COLLEGE OF COMPUTING & INFORMATICS (CCI)

SEMESTER 2, 2023/24

BUSINESS PROGRAMMING

(CISB4223)

Section 01

MINI PROJECT : HOTEL RESERVATION SYSTEM [GENTING SKYWORLD HOTEL]

Group Members:	**Ahmad Amirul Aizad Rosmadi**	**(IS01082507)**

SUBMISSION DATE : 2 JUNE 2024

Contents

Company Background

Genting Group is a prominent Malaysian conglomerate with interests in leisure and hospitality, as well as other sectors such as gaming, entertainment, and plantations. The group was founded by the late Tan Sri Lim Goh Tong in 1965. One of the key highlights of Genting Group's hospitality portfolio is Resorts World Genting, which is located in the Genting Highlands, a mountain resort in Malaysia.

Resorts World Genting is one of the largest and most popular integrated resorts in Southeast Asia. It features multiple hotels, a wide array of entertainment options, a theme park, shopping malls, restaurants, and a casino. The resort complex has been a major tourist destination for decades, attracting visitors from Malaysia and around the world.

Genting Skyworld Hotel located right at the doorsteps of the Arena of Stars and SkyAvenue, Genting Skyworlds Hotel is a stone's throw away from most of Genting Highland's popular attractions. Besides being walking distance to the resort's lifestyle mall SkyAvenue, it also provides direct access to the brand-new Genting Skyworlds Outdoor Theme Park**.** The theme park's entrance and ticketing counters are literally right outside the hotel!

Genting Skyworlds Hotel offers close to 300 rooms with four room types that can cater to different groups of travellers. The largest room houses three queen-sized beds and can fit up to 6 guests, while the most luxurious is a 550 sqft spacious suite that comes with an indulgent bathtub.

Take your pick from these four room types available:

Room Type	Sleeping Arrangement	Price per Night
Quad Room	2 Queen Beds	RM 308
Sixer Room	3 Queen Beds	RM 340
King Room	1 King Bed	RM 400
Theme Park Hotel Suite	1 King Bed	RM 900

All rooms at Genting Skyworlds Hotel are simply yet tastefully furnished with colourful decor, family-inspired graffiti, and contemporary furniture. They also all come with satellite TVs, attached bathrooms, coffee and tea-making facilities, and complimentary wi-fi.

System Objective

This system was developed targeting a simple hotel reservation and management system. This system will be useful to hotel employees & staff to do their work. Thus, creating efficiency for customers to stay at the hotel.

Its main objectives include:

- **Check-In**: Allows users to check-in guests by providing their details and assigning them available rooms based on the selected room type.
- **Room Service**: Provides guests with the option to order room service items from a menu. The system records the room service charges for each guest.
- **Display Occupied Rooms**: Shows a list of currently occupied rooms along with guest details
- **Search Room**: Allows users to search for specific room details based on the room number.
- **Edit Room Details**: Enables users to edit guest details (name, address, and phone number) for a specific room.
- **Check-Out**: Facilitates the check-out process for guests, calculating and displaying the bill including room charges and room service charges.
- **Save and Load Data**: The system can save and load data from a file, allowing it to maintain records of occupied rooms, available rooms, and guest details even after restarting the application.

Overall, the system aims to streamline the hotel management process, from guest check-in to check-out, while providing functionalities for room service, data management, and guest interaction.

Source Code with Detailed IPO (Input, Process, Output)

Complete Source Code with Description

```
from datetime import date
import json
```

- Import necessary modules for date handling (date from datetime module) and JSON serialization/deserialization (json).

```
class Hotel:
    def __init__(self):
        self.rooms = {}
        self.available_rooms = {'std':[101,102,103], 'delux':[201,202,203],
'execu':[301,302,303], 'suite':[401,402,403]}
        self.roomprice = {1:308, 2:340, 3:400, 4:900}
        self.load_data()
```

- Define a class Hotel with attributes to manage hotel rooms, available rooms, room prices, and initialize by loading existing data.

```
def save_data(self):
        data = {
            'rooms': self.rooms,
            'available_rooms': self.available_rooms
        }
        with open('Learning/Mini Project/hotel_data.txt', 'w') as file:
            json.dump(data, file)
```

- Method save_data serializes the hotel data (occupied and available rooms) into a JSON format and saves it to a file named hotel_data.txt.

```
def load_data(self):
        try:
            with open('Learning/Mini Project/hotel_data.txt', 'r') as file:
                data = json.load(file)
                self.rooms = {int(k): v for k, v in data['rooms'].items()}
                self.available_rooms = data['available_rooms']
        except FileNotFoundError:
            pass
```

- Method load_data loads existing hotel data from the hotel_data.txt file if it exists, otherwise initializes the hotel attributes.

```
def check_in(self, name, address, phone):
        try:
            roomtype = int(input("\nRoom types: \n1. Quad Room \n2. Sixer
Room \n3. King Room \n4. Theme Park Hotel Suite \n\nSelect a room type: "))

            if roomtype == 1:
                if self.available_rooms['std']:
                    room_no = self.available_rooms['std'].pop(0)
                else:
                    print("Sorry, Standard room not available")
                    return

            elif roomtype == 2:
                if self.available_rooms['delux']:
                    room_no = self.available_rooms['delux'].pop(0)
                else:
                    print("Sorry, Deluxe room not available")
                    return

            elif roomtype == 3:
                if self.available_rooms['execu']:
                    room_no = self.available_rooms['execu'].pop(0)
                else:
                    print("Sorry, Executive room not available")
                    return

            elif roomtype == 4:
                if self.available_rooms['suite']:
                    room_no = self.available_rooms['suite'].pop(0)
                else:
                    print("Sorry, Suite room not available")
                    return

            else:
                print("Choose a valid room type")
                return

            d, m, y = map(int, input("Enter check-in date in day-month-year
format: ").split())

            try:
                check_in = date(y, m, d)
            except ValueError:
                print("Invalid date format. Please enter in day-month-year
format.")
                return

            self.rooms[room_no] = {
```

```
                'name': name,
                'address': address,
                'phone': phone,
                'check_in_date': check_in.strftime('%Y-%m-%d'),
                'room_type': roomtype,
                'roomservice': 0
            }

            self.save_data()
            print(f"\n~ - ~ - Checked in {name}, {address}, to room:
{room_no} on {check_in} ~ - ~ -")

        except ValueError:
            print("Invalid choice. Please try again. ")
```

- Method check_in prompts the user to select a room type for check-in.
- If the selected room type is Quad Room [standard (std)] (roomtype == 1), it assigns a room number from the available Quad rooms, else notifies the unavailability.
- If the selected room type is Sixer Room [deluxe (delux)] (roomtype == 2), it assigns a room number from the available Sixer rooms, else notifies the unavailability.
- If the selected room type is King Room [executive (execu)] (roomtype == 3), it assigns a room number from the available King rooms, else notifies the unavailability.
- If the selected room type is Theme Park Hotel Suite [suite (suite)] (roomtype == 4), it assigns a room number from the available Theme Park Hotel Suite, else notifies the unavailability.
- If the user selects an invalid room type, it prompts to choose a valid one.
- Prompt user for check-in date and create a date object.
- Store check-in details including client information, check-in date, room type, and initialize room service charges to 0.
- Save data after check-in and print a confirmation message.

```
def room_service(self, room_no):
        if room_no in self.rooms:
            print("\n***** Genting Restaurant Menu *****")
            print("1. Tea/Coffee: RM 10 \t2. Dessert: RM 20 \t3. Breakfast:
RM 50 \t4. Lunch: RM 80 \t5. Dinner: RM 120 \t6. Exit")

            while True:
                c = int(input("Select your choice: "))
```

```
                if c == 1:
                    q = int(input("Enter the quantity: "))
                    self.rooms[room_no]['roomservice'] += 10 * q

                elif c == 2:
                    q = int(input("Enter the quantity: "))
                    self.rooms[room_no]['roomservice'] += 20 * q

                elif c == 3:
                    q = int(input("Enter the quantity: "))
                    self.rooms[room_no]['roomservice'] += 50 * q

                elif c == 4:
                    q = int(input("Enter the quantity: "))
                    self.rooms[room_no]['roomservice'] += 80 * q

                elif c == 5:
                    q = int(input("Enter the quantity: "))
                    self.rooms[room_no]['roomservice'] += 120 * q

                elif c == 6:
                    break

                else:
                    print("Invalid option")

            self.save_data()
            print("Room Service RM:", self.rooms[room_no]['roomservice'],
"\n")

        else:
            print("Not occupied /Invalid room number")
```

- Method room_service checks if the provided room number is occupied.
- Display the menu options for room service.
- Enter a loop to process room service choices until the user chooses to exit.
- If the user selects option 1 (Tea/Coffee), prompt for quantity and update room service charges accordingly.
- If the user selects option 2 (Dessert), prompt for quantity and update room service charges accordingly.
- If the user selects option 3 (Breakfast), prompt for quantity and update room service charges accordingly.

- If the user selects option 4 (Lunch), prompt for quantity and update room service charges accordingly.
- If the user selects option 5 (Dinner), prompt for quantity and update room service charges accordingly.
- If the user selects option 6 (Exit), break out of the loop.
- If the user selects an invalid option, notify the message, and continue the loop.
- After exiting the loop, save the updated data and print the total room service charges for the room.
- If the provided room number is not occupied, notify the user.

```
def display_occupied(self):
        while True:
            print("\nOptions:")
            print("1. View Occupied Rooms")
            print("2. Search Room")
            print("3. Edit Room Details")
            print("4. Back to Main Menu")

            choice = input("\nEnter your choice (1-4): ")

            if choice == "1":
                self.show_occupied_rooms()
            elif choice == "2":
                self.search_room()
            elif choice == "3":
                self.edit_room()
            elif choice == "4":
                break
            else:
                print("Invalid choice. Please try again.")
```

- Method display_occupied presents a menu with options to view occupied rooms, search for a specific room, edit room details, or return to the main menu.
- If the user selects option 1, it calls the show_occupied_rooms method.
- If the user selects option 2, it calls the search_room method.
- If the user selects option 3, it calls the edit_room method.
- If the user selects option 4, it breaks out of the loop to return to the main menu.
- If the user enters an invalid choice, it notifies the user and continues the loop.

```
def show_occupied_rooms(self):
        if not self.rooms:
            print("\nNo rooms are occupied at the moment.")
        else:
            print("\nOccupied Rooms: ")
            print("--------------------------")
            print('Room no.     Name    Phone')
            print("--------------------------")

            for room_number, details in self.rooms.items():
                print(room_number, '\t', details['name'], '\t',
details['phone'])
```

- Method show_occupied_rooms display a list of occupied rooms with their numbers, client names, and phone numbers.
- The list that will be displayed it gets from hotel_data.txt

```
def search_room(self):
        search_room_no = int(input("Enter room number to search: "))
        if search_room_no in self.rooms:
            details = self.rooms[search_room_no]
            print(f"\nRoom No: {search_room_no}")
            print(f"Name: {details['name']}")
            print(f"Address: {details['address']}")
            print(f"Phone: {details['phone']}")
            print(f"Check-in Date: {details['check_in_date']}")
            print(f"Room Type: {details['room_type']}")
            print(f"Room Service: {details['roomservice']}")
        else:
            print(f"\nRoom {search_room_no} is not occupied.")
```

- Method search_room prompts the user to enter a room number to search for.
- If the room is occupied, it displays the details of the room including client name, address, phone number, check-in date, room type, and room service charges.
- If the room is not occupied, it notifies the user.

```
def edit_room(self):
        edit_room_no = int(input("Enter room number to edit: "))
        if edit_room_no in self.rooms:
            details = self.rooms[edit_room_no]
            print("\nEditing Room Details (leave blank to keep current
value):")
            name = input(f"Name ({details['name']}): ")
            address = input(f"Address ({details['address']}): ")
            phone = input(f"Phone ({details['phone']}): ")
            if name:
                details['name'] = name
            if address:
                details['address'] = address
            if phone:
                details['phone'] = phone
            self.save_data()
            print(f"\n~ ~ ~ ~ ~ Room {edit_room_no} details updated. ~ ~ ~ ~
~")
        else:
            print(f"\nRoom {edit_room_no} is not occupied.")
```

- Method edit_room prompts the user to enter a room number to edit its details.
- If the room is occupied, it allows the user to edit the client's name, address, and phone number. Changes are saved to the data file.
- If the room is not occupied, it notifies the user.

```
def check_out(self, room_number):
        if room_number in self.rooms:
            check_out_date = date.today()
            check_in_date =
date.fromisoformat(self.rooms[room_number]['check_in_date'])
            duration = (check_out_date - check_in_date).days

            roomtype = self.rooms[room_number]['room_type']

            if roomtype == 1:
                self.available_rooms['std'].append(room_number)

            elif roomtype == 2:
                self.available_rooms['delux'].append(room_number)

            elif roomtype == 3:
                self.available_rooms['execu'].append(room_number)
```

```
            elif roomtype == 4:
                self.available_rooms['suite'].append(room_number)

            print("\n-----------------------------------------------------------
--")
            print("|    Genting Hotel Receipt\t\t\t\t|")
            print("| \t\t\t\t\t\t\t|")
            print(f"|    Name:
{self.rooms[room_number]['name']}\t\t\t\t\t|")
            print(f"|    Address:
{self.rooms[room_number]['address']}\t\t\t\t|")
            print(f"|    Phone:
{self.rooms[room_number]['phone']}\t\t\t\t\t|")
            print(f'|    Room Number: {room_number}\t\t\t\t\t|')
            print(f"|    Check-In date: {check_in_date.strftime('%d %B
%Y')}\t\t\t\t|")
            print(f'|    Check-Out date: {check_out_date.strftime("%d %B
%Y")}\t\t\t|')
            print(f"|    No. of Days: {duration}\tPrice per day: RM
{self.roomprice[roomtype]}\t\t|")
            print("| \t\t\t\t\t\t\t|")

            roombill = self.roomprice[roomtype] * duration
            roomservice = self.rooms[room_number]['roomservice']

            print('|    Room bill: RM ', roombill,'\t\t\t\t|')
            print('|    Room service: RM ', roomservice,'\t\t\t\t|')
            print('|    Total bill: RM ', roombill +
roomservice,'\t\t\t\t|')
            print("-----------------------------------------------------------
")

            del self.rooms[room_number]
            self.save_data()

        else:
            print(f"Room {room_number} is not occupied.")
```

- Method check_out checks if the provided room number is occupied.
- Calculates the duration of stay and marks the room as available for the respective room type.
- Updates the available rooms list with the checked-out room number.
- Display a detailed receipt including client details, room charges, room service charges, and total bill.

- *del self.rooms* function - Remove the checked-out room from the occupied rooms list and save the updated data to the file.
- If the provided room number is not occupied, notify the user.

```
def start_app(self):
        while True:
            print("\n-----------------------------------------------")
            print("Welcome To Genting Skyworld Hotel Reservation")
            print("1. Check-In")
            print("2. Room Service")
            print("3. Display Occupied Room")
            print("4. Check-Out")
            print("5. Exit\n")

            choice = input("Enter your choice (1-5): ")

            if choice == "1":
                name = input("\nEnter Client Name: ")
                address = input("Enter Address: ")
                phone = input("Enter Contact No: ")
                self.check_in(name, address, phone)

            elif choice == "2":
                try:
                    room_no = int(input("\nEnter the room number: "))
                    self.room_service(room_no)

                except ValueError:
                    print("Invalid choice. Please try again. ")

            elif choice == "3":
                self.display_occupied()

            elif choice == "4":
                try:
                    room_number = int(input("\nEnter the room number: "))
                    self.check_out(room_number)
                except ValueError:
                    print("Invalid input. Please try again. ")

            elif choice == "5":
                break

            else:
                print("\nInvalid choice. Please try again.")
```

- Method start_app presents a main menu with options for check-in, room service, display of occupied rooms, check-out, and exit.
- The list function displays and provides a handle user input based on the selected menu option:
    - If the user selects option 1, it prompts for client information and calls the check_in method.
    - If the user selects option 2, it prompts for the room number and calls the room_service method.
    - If the user selects option 3, it calls the display_occupied method to show occupied rooms.
    - If the user selects option 4, it prompts for the room number and calls the check_out method.
    - If the user selects option 5, it breaks out of the loop to exit the application.
    - If the user enters an invalid choice, it notifies the user and continues the loop.

```
h = Hotel()
h.start_app()
```

Create an instance of the Hotel class and start the hotel reservation application by calling the start_app method.

IPO Details

This runs the hotel booking application, allowing users to check in, request room service, display occupied rooms, check out or exit the application. So below is the **input, process, and output (IPO)** for the developed code.

Input:

1. Check-In:
    - Client information: Name, address, phone number.
    - Selection of room type.
    - Check-in date (day-month-year format).
2. Room Service:
    - Room number for which the service is requested.
    - Choice of items from the menu and their quantities.
3. Display Occupied Room:
    - Option selection (view occupied rooms, search room, edit room details, back to the main menu).
    - Specific room number (for search and edit).
4. Check-Out:
    - Room number for check-out.

Process:

1. Initialization:
    - Initialize hotel rooms, available rooms, room prices, and load data from a file if available.
2. Check-In:
    - Prompt user for client information, room type selection, and check-in date.
    - Assign a room if available, else notify the unavailability.
    - Store check-in details including client information, room number, and check-in date.
    - Save data to a file after check-in.
3. Room Service:
    - Display a menu of services and their prices.

- Accept user input for service selection and quantity.
- Update room service charges for the specified room.
- Save data after updating room service details.

4. Display Occupied Room:
    - Provide options to view occupied rooms, search for a specific room, edit room details, or return to the main menu.
    - Implement functionalities accordingly:
        - Show occupied rooms with their numbers, client names, and phone numbers.
        - Search for a room by its number and display its details if occupied.
        - Edit details of an occupied room.
5. Check-Out:
    - Prompt user for the room number to check out.
    - Calculate the duration of stay and total bill including room charges and room service charges.
    - Mark the room as available for the specified room type.
    - Generate and display a receipt with client details, room charges, and total bill.
    - Remove the room from the occupied rooms list.
    - Save data after check-out.
6. Start Application:
    - Present a menu with options for check-in, room service, display of occupied rooms, check-out, and exit.
    - Handle user input and execute the corresponding functionality.
    - Continue until the user chooses to exit.

Output:

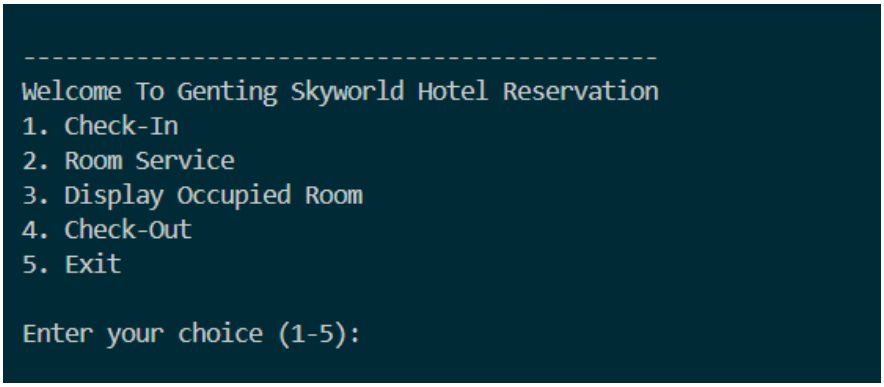
1. Check-In:
    - Confirmation message with client details, assigned room number, and check-in date.
2. Room Service:
    - Updated room service charges for the specified room.

3. Display Occupied Room:
    - List of occupied rooms with their numbers, client names, and phone numbers.
    - Details of a specific room if searched.
4. Check-Out:
    - Detailed receipt including client details, room charges, room service charges, and total bill.
5. Start Application:
    - Menu options for different functionalities.

Sample Output

This particular example also shows the features of the hotel reservation system with activities such as checking in, ordering services, viewing which rooms are taken, and checking out. It highlights the dialogue with the user and the output that the system provides in return.

1. Main Interface



The application starts with a welcome message and presents the main menu with five options: Call made to distinguish Check-In, Room Service, Display Occupied Room, Check-Out, and Exit.

Error Handling

The program will tell the user to double check their input and the program loop and the main interface like the first image will be displayed.

2. Check In

```
Enter Client Name: John Wahid
Enter Address: 123 Street Fight
Enter Contact No: 123456789

Room types:
1. Quad Room
2. Sixer Room
3. King Room
4. Theme Park Hotel Suite

Select a room type: 1
Enter check-in date in day-month-year format: 29 5 2023

~ - ~ - Checked in John Wahid, 123 Street Fight, to room: 103 on 2023-05-29 ~ - ~ -
```

The user enters input 1 (Check In) to accommodate at the Hotel.

Option 1 is selected for check-in and details need to be filled in by the user for verification and information to be stored in the hotel's data store.

- For instance, a client namely “John Wahid” with an address “123 Street Fight” contact number “123456789”.
- The user enters the option 1 to choose the type of the room as Quad Room (Standard) and then enters the check in date.
- After entering the check-in particulars John Wahid gets the check-in confirmation with the room number 103 allocated to him.

Error Handling

```
Enter Client Name: Jamal
Enter Address: Excel Street
Enter Contact No: 987654321

Room types:
1. Quad Room
2. Sixer Room
3. King Room
4. Theme Park Hotel Suite

Select a room type: 1
Sorry, Standard room not available
```

If the room capacity is full, this message will be displayed to inform the user that their preferred room is not currently available. Therefore, the information entered will not be forwarded to the hotel's data store.

```
Enter your choice (1-5): 1

Enter Client Name: Jamal
Enter Address: Street Dance
Enter Contact No: 987654321

Room types:
1. Quad Room
2. Sixer Room
3. King Room
4. Theme Park Hotel Suite

Select a room type: 10
Choose a valid room type
```

```
Enter your choice (1-5): 1

Enter Client Name: Jamal
Enter Address: Street Dance
Enter Contact No: 987654321

Room types:
1. Quad Room
2. Sixer Room
3. King Room
4. Theme Park Hotel Suite

Select a room type: u
Invalid choice. Please try again.
```

If user enter incorrect input, the program will tell the user to double check their input and the program will be back to main interface.

3. Room Service

```
Enter your choice (1-5): 2

Enter the room number: 103

***** Genting Restaurant Menu *****
1. Tea/Coffee: RM 10    2. Dessert: RM 20      3. Breakfast: RM 50    4. Lunch: RM 80        5. Dinner: RM 120      6. Exit
Select your choice: 1
Enter the quantity: 2
Select your choice: 2
Enter the quantity: 1
Select your choice: 6
Room Service RM: 40
```

The user enters input 2 (Room Service) to use the room-to-room service provided by the Hotel.

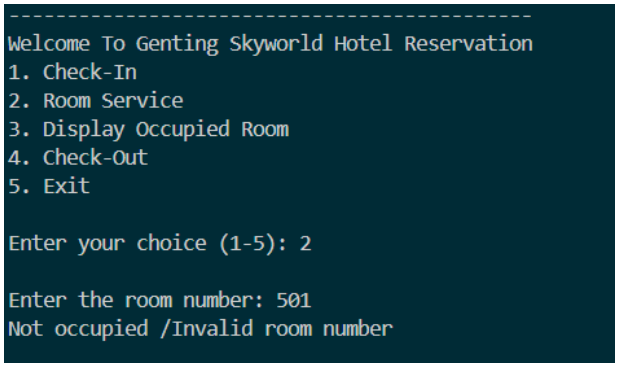
The user presses the number 2 for room service, and enters the room number, 103.

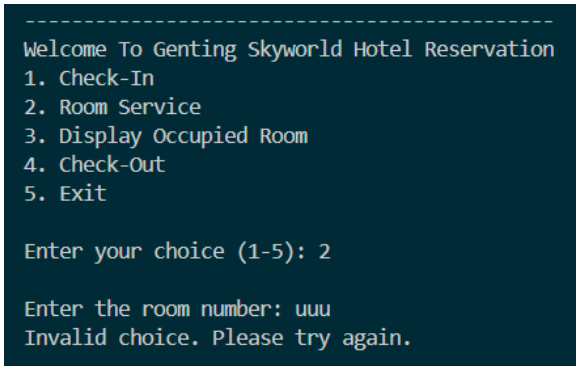
The system displays a new dialog with the options that contain tea/coffee, dessert, breakfast, lunch, dinner, or quit.

In this case, tea/coffee is selected, and the quantity is 2 from option 1. Additionally, dessert is selected, and the quantity is 1 from option 2.

The program revises the room service charges for room 103 to RM 40 in the system.

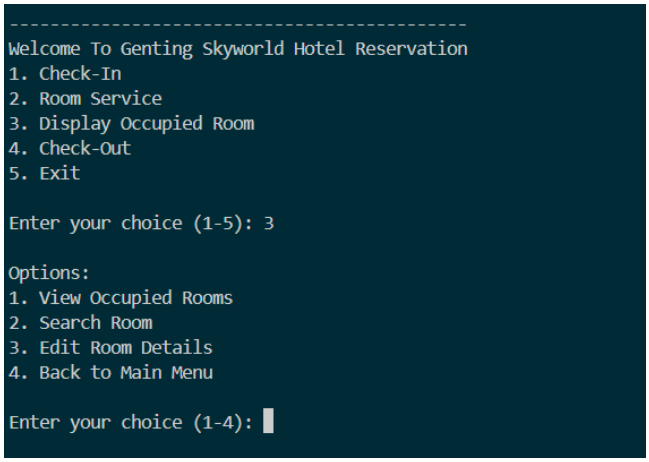
Error Handling

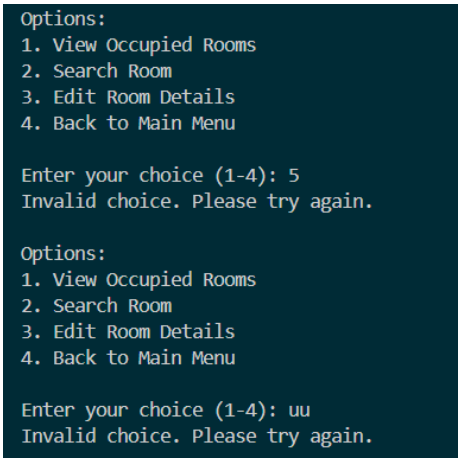




If user enter incorrect input, the program will tell the user to double check their input and the program will be back to main interface.

4. Display Occupied Room





Option 3 is selected by the user in order to see the occupied rooms.

However, there are another 3 function that can be use by the user.

Error Handling

If user enter incorrect input, the program will tell the user to double check their input and the program will be back to main interface.

I. View Occupied Rooms

```
Options:
1. View Occupied Rooms
2. Search Room
3. Edit Room Details
4. Back to Main Menu

Enter your choice (1-4): 1

Occupied Rooms:
-------------------------
Room no.     Name    Phone
-------------------------
103      John Wahid      123456789
101      ewe     23421242
102      Haaxs   21431232
```

It shows the list rooms that are already booked, in a format of room number, the client's name as well as the client's phone number.

II. Search Room

```
Options:
1. View Occupied Rooms
2. Search Room
3. Edit Room Details
4. Back to Main Menu

Enter your choice (1-4): 2
Enter room number to search: 103

Room No: 103
Name: John Wahid
Address: 123 Street Fight
Phone: 123456789
Check-in Date: 2023-05-29
Room Type: 1
Room Service: 40
```

The search function will only show the room details of the room number entered in the field. All about customer information will be displayed including check-in date.

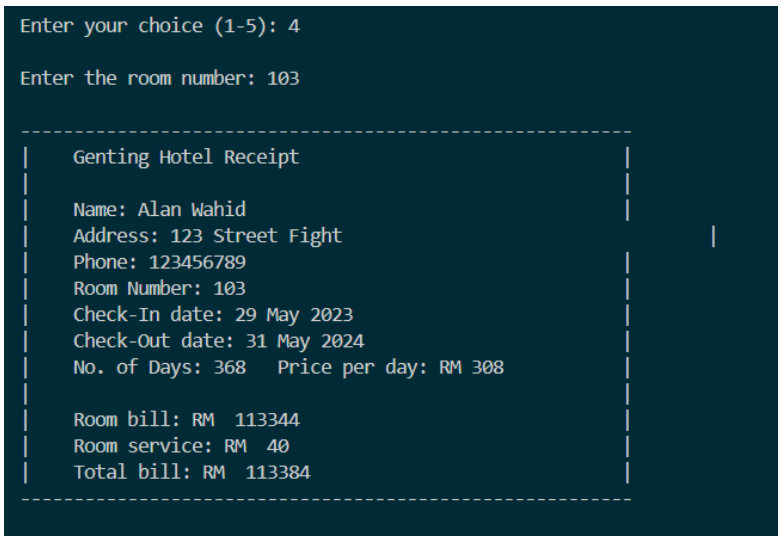
III. Edit Room Details

```
Options:
1. View Occupied Rooms
2. Search Room
3. Edit Room Details
4. Back to Main Menu

Enter your choice (1-4): 3
Enter room number to edit: 103

Editing Room Details (leave blank to keep current value):
Name (John Wahid): Alan Wahid
Address (123 Street Fight):
Phone (123456789):

~ ~ ~ ~ ~ Room 103 details updated. ~ ~ ~ ~ ~
```

The edit function requires the user to enter the room number. If the room number is already occupied, the system will proceed to edit the form page. But if the room is not occupied the system will notify a message. After completing the update, a notification will alert the user to say that the update was successful.

Then, lastly you can return to the main menu by pressing number 4.

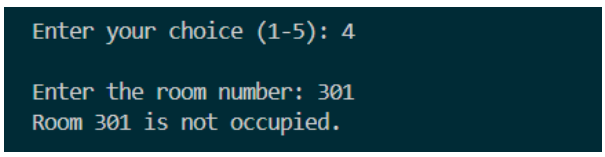
5. Check Out

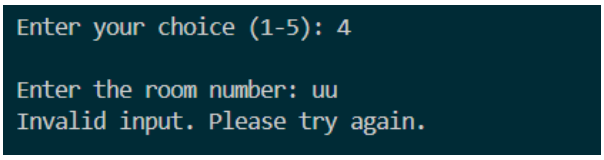


The user enters '4' to check out a client from room 103.

It also produces an itemised receipt in the check-out process that contains the client's information, room prices, room service prices, as well as the total amount charged.

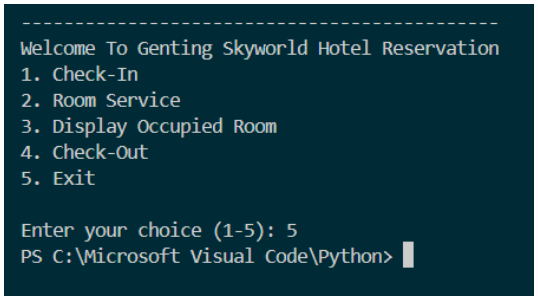
Error Handling





If user enter incorrect input, the program will tell the user to double check their input and the program will be back to main interface.

6. Exit



Last but not the least, the user chooses to exit the application by entering option 5. Thus, the program will terminate and terminate the process.