Seamless Guest Experience at Our Hotel Reception

Explore the heart of our hotel's hospitality and efficiency.

Software Design Description (SDD) for Hotel Management System (HMS)

1- Frontispiece

Date of Issue and Status

- **Date of Issue**: June 22, 2024
- **Status**: Final

Issuing Organization

- **Organization**: FATIMA JINNAH WOMEN UNIVRSITY

Authorship

1. Syeda Farwa Batool (2022-BSE-071)
2. Urooj fatima(2022-BSE-074)
3. Zainab (2022-BSE-076)

Change History:

Date	Version	Description	Author
2024-05-10	1.0	Initial creation	Zainab
2024-06-20	1.1	Final version for submission	Syeda farwa batool , urooj fatima
2024-06-20	1.2	Final review and corrections	Syeda farwa batool , urooj fatima

2- Introduction

2.1 Purpose:

The purpose of this project is to develop a hotel management system that automates the process of managing room reservations, customer check-ins, and check-outs. This system will help hotel staff manage rooms efficiently and keep track of customer information and room availability.

2.2 Scope:

The system covers the following functionalities:

- Room management: Adding, searching, and deleting rooms.
- Customer management: Checking in, checking out, and searching for customers.
- Reporting: Generating guest summary reports.
- Room availability: Displaying available rooms.

2.3 Context:

The Hotel Management System (HMS) automates hotel operations including room reservations, check-ins, and guest management. It employs object-oriented programming for scalability and maintenance ease, featuring classes like RoomBase and Customer. The system offers a user-friendly console interface for varied staff expertise, supporting tasks such as room management, booking, and reporting. Robust error handling and concurrency management ensure reliability and security. Overall, the HMS enhances hotel efficiency, improves guest satisfaction, and streamlines operational workflows.

2.4 Summary:

The hotel management system is designed to simplify the process of managing hotel operations. The system's architecture includes classes for rooms and customers, along with functionalities for checking in and out of rooms, managing room availability, and generating reports.

3- References

- IEEE Std 1016™-2009 for Software Design Descriptions.
- C++ Standard Library Documentation.

4- Glossary

- **RoomBase**: Abstract base class for room operations.
- **Customer**: Class representing a hotel customer.
- **Room**: Class representing a hotel room, inherits from RoomBase.
- **HotelMgnt**: Class representing hotel management functionalities, inherits from Room.
- **Status**: Indicates whether a room is available or reserved.
- **Booking ID**: Unique identifier for a customer's booking.
- **Rent**: The daily rent of a room.
- **Check-in**: Process of registering a customer into a room.
- **Check-out**: Process of a customer leaving the room and settling any dues.

5- Body

5.1 Identified Stakeholders and Design Concerns:

Stakeholders:

- **Hotel Staff:** Primary users of the system, responsible for managing room reservations and customer information.
- **Hotel Management:** Interested in reports and overall room utilization data.
- **Software Development Team:** Responsible for implementing and maintaining the system.
- **Course Instructor:** Evaluates the project based on design principles and implementation.

Design Concerns:

- **Usability:** The system should be easy to use for hotel staff with minimal training.
- **Reliability:** The system should reliably handle room reservations and prevent double booking.
- **Performance:** The system should perform efficiently, handling multiple operations without significant delays.
- **Maintainability:** The code should be modular and well-documented to facilitate future maintenance and updates.

5.2 System Architecture:

The system architecture for the hotel management system is designed to follow a modular approach, ensuring separation of concerns and ease of maintenance. The key components are:

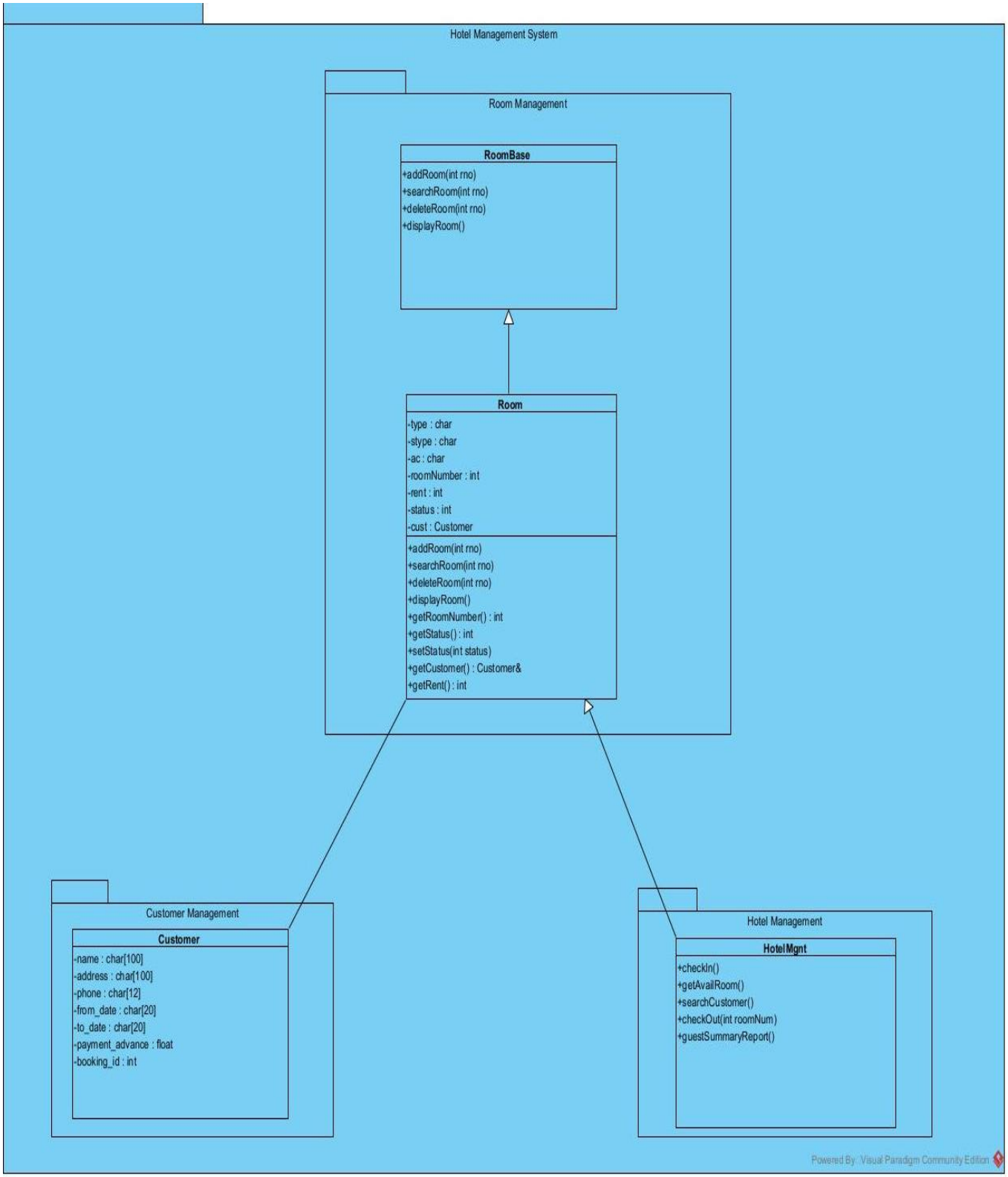
1. **RoomBase**: An abstract base class defining the interface for room operations.
2. **Customer**: A class representing hotel customers, storing details like name, address, phone number, booking dates, and payment information.
3. **Room**: A class inheriting from RoomBase, representing hotel rooms and implementing room operations such as adding, searching, and displaying room details.
4. **HotelMgnt**: A class inheriting from Room, encapsulating the core functionalities for managing hotel operations, such as check-in, check-out, and generating guest summary reports.

5.3 Design Viewpoints and Views

5.3.1 Architectural Viewpoint

Design View 1: System Architecture

Diagram: Package Diagram

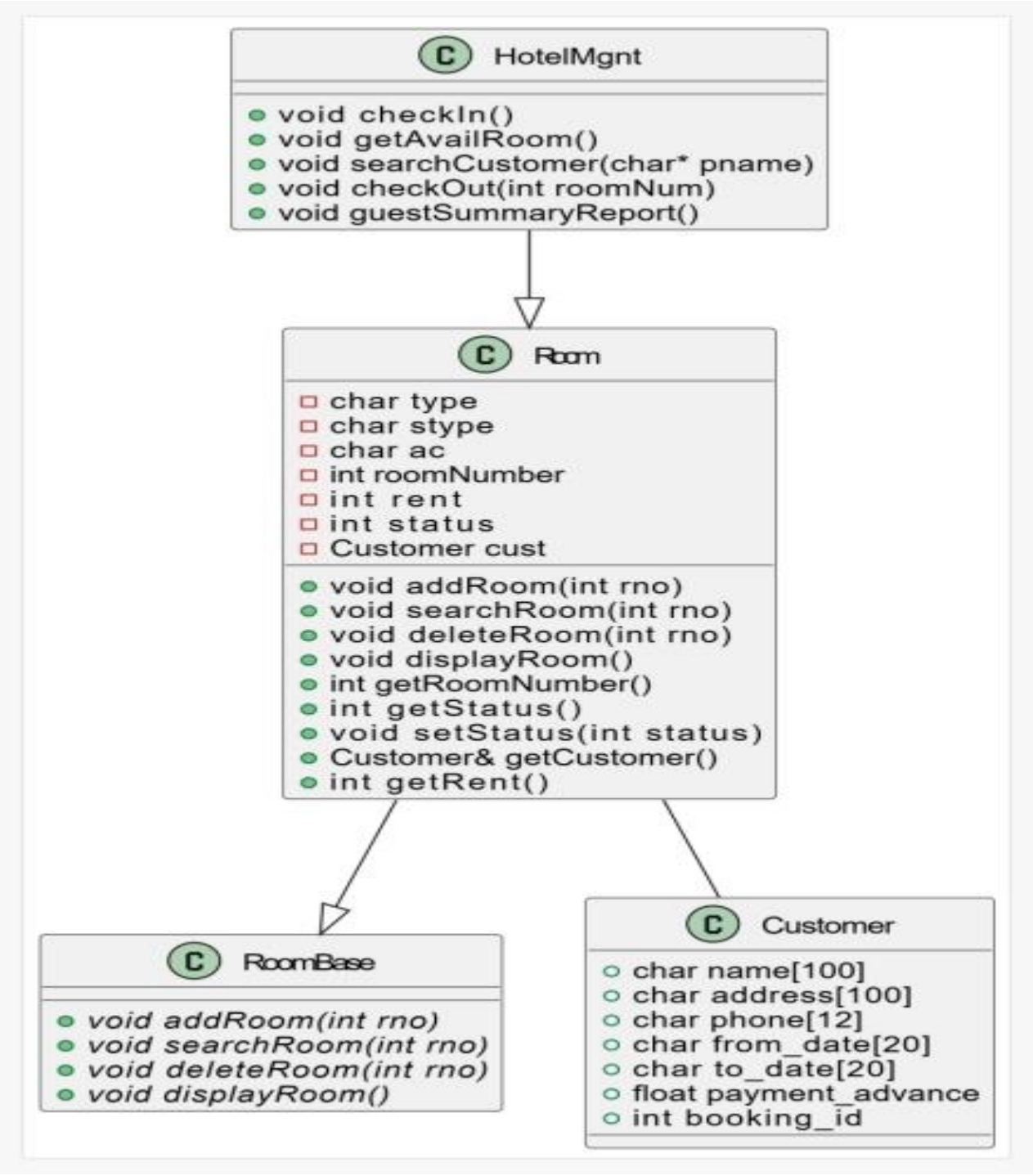


Description: The package diagram illustrates the high-level architecture of the hotel management system. It includes the Room Management, Customer Management, and Hotel Management packages, each containing relevant classes. The Room class inherits from the abstract base class RoomBase and contains an instance of the Customer class. The HotelMgnt class extends the Room class to incorporate hotel management functionalities

5.3.2 Structural Viewpoint

Design View 2: Class Diagram

- **Diagram: Class Diagram**

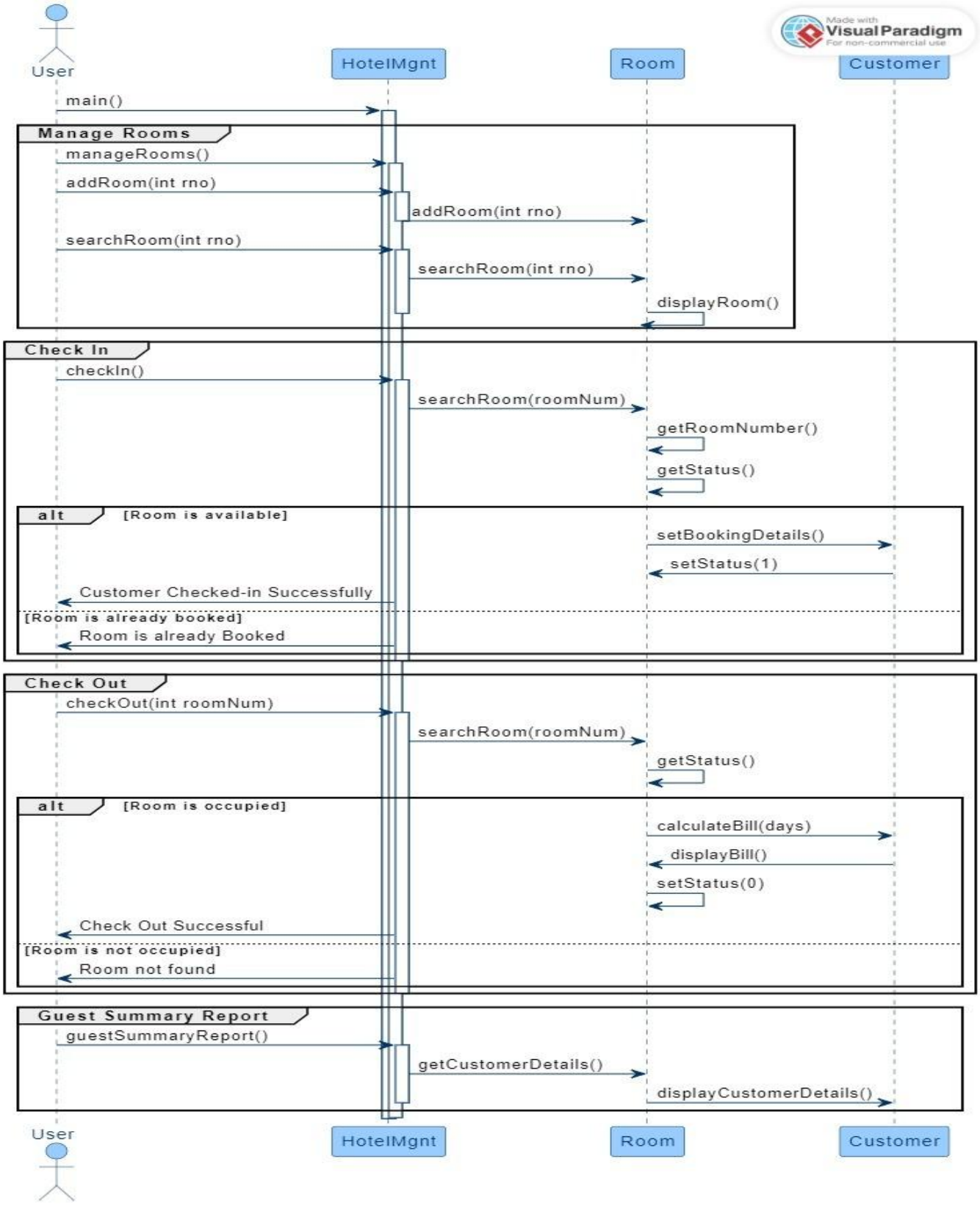


Description: The class diagram details the structure of the hotel management system, showing the relationships between classes. RoomBase is an abstract base class with virtual functions for room operations. The Room class extends RoomBase, implementing these operations and including attributes like type, stype, ac, roomNumber, rent, status, and a Customer instance. The HotelMgnt class extends Room to provide hotel management functionalities such as check-in, check-out, and room availability.

5.3.3 Behavioral Viewpoint

Design View 3: Sequence Diagram

- **Diagram: Sequence Diagram**



Description: The sequence diagram illustrates the interactions between the staff and the hotel management system for key operations such as check-in, check-out, getting available rooms, and searching for a customer. It shows how the HotelMgnt class interacts with Room and Customer classes to perform these operations.

5.4 Design Rationale:

The design of the hotel management system is driven by the need for modularity, maintainability, and clear separation of concerns. The use of an abstract base class (RoomBase) ensures that room-related operations are defined consistently, while the Room class implements these operations and manages room-specific data. The HotelMgnt class extends the Room class to encapsulate hotel management functions, keeping the responsibilities well-defined and manageable.

6- Dynamic Aspects

The dynamic aspects of the system can be illustrated by the sequence of interactions for various operations:

Check-In Process:

- The staff initiates the check-in process.
- The system verifies the room number and status.
- If available, the system updates the room status and records customer details.

Check-Out Process:

- The staff initiates the check-out process.
- The system verifies the room number and status.
- The system calculates the total bill and updates the room status.

Getting Available Rooms:

- The staff requests a list of available rooms.
- The system iterates through the rooms and displays those that are available.

Searching for a Customer:

- The staff searches for a customer by name.
- The system checks each room's status and retrieves the customer details if found.

7- Code implementation

```
 #include<iostream>
#include<string.h>
#include<fstream>
#define MAX 100
using namespace std;

// Abstract base class for Room
class RoomBase {
public:
    virtual void addRoom(int rno) = 0;
```

```
    virtual void searchRoom(int rno) = 0;
    virtual void deleteRoom(int rno) = 0;
    virtual void displayRoom() = 0;
};

// Class Customer
class Customer {
public:
    char name[100];
    char address[100];
    char phone[12];
    char from_date[20];
    char to_date[20];
    float payment_advance;
    int booking_id;
};

// Class Room
class Room : public RoomBase {
private:
    char type;
    char stype;
    char ac;
    int roomNumber;
    int rent;
    int status;
    Customer cust;

public:
    void addRoom(int rno) override;
    void searchRoom(int rno) override;
    void deleteRoom(int rno) override;
    void displayRoom() override;

    // Getter and setter methods
    int getRoomNumber()
            { return roomNumber; }
    int getStatus()
             { return status; }
    void setStatus(int status)
            { this->status = status; }
    Customer& getCustomer()
            { return cust; }
    int getRent()
            { return rent; } // Added getter for rent
};

// Global Declarations
Room rooms[MAX];
int count = 0;

void Room::addRoom(int rno) {
                    fstream f;
           f.open("E:AddedRooms_file.txt",ios::app);
if(f.is_open())
```

```
{

   roomNumber = rno;
   cout << "\nType AC/Non-AC (A/N) : ";
   cin >> ac;
   cout << "\nType Comfort (S/N) : ";
   cin >> type;
   cout << "\nType Size (B/S) : ";
   cin >> stype;
   cout << "\nDaily Rent : ";
   cin >> rent;
   f <<ac<<"\t"<<type<<"\t"<<stype<<"\t"<<rent<<"\t"<<rno<<"\t"<<"\n";
   status = 0;

   cout << "\nRoom Added Successfully!";
   }
else
{
cout<<"File is not found";}
f.close();

}

void Room::searchRoom(int rno) {
   int found = 0;
   for (int i = 0; i < count; i++) {
      if (rooms[i].getRoomNumber() == rno) {
         found = 1;
         cout << "Room Details\n";
         if (rooms[i].getStatus() == 1) {
            cout << "\nRoom is Reserved";
         } else {
            cout << "\nRoom is available";
         }
         rooms[i].displayRoom();
         break;
      }
   }
   if (!found) {
      cout << "\nRoom not found";
   }
}

void Room::displayRoom() {
   cout << "\nRoom Number: \t" << roomNumber;
   cout << "\nType AC/Non-AC (A/N): " << ac;
   cout << "\nType Comfort (S/N): " << type;
   cout << "\nType Size (B/S): " << stype;
   cout << "\nRent: " << rent;
}

void Room::deleteRoom(int rno) {
   // Implement deletion logic if needed
}
```

```
// Hotel management class
class HotelMgnt : public Room {
public:
   void checkIn();
   void getAvailRoom();
   void searchCustomer(char* pname);
   void checkOut(int roomNum);
   void guestSummaryReport();
};

void HotelMgnt::guestSummaryReport() {
   if (count == 0) {
      cout << "\n No Guest in Hotel !!";
   }
   for (int i = 0; i < count; i++) {
      if (rooms[i].getStatus() == 1) {
         cout << "\nCustomer First Name: " << rooms[i].getCustomer().name;
         cout << "\nRoom Number: " << rooms[i].getRoomNumber();
         cout << "\nAddress (only city): " << rooms[i].getCustomer().address;
         cout << "\nPhone: " << rooms[i].getCustomer().phone;
         cout << "\n---------------------------------------";
      }
   }
}

void HotelMgnt::checkIn() {
   int rno, found = 0;

   cout << "\nEnter Room number: ";
   cin >> rno;
   for (int i = 0; i < count; i++) {
      if (rooms[i].getRoomNumber() == rno) {
         found = 1;
         if (rooms[i].getStatus() == 1) {
            cout << "\nRoom is already Booked";
            return;
         }

         cout << "\nEnter booking id: ";
         cin >> rooms[i].getCustomer().booking_id;

         cout << "\nEnter Customer Name (First Name): ";
         cin >> rooms[i].getCustomer().name;

         cout << "\nEnter Address (only city): ";
         cin >> rooms[i].getCustomer().address;

         cout << "\nEnter Phone: ";
         cin >> rooms[i].getCustomer().phone;

         cout << "\nEnter From Date: ";
         cin >> rooms[i].getCustomer().from_date;

         cout << "\nEnter to Date: ";
         cin >> rooms[i].getCustomer().to_date;
```

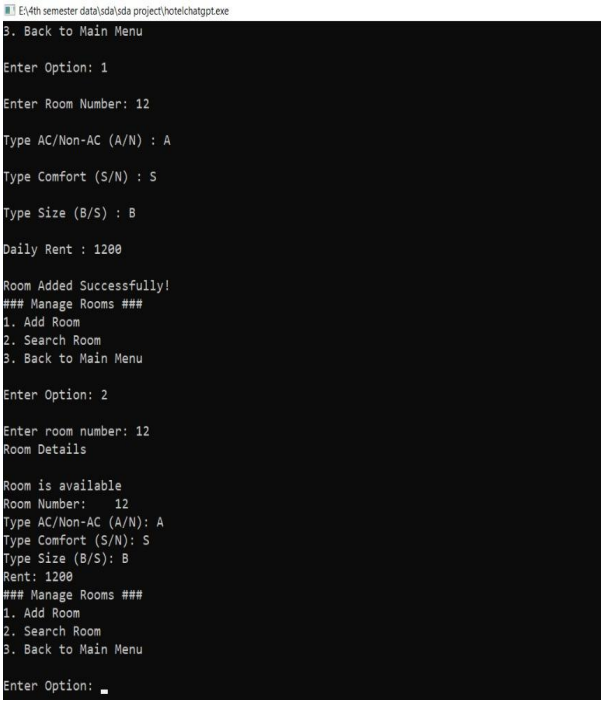
```
            cout << "\nEnter Advance Payment: ";
            cin >> rooms[i].getCustomer().payment_advance;

            rooms[i].setStatus(1);

            cout << "\nCustomer Checked-in Successfully..";
            break;
        }
    }
    if (!found) {
        cout << "\nRoom not found";
    }
}

void HotelMgnt::getAvailRoom() {
    int found = 0;
    for (int i = 0; i < count; i++) {
        if (rooms[i].getStatus() == 0) {
            rooms[i].displayRoom();
            cout << "\n\nPress enter for next room";
            found = 1;
        }
    }
    if (found == 0) {
        cout << "\nAll rooms are reserved";
    }
}

void HotelMgnt::searchCustomer(char* pname) {
    int found = 0;
    for (int i = 0; i < count; i++) {
        if (rooms[i].getStatus() == 1 && strcmp(rooms[i].getCustomer().name, pname) == 0) {
            cout << "\nCustomer Name: " << rooms[i].getCustomer().name;
            cout << "\nRoom Number: " << rooms[i].getRoomNumber();

            cout << "\n\nPress enter for next record\n";
            found = 1;
        }
    }
    if (found == 0) {
        cout << "\nPerson not found.\n";
    }
}

void HotelMgnt::checkOut(int roomNum) {
    int found = 0, days;
    float billAmount = 0;
    for (int i = 0; i < count; i++) {
        if (rooms[i].getStatus() == 1 && rooms[i].getRoomNumber() == roomNum) {
            found = 1;
            cout << "\nEnter Number of Days:\t";
            cin >> days;
            billAmount = days * rooms[i].getRent();
```

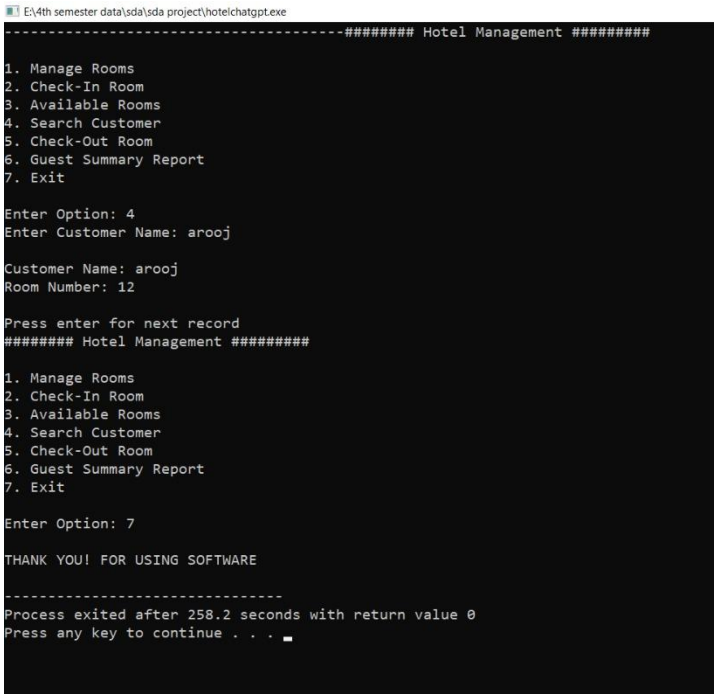
```
            cout << "\n\t######## CheckOut Details ########\n";
            cout << "\nCustomer Name : " << rooms[i].getCustomer().name;
            cout << "\nRoom Number : " << rooms[i].getRoomNumber();
            cout << "\nAddress : " << rooms[i].getCustomer().address;
            cout << "\nPhone : " << rooms[i].getCustomer().phone;
            cout << "\nTotal Amount Due : " << billAmount << " /";
            cout << "\nAdvance Paid: " << rooms[i].getCustomer().payment_advance << " /";
            cout << "\n*** Total Payable: " << billAmount - rooms[i].getCustomer().payment_advance << "/
only";

            rooms[i].setStatus(0);
            break;
        }
    }
    if (!found) {
        cout << "\nRoom not found";
    }
}

void manageRooms() {
    Room room;
    int opt, rno, flag = 0;

    do {
        cout << "\n### Manage Rooms ###";
        cout << "\n1. Add Room";
        cout << "\n2. Search Room";
        cout << "\n3. Back to Main Menu";
        cout << "\n\nEnter Option: ";
        cin >> opt;

        switch (opt) {
        case 1:
            cout << "\nEnter Room Number: ";
            cin >> rno;
            for (int i = 0; i < count; i++) {
                if (rooms[i].getRoomNumber() == rno) {
                    flag = 1;
                }
            }
            if (flag == 1) {
                cout << "\nRoom Number is Present.\nPlease enter unique Number";
                flag = 0;
            } else {
                rooms[count].addRoom(rno);
                count++;
            }
            break;
        case 2:
            cout << "\nEnter room number: ";
            cin >> rno;
            room.searchRoom(rno);
            break;
        case 3:
            // nothing to do
```

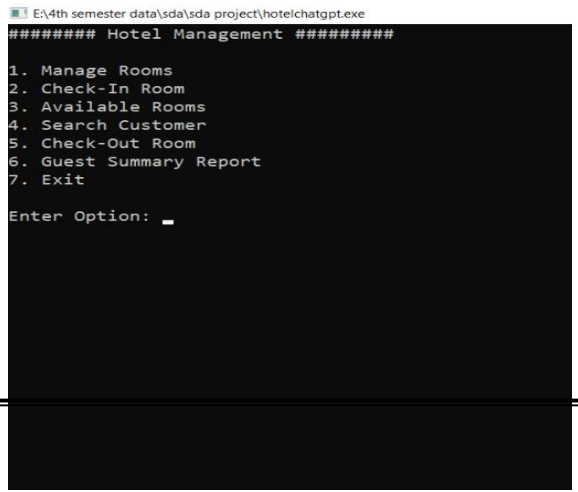
```
            break;
        default:
            cout << "\nPlease Enter correct option";
            break;
        }
    } while (opt != 3);
}

int main() {
    HotelMgnt hm;
    int opt, rno;
    char pname[100];

    do {
        cout << "######## Hotel Management #########\n";
        cout << "\n1. Manage Rooms";
        cout << "\n2. Check-In Room";
        cout << "\n3. Available Rooms";
        cout << "\n4. Search Customer";
        cout << "\n5. Check-Out Room";
        cout << "\n6. Guest Summary Report";
        cout << "\n7. Exit";
        cout << "\n\nEnter Option: ";
        cin >> opt;

        switch (opt) {
        case 1:
            manageRooms();
            break;
        case 2:
            if (count == 0) {
                cout << "\nRooms data is not available.\nPlease add the rooms first.";
            } else {
                hm.checkIn();
            }
            break;
        case 3:
            if (count == 0) {
                cout << "\nRooms data is not available.\nPlease add the rooms first.";
            } else {
                hm.getAvailRoom();
            }
            break;
        case 4:
            if (count == 0) {
                cout << "\nRooms are not available.\nPlease add the rooms first.";
            } else {
                cout << "Enter Customer Name: ";
                cin >> pname;
                hm.searchCustomer(pname);
            }
            break;
        case 5:
            if (count == 0) {
                cout << "\nRooms are not available.\nPlease add the rooms first.";
```

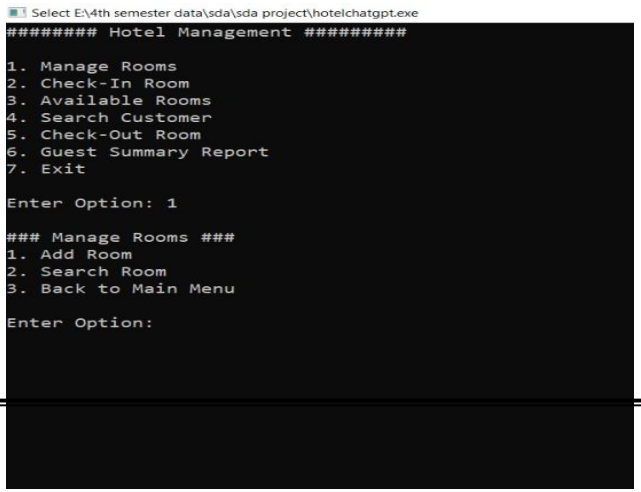
```
        } else {
            cout << "Enter Room Number : ";
            cin >> rno;
            hm.checkOut(rno);
        }
        break;
    case 6:
        hm.guestSummaryReport();
        break;
    case 7:
        cout << "\nTHANK YOU! FOR USING SOFTWARE\n";
        break;
    default:
        cout << "\nPlease Enter correct option";
        break;
    }
  } while (opt != 7);

  return 0;
}
```

OUTPUT :









E:\4th semester data\sda\sda project\hotelchatgpt.exe

```
######### Hotel Management ##########

1. Manage Rooms
2. Check-In Room
3. Available Rooms
4. Search Customer
5. Check-Out Room
6. Guest Summary Report
7. Exit

Enter Option: 1

### Manage Rooms ###
1. Add Room
2. Search Room
3. Back to Main Menu

Enter Option: 1

Enter Room Number: 12

Type AC/Non-AC (A/N) : A

Type Comfort (S/N) : S

Type Size (B/S) : B

Daily Rent : 1200

Room Added Successfully!
### Manage Rooms ###
1. Add Room
2. Search Room
3. Back to Main Menu

Enter Option:
```

E:\4th semester data\sda\sda project\hotelchatgpt.exe

```
### Manage Rooms ###
1. Add Room
2. Search Room
3. Back to Main Menu

Enter Option: 3
######## Hotel Management #########

1. Manage Rooms
2. Check-In Room
3. Available Rooms
4. Search Customer
5. Check-Out Room
6. Guest Summary Report
7. Exit

Enter Option: 3

Room Number:    12
Type AC/Non-AC (A/N): A
Type Comfort (S/N): S
Type Size (B/S): B
Rent: 1200

Press enter for next room######### Hotel Management #########

1. Manage Rooms
2. Check-In Room
3. Available Rooms
4. Search Customer
5. Check-Out Room
6. Guest Summary Report
7. Exit

Enter Option:
```

E:\4th semester data\sda\sda project\hotelchatgpt.exe

```
Enter Option: 3

Room Number:    12
Type AC/Non-AC (A/N): A
Type Comfort (S/N): S
Type Size (B/S): B
Rent: 1200

Press enter for next room######## Hotel Management ########

1. Manage Rooms
2. Check-In Room
3. Available Rooms
4. Search Customer
5. Check-Out Room
6. Guest Summary Report
7. Exit

Enter Opti
```

AddedRooms_file.txt - Notepad

File Edit Format View Help

```
A       S       B       1200    12
```

```
3. Available Rooms
4. Search Customer
5. Check-Out Room
6. Guest Summary Report
7. Exit

Enter Option: 3

All rooms are reserved######## Hotel Management #########

1. Manage Rooms
2. Check-In Room
3. Available Rooms
4. Search Customer
5. Check-Out Room
6. Guest Summary Report
7. Exit
```

```
me: arooj

y): harley

----------######## Hotel Management #########

Report
```

Type here to search

8- Conclusion

The hotel management system designed and implemented in this project demonstrates the application of software design principles and UML modeling. The system architecture ensures modularity and maintainability, while the class and sequence diagrams provide clear structural and behavioral views of the system. The C++ implementation aligns with the design, showcasing the practical application of the design principles. The project fulfills the requirements of the Software Design and Architecture course and provides a functional system for managing hotel operations.