Adds a new room to the system.
  - update_room_status(room_id, status): Updates the status (available, booked, etc.) of a room.
  - check_room_availability(room_id, check_in, check_out): Checks if a room is available for the specified dates.

3. **Booking Management Module: -**

- **Description**: Manages the booking process, including creating and cancelling bookings.
- **Key Functions**:
  - create_booking(customer_id, room_id, check_in_date, check_out_date): Creates a new booking.
  - cancel_booking(booking_id): Cancels an existing booking.
  - get_booking_details(booking_id): Retrieves details of a specific booking.

4. **Payment Processing Module: -**

- **Description**: Handles payment processing, including capturing payment details and processing refunds.
- **Key Functions**:
    - process_payment(booking_id, amount, payment_method): Processes the payment for a booking.
    - refund_payment(payment_id): Processes a refund for a payment.
    - get_payment_details(payment_id): Retrieves details of a specific payment.

5. **Reporting Module: -**

- **Description**: Generates reports related to bookings, payments, and room occupancy.
- **Key Functions**:
    - generate_booking_report(start_date, end_date): Generates a report of bookings between specified dates.
    - generate_payment_report(start_date, end_date): Generates a report of payments made during a specified period.
    - generate_room_occupancy_report(): Provides a report on room occupancy rates.

6. **Database Module: -**

- **Description**: Handles all database interactions, including connection management and executing queries.
- **Key Functions**:
    - connect_db(): Establishes a connection to the database.
    - execute_query(query): Executes a given SQL query.
    - fetch_results(query): Fetches results from a query execution.

7. **User Interface Module: -**

- **Description**: Manages user interactions, including a command-line interface or a graphical user interface.
- **Key Functions**:
  - display menu(): Displays the main menu for user interaction.
  - get_user_input(): Captures user input for various actions.
  - show_message(message): Displays messages to the user.

8. **Utility Module: -**

- **Description**: Contains helper functions and utilities used across different modules.
- **Key Functions**:
  - validate_email(email): Validates the format of an email address.
  - calculate_total_price(room_rate, nights): Calculates the total price based on the room rate and number of nights.

CONCLUSION: -

The Hotel Booking Management System (**RR Hotels**) developed in Python represents a comprehensive solution for managing hotel operations efficiently. By leveraging Python's robust features and a modular design approach, the system offers a user-friendly interface and powerful backend functionality. Here are some key points to summarize the effectiveness and significance of the project:

1. **Modular Architecture**: The system's modular structure facilitates easier maintenance and scalability. Each module—Customer Management, Room Management, Booking Management, Payment Processing, Reporting, Database Management, User Interface, and Utility—encapsulates specific functionalities, making it easy to update or expand individual components as needed.

2. **User-Friendly Interface**: By providing an intuitive user interface, the system enhances user experience for both hotel staff and customers. This ease of use encourages efficient interactions, from making reservations to managing bookings.
3. **Automation of Operations**: The system automates critical processes such as room availability checks, booking confirmations, payment processing, and report generation. This automation minimizes human error and optimizes operational efficiency, allowing staff to focus on enhancing customer service.
4. **Data Management and Reporting**: The integrated reporting module enables hotel management to analyse booking trends, occupancy rates, and revenue streams effectively. This data-driven approach empowers decision-makers to devise strategies for maximizing profitability and improving service offerings.
5. **Payment Flexibility**: By supporting multiple payment methods, including cash and digital payments, the system caters to a diverse clientele. This flexibility enhances customer satisfaction and encourages seamless transactions.
6. **Scalability**: Designed to handle varying scales of operation, the system can adapt to small boutique hotels or large chain establishments. Its scalability ensures that it remains effective as business needs evolve.
7. **Future Enhancements**: The foundational structure of the system allows for future enhancements, such as integrating third-party services (e.g., payment gateways, loyalty programs) or expanding functionalities to include features like customer reviews and personalized marketing.
8. **Community and Support**: Python's extensive community and library support mean that developers can access a wealth of resources, facilitating ongoing development and troubleshooting.

In summary, the Hotel Booking Management System **(RR Hotels**) built using Python is a powerful tool for enhancing operational efficiency in the hospitality industry. By focusing on user experience, data management, and automation, the system not only streamlines booking processes but also contributes to improved customer satisfaction and business growth. The project serves as a strong foundation for future enhancements and adaptations, ensuring its relevance in a dynamic market.

APPENDICES

A) References: -

This appendix provides references and resources specific to implementing a Hotel Booking System entirely using Python. The following books, websites, and resources focus on Python's features and libraries for building a full-fledged hotel management system, covering everything from database handling to user interfaces.

Appendix A: Reference Books: -

1. **"Python Programming: An Introduction to Computer Science" by John Zelle**
    - **Description**: A foundational book that introduces Python programming concepts and object-oriented principles, which are crucial for building a modular hotel booking system.
2. **"Python Cookbook" by David Beazley and Brian K. Jones**
    - **Description**: This book provides practical solutions for common Python programming challenges. It covers essential techniques, including file handling, database connections, and custom exceptions, which are relevant to the hotel booking system.
3. **"Automate the Boring Stuff with Python" by Al Sweigart**
    - **Description**: This book is ideal for automating repetitive tasks like generating reports, sending emails, and managing data in the hotel booking system. It focuses on practical Python programming for day-to-day use cases.
4. **"Mastering Object-Oriented Python" by Steven Lott**
    - **Description**: This book offers an in-depth exploration of object-oriented programming (OOP) in Python. It helps in designing a hotel booking system where entities like Customer, Room, and Booking are implemented as objects.

5. **"Learn Python 3 the Hard Way" by Zed Shaw**
    - **Description**: A hands-on guide that walks through Python basics with exercises, covering fundamental programming concepts needed for any project, including hotel management systems.

Appendix B: Websites and Online Resources: -

1. **Python Official Documentation**
    - **URL**: https://docs.python.org/3/
    - **Description**: This is the official Python documentation. It is the most authoritative reference for all standard Python libraries used in the project, such as sqlite3 for databases and tkinter for user interfaces.
2. **Python SQLite Tutorial**
    - **URL**: https://docs.python.org/3/library/sqlite3.html
    - **Description**: This Python documentation explains how to interact with SQLite databases directly within Python, which is useful for managing customer, room, and booking data.
3. **Real Python - Python SQLite Databases**
    - **URL**: https://realpython.com/python-sql-libraries/
    - **Description**: Real Python offers detailed tutorials and practical examples of how to use Python's SQLite library. It is beneficial for understanding how to design, query, and update databases in the hotel booking system.
4. **PythonAnywhere**
    - **URL**: https://www.pythonanywhere.com/
    - **Description**: This cloud platform allows hosting Python applications. It's useful for deploying and testing your hotel booking system in a live environment without needing extensive server infrastructure.
5. **Geeks for Geeks - Python OOP**
    - **URL: https://www.geeksforgeeks.org/python-oops-concepts/Description:** This resource provides a detailed introduction to object-oriented programming (OOP) in Python, which is essential for designing classes like Customer, Room, and Booking in the hotel system.

6. **Tkinter Documentation (Python's GUI Library)**
    - **URL**: https://docs.python.org/3/library/tkinter.html
    - **Description**: The Tkinter library is used to create graphical user interfaces (GUIs) in Python. This is a comprehensive resource for building a simple user interface for the hotel booking system.
7. **Python 3 Exception Handling**
    - **URL**: https://docs.python.org/3/tutorial/errors.html
    - **Description**: This guide explains how to manage errors and exceptions in Python, which is critical for ensuring the robustness of the hotel booking system during user input and database operations.

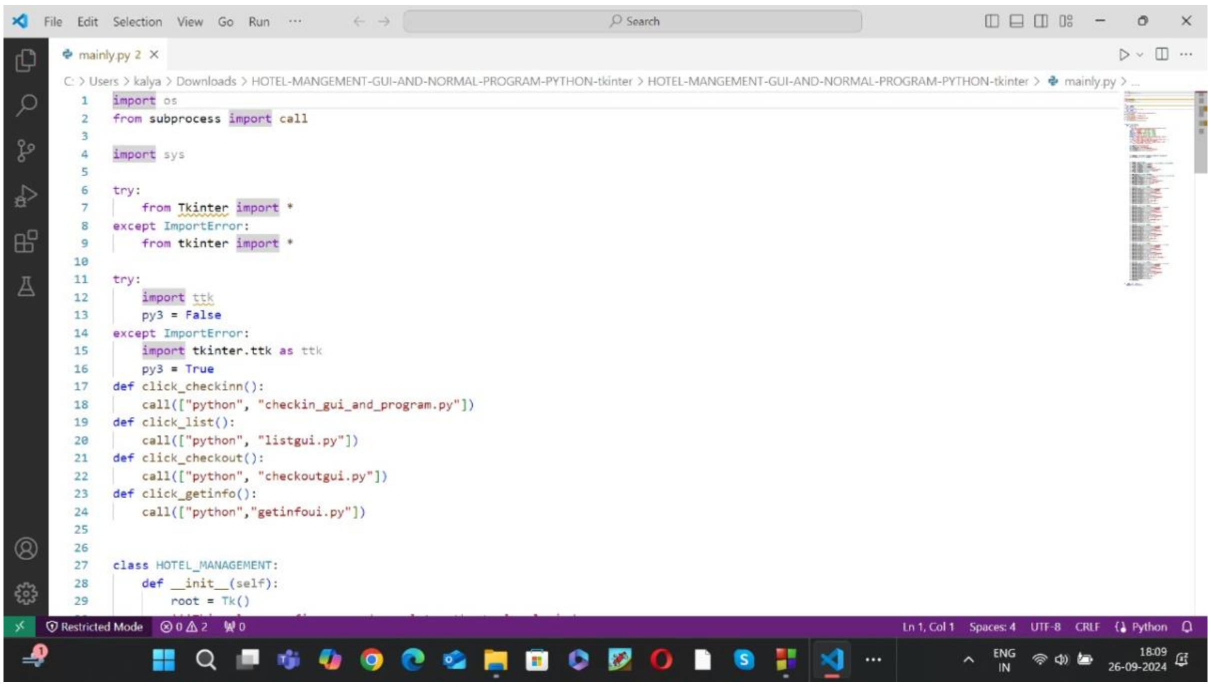
Appendix C: Python Libraries and Modules Used: -

1. **SQLite3 (Built-in Python Database)**
    - **Library Name**: sqlite3
    - **Description**: This is a built-in Python module that provides an SQL database engine. It's perfect for managing customer, room, and booking information in the hotel system.
    - **Reference**: SQLite3 Documentation
2. **Tkinter (Python GUI Library)**
    - **Library Name**: tkinter
    - **Description**: Tkinter is a standard Python library used for creating basic GUIs. It can be used to build the frontend of the hotel booking system, allowing users to interact with the system through graphical elements like buttons and forms.
    - **Reference**: Tkinter Documentation
3. **Datetime (Date and Time Handling)**
    - **Library Name**: datetime
    - **Description**: The datetime module allows you to manage dates and times efficiently. It is particularly useful in handling check-in and check-out dates in the hotel booking system.
    - **Reference**: Datetime Documentation

4. **OS (Interacting with the Operating System)**
    - **Library Name**: OS
    - **Description**: This module is used for file handling operations such as reading and writing data. It can be used for exporting reports or managing system files in the hotel booking system.
    - **Reference**: OS Module Documentation

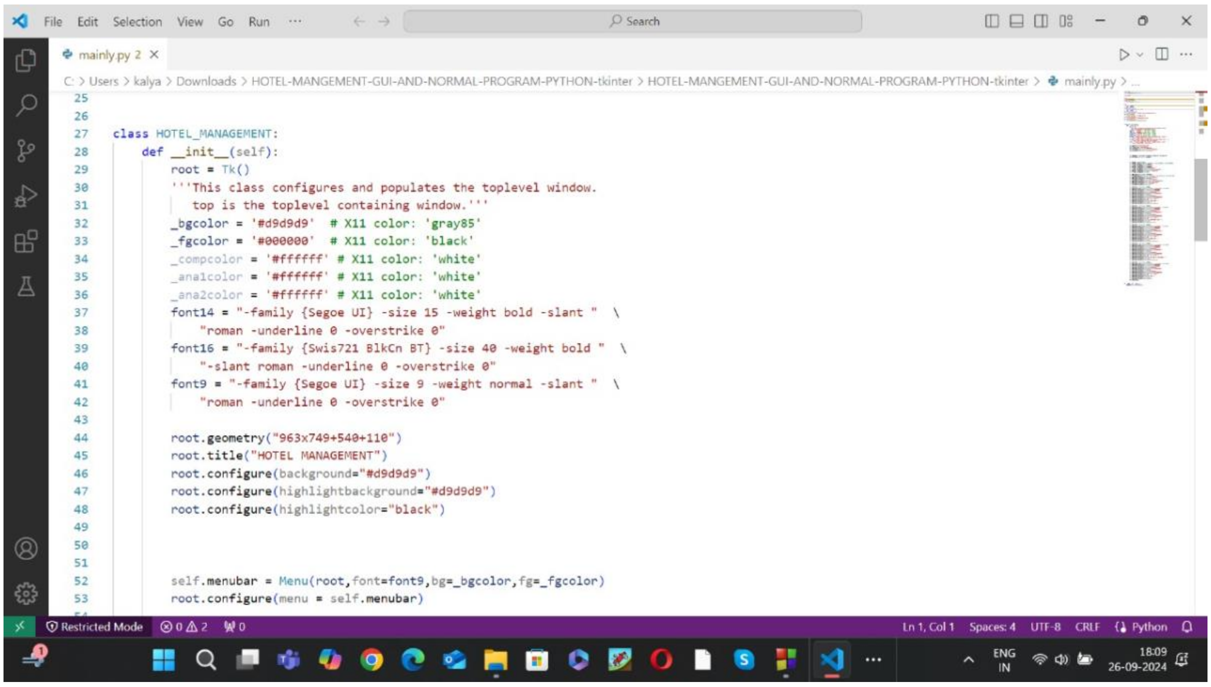
5. **Logging (Python Logging Module)**
    - **Library Name**: logging
    - **Description**: The logging module provides a way to track events and errors during program execution, making it easier to debug and maintain the system.
    - **Reference**: Logging Documentation

B) Screenshots: -



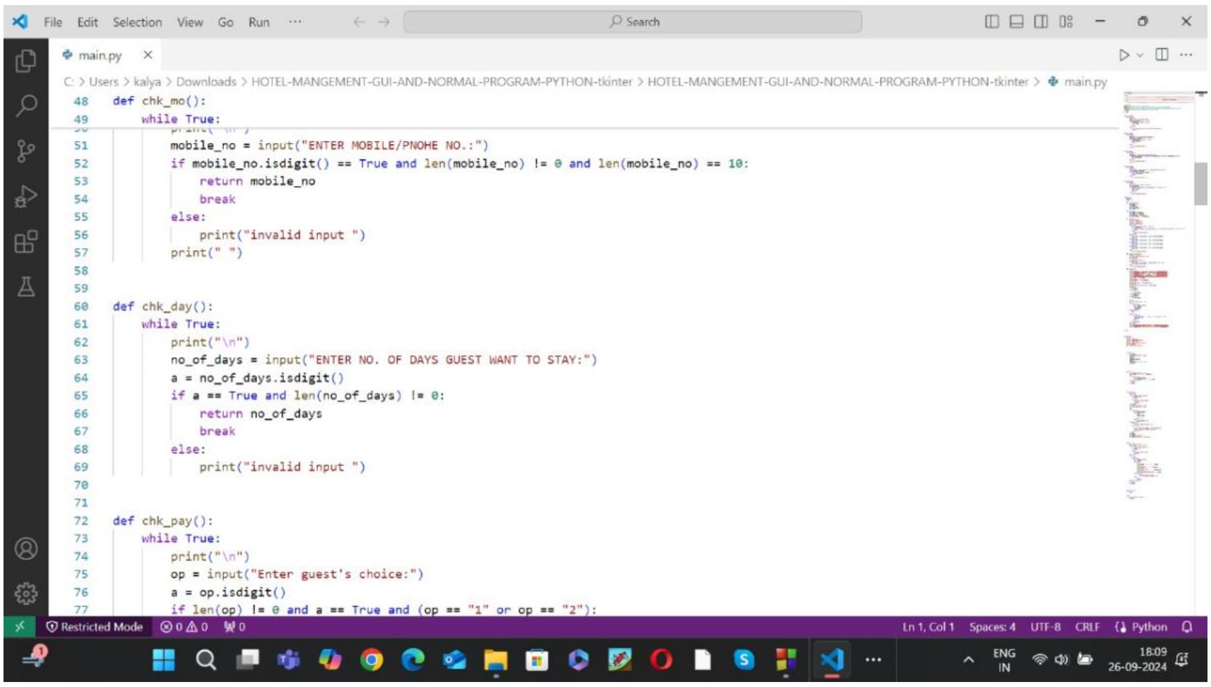
C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > mainly.py

```python
import os
from subprocess import call

import sys

try:
    from Tkinter import *
except ImportError:
    from tkinter import *

try:
    import ttk
    py3 = False
except ImportError:
    import tkinter.ttk as ttk
    py3 = True
def click_checkinn():
    call(["python", "checkin_gui_and_program.py"])
def click_list():
    call(["python", "listgui.py"])
def click_checkout():
    call(["python", "checkoutgui.py"])
def click_getinfo():
    call(["python","getinfoui.py"])


class HOTEL_MANAGEMENT:
    def __init__(self):
        root = Tk()
```



C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > mainly.py

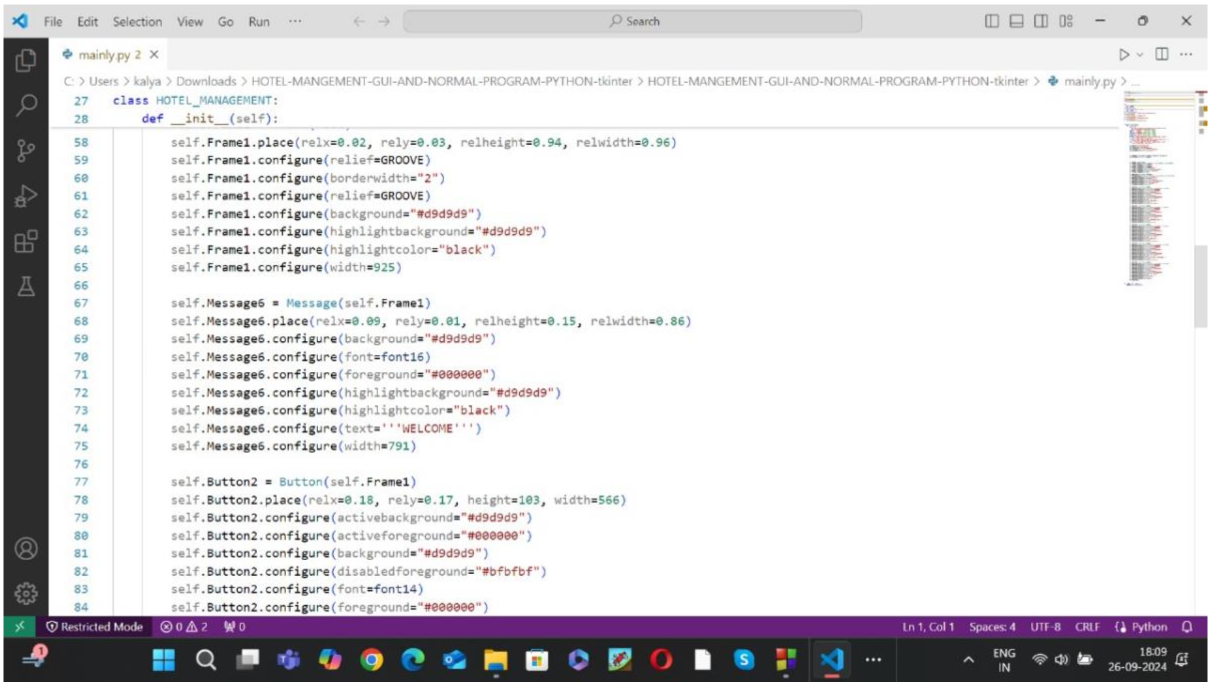
```python


class HOTEL_MANAGEMENT:
    def __init__(self):
        root = Tk()
        '''This class configures and populates the toplevel window.
           top is the toplevel containing window.'''
        _bgcolor = '#d9d9d9'  # X11 color: 'gray85'
        _fgcolor = '#000000'  # X11 color: 'black'
        _compcolor = '#ffffff' # X11 color: 'white'
        _ana1color = '#ffffff' # X11 color: 'white'
        _ana2color = '#ffffff' # X11 color: 'white'
        font14 = "-family {Segoe UI} -size 15 -weight bold -slant "  \
            "roman -underline 0 -overstrike 0"
        font16 = "-family {Swis721 BlkCn BT} -size 40 -weight bold "  \
            "-slant roman -underline 0 -overstrike 0"
        font9 = "-family {Segoe UI} -size 9 -weight normal -slant "  \
            "roman -underline 0 -overstrike 0"

        root.geometry("963x749+540+110")
        root.title("HOTEL MANAGEMENT")
        root.configure(background="#d9d9d9")
        root.configure(highlightbackground="#d9d9d9")
        root.configure(highlightcolor="black")



        self.menubar = Menu(root,font=font9,bg=_bgcolor,fg=_fgcolor)
        root.configure(menu = self.menubar)
```


main.py

C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > main.py

```python
def chk_mo():
    while True:
        mobile_no = input("ENTER MOBILE/PNOHE NO.:")
        if mobile_no.isdigit() == True and len(mobile_no) != 0 and len(mobile_no) == 10:
            return mobile_no
            break
        else:
            print("invalid input ")
        print(" ")


def chk_day():
    while True:
        print("\n")
        no_of_days = input("ENTER NO. OF DAYS GUEST WANT TO STAY:")
        a = no_of_days.isdigit()
        if a == True and len(no_of_days) != 0:
            return no_of_days
            break
        else:
            print("invalid input ")


def chk_pay():
    while True:
        print("\n")
        op = input("Enter guest's choice:")
        a = op.isdigit()
        if len(op) != 0 and a == True and (op == "1" or op == "2"):
```

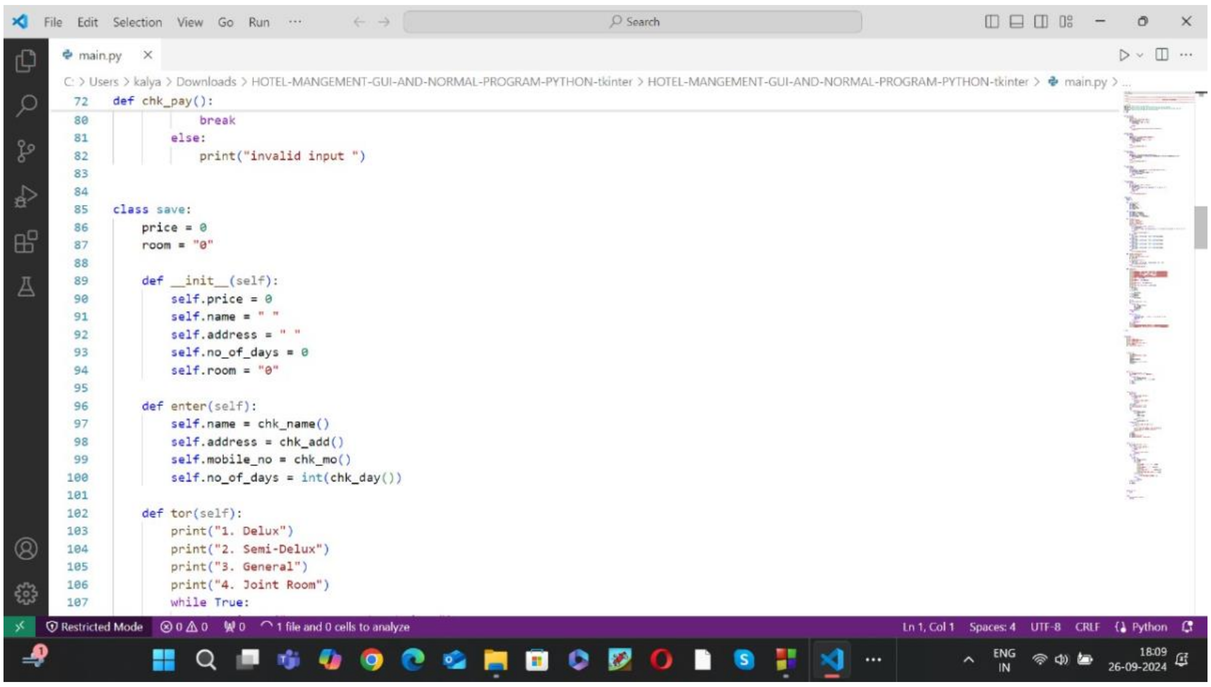
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python


mainly.py 2

C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > mainly.py > ...

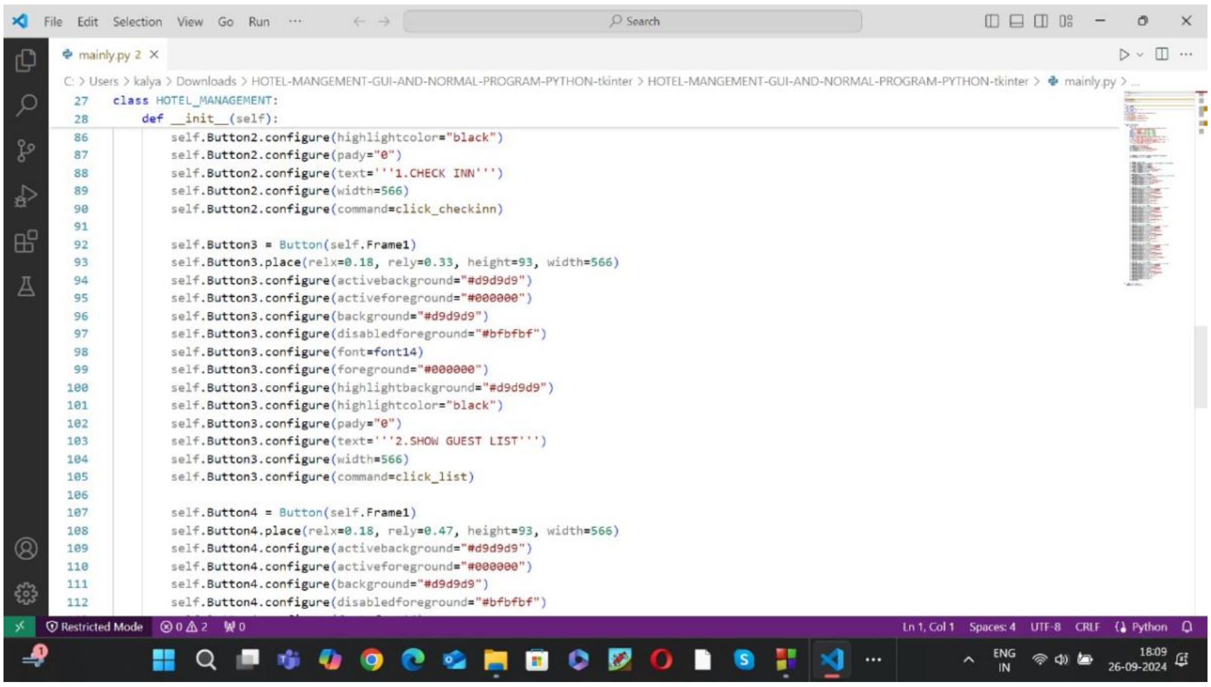
```python
class HOTEL_MANAGEMENT:
    def __init__(self):
        self.Frame1.place(relx=0.02, rely=0.03, relheight=0.94, relwidth=0.96)
        self.Frame1.configure(relief=GROOVE)
        self.Frame1.configure(borderwidth="2")
        self.Frame1.configure(relief=GROOVE)
        self.Frame1.configure(background="#d9d9d9")
        self.Frame1.configure(highlightbackground="#d9d9d9")
        self.Frame1.configure(highlightcolor="black")
        self.Frame1.configure(width=925)

        self.Message6 = Message(self.Frame1)
        self.Message6.place(relx=0.09, rely=0.01, relheight=0.15, relwidth=0.86)
        self.Message6.configure(background="#d9d9d9")
        self.Message6.configure(font=font16)
        self.Message6.configure(foreground="#000000")
        self.Message6.configure(highlightbackground="#d9d9d9")
        self.Message6.configure(highlightcolor="black")
        self.Message6.configure(text='''WELCOME''')
        self.Message6.configure(width=791)

        self.Button2 = Button(self.Frame1)
        self.Button2.place(relx=0.18, rely=0.17, height=103, width=566)
        self.Button2.configure(activebackground="#d9d9d9")
        self.Button2.configure(activeforeground="#000000")
        self.Button2.configure(background="#d9d9d9")
        self.Button2.configure(disabledforeground="#bfbfbf")
        self.Button2.configure(font=font14)
        self.Button2.configure(foreground="#000000")
```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python



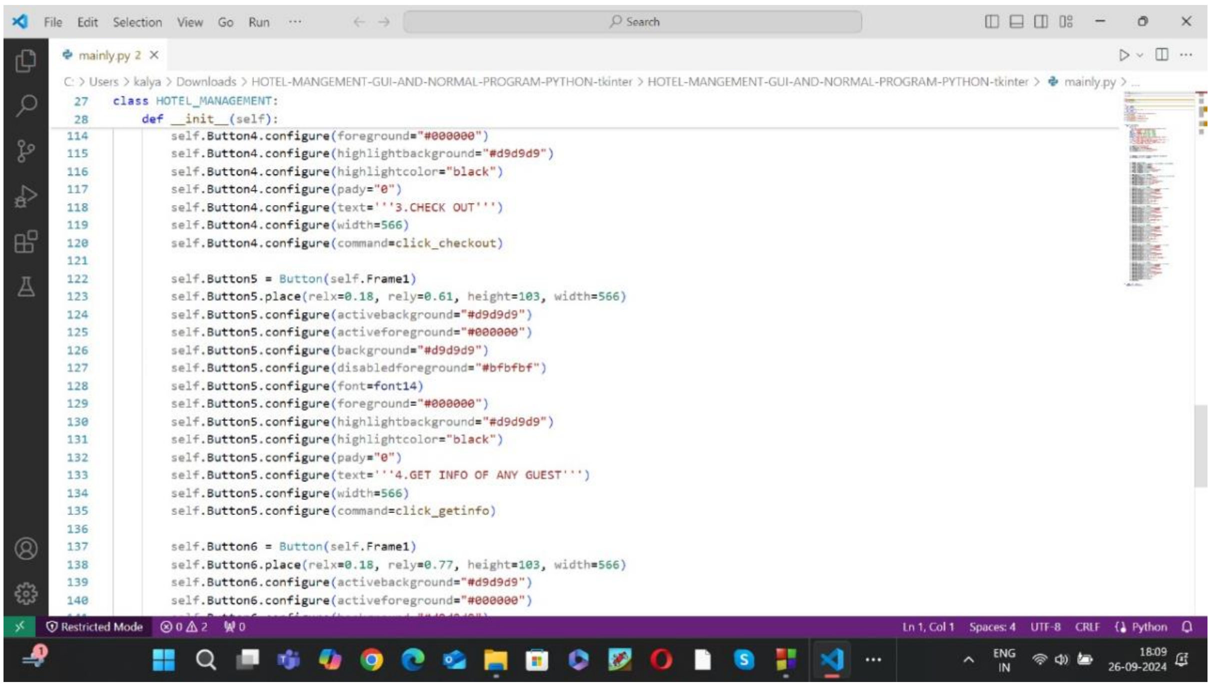
C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > main.py

```python
def chk_pay():
            break
        else:
            print("invalid input ")


class save:
    price = 0
    room = "0"

    def __init__(self):
        self.price = 0
        self.name = " "
        self.address = " "
        self.no_of_days = 0
        self.room = "0"

    def enter(self):
        self.name = chk_name()
        self.address = chk_add()
        self.mobile_no = chk_mo()
        self.no_of_days = int(chk_day())

    def tor(self):
        print("1. Delux")
        print("2. Semi-Delux")
        print("3. General")
        print("4. Joint Room")
        while True:
```



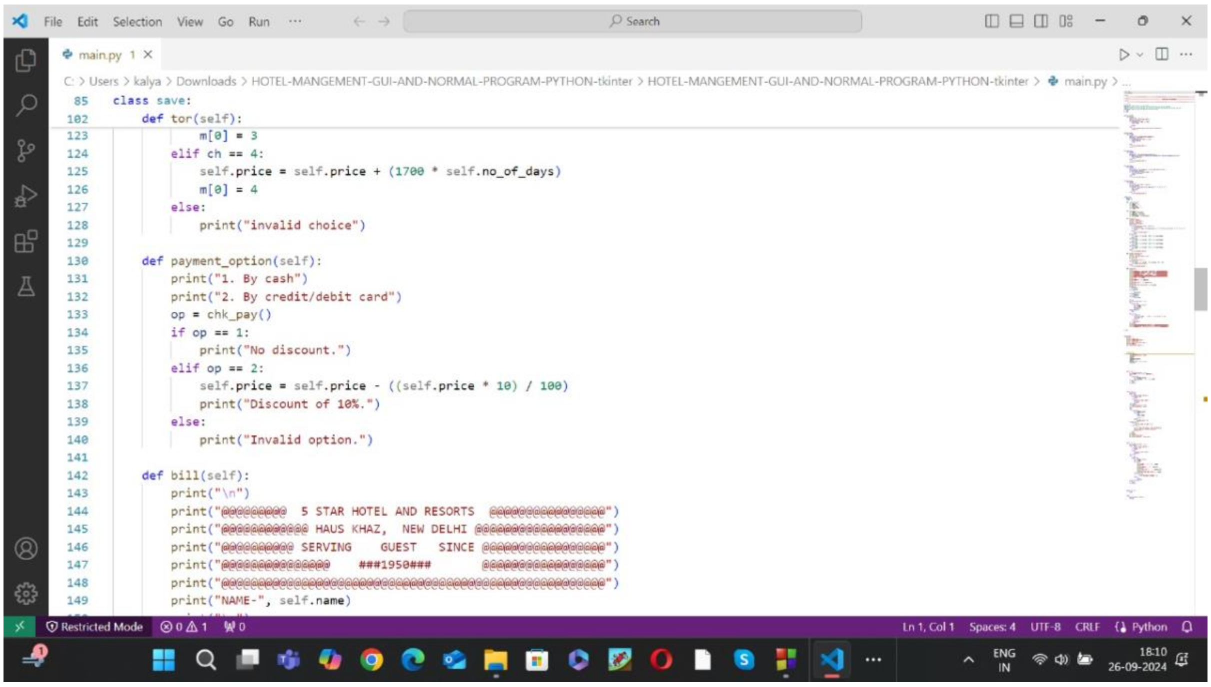
C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > mainly.py

```python
class HOTEL_MANAGEMENT:
    def __init__(self):
        self.Button2.configure(highlightcolor="black")
        self.Button2.configure(pady="0")
        self.Button2.configure(text='''1.CHECK INN''')
        self.Button2.configure(width=566)
        self.Button2.configure(command=click_checkinn)

        self.Button3 = Button(self.Frame1)
        self.Button3.place(relx=0.18, rely=0.33, height=93, width=566)
        self.Button3.configure(activebackground="#d9d9d9")
        self.Button3.configure(activeforeground="#000000")
        self.Button3.configure(background="#d9d9d9")
        self.Button3.configure(disabledforeground="#bfbfbf")
        self.Button3.configure(font=font14)
        self.Button3.configure(foreground="#000000")
        self.Button3.configure(highlightbackground="#d9d9d9")
        self.Button3.configure(highlightcolor="black")
        self.Button3.configure(pady="0")
        self.Button3.configure(text='''2.SHOW GUEST LIST''')
        self.Button3.configure(width=566)
        self.Button3.configure(command=click_list)

        self.Button4 = Button(self.Frame1)
        self.Button4.place(relx=0.18, rely=0.47, height=93, width=566)
        self.Button4.configure(activebackground="#d9d9d9")
        self.Button4.configure(activeforeground="#000000")
        self.Button4.configure(background="#d9d9d9")
        self.Button4.configure(disabledforeground="#bfbfbf")
```


File Edit Selection View Go Run
Search
mainly.py 2
Restricted Mode
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python
ENG IN
18:09 26-09-2024

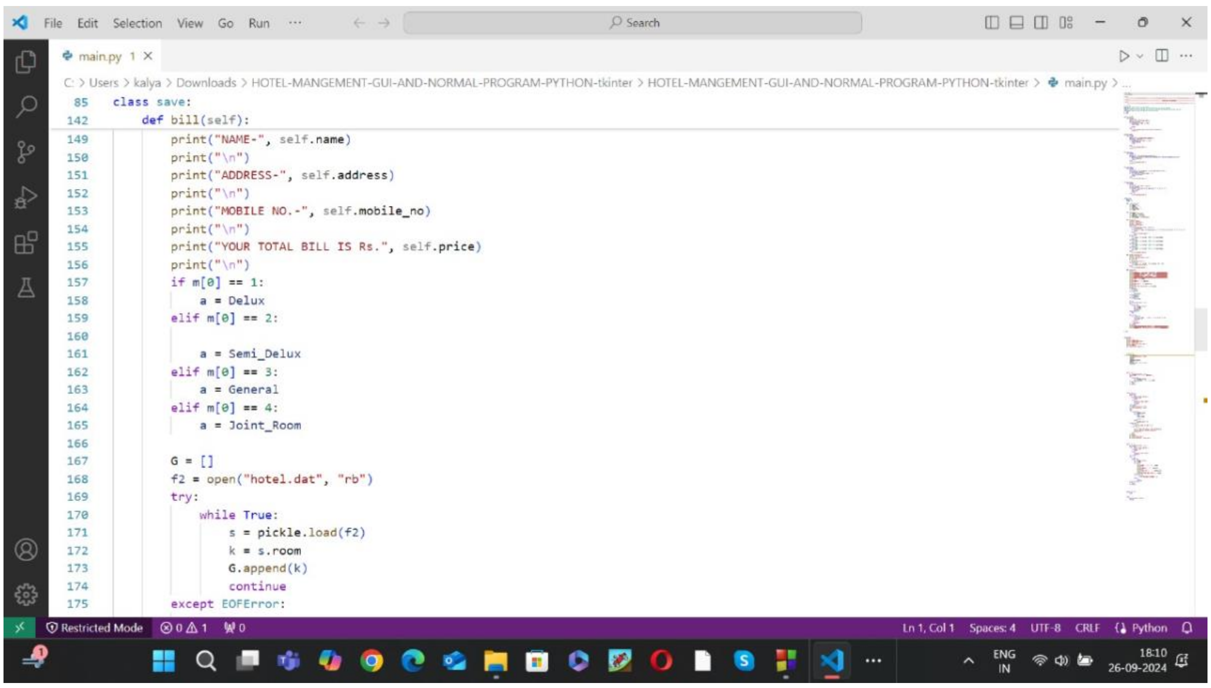
C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > mainly.py > ...

```python
class HOTEL_MANAGEMENT:
    def __init__(self):
        self.Button4.configure(foreground="#000000")
        self.Button4.configure(highlightbackground="#d9d9d9")
        self.Button4.configure(highlightcolor="black")
        self.Button4.configure(pady="0")
        self.Button4.configure(text='''3.CHECK OUT''')
        self.Button4.configure(width=566)
        self.Button4.configure(command=click_checkout)

        self.Button5 = Button(self.Frame1)
        self.Button5.place(relx=0.18, rely=0.61, height=103, width=566)
        self.Button5.configure(activebackground="#d9d9d9")
        self.Button5.configure(activeforeground="#000000")
        self.Button5.configure(background="#d9d9d9")
        self.Button5.configure(disabledforeground="#bfbfbf")
        self.Button5.configure(font=font14)
        self.Button5.configure(foreground="#000000")
        self.Button5.configure(highlightbackground="#d9d9d9")
        self.Button5.configure(highlightcolor="black")
        self.Button5.configure(pady="0")
        self.Button5.configure(text='''4.GET INFO OF ANY GUEST''')
        self.Button5.configure(width=566)
        self.Button5.configure(command=click_getinfo)

        self.Button6 = Button(self.Frame1)
        self.Button6.place(relx=0.18, rely=0.77, height=103, width=566)
        self.Button6.configure(activebackground="#d9d9d9")
        self.Button6.configure(activeforeground="#000000")
```

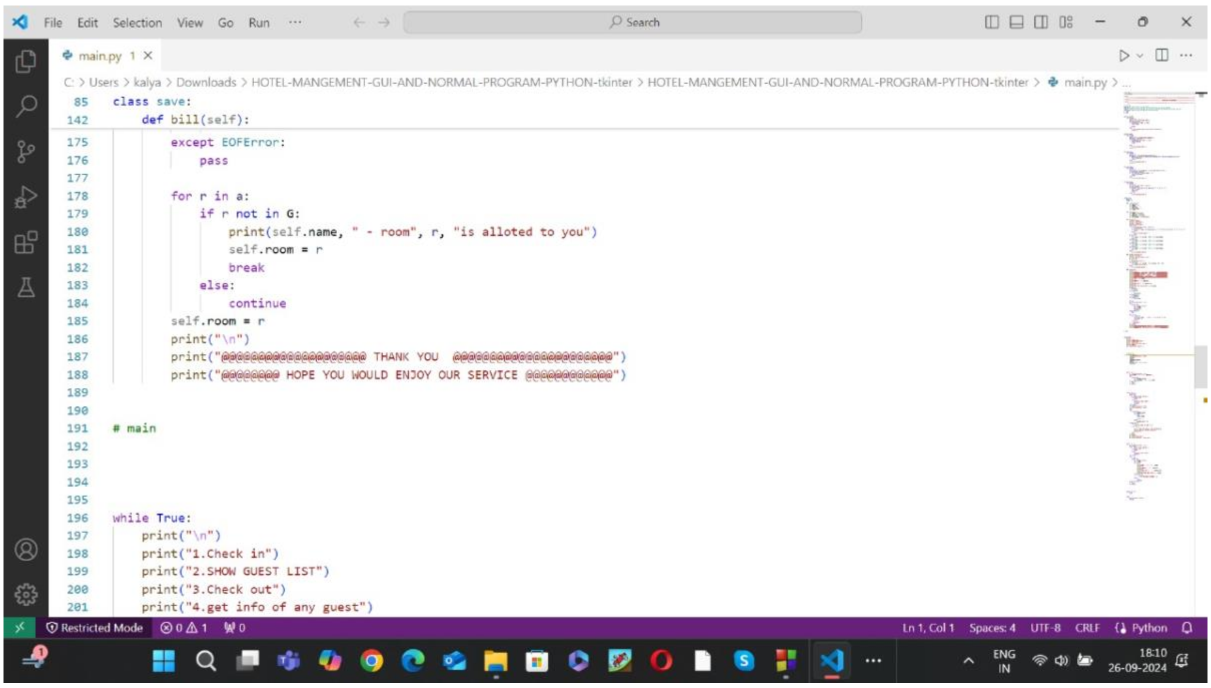

File Edit Selection View Go Run
Search
main.py 1
Restricted Mode
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python
ENG IN
18:10 26-09-2024

C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > main.py > ...

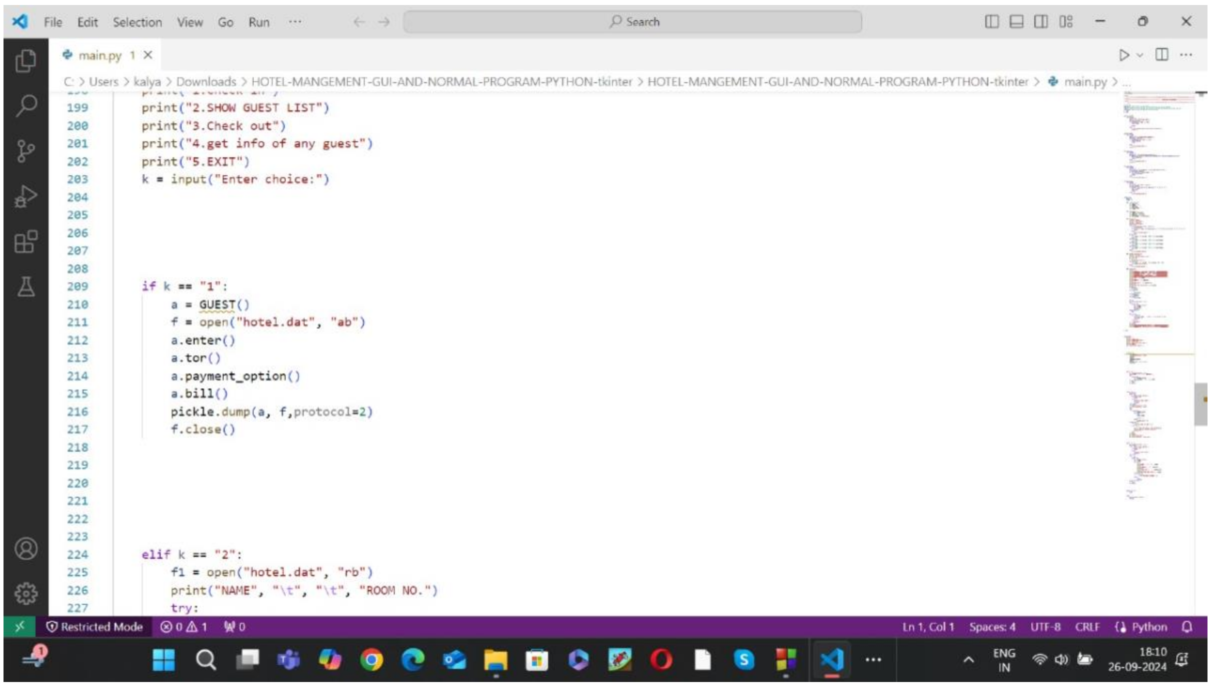
```python
class save:
    def tor(self):
            m[0] = 3
        elif ch == 4:
            self.price = self.price + (1700 * self.no_of_days)
            m[0] = 4
        else:
            print("invalid choice")

    def payment_option(self):
        print("1. By cash")
        print("2. By credit/debit card")
        op = chk_pay()
        if op == 1:
            print("No discount.")
        elif op == 2:
            self.price = self.price - ((self.price * 10) / 100)
            print("Discount of 10%.")
        else:
            print("Invalid option.")

    def bill(self):
        print("\n")
        print("@@@@@@@@@  5 STAR HOTEL AND RESORTS  @@@@@@@@@@@@@@@@@")
        print("@@@@@@@@@@@@ HAUS KHAZ,  NEW DELHI @@@@@@@@@@@@@@@@@@@")
        print("@@@@@@@@@@ SERVING    GUEST   SINCE @@@@@@@@@@@@@@@@@@")
        print("@@@@@@@@@@@@@@@    ###1950###       @@@@@@@@@@@@@@@@@@")
        print("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@")
        print("NAME-", self.name)
```


File Edit Selection View Go Run
Search
main.py 1
C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > main.py > ...
Restricted Mode
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python
ENG IN
18:10 26-09-2024

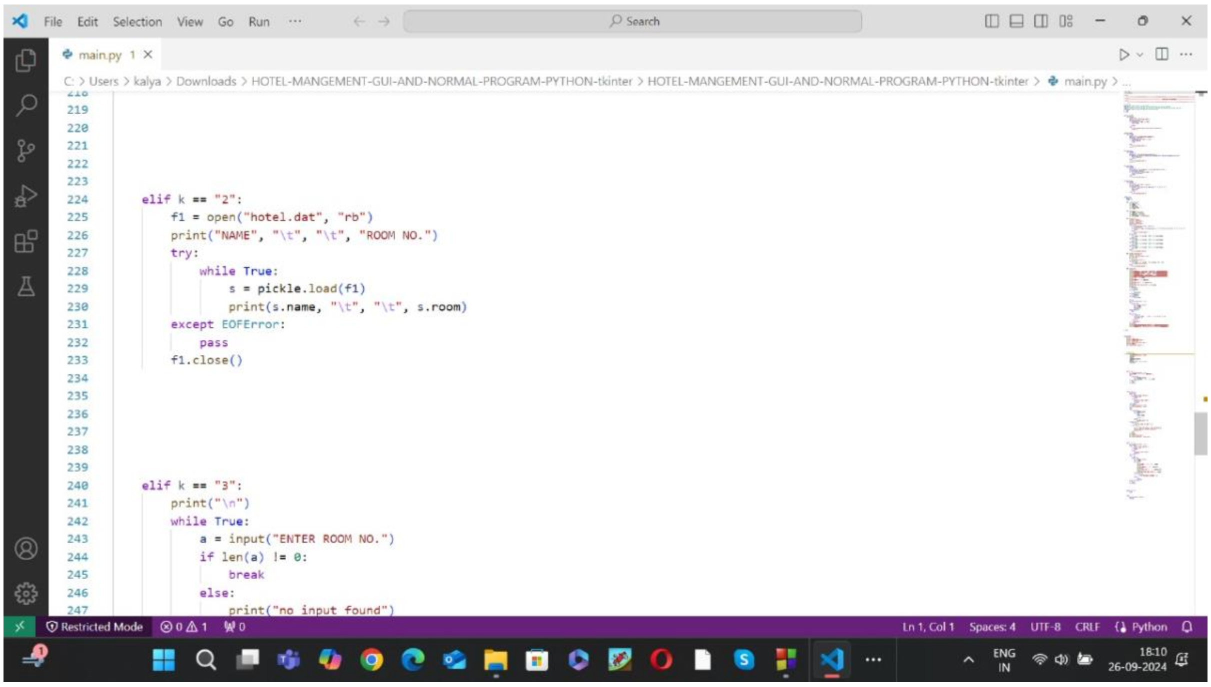
```python
class save:
    def bill(self):
        print("NAME-", self.name)
        print("\n")
        print("ADDRESS-", self.address)
        print("\n")
        print("MOBILE NO.-", self.mobile_no)
        print("\n")
        print("YOUR TOTAL BILL IS Rs.", self.price)
        print("\n")
        if m[0] == 1:
            a = Delux
        elif m[0] == 2:

            a = Semi_Delux
        elif m[0] == 3:
            a = General
        elif m[0] == 4:
            a = Joint_Room

        G = []
        f2 = open("hotel.dat", "rb")
        try:
            while True:
                s = pickle.load(f2)
                k = s.room
                G.append(k)
                continue
        except EOFError:
```


File Edit Selection View Go Run
Search
main.py 1
C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > main.py > ...
Restricted Mode
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python
ENG IN
18:10 26-09-2024

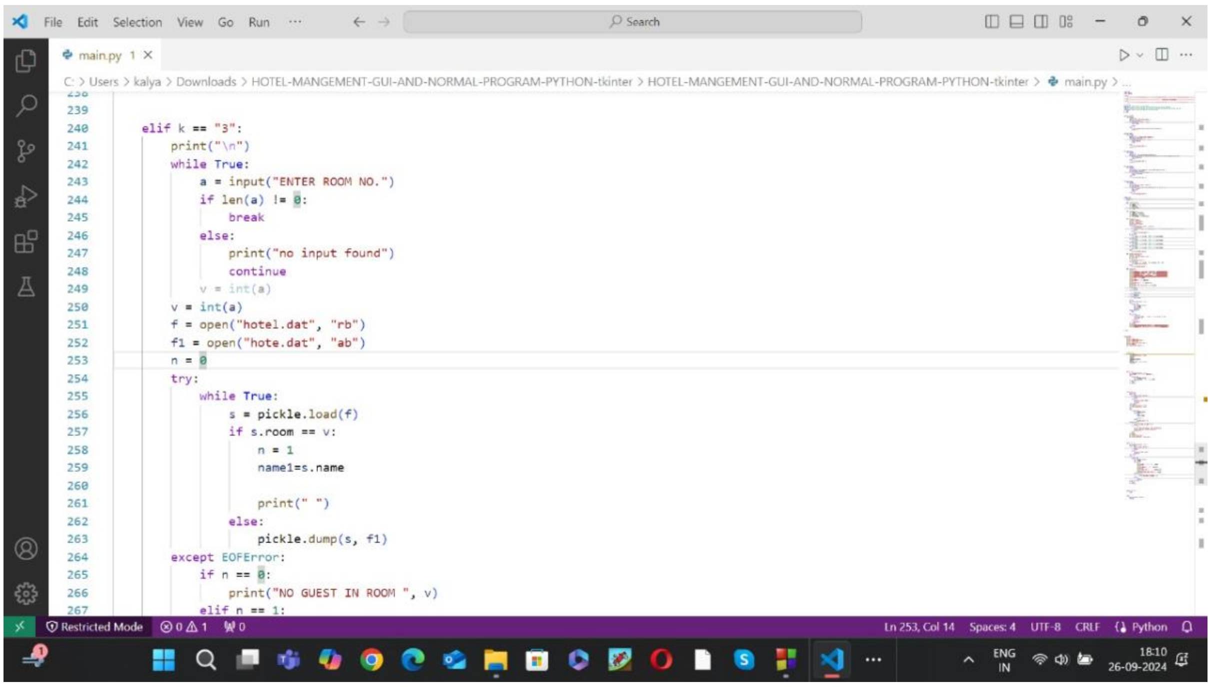
```python
class save:
    def bill(self):
        except EOFError:
            pass

        for r in a:
            if r not in G:
                print(self.name, " - room", r, "is alloted to you")
                self.room = r
                break
            else:
                continue
        self.room = r
        print("\n")
        print("@@@@@@@@@@@@@@@@@@@@ THANK YOU  @@@@@@@@@@@@@@@@@@@@@@@")
        print("@@@@@@@@ HOPE YOU WOULD ENJOY OUR SERVICE @@@@@@@@@@@@")


# main




while True:
    print("\n")
    print("1.Check in")
    print("2.SHOW GUEST LIST")
    print("3.Check out")
    print("4.get info of any guest")
```


File Edit Selection View Go Run
main.py 1
C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > main.py
Restricted Mode
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python
ENG IN 18:10 26-09-2024

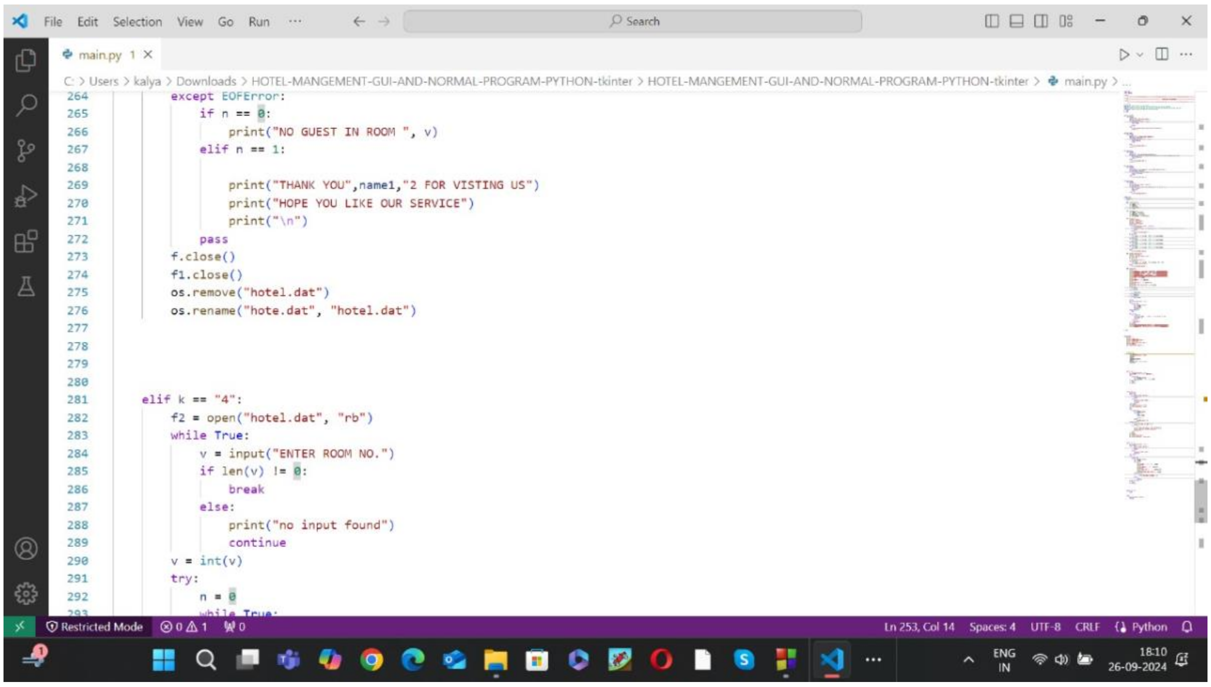
```
    print("2.SHOW GUEST LIST")
    print("3.Check out")
    print("4.get info of any guest")
    print("5.EXIT")
    k = input("Enter choice:")




    if k == "1":
        a = GUEST()
        f = open("hotel.dat", "ab")
        a.enter()
        a.tor()
        a.payment_option()
        a.bill()
        pickle.dump(a, f,protocol=2)
        f.close()






    elif k == "2":
        f1 = open("hotel.dat", "rb")
        print("NAME", "\t", "\t", "ROOM NO.")
        try:
```


File Edit Selection View Go Run
main.py 1
C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > main.py
Restricted Mode
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python
ENG IN 18:10 26-09-2024

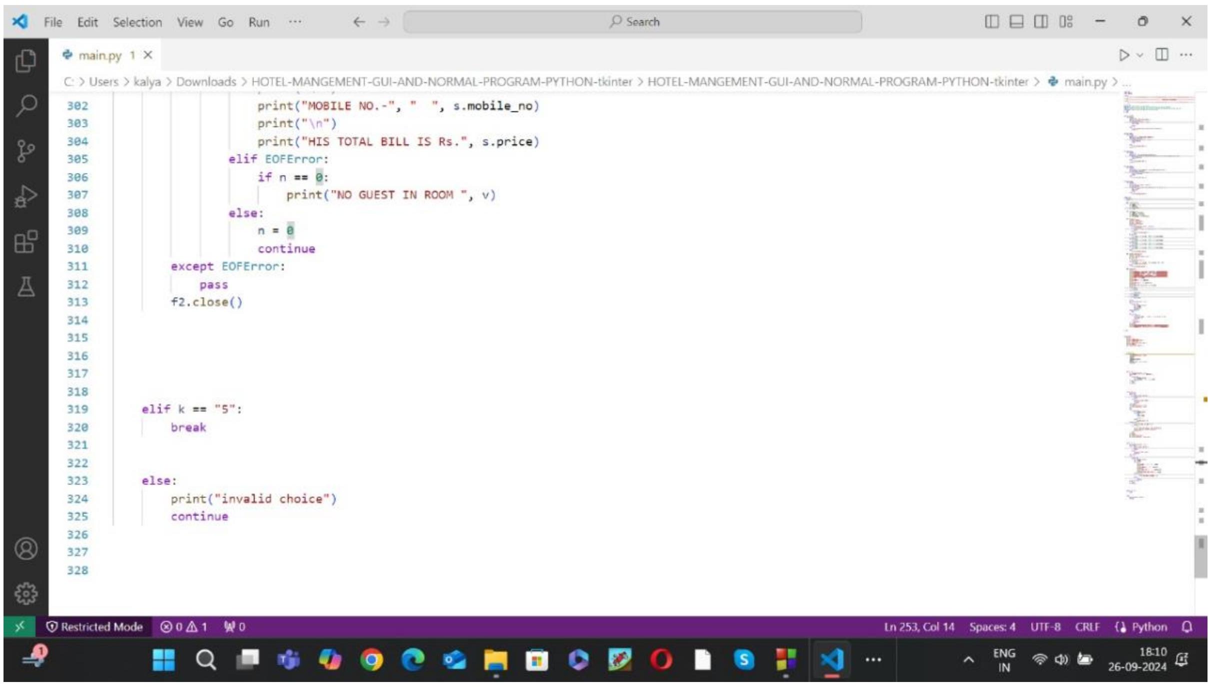
```





    elif k == "2":
        f1 = open("hotel.dat", "rb")
        print("NAME", "\t", "\t", "ROOM NO.")
        try:
            while True:
                s = pickle.load(f1)
                print(s.name, "\t", "\t", s.room)
        except EOFError:
            pass
        f1.close()






    elif k == "3":
        print("\n")
        while True:
            a = input("ENTER ROOM NO.")
            if len(a) != 0:
                break
            else:
                print("no input found")
```


File Edit Selection View Go Run ···
Search
main.py 1
C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > main.py > ...
Restricted Mode ⊗ 0 ⚠ 1 ₩ 0
Ln 253, Col 14 Spaces: 4 UTF-8 CRLF {} Python
ENG IN
18:10 26-09-2024

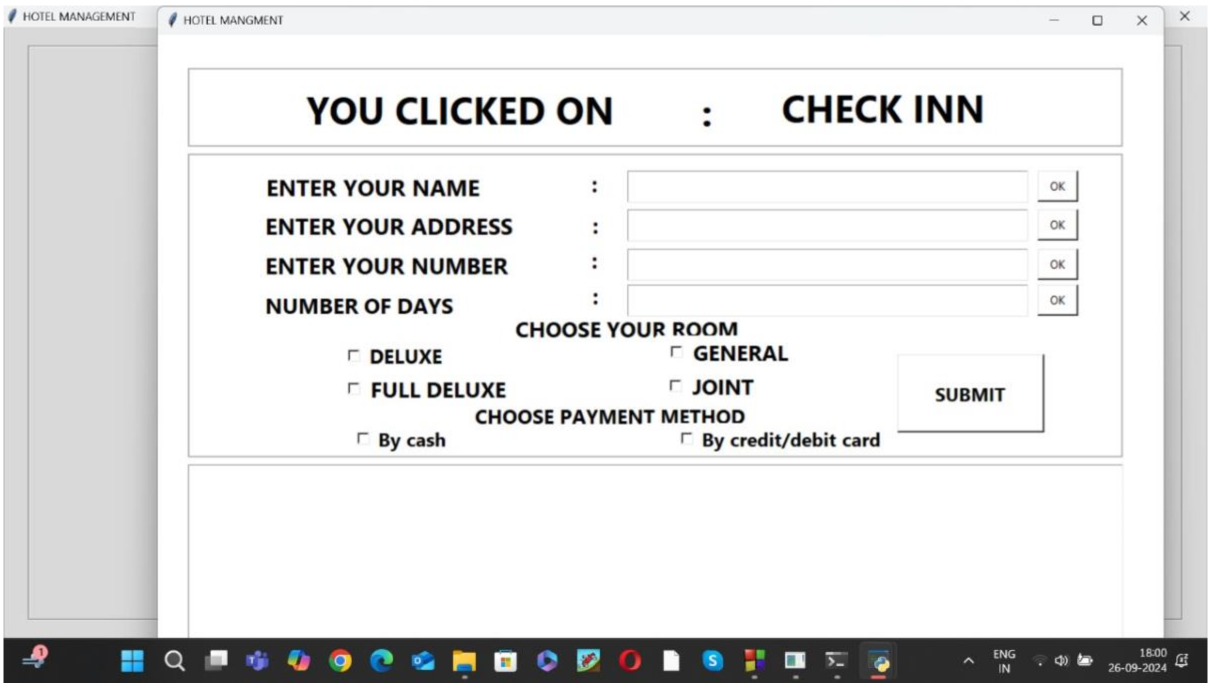
```

    elif k == "3":
        print("\n")
        while True:
            a = input("ENTER ROOM NO.")
            if len(a) != 0:
                break
            else:
                print("no input found")
                continue
            v = int(a)
        v = int(a)
        f = open("hotel.dat", "rb")
        f1 = open("hote.dat", "ab")
        n = 0
        try:
            while True:
                s = pickle.load(f)
                if s.room == v:
                    n = 1
                    name1=s.name

                    print(" ")
                else:
                    pickle.dump(s, f1)
        except EOFError:
            if n == 0:
                print("NO GUEST IN ROOM ", v)
            elif n == 1:
```


File Edit Selection View Go Run ···
Search
main.py 1
C: > Users > kalya > Downloads > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > HOTEL-MANGEMENT-GUI-AND-NORMAL-PROGRAM-PYTHON-tkinter > main.py > ...
Restricted Mode ⊗ 0 ⚠ 1 ₩ 0
Ln 253, Col 14 Spaces: 4 UTF-8 CRLF {} Python
ENG IN
18:10 26-09-2024

```
        except EOFError:
            if n == 0:
                print("NO GUEST IN ROOM ", v)
            elif n == 1:

                print("THANK YOU",name1,"2 FOR VISTING US")
                print("HOPE YOU LIKE OUR SERVICE")
                print("\n")
            pass
        f.close()
        f1.close()
        os.remove("hotel.dat")
        os.rename("hote.dat", "hotel.dat")




    elif k == "4":
        f2 = open("hotel.dat", "rb")
        while True:
            v = input("ENTER ROOM NO.")
            if len(v) != 0:
                break
            else:
                print("no input found")
                continue
        v = int(v)
        try:
            n = 0
            while True:
```



```python
                print("MOBILE NO.-", "  ", s.mobile_no)
                print("\n")
                print("HIS TOTAL BILL IS Rs.", s.price)
            elif EOFError:
                if n == 0:
                    print("NO GUEST IN ROOM ", v)
            else:
                n = 0
                continue
    except EOFError:
        pass
    f2.close()




elif k == "5":
    break


else:
    print("invalid choice")
    continue
```



YOU CLICKED ON : CHECK INN

ENTER YOUR NAME	:	OK
ENTER YOUR ADDRESS	:	OK
ENTER YOUR NUMBER	:	OK
NUMBER OF DAYS	:	OK

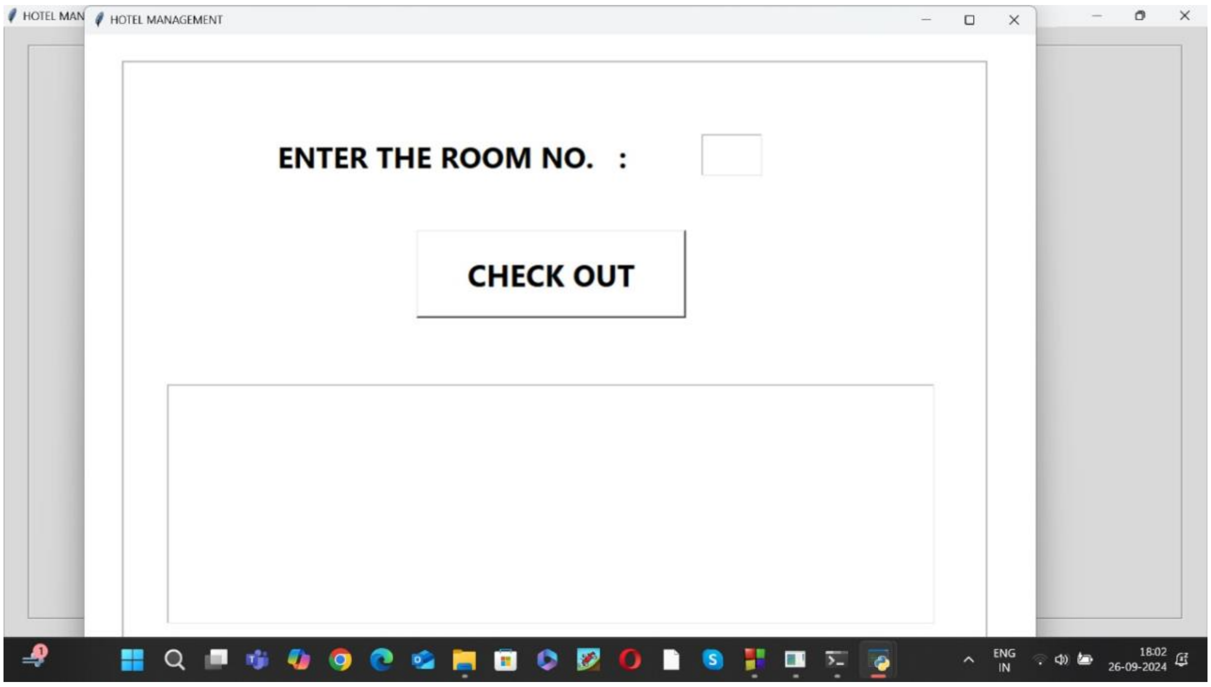
CHOOSE YOUR ROOM

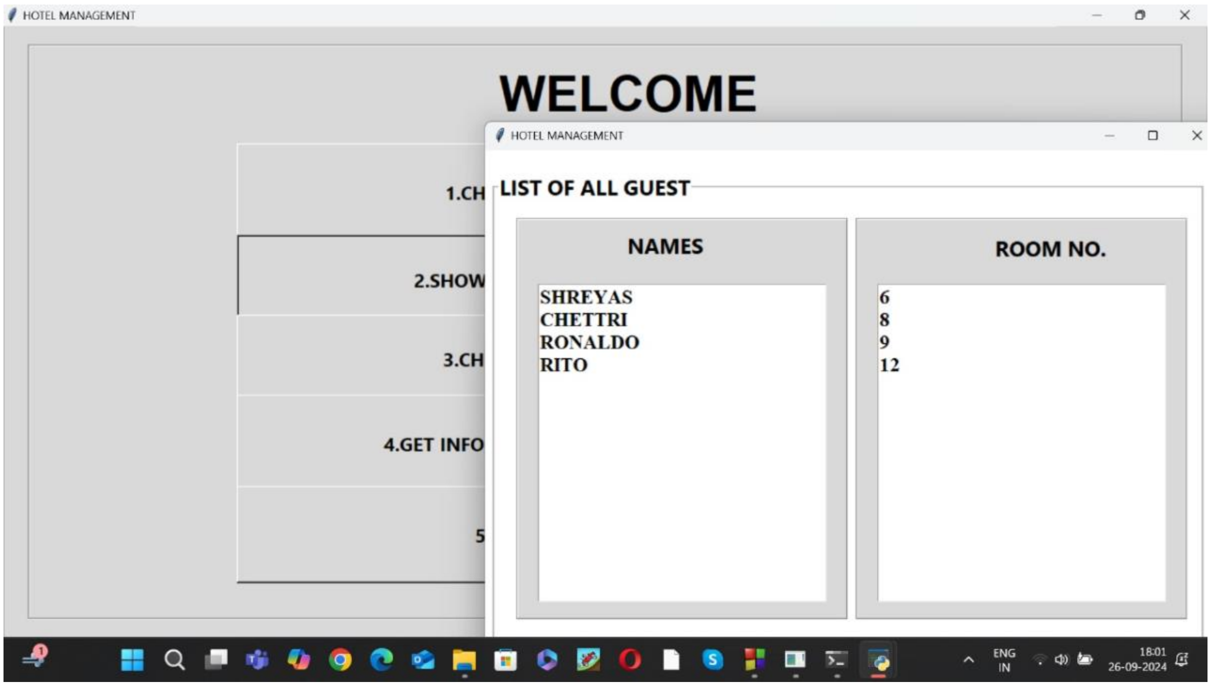
☐ DELUXE ☐ GENERAL

☐ FULL DELUXE ☐ JOINT

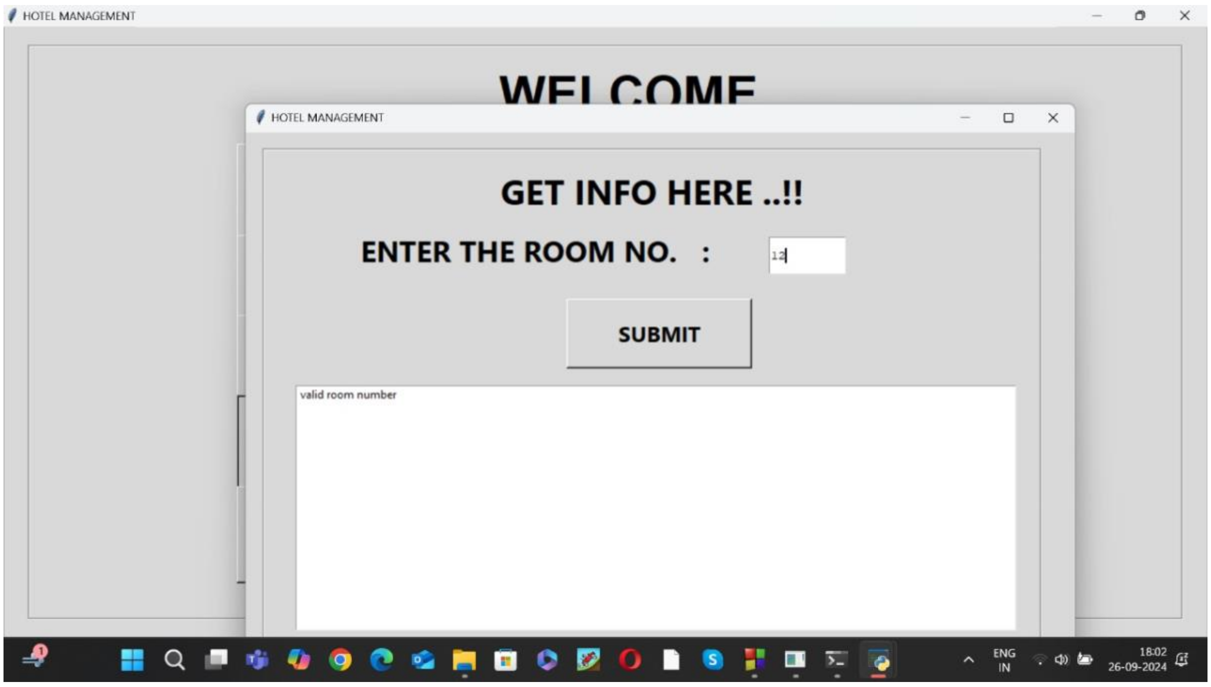
CHOOSE PAYMENT METHOD

☐ By cash ☐ By credit/debit card





NAMES	ROOM NO.
SHREYAS	6
CHETTRI	8
RONALDO	9
RITO	12



valid room number

C) Sample code: - (Simple sample code of RR hotel).

```
import sqlite3
from datetime import datetime

conn = sqlite3.connect('rr_hotels.db')
cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS customers (
                    customer_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    name TEXT,
                    email TEXT,
                    phone TEXT
                  )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS rooms (
                    room_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    room_type TEXT,
                    price REAL,
                    status TEXT
                  )''')
```

```
cursor.execute('''CREATE TABLE IF NOT EXISTS bookings (
                    booking_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    customer_id INTEGER,
                    room_id INTEGER,
                    check_in_date TEXT,
                    check_out_date TEXT,
                    FOREIGN KEY (customer_id) REFERENCES customers (customer_id),
                    FOREIGN KEY (room_id) REFERENCES rooms (room_id)
                  )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS payments (
                    payment_id INTEGER PRIMARY KEY AUTOINCREMENT,
                    booking_id INTEGER,
                    amount REAL,
                    payment_method TEXT,
                    FOREIGN KEY (booking_id) REFERENCES bookings (booking_id)
                  )''')
conn.commit()

class Customer:
    def __init__(self, name, email, phone):
        self.name = name
        self.email = email
        self.phone = phone

    def register(self):
        cursor.execute('INSERT INTO customers (name, email, phone) VALUES (?, ?, ?)',
                       (self.name, self.email, self.phone))
        conn.commit()
        print(f"Customer {self.name} registered successfully!")
```

```
class Room:
    def __init__(self, room_type, price, status="available"):
        self.room_type = room_type
        self.price = price
        self.status = status

    @staticmethod
    def check_availability(room_type):
        cursor.execute('SELECT * FROM rooms WHERE room_type = ? AND status =
"available"', (room_type,))
        available_rooms = cursor.fetchall()
        return available_rooms

    @staticmethod
    def update_status(room_id, status):
        cursor.execute('UPDATE rooms SET status = ? WHERE room_id = ?', (status,
room_id))
        conn. commit()

class Booking:
    def __init__(self, customer_id, room_id, check_in, check_out):
        self.customer_id = customer_id
        self.room_id = room_id
        self.check_in = check_in
        self.check_out = check_out

    def create_booking(self):
        cursor.execute('INSERT INTO bookings (customer_id, room_id, check_in_date,
check_out_date) VALUES (?, ?, ?, ?)',
                       (self.customer_id, self.room_id, self.check_in, self.check_out))
        conn.commit()
        Room.update_status(self.room_id, "booked")
        print(f"Booking for customer {self.customer_id} created successfully!")
```

```
class Payment:
    def __init__(self, booking_id, amount, payment_method):
        self.booking_id = booking_id
        self.amount = amount
        self.payment_method = payment_method

    def process_payment(self):
        cursor.execute('INSERT INTO payments (booking_id, amount, payment_method)
VALUES (?, ?, ?)',
                       (self.booking_id, self.amount, self.payment_method))
        conn.commit()
        print(f"Payment of {self.amount} by {self.payment_method} processed
successfully!")



def add_sample_rooms():
    rooms = [
        ('Single', 2000.0, 'available'),
        ('Double', 3000.0, 'available'),
        ('Suite', 5000.0, 'available')
    ]
    cursor.executemany('INSERT INTO rooms (room_type, price, status) VALUES (?, ?,
?)', rooms)
    conn.commit()

if __name__ == "__main__":
    print("Welcome to RR Hotels Booking System")

    customer = Customer("John Doe", "johndoe@example.com", "1234567890")
    customer.register()

    room_type = "Single"
    available_rooms = Room.check_availability(room_type)
    if available_rooms:
        print(f"Available {room_type} rooms: {available_rooms}")
        selected_room_id = available_rooms[0][0]

            check_in = "2024-09-27"
        check_out = "2024-09-30"
        booking = Booking(1, selected_room_id, check_in, check_out)
        booking.create_booking()

        amount = available_rooms[0][2] * 3  # 3 days stay
        payment_method = "gpay"
        payment = Payment(1, amount, payment_method)
        payment.process_payment()
    else:
        print(f"No {room_type} rooms available!")

conn.close()
```