PROJECT REPORT

“HOTEL MANAGEMENT SYSTEM (HMS)”

Course Code & Name:
CSC 284 - Programming In C++ Lab

Instructor:
ASM Shakil Ahamed
Lecturer
CSE - IUBAT

Developers:
Md. Mahedi Zaman Zaber
Student ID: 23203134

Shanjida Afrin
Student ID: 23103290

Sumayia Akter
Student ID: 22303423

Sumiaya Afrin
Student ID: 23203130

Afsana Meem
Student ID: 23203063

Department of Computer Science and Engineering
College of Engineering and Technology
IUBAT – International University of Business Agriculture and Technology

Spring 2024

Letter of Transmittal

20 May 2024
The Course Instructor
ASM Shakil Ahamed
Lecturer
Department of Computer Science and Engineering
IUBAT–International University of Business Agriculture and Technology
4 Embankment Drive Road, Sector 10, Uttara Model Town
Dhaka 1230, Bangladesh

Subject: Letter of Transmittal

Dear Sir,

I am pleased to submit the final project report for our Hotel Management System, developed as part of the Programming in C++ Lab (CSC 284) at IUBAT - International University of Business Agriculture and Technology. This project, titled "Hotel Management System (HMS)," represents a comprehensive solution designed to facilitate efficient management of hotel operations.

The enclosed report contains a detailed account of our project's design and implementation. It includes descriptions of the system architecture, class diagrams, data flow diagrams, user interface design, and core functionalities such as room and customer management, booking systems, and super admin features.

This project was developed by Mahedi Zaman Zaber, Afsana Meem, Shanjida Afrin, Sumayia Akter, and Sumiaya Afrin under the guidance of our esteemed faculty member, ASM Shakil Ahamed. We hope that this report effectively showcases the practical application of the skills and knowledge we have acquired during this course.

We sincerely appreciate the support and guidance provided throughout the development of this project. We look forward to any feedback or suggestions you may have.

Thank you for your time and consideration.

Yours Obediently,

Md. Mahedi Zaman Zaber
Email: mahedizaber51@gmail.com
Student ID: 23203134, Phone: 01302543112

Abstract

The Hotel Management System (HMS) project is a comprehensive solution designed to streamline and automate the management of hotel operations. Developed as part of the Programming in C++ Lab (CSC 284) at IUBAT, this system caters to the needs of hotel administrators by offering strong functionalities for room management, customer management, and booking processes.

Key features of the HMS include the ability for multiple admins to manage rooms, book rooms for customers, edit room prices, and vacate rooms upon customer departure. A unique aspect of the system is the super admin role, which has exclusive privileges to add new admins, ensuring a secure and hierarchical management structure.

The system is built using C++ and employs a user-friendly interface that guides admins through various operations efficiently. It includes class-based design principles with distinct classes for Room, Customer, Hotel, and Admin, each encapsulating relevant attributes and methods to perform specific tasks.

Testing of the HMS demonstrates its capability to handle typical hotel management scenarios, ensuring reliability and user satisfaction. The system's design and implementation are documented thoroughly in this report, providing insights into its architecture, data flow, and user interface.

Overall, the Hotel Management System is a practical and effective tool for modern hotel management, combining theoretical knowledge and practical application to deliver a solution that meets the demands of the industry.

Table of Contents

Table of Contents **1**
1. Introduction **3**
1.1 Project Overview 3
1.2 Purpose and Scope 3
1.3 Technologies Used 3
1.4 Development Team 4
2. System Requirements **5**
2.1 Hardware Requirements 5
2.2 Software Requirements 5
3. System Design **6**
3.1 System Architecture 6
3.2 Class Diagrams 6
3.3 Data Flow Diagrams 7
4. Implementation **9**
4.1 Class Descriptions 9
4.1.1 Room Class 9
4.1.2 Customer Class 10
4.1.3 Hotel Class 10
4.1.4 Admin Class 11
4.2 Core Functionalities 12
4.2.1 Room Management 12
4.2.2 Customer Management 12
4.2.3 Booking System 12
4.2.4 Super Admin Features 12
4.3 Code Listings 12
4.3.01 Room Class Implementation 12
4.3.02 Customer Class Implementation 14
4.3.03 Customer Related Functions 15
4.3.04 Hotel Class Implementation 16
4.3.05 Admin Class Implementation 19
4.3.06 Admin Related Functions 20
4.3.07 Process Continue Function 21
4.3.08 Login Screen Implementation 21
4.3.09 Main Menu Function 21
4.3.10 Main Function 25
5. User Interface **28**
5.1 Welcome Screen 28
5.2 First Time Setup Screen 28
5.1 Login Screen 29
5.2 Admin Panel 30
5.3 Super Admin Panel 31

5.4 Admin Panel Option - All Rooms..... 32
5.5 Admin Panel Option - All Customers.....32
5.08 Admin Panel Option - Room Details.....33
5.09 Admin Panel Option - Edit Room Price..... 34
5.10 Admin Panel Option - Book A Room..... 35
5.11 Admin Panel Option - Vacate A Room..... 36
5.12 Admin Panel Option - Add New Admin..... 37
5.13 Admin Panel Option - Logout..... 38
6. Conclusion.....39
6.1 Summary of Achievements..... 39
6.2 Future Enhancements.....39
7. Appendices.....40
7.1 Full Code Listings.....40
8.3 References.....69

1. Introduction

1.1 Project Overview

The Hotel Management System (HMS) is designed to provide a complete and efficient solution for managing a hotel. The system supports multiple administrative roles, including a super admin who has exclusive rights to add new admins. All admins are equipped to manage rooms, book them for customers, check room statuses, edit room prices, add new customers, and vacate rooms when necessary.

The goal of the HMS is to streamline hotel operations, improve customer service, and enhance overall efficiency.

1.2 Purpose and Scope

The primary purpose of the Hotel Management System is to automate and simplify the various tasks involved in hotel management. This includes:

- **Room Management:** Administrators can manage room availability, prices, and details.
- **Booking System:** The system allows for quick and efficient booking of rooms for customers.
- **Customer Management:** Details of customers are stored and managed for better service.
- **Admin Management:** Super admins can add and manage multiple admins, ensuring robust management of hotel operations.

The scope of the project includes the development of a user-friendly interface for admins, a secure login system, and comprehensive room and customer management functionalities.

1.3 Technologies Used

The HMS is developed using C++, using its object-oriented features to create a modular and maintainable codebase.

The ANSI color codes are utilized to enhance the user interface in the console environment, providing a more engaging and better experience for the users.

1.4 Development Team

This Hotel Management System is developed by a group of dedicated students from the International University of Business Agriculture and Technology (IUBAT) as a project for the Programming in C++ Lab (CSC 284) course. The team members include:

- **Name:** Mahedi Zaman Zaber
  ID: 23203134
- **Name:** Shanjida Afrin
  ID: 23103290
- **Name:** Sumayia Akter
  ID: 22303423
- **Name:** Sumiaya Afrin
  ID: 23203130
- **Name:** Afsana Meem
  ID: 23203063

Under the guidance of our esteemed faculty member, **ASM Shakil Ahamed**, the group has worked collaboratively to deliver a strong and functional system that meets the needs of modern hotel management.

2. System Requirements

2.1 Hardware Requirements

To run the Hotel Management System efficiently, the following hardware requirements are recommended:

- **Processor:** Intel Core i3 or equivalent
- **RAM:** 2 GB or more
- **Hard Disk:** Minimum 100 MB of free space
- **Display:** 1024x768 resolution or higher
- **Keyboard and Mouse:** Standard

2.2 Software Requirements

The following software components are required to develop, compile, and run the Hotel Management System:

- **Operating System:** Windows 7 or higher, Linux, or macOS
- **Compiler:** GCC (GNU Compiler Collection) for Linux/macOS or MinGW for Windows
- **IDE (Integrated Development Environment):** Code::Blocks, Visual Studio, or any other C++ compatible IDE
- **Libraries:** Standard C++ libraries

3. System Design

3.1 System Architecture

The system architecture of the Hotel Management System is based on a layered approach, ensuring modularity and separation of concerns.

The key components of the architecture include:

1. **User Interface Layer:** Handles the interaction between the user and the system. It includes login screens, admin panels, and interfaces for room and customer management.

2. **Application Logic Layer:** Contains the core functionalities of the system, including room booking, customer management, and admin management.

3. **Data Layer:** Manages the data storage and retrieval processes. In this system, data is managed in memory using appropriate data structures.

3.2 Class Diagrams

The class diagram represents the structure of the system by showing the system's classes, their attributes, methods, and the relationships among objects.

The key classes in the Hotel Management System include:

1. **Room Class:**

    - **Attributes:**
      roomNumber, isBooked, customerID, roomPrice, roomFloor

    - **Methods:**
      getRoomNumber(), getCustomerID(), getPrice(), setPrice(), getIsBooked(), book(), vacate(), getDetails()

2. **Customer Class:**

    - **Attributes:**
      customerID, customerName, customerAddress, customerPhone, customerEmail

    - **Methods:**
      getName(), getID(), getDetails(), getDetailsFlat()

3. **Hotel Class:**

    - **Attributes:**
      hotelName, rooms, totalFloors, eachFloorRoomCount

    - **Methods:**
      isRoomAvailable(), bookRoom(), vacateRoom(), getRoomPrice(), checkRoom(), editRoom(), viewAllRooms()

4. **Admin Class:**

    - **Attributes:**
      username, password, superAdmin

    - **Methods:**
      getUsername(), getPassword(), isSuper()

3.3 Data Flow Diagrams

The data flow diagrams (DFDs) illustrate how data flows through the system. The key processes in the Hotel Management System and their data flows are:

1. **User Login:**

    - **Input:** Username, Password
    - **Process:** Authenticate user credentials
    - **Output:** Access granted/denied

2. **Room Management:**

    - **Input:** Room details, Customer details
    - **Process:** Check room availability, book room, vacate room, edit room details
    - **Output:** Updated room status, booking confirmation

3. **Customer Management:**

    - **Input:** Customer details
    - **Process:** Add new customer, view customer details
    - **Output:** Customer ID, customer list

4. **Admin Management (Super Admin Only):**

    - **Input:** Admin details
    - **Process:** Add new admin
    - **Output:** Updated admin list

4. Implementation

4.1 Class Descriptions

4.1.1 Room Class

The Room class models a hotel room with its attributes and operations. It handles room details, booking, and vacating operations.

Attributes:

- `roomNumber`: The room number.
- `isBooked`: A boolean indicating if the room is booked.
- `customerID`: The ID of the customer who booked the room.
- `roomPrice`: The price of the room.
- `roomFloor`: The floor number where the room is located.

Methods:

- `Room(string number, double price, int floor)`
  Constructor to initialize room details.

- `getRoomNumber()`
  Returns the room number.

- `getCustomerID()`
  Returns the customer ID if booked.

- `getPrice()`
  Returns the room price.

- `setPrice(double price)`
  Sets a new price for the room.

- `getIsBooked()`
  Returns booking status.

- `book(int customer)`
  Books the room for a customer.

- `vacate()`
  Vacates the room.

- `getDetails()`
  Prints the room details.

4.1.2 Customer Class

The Customer class models a customer with their personal details and operations.

Attributes:

- `customerID`: The unique ID of the customer.
- `customerName`: The name of the customer.
- `customerAddress`: The address of the customer.
- `customerPhone`: The phone number of the customer.
- `customerEmail`: The email of the customer.

Methods:

- `Customer()`
  Constructor to initialize and register customer details.

- `getName()`
  Returns the customer name.

- `getID()`
  Returns the customer ID.

- `getDetails()`
  Prints the customer details.

- `getDetailsFlat()`
  Prints the customer details in a flat format for listing.

4.1.3 Hotel Class

The Hotel class models a hotel and manages its rooms and operations.

Attributes:

- `hotelName`: The name of the hotel.
- `rooms`: A vector containing all the rooms in the hotel.
- `totalFloors`: Total number of floors in the hotel.
- `eachFloorRoomCount`: Number of rooms on each floor.

Methods:

- `Hotel(string name, int floors, int roomEach, double price)`
  Constructor to initialize hotel details and rooms.

- `isRoomAvailable(string roomNumber)`
  Checks if a room is available.

- `bookRoom(string roomNumber, Customer& customer)`
  Books a room for a customer.

- `vacateRoom(string roomNumber)`
  Vacates a room.

- `getRoomPrice(string roomNumber)`
  Gets the price of a room.

- `checkRoom(string roomNumber)`
  Prints room details and returns customer ID if booked.

- `editRoom(string roomNumber)`
  Edits the room price.

- `viewAllRooms()`
  Displays the details of all rooms in the hotel.

4.1.4 Admin Class

The Admin class models an admin user with authentication capabilities.

Attributes:

- `username`: The username of the admin.
- `password`: The password of the admin.
- `superAdmin`: A boolean indicating if the admin is a super admin.

Methods:

- `Admin(bool super = false)`
  Constructor to initialize admin details.

- `getUsername()`
  Returns the username.

- `getPassword()`
  Returns the password.

- `isSuper()`
  Returns if the admin is a super admin.

4.2 Core Functionalities

4.2.1 Room Management

Room management includes viewing all rooms, checking room details, editing room prices, and vacating rooms.

4.2.2 Customer Management

Customer management includes adding new customers and viewing customer details.

4.2.3 Booking System

The booking system handles room availability checks, booking rooms for customers, and listing booked rooms.

4.2.4 Super Admin Features

Super admin features include adding new admins and managing all admin activities.

4.3 Code Listings

4.3.01 Room Class Implementation

```
class Room
{
    string roomNumber;
    bool isBooked;
    int customerID;
    double roomPrice;
    int roomFloor;
public:
    // Constructor
    Room(string number,double price,int floor)
```

```
{
    roomNumber= number;
    isBooked = false;
    customerID = -1;
    roomPrice = price;
    roomFloor = floor;
}

string getRoomNumber()
{
    return roomNumber;
}

int getCustomerID()
{
    return customerID;
}
double getPrice()
{
    return roomPrice;
}
void setPrice(double price)
{
    roomPrice = price;
}

bool getIsBooked()
{
    return isBooked;
}

void book(int customer)
{
    isBooked = true;
    customerID = customer;
}

void vacate()
{
    isBooked = false;
    customerID = -1;
}
```

```
    void getDetails()
    {
        cout << "Room No: "<< roomNumber<<endl;
        cout << "Floor No: "<< roomFloor<<endl;
        cout << "Price ($): "<< roomPrice<<endl;
        cout << "Available: ";
        if(isBooked)
        {
            cout <<RED<< "No"<<RESET<<endl;
        }
        else
        {
            cout <<GREEN<< "Yes"<<RESET<<endl;
        }
    }
};
```

4.3.02 Customer Class Implementation

```
class Customer
{
    static int lastID;
    int customerID;
    string customerName;
    string customerAddress;
    string customerPhone;
    string customerEmail;
public:
    // Constructor
    Customer()
    {
        customerID = ++lastID;
        cout << "Customer ID: " << customerID <<endl;
        cin.ignore();
        cout << "Customer Name: ";
        getline(cin,customerName);
        cout << "Customer Address: ";
        getline(cin,customerAddress);
        cout << "Customer Phone: ";
        cin >> customerPhone;
        cout << "Customer Email: ";
```

```
        cin >> customerEmail;
    }

    string getName()
    {
        return customerName;
    }

    int getID()
    {
        return customerID;
    }
    void getDetails()
    {
        cout << "Customer ID: "<< customerID <<endl;
        cout << "Customer Name: "<< customerName<<endl;
        cout << "Customer Address: "<< customerAddress<<endl;
        cout << "Customer Phone: "<< customerPhone<<endl;
        cout << "Customer Email: "<< customerEmail<<endl;
    }

    // To Show as a List
    void getDetailsFlat()
    {
        cout <<CYAN<< "\n["<<(customerID < 10 ? "00" :
(customerID < 100 ? "0" : ""))<<customerID<< "] "<<RESET;
        cout <<BOLD<<"NAME\t: "<<RESET << customerName <<"\n
"<<BOLD<<"ADDRESS\t: "<<RESET<< customerAddress ;
        cout <<"\n      "<<BOLD<<"PHONE\t: "<<RESET<<
customerPhone<<"\n      "<<BOLD<<"EMAIL\t: "<<RESET<<
customerEmail;
        cout <<endl;
    }
};

// Static Value Declaration (Auto Increment ID)
int Customer::lastID = 0;
```

4.3.03 Customer Related Functions

```
// To get customer By ID
Customer* checkCustomer(vector<Customer>& customers, int id)
```

```
{
    for (int i =0;i < customers.size();i++)
    {
        if (customers[i].getID() == id)
        {
            customers[i].getDetails();
            return &customers[i];
        }
    }
    return nullptr;
}

// Get the list of Customers
void viewAllCustomers(vector<Customer>& customers)
{
    if(customers.size()==0){
        cout << "\nNo Registered Customers Yet!";
    } else {
        for (int i = 0; i < customers.size(); i++)
        {
            customers[i].getDetailsFlat();
        }
    }
}
```

4.3.04 Hotel Class Implementation

```
class Hotel
{
    string hotelName;
    vector<Room> rooms;
    int totalFloors;
    int eachFloorRoomCount;
public:
    // Constructor
    Hotel(string name,int floors, int roomEach,double price)
    {
        hotelName = name;
        totalFloors = floors;
        eachFloorRoomCount = roomEach;

        char floorChar = 'A';
```

```
        int floorExt = 1;
        for (int floor = 1; floor <= totalFloors; ++floor)
        {
            for(int roomNum = 1; roomNum <=
eachFloorRoomCount; ++roomNum)
            {
                // Room number format: Character Representing
Floor + Floor Extension + 00 or 0 + Room Number
                string roomNumber = floorChar +
to_string(floorExt)
                                    + (roomNum < 10 ? "00" :
(roomNum < 100 ? "0" : ""))
                                    + to_string(roomNum);


rooms.push_back(Room(roomNumber,price,floor));
            }
            // Reset To Character Representing Floor To 'A' if
it Exceeds 'Z'
            // Increase Floor Extension If Character
Representing Floor Exceeds 'Z'
            if (floorChar == 'Z')
            {
                floorChar = 'A';
                floorExt++;
            }
            else
            {
                floorChar++;
            }
        }
    }

    // Check if a room is available
    bool isRoomAvailable(string roomNumber)
    {
        for (int i =0;i < rooms.size();i++)
        {
            if (rooms[i].getRoomNumber() == roomNumber &&
!rooms[i].getIsBooked())
            {
                return true;
            }
```

```
        }
        return false;
    }

    // Book a room for a customer
    bool bookRoom(string roomNumber, Customer& customer)
    {
        for (int i =0;i < rooms.size();i++)
        {
            if (rooms[i].getRoomNumber() == roomNumber &&
!rooms[i].getIsBooked())
            {
                rooms[i].book(customer.getID());
                cout <<GREEN<< "Room " << roomNumber << "
Booked For " << customer.getName()<<RESET << endl;
                return true;
            }
        }
        cout <<RED<< "Room " << roomNumber << " Is Not
Available!"<<RESET << endl;
        return false;
    }

    // Vacate a room
    void vacateRoom(string roomNumber)
    {
        for (int i =0;i < rooms.size();i++)
        {
            if (rooms[i].getRoomNumber() == roomNumber)
            {
                if(rooms[i].getIsBooked()){
                    rooms[i].vacate();
                    cout <<GREEN<< "Room " << roomNumber << "
Vacated!"<<RESET << endl;
                } else {
                    cout <<GREEN<< "Room " << roomNumber << "
Is Available To Book!"<<RESET << endl;
                }

                return;
            }
        }
```

```
        cout <<RED<< "Room " << roomNumber << " Doesn\'t
Exist!"<<RESET << endl;
    }

    // Get Room Price
    double getRoomPrice(string roomNumber)
    {
        for (int i =0;i < rooms.size();i++)
        {
            if (rooms[i].getRoomNumber() == roomNumber)
            {
                return rooms[i].getPrice();
            }
        }
        cout <<RED<< "Room " << roomNumber << " Doesn\'t
Exist!"<<RESET << endl;
        return -1;
    }

    //get room info and return customer id if booked
    int checkRoom(string roomNumber)
    {
        for (int i =0;i < rooms.size();i++)
        {
            if (rooms[i].getRoomNumber() == roomNumber)
            {
                rooms[i].getDetails();
                return rooms[i].getCustomerID();
            }
        }
        cout <<RED<< "Room " << roomNumber << " Doesn\'t
Exist!"<<RESET << endl;
        return -1;
    }

    // Set Room Price
    bool editRoom(string roomNumber)
    {
        for (int i =0;i < rooms.size();i++)
        {
            if (rooms[i].getRoomNumber() == roomNumber)
            {
                double price;
```

```
                cout << "New Price ($):";
                cin >> price;
                rooms[i].setPrice(price);
                cout <<GREEN<< "Room " << roomNumber << "
Price Updated To " << price<<RESET << endl;
                return true;
            }
        }
        cout <<RED<< "Room " << roomNumber << " Doesn\'t
Exist!"<<RESET << endl;
        return false;
    }


    // Get The List of All Rooms
    void viewAllRooms()
    {
        char floorChar = 'A';
        int floorExt = 1;
        for (int floor = 1; floor <= totalFloors; ++floor)
        {
            cout << "FLOOR "<<(floor < 10 ? "00" : (floor <
100 ? "0" : "")) <<floor;
            cout << ":\n----------\n";
            for(int roomNum = 1; roomNum <=
eachFloorRoomCount; ++roomNum)
            {
                // Room number format: Character Representing
Floor + Floor Extension + 00 or 0 + Room Number
                string roomNumber = floorChar +
to_string(floorExt)
                                    + (roomNum < 10 ? "00" :
(roomNum < 100 ? "0" : ""))+ to_string(roomNum);
                if (isRoomAvailable(roomNumber))
                {
                    cout <<"["<<GREEN<< roomNumber << RESET<<
"] ";
                }
                else
                {
                    cout <<"["<< RED << roomNumber << RESET<<
"] ";
                }
```

```
            }
            cout <<endl<< endl;
            // Reset To Character Representing Floor To 'A' if
it Exceeds 'Z'
            // Increase Floor Extension If Character
Representing Floor Exceeds 'Z'
            if (floorChar == 'Z')
            {
                floorChar = 'A';
                floorExt++;
            }
            else
            {
                floorChar++;
            }
        }
        cout << endl;
    }
};
```

4.3.05 Admin Class Implementation

```
class Admin
{
    string username;
    string password;
    bool superAdmin;
public:
    // Constructor
    Admin(bool super = false)
    {
        cout << "Username: ";
        cin >> username;
        cout << "Password: ";
        cin >> password;
        superAdmin = super;
    }
    string getUsername()
    {
        return username;
    }
```

```
    string getPassword()
    {
        return password;
    }
    bool isSuper(){
        return superAdmin;
    }
};
```

4.3.06 Admin Related Functions

```
// Check If Admin Credentials Match
bool checkAdmin(vector<Admin>& admins, string username,
string password)
{
    for (int i =0;i < admins.size();i++)
    {
        if (admins[i].getUsername() == username &&
admins[i].getPassword() == password)
        {
            if(admins[i].isSuper()){
                ::superAdmin = true;
            } else {
                ::superAdmin = false;
            }
            return true;
        }
    }
    return false;
}
```

4.3.07 Process Continue Function

```
// Enter to Continue
void continueProcess()
{
    cin.ignore();
    cout << "\n\nPress Enter To Continue...";
    cin.get();
    system("cls");
}
```

4.3.08 Login Screen Implementation

```
// Login Screen
bool loginScreen(vector<Admin>& admins)
{
    string username,password;
    cout << "\n\nLogin To Panel\n=================\n";
    cout << "Username: ";
    cin >> username;
    cout << "Password: ";
    cin >> password;
    if(checkAdmin(admins,username,password))
    {
        cin.get();
        system("cls");
        return true;
    }
    else
    {
        return false;
    }
}
```

4.3.09 Main Menu Function

```
// Menu Screen
void menuScreen(vector<Admin>& admins,vector<Customer>&
customers,Hotel& hotel)
{
    // Temporary Variables
    int option;
    string id;
    int custType;
    int cust;
    Customer* tempCust;


    // Admin Panel UI
    cout << "Admin Panel\n=================\n";
    cout <<"Select An Option (1-6)\n\n";
    cout <<MAGENTA<< "[1] "<<RESET<< "All Rooms"<<endl;
```

```
    cout <<MAGENTA<< "[2] "<<RESET<< "All Customers"<<endl;
    cout <<MAGENTA<< "[3] "<<RESET<< "Room Details"<<endl;
    cout <<MAGENTA<< "[4] "<<RESET<< "Edit Room Price"<<endl;
    cout <<MAGENTA<< "[5] "<<RESET<< "Book A Room"<<endl;
    cout <<MAGENTA<< "[6] "<<RESET<< "Vacate A Room"<<endl;
    if(::superAdmin){
        cout <<MAGENTA<< "[7] "<<RESET<< "Add New
Admin"<<endl;
        cout <<MAGENTA<< "[8] "<<RESET<< "Logout"<<endl;
    } else {
        cout <<MAGENTA<< "[7] "<<RESET<< "Logout"<<endl;
    }
    cout << "\n> ";
    cin >> option;
    system("cls");


    // After Menu Selection
    switch(option)
    {
    case 1:
        // Check All Rooms
        cout << "All Rooms\n=================\nList of all
rooms in the hotel\n\n";
        hotel.viewAllRooms();
        continueProcess();
        break;
    case 2:
        // Check All Registered Customers
        cout << "All Customers\n=================\nList of
all registered customers in the hotel\n\n";
        viewAllCustomers(customers);
        continueProcess();
        break;
    case 3:
        // Check Room Details
        cout << "Room Details\n=================\nEnter a
Room Number below to get the room details\n\n";
        cout << "Room Number: ";
        cin >> id;
        cout << endl<< endl;
        cust = hotel.checkRoom(id);
        checkCustomer(customers,cust);
```

```
        continueProcess();
        break;

    case 4:
        // Edit Room Price
        cout << "Edit Room Price\n=================\nEnter a
Room Number below to edit it\'s details\n\n";
        cout << "Room No: ";
        cin >> id;
        hotel.editRoom(id);
        continueProcess();
        break;

    case 5:
        // Book A Room
        cout << "Book A Room\n=================\nEnter a Room
Number below to book a room\n\n";
        cout << "Room No:";
        cin >> id;
        if(!hotel.isRoomAvailable(id))
        {
            cout <<RED<< "Room " << id << " Is Not
Available!"<<RESET << endl;
            continueProcess();
            break;
        }
        cout << "Room Price ($): " << hotel.getRoomPrice(id)
<<endl;

        // Register A Customer or Use Existing
        cout << "\nCustomer Type:\n"<<MAGENTA<<
"[1]"<<RESET<<" Existing\n"<<MAGENTA<<"[2]"<<RESET<<" New
Customer\n\n> ";
        cin >> custType;
        if(custType == 1)
        {
            cout << "Customer ID:";
            cin >> cust;
            tempCust = checkCustomer(customers,cust);
            if(tempCust != nullptr)
            {
                hotel.bookRoom(id, *tempCust);
            }
```

```
            else
            {
                cout <<RED<< "\nInvalid Customer ID!"<<RESET;
            }
        }
        else if(custType == 2)
        {
            customers.push_back(Customer());
            hotel.bookRoom(id, customers.back());
        }
        else
        {
            cout <<RED<< "\nInvalid Choice!"<<RESET;
        }
        continueProcess();
        break;

    case 6:
        // Make A Room Available
        cout << "Vacant A Room\n=================\nEnter a
Room Number below to make a room available\n\n";
        cout << "Room No:";
        cin >> id;
        hotel.vacateRoom(id);
        continueProcess();
        break;
    case 7:
        if(::superAdmin){
            cout << "Add An Admin\n=================\nEnter a
login credentials of the new admin\n\n";
            admins.push_back(Admin());
            continueProcess();
        } else {
            ::loggedIn = false;
            system("cls");
            cout <<GREEN<< "Logged Out
Successfully!...\n\n"<<RESET;
        }
        break;
    case 8:
        if(::superAdmin){
            ::loggedIn = false;
            system("cls");
```

```
            cout <<GREEN<< "Logged Out
Successfully!...\n\n"<<RESET;
        } else {
            cout <<RED<< "Invalid Option! Try
again...\n\n"<<RESET;
            continueProcess();
        }
        break;
    default:
        cout <<RED<< "Invalid Option! Try
again...\n\n"<<RESET;
        continueProcess();
        break;
    }
}
```

4.3.10 Main Function

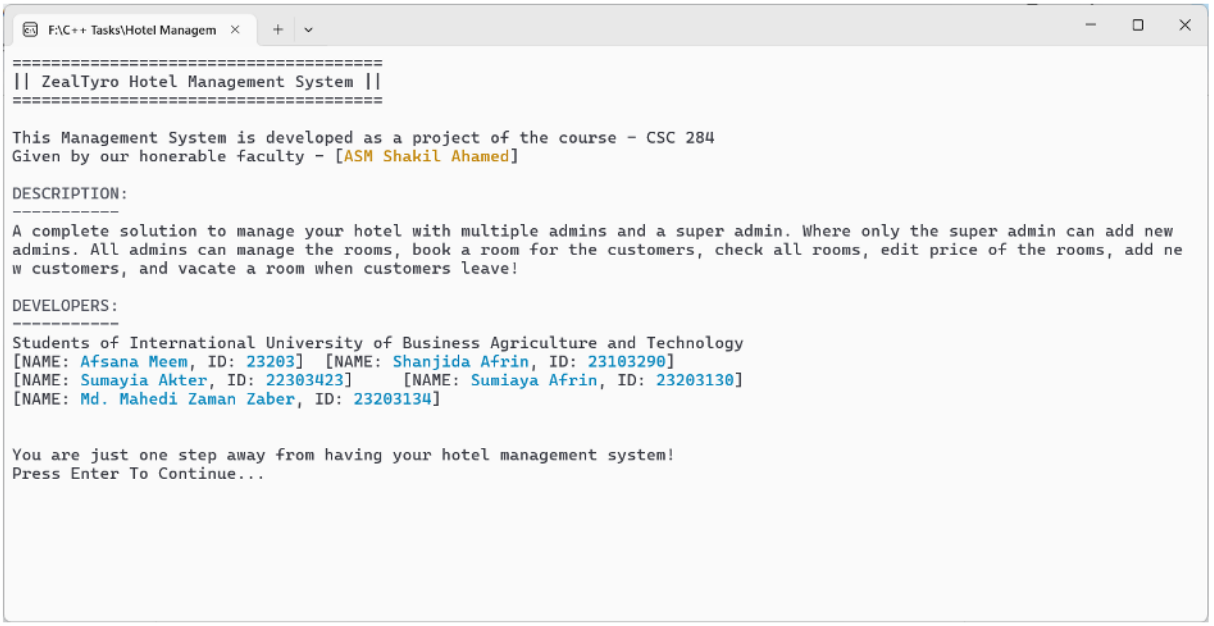
```
int main()
{
    string tempStr1,tempStr2;
    int tempInt1,tempInt2;
    double tempDouble1;
    bool loggedIn = false;


    // Intro Screen
    cout << "======================================"<<endl;
    cout << "|| ZealTyro Hotel Management System ||"<<endl;
    cout <<
"======================================"<<endl<<endl;
    cout << "This Management System is developed as a project
of the course - CSC 284\n";
    cout << "Given by our honerable faculty -
["<<YELLOW<<"ASM Shakil Ahamed"<<RESET<<"]\n\n";
    cout << BOLD << "DESCRIPTION:\n----------\n" << RESET;
    cout << "A complete solution to manage your hotel with
multiple admins and a super admin. Where only the super
admin"
         " can add new admins. All admins can manage the
rooms, book a room for the customers, check all rooms, "
```

```
        "edit price of the rooms, add new customers, and
vacate a room when customers leave! \n\n";
    cout << BOLD << "DEVELOPERS:\n----------\n" << RESET;
    cout << "Students of International University of Business
Agriculture and Technology";
    cout << "\n[NAME:"<<BLUE<<" Afsana Meem"<<RESET<<",
ID:"<<BLUE<<" 23203063"<<RESET<<"]\t";
    cout << "[NAME:"<<BLUE<<" Shanjida Afrin"<<RESET<<",
ID:"<<BLUE<<" 23103290"<<RESET<<"]\t";
    cout << "\n[NAME:"<<BLUE<<" Sumayia Akter"<<RESET<<",
ID:"<<BLUE<<" 22303423"<<RESET<<"]\t";
    cout << "[NAME:"<<BLUE<<" Sumiaya Afrin"<<RESET<<",
ID:"<<BLUE<<" 23203130"<<RESET<<"]\t";
    cout << "\n[NAME:"<<BLUE<<" Md. Mahedi Zaman
Zaber"<<RESET<<", ID:"<<BLUE<<"
23203134"<<RESET<<"]\n"<<endl;
    cout << "\nYou are just one step away from having your
hotel management system!";
    cout << "\nPress Enter To Continue...";
    cin.get();
    system("cls");


    // First Time Hotel Setup
    cout << "First Time Setup\n==================\n";
    cout << "Before we can continue, let\'s setup your hotel
first. "
        "Please fill the form below:\n\n";
    cout << "Hotel Name:";
    getline(cin, tempStr1);
    cout << "Total Floors:";
    cin >> tempInt1;
    cout << "Rooms In Each Floors:";
    cin >> tempInt2;
    cout << "Default Room Cost ($):";
    cin >> tempDouble1;
    Hotel hotel(tempStr1,tempInt1,tempInt2,tempDouble1);
    vector<Customer> customers;
    cout <<GREEN<< "Hotel Information Saved!"<<RESET;


    // Super Admin Setup
    cout << "\n\nSuper Admin\n==================\n";
```

```
    cout << "Let\'s setup a super admin of your hotel. Only
super admins have the power to"
         " add more admins and have all admin powers\n\n";
    vector<Admin> admins;
    admins.push_back(Admin(true));
    cout << "\nSuper Admin created! Now continue to the panel
using your username "
         "and password to start managing your hotel";
    continueProcess();

    // Admin Panel
    while(true)
    {
        if(::loggedIn)
        {
            //Main Menu
            menuScreen(admins,customers,hotel);


        }
        else
        {
            // Login Screen
            if(loginScreen(admins))
            {
                ::loggedIn = true;
            }
            else
            {
                system("cls");
                cout <<RED<< "Invalid Iogin
Credentials!"<<RESET;

            }
        }
    }
    return 0;
}
```

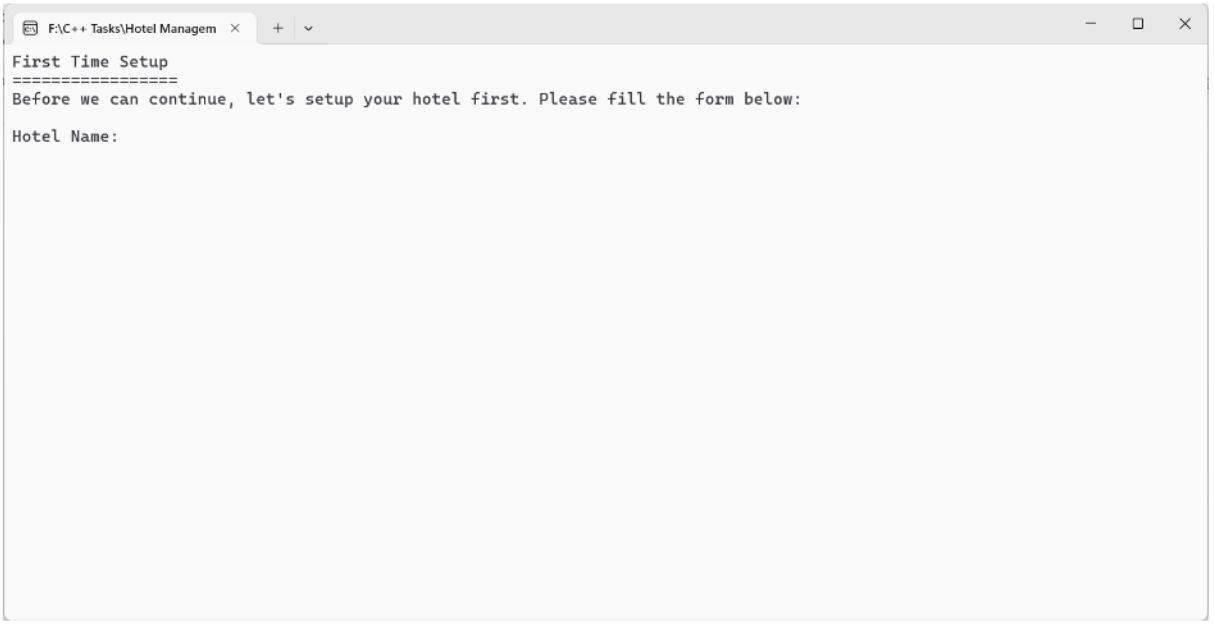
5. User Interface

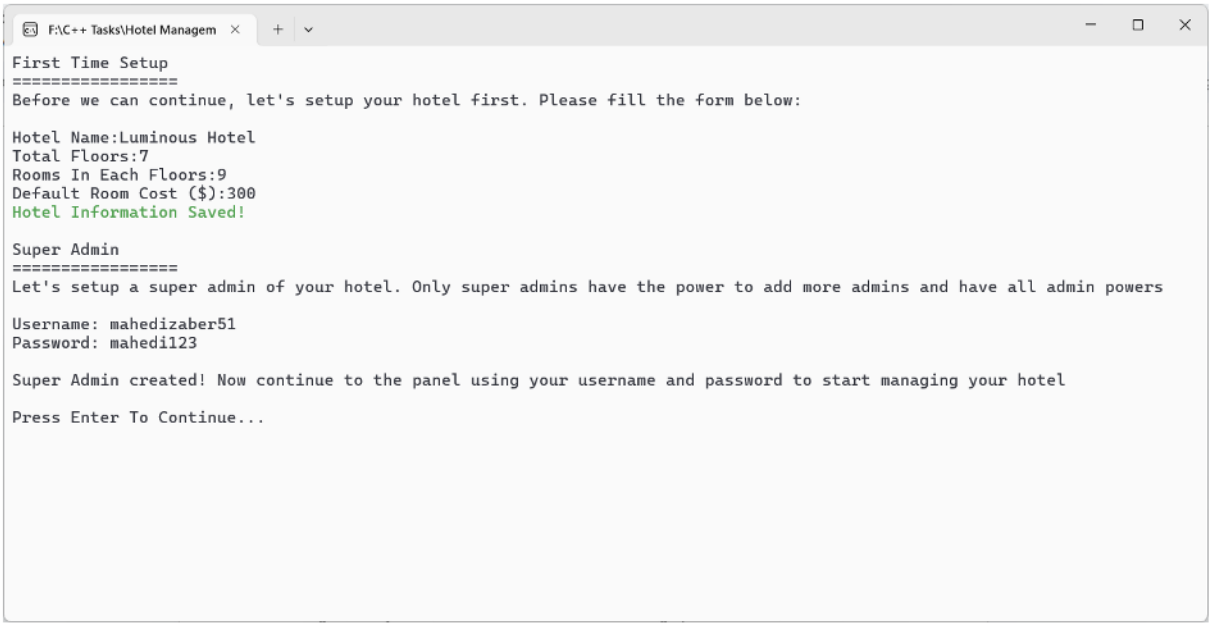
5.1 Welcome Screen



The welcome screen is the initial interface presented to users when they access the hotel management system. It provides a brief introduction to the system and guides users to proceed with login or setup.

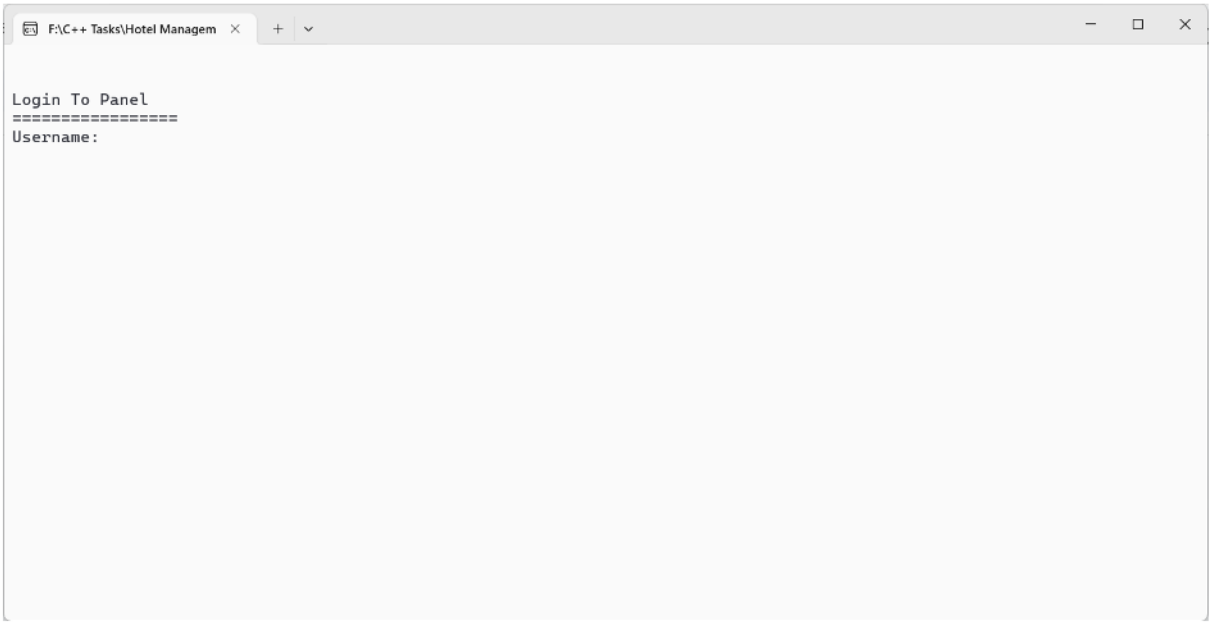
5.2 First Time Setup Screen





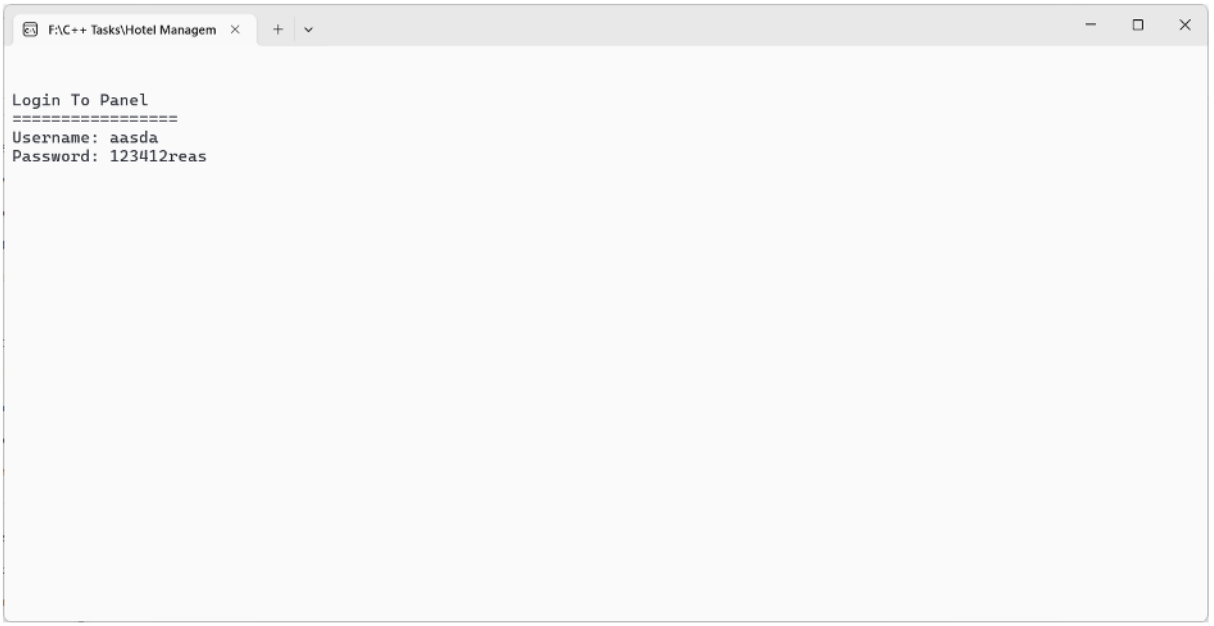
The first-time setup screen is designed for the initial configuration of the hotel management system. This screen is crucial for setting up essential details before the system can be fully operational.

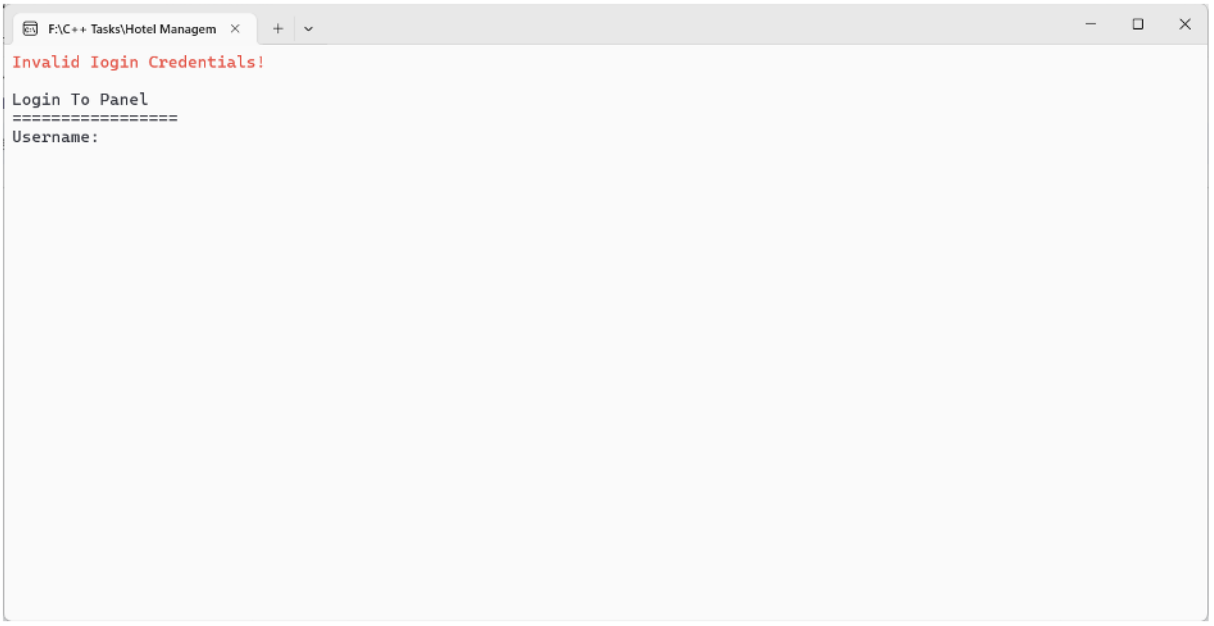
5.1 Login Screen



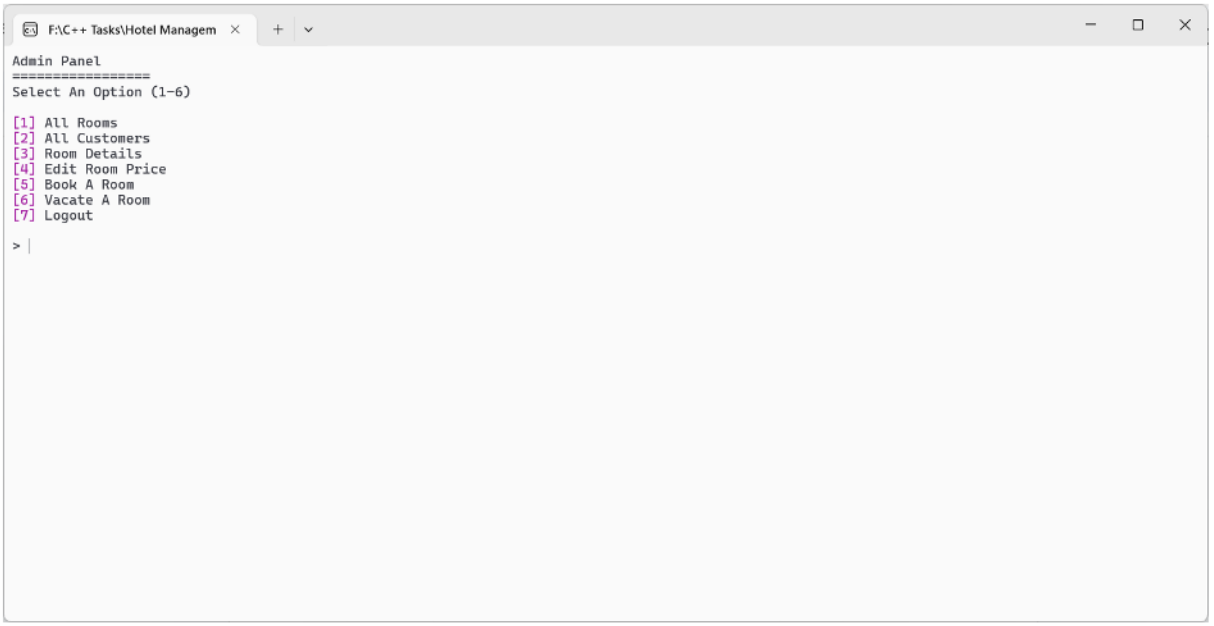
The login screen is the gateway to the system, where users (admins and super admins) authenticate themselves. It requires the input of a username and password. Upon successful authentication, users are redirected to their respective panels.

When a user uses the wrong login details to login:



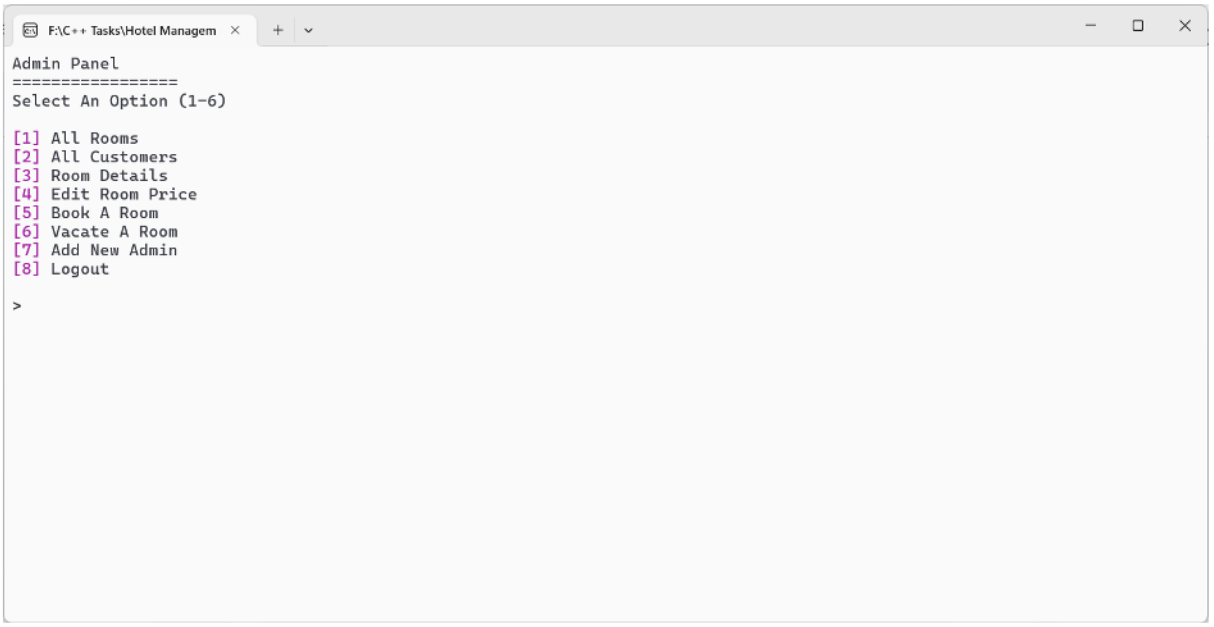


5.2 Admin Panel



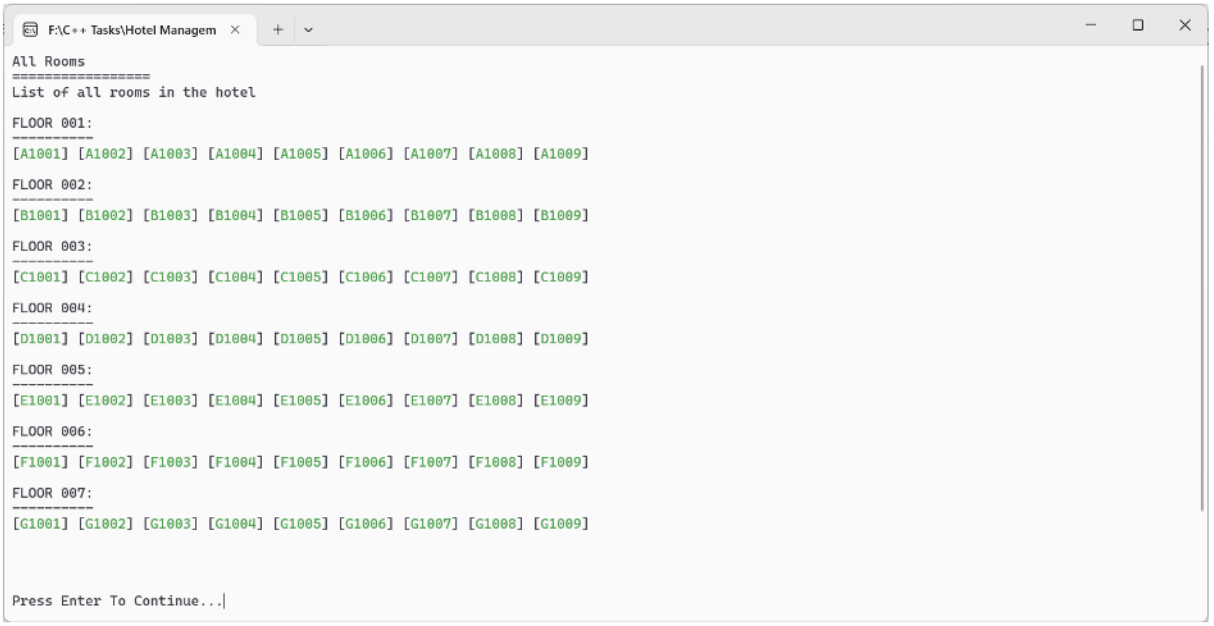
The admin panel is designed for hotel administrators to manage rooms, customers, and bookings. It provides a user-friendly interface to access various functionalities.

5.3 Super Admin Panel



The super admin panel includes all the features available to regular admins and additional functionalities for managing other admins.

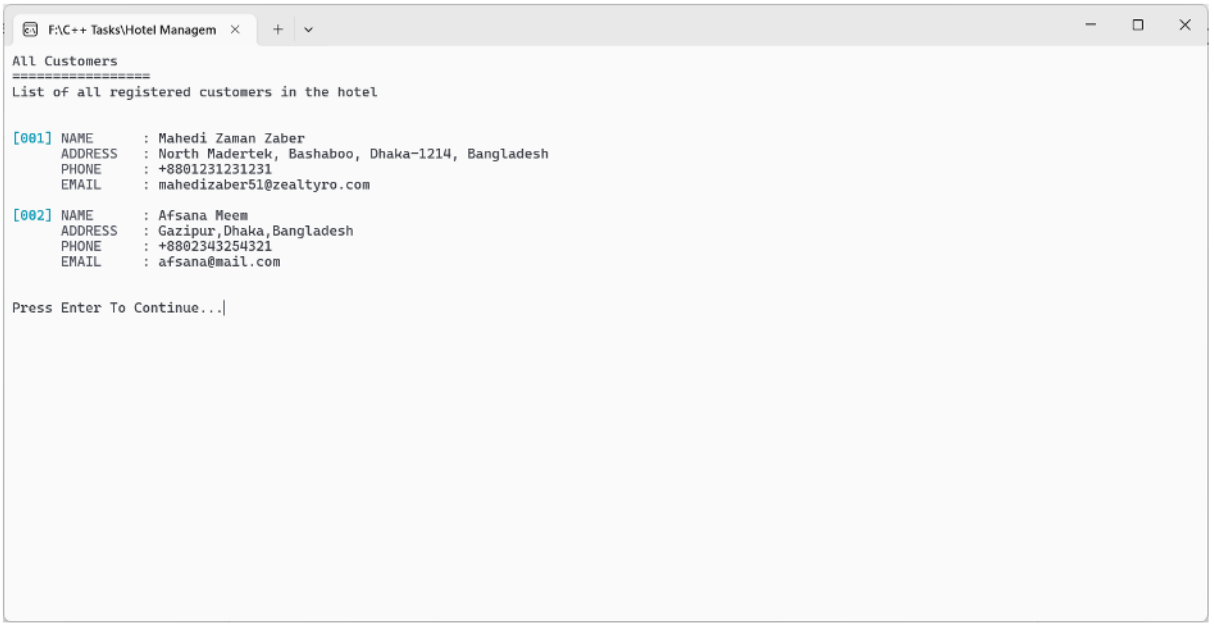
5.4 Admin Panel Option - All Rooms



This option allows admins to view a list of all rooms in the hotel. It displays details such as room number, type, status (available/booked), and price.

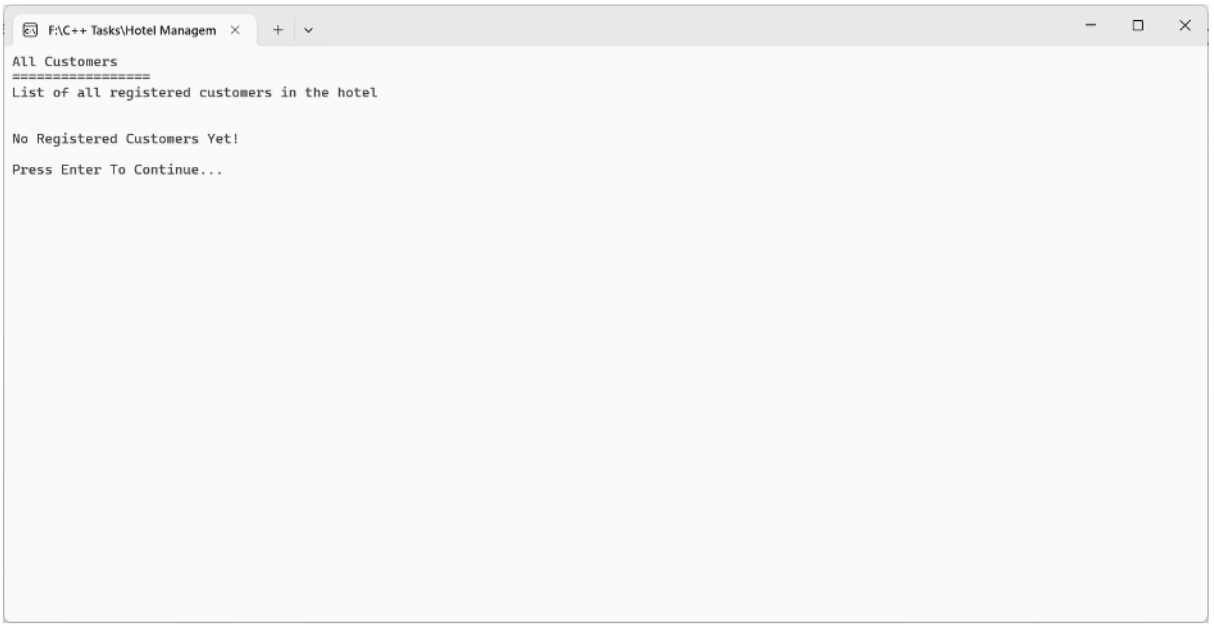
When a room is not available it shows the room number in red color:

5.5 Admin Panel Option - All Customers

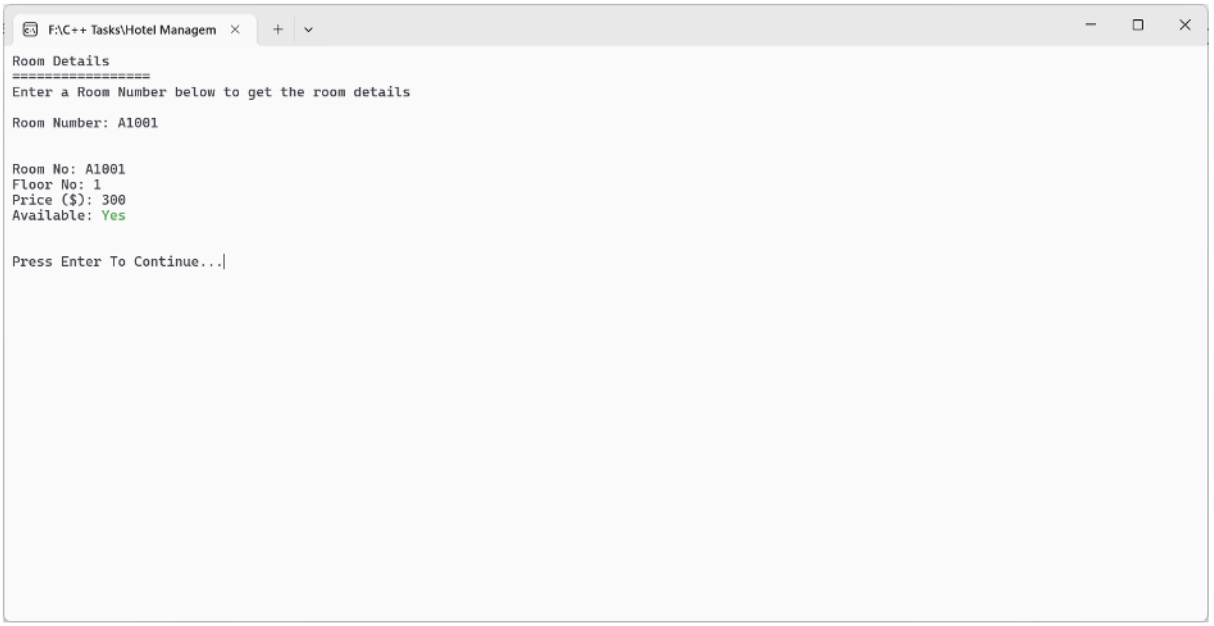


This option provides a comprehensive list of all customers, including their details like name, contact information, and booking history.

If there's no customer registered yet:

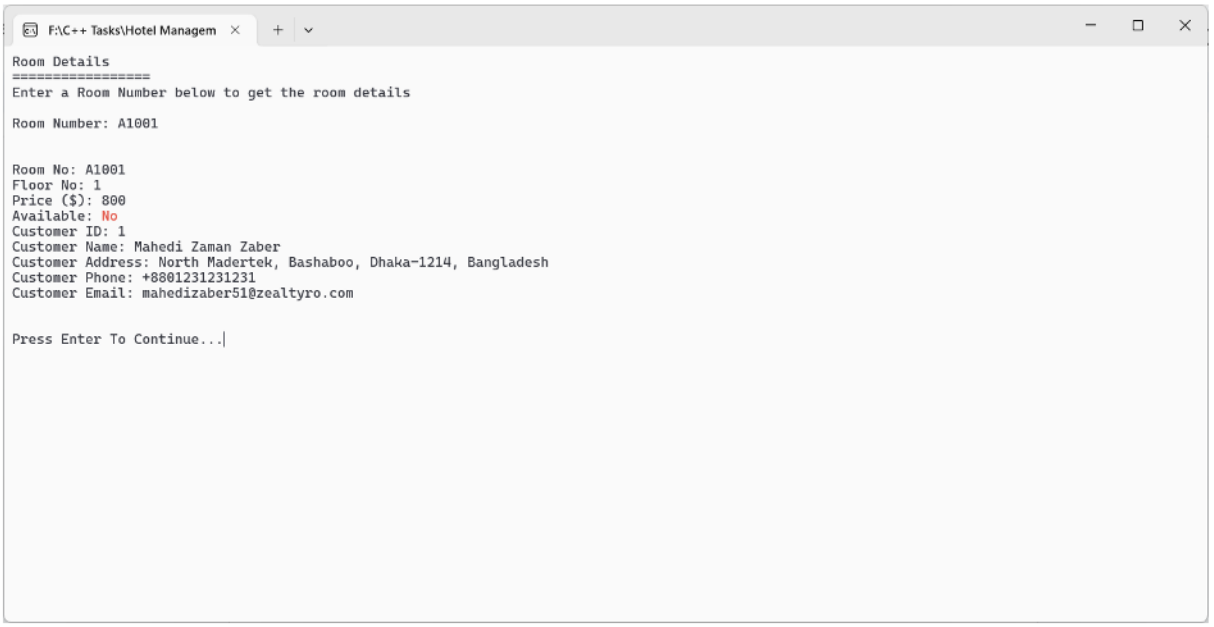


5.08 Admin Panel Option - Room Details

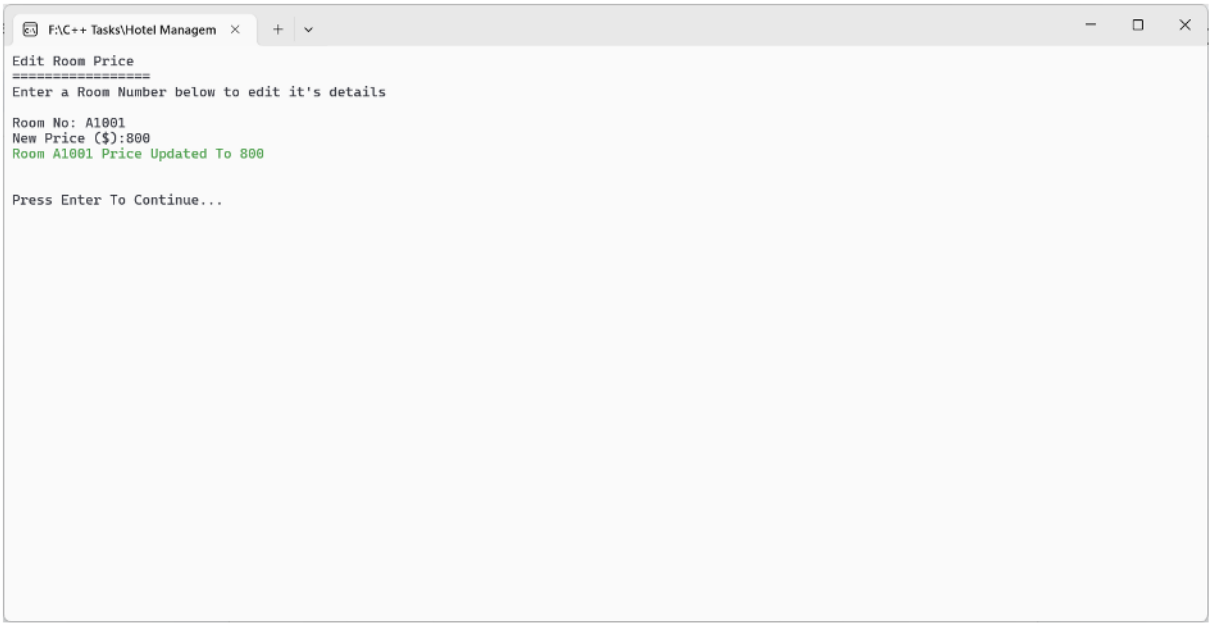


Selecting a specific room from the list allows the admin to view detailed information about that room.

When the room is not available and booked by a customer:

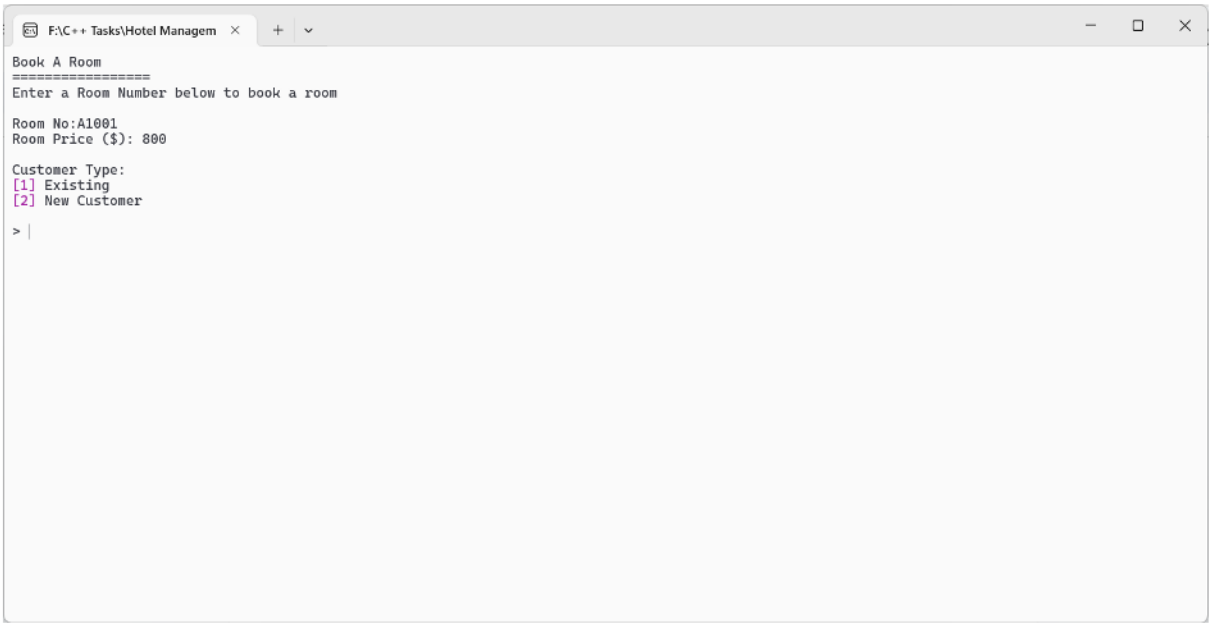


5.09 Admin Panel Option - Edit Room Price



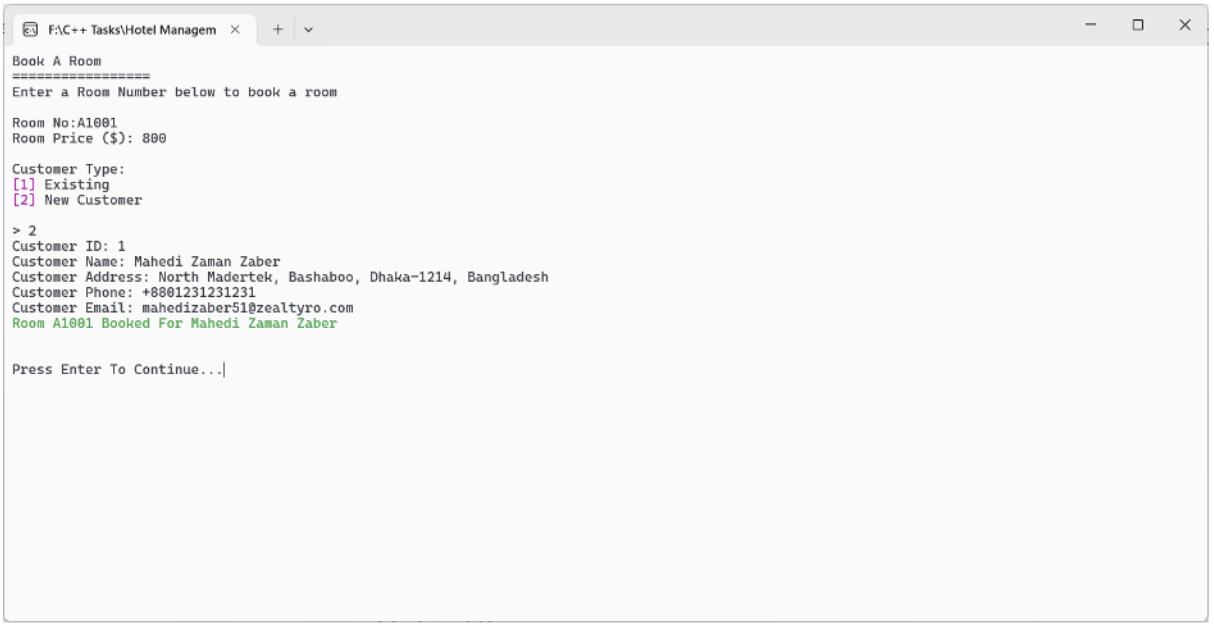
Admins can modify the price of rooms through this option. It ensures that room pricing is flexible and can be updated according to demand or season.

5.10 Admin Panel Option - Book A Room

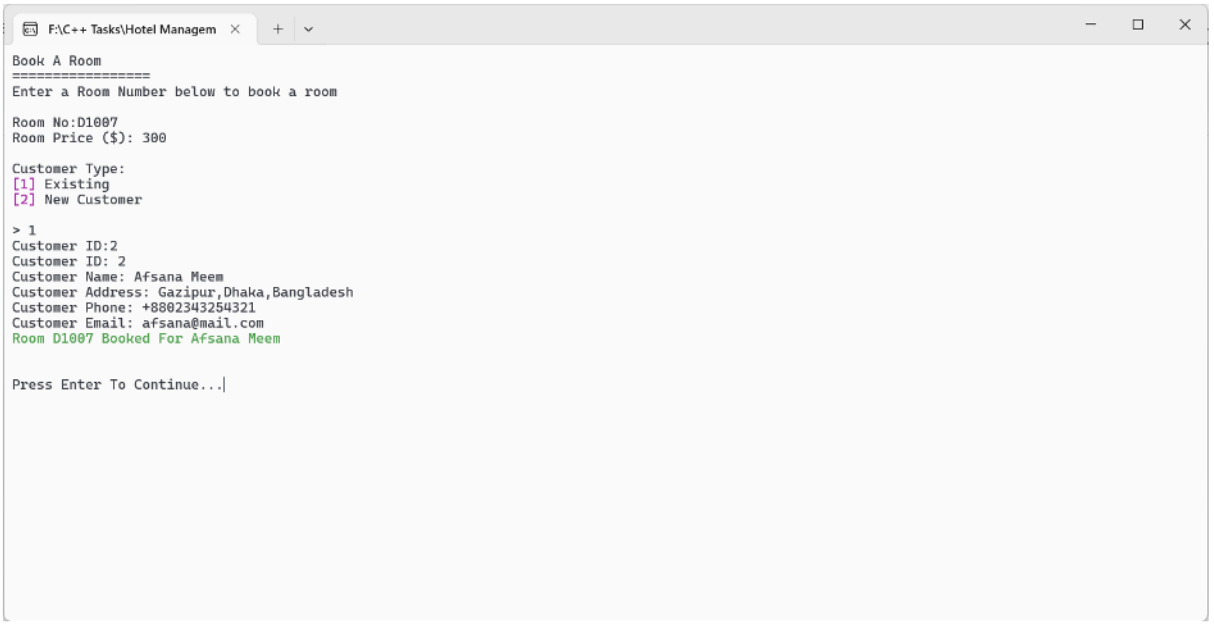


This option allows admins to book rooms for customers, entering necessary details such as customer ID, room number, and booking dates.

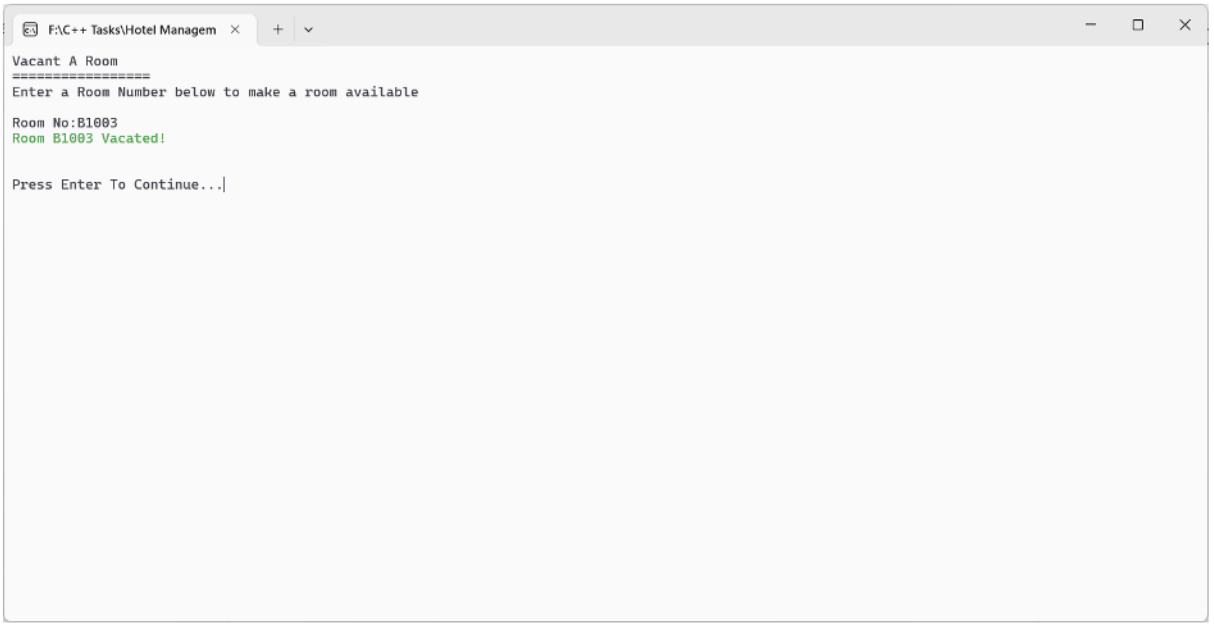
Registering a new customer while booking the room:



Using Existing customer while booking the room:

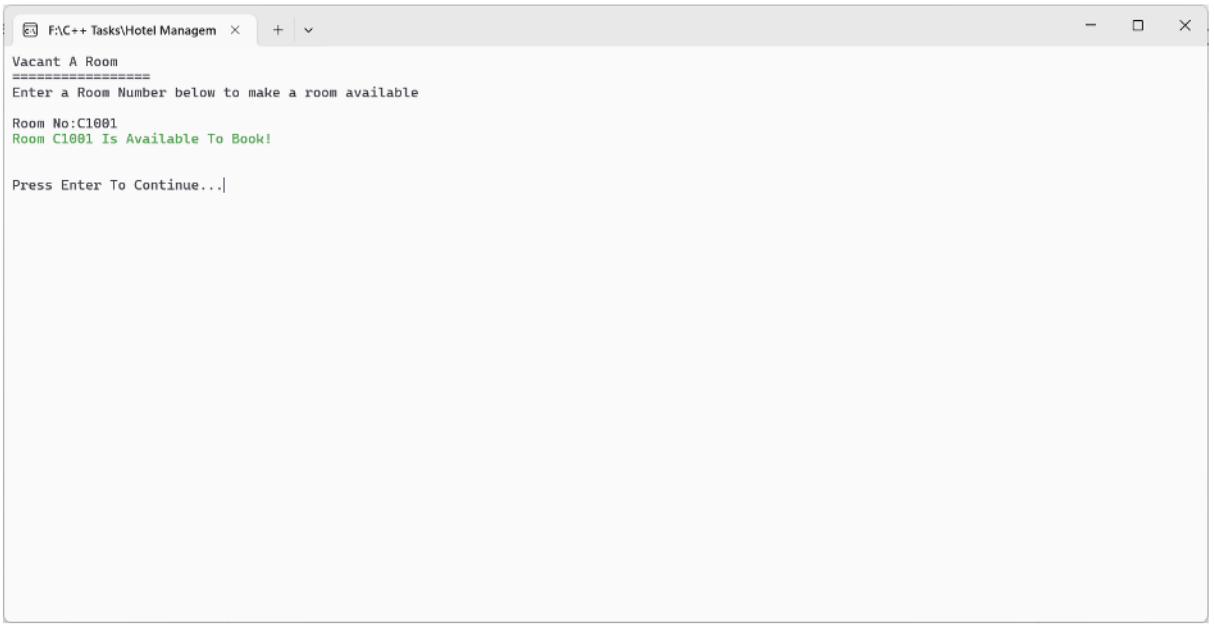


5.11 Admin Panel Option - Vacate A Room

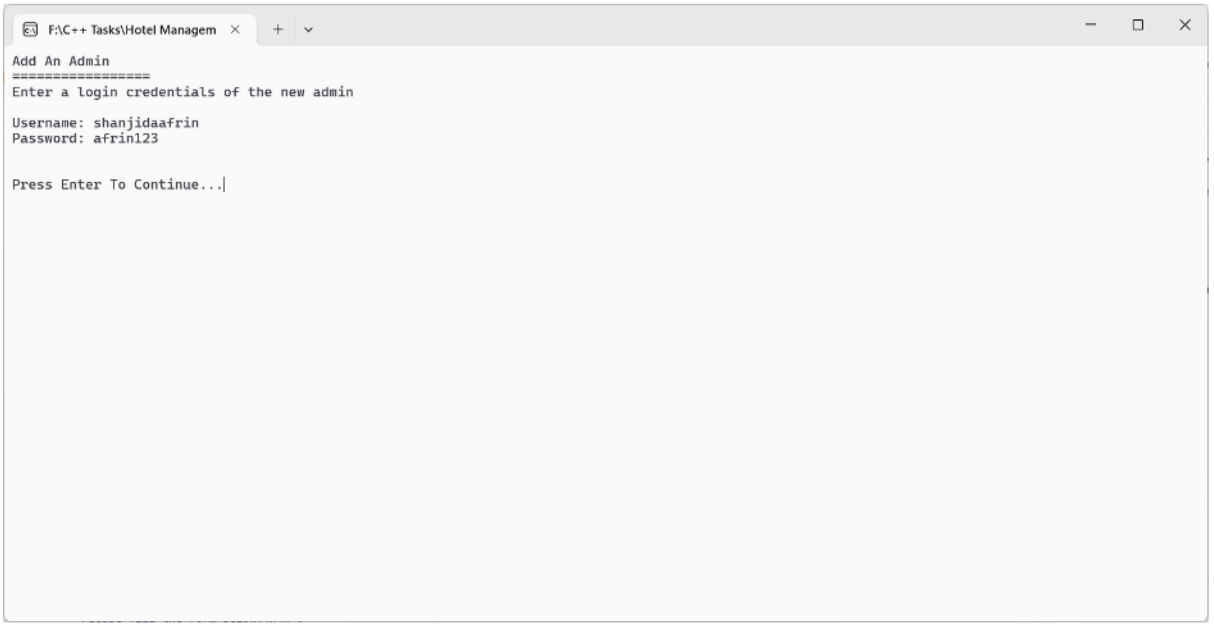


Admins can mark rooms as vacated once a customer checks out. This updates the room status to available.

When the room is already available:

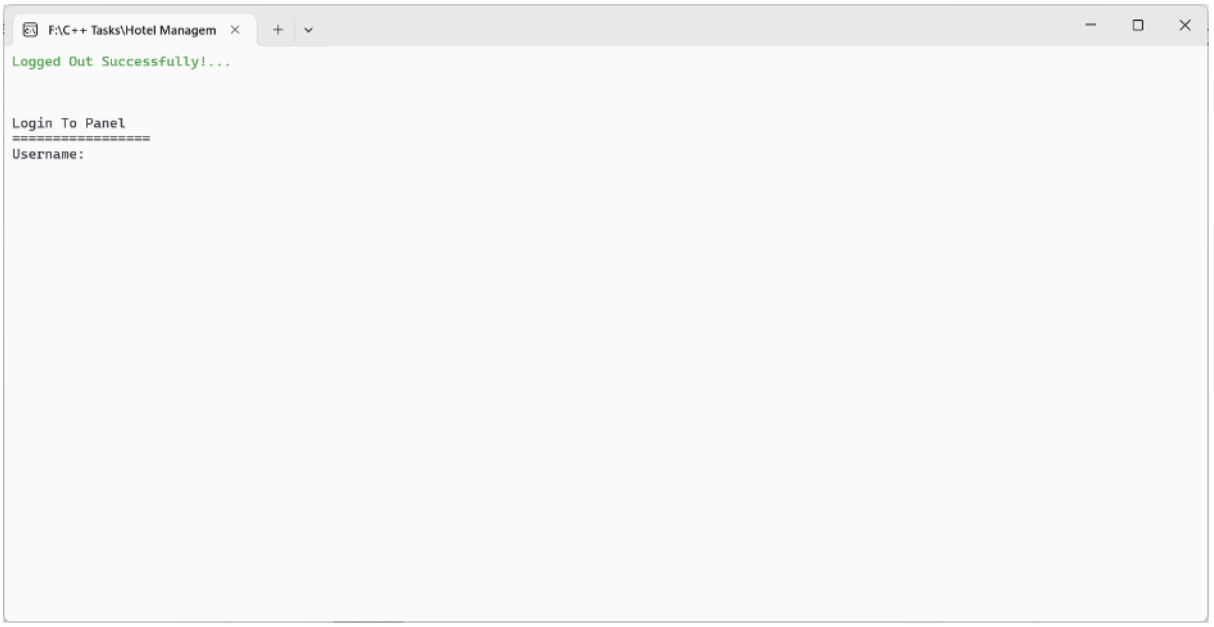


5.12 Admin Panel Option - Add New Admin



This feature is exclusive to the super admin panel. It allows super admins to add new admin users to the system.

5.13 Admin Panel Option - Logout



Admins and super admins can log out of their accounts using this option, which securely ends the session and redirects them to the login screen.

6. Conclusion

6.1 Summary of Achievements

The Hotel Management System was successfully developed and tested, achieving the following:

- Efficient management of rooms, customers, and bookings.
- A strong admin interface with super admin capabilities.
- Comprehensive testing to ensure functional integrity.

6.2 Future Enhancements

Potential enhancements for the system include:

- Implement Hotel Staff Management
- Integration with online booking platforms.
- Advanced analytics for hotel performance.
- Mobile application for hotel management on the go.
- Enhanced security features such as multi-factor authentication.

7. Appendices

7.1 Full Code Listings

The full code listings for all classes and main functions are provided below:

```
/*

    * Project Name: Hotel Management System

    * Description : A complete solution to manage your hotel
with multiple admins and a super admin.

                    Where only the super admin can add new
admins. All admins can manage the rooms,

                    book a room for the customers, check all
rooms, edit price of the rooms, add new

                    customers, and vacate a room when
customers leave!

    * Developers  : Mahedi Zaman Zaber, Afsana Meem, Shanjida
Afrin, Sumayia Akter, Sumiaya Afrin

    * University  : IUBAT - International University of
Business Agriculture and Technology

    * Course      : Programming in C++ Lab (CSC 284)

*/



#include <iostream>

#include <vector>

#include <string>

using namespace std;
```

```
// ANSI Colors
#define RESET "\033[0m"
#define BOLD "\033[1m"
#define RED "\033[31m"
#define GREEN "\033[32m"
#define YELLOW "\033[33m"
#define BLUE "\033[34m"
#define MAGENTA "\033[35m"
#define CYAN "\033[36m"
#define WHITE "\033[37m"



// Global Variable
bool loggedIn = false;
bool superAdmin = false;

class Room
{
    string roomNumber;
    bool isBooked;
    int customerID;
```

```
    double roomPrice;
    int roomFloor;
public:
    // Constructor
    Room(string number,double price,int floor)
    {
        roomNumber= number;
        isBooked = false;
        customerID = -1;
        roomPrice = price;
        roomFloor = floor;
    }

    string getRoomNumber()
    {
        return roomNumber;
    }

    int getCustomerID()
    {
        return customerID;
    }
```

```
double getPrice()
{
    return roomPrice;
}
void setPrice(double price)
{
    roomPrice = price;
}

bool getIsBooked()
{
    return isBooked;
}

void book(int customer)
{
    isBooked = true;
    customerID = customer;
}

void vacate()
{
```

```
        isBooked = false;
        customerID = -1;
    }

    void getDetails()
    {
        cout << "Room No: "<< roomNumber<<endl;
        cout << "Floor No: "<< roomFloor<<endl;
        cout << "Price ($): "<< roomPrice<<endl;
        cout << "Available: ";
        if(isBooked)
        {
            cout <<RED<< "No"<<RESET<<endl;
        }
        else
        {
            cout <<GREEN<< "Yes"<<RESET<<endl;
        }
    }
};
```

```
class Customer
{
    static int lastID;
    int customerID;
    string customerName;
    string customerAddress;
    string customerPhone;
    string customerEmail;
public:
    // Constructor
    Customer()
    {
        customerID = ++lastID;
        cout << "Customer ID: " << customerID <<endl;
        cin.ignore();
        cout << "Customer Name: ";
        getline(cin,customerName);
        cout << "Customer Address: ";
        getline(cin,customerAddress);
        cout << "Customer Phone: ";
        cin >> customerPhone;
```

```
        cout << "Customer Email: ";
        cin >> customerEmail;
    }

    string getName()
    {
        return customerName;
    }

    int getID()
    {
        return customerID;
    }
    void getDetails()
    {
        cout << "Customer ID: "<< customerID <<endl;
        cout << "Customer Name: "<< customerName<<endl;
        cout << "Customer Address: "<< customerAddress<<endl;
        cout << "Customer Phone: "<< customerPhone<<endl;
        cout << "Customer Email: "<< customerEmail<<endl;
    }
```

```
    // To Show as a List

    void getDetailsFlat()

    {

        cout <<CYAN<< "\n["<<(customerID < 10 ? "00" :
(customerID < 100 ? "0" : ""))<<customerID<< "] "<<RESET;

        cout <<BOLD<<"NAME\t: "<<RESET << customerName <<"\n
"<<BOLD<<"ADDRESS\t: "<<RESET<< customerAddress ;

        cout <<"\n      "<<BOLD<<"PHONE\t: "<<RESET<<
customerPhone<<"\n      "<<BOLD<<"EMAIL\t: "<<RESET<<
customerEmail;

        cout <<endl;

    }

};



// Static Value Declaration (Auto Increment ID)

int Customer::lastID = 0;



// To get customer By ID

Customer* checkCustomer(vector<Customer>& customers, int id)

{

    for (int i =0;i < customers.size();i++)

    {

        if (customers[i].getID() == id)

        {
```

```
            customers[i].getDetails();
            return &customers[i];
        }
    }
    return nullptr;
}

// Get the list of Customers
void viewAllCustomers(vector<Customer>& customers)
{
    if(customers.size()==0){
        cout << "\nNo Registered Customers Yet!";
    } else {
        for (int i = 0; i < customers.size(); i++)
        {
            customers[i].getDetailsFlat();
        }
    }
}



class Hotel
```

```
{
    string hotelName;
    vector<Room> rooms;
    int totalFloors;
    int eachFloorRoomCount;
public:
    // Constructor
    Hotel(string name,int floors, int roomEach,double price)
    {
        hotelName = name;
        totalFloors = floors;
        eachFloorRoomCount = roomEach;

        char floorChar = 'A';
        int floorExt = 1;
        for (int floor = 1; floor <= totalFloors; ++floor)
        {
            for(int roomNum = 1; roomNum <=
eachFloorRoomCount; ++roomNum)
            {
                // Room number format: Character Representing
Floor + Floor Extension + 00 or 0 + Room Number
```

```
                string roomNumber = floorChar +
to_string(floorExt)

                                    + (roomNum < 10 ? "00" :
(roomNum < 100 ? "0" : ""))

                                    + to_string(roomNum);



rooms.push_back(Room(roomNumber,price,floor));

            }

            // Reset To Character Representing Floor To 'A' if
it Exceeds 'Z'

            // Increase Floor Extension If Character
Representing Floor Exceeds 'Z'

            if (floorChar == 'Z')

            {

                floorChar = 'A';

                floorExt++;

            }

            else

            {

                floorChar++;

            }

        }

    }
```

```
    // Check if a room is available
    bool isRoomAvailable(string roomNumber)
    {
        for (int i =0;i < rooms.size();i++)
        {
            if (rooms[i].getRoomNumber() == roomNumber &&
!rooms[i].getIsBooked())
            {
                return true;
            }
        }
        return false;
    }

    // Book a room for a customer
    bool bookRoom(string roomNumber, Customer& customer)
    {
        for (int i =0;i < rooms.size();i++)
        {
            if (rooms[i].getRoomNumber() == roomNumber &&
!rooms[i].getIsBooked())
            {
```

```
                rooms[i].book(customer.getID());

                cout <<GREEN<< "Room " << roomNumber << "
Booked For " << customer.getName()<<RESET << endl;

                return true;

            }

        }

        cout <<RED<< "Room " << roomNumber << " Is Not
Available!"<<RESET << endl;

        return false;

    }


    // Vacate a room

    void vacateRoom(string roomNumber)

    {

        for (int i =0;i < rooms.size();i++)

        {

            if (rooms[i].getRoomNumber() == roomNumber)

            {

                if(rooms[i].getIsBooked()){

                    rooms[i].vacate();

                    cout <<GREEN<< "Room " << roomNumber << "
Vacated!"<<RESET << endl;

                } else {
```

```
                    cout <<GREEN<< "Room " << roomNumber << "
Is Available To Book!"<<RESET << endl;

                }



                return;

            }

        }

        cout <<RED<< "Room " << roomNumber << " Doesn\'t
Exist!"<<RESET << endl;

    }



    // Get Room Price

    double getRoomPrice(string roomNumber)

    {

        for (int i =0;i < rooms.size();i++)

        {

            if (rooms[i].getRoomNumber() == roomNumber)

            {

                return rooms[i].getPrice();

            }

        }

        cout <<RED<< "Room " << roomNumber << " Doesn\'t
Exist!"<<RESET << endl;
```

```
        return -1;
    }

    //get room info and return customer id if booked
    int checkRoom(string roomNumber)
    {
        for (int i =0;i < rooms.size();i++)
        {
            if (rooms[i].getRoomNumber() == roomNumber)
            {
                rooms[i].getDetails();
                return rooms[i].getCustomerID();
            }
        }
        cout <<RED<< "Room " << roomNumber << " Doesn\'t
Exist!"<<RESET << endl;
        return -1;
    }

    // Set Room Price
    bool editRoom(string roomNumber)
    {
```

```
        for (int i =0;i < rooms.size();i++)
        {
            if (rooms[i].getRoomNumber() == roomNumber)
            {
                double price;
                cout << "New Price ($):";
                cin >> price;
                rooms[i].setPrice(price);
                cout <<GREEN<< "Room " << roomNumber << "
Price Updated To " << price<<RESET << endl;
                return true;
            }
        }
        cout <<RED<< "Room " << roomNumber << " Doesn\'t
Exist!"<<RESET << endl;
        return false;
    }



    // Get The List of All Rooms
    void viewAllRooms()
    {
        char floorChar = 'A';
```

```
        int floorExt = 1;

        for (int floor = 1; floor <= totalFloors; ++floor)

        {

            cout << "FLOOR "<<(floor < 10 ? "00" : (floor <
100 ? "0" : "")) <<floor;

            cout << ":\n----------\n";

            for(int roomNum = 1; roomNum <=
eachFloorRoomCount; ++roomNum)

            {

                // Room number format: Character Representing
Floor + Floor Extension + 00 or 0 + Room Number

                string roomNumber = floorChar +
to_string(floorExt)

                                    + (roomNum < 10 ? "00" :
(roomNum < 100 ? "0" : ""))+ to_string(roomNum);

                if (isRoomAvailable(roomNumber))

                {

                    cout <<"["<<GREEN<< roomNumber << RESET<<
"] ";

                }

                else

                {

                    cout <<"["<< RED << roomNumber << RESET<<
"] ";

                }
```

```
            }

            cout <<endl<< endl;

            // Reset To Character Representing Floor To 'A' if
it Exceeds 'Z'

            // Increase Floor Extension If Character
Representing Floor Exceeds 'Z'

            if (floorChar == 'Z')

            {

                floorChar = 'A';

                floorExt++;

            }

            else

            {

                floorChar++;

            }

        }

        cout << endl;

    }

};



// Admin Class

class Admin
```

```
{
    string username;
    string password;
    bool superAdmin;
public:
    // Constructor
    Admin(bool super = false)
    {
        cout << "Username: ";
        cin >> username;
        cout << "Password: ";
        cin >> password;
        superAdmin = super;
    }
    string getUsername()
    {
        return username;
    }
    string getPassword()
    {
        return password;
    }
```

```
    bool isSuper(){

        return superAdmin;

    }

};



// Check If Admin Credentials Match

bool checkAdmin(vector<Admin>& admins, string username,
string password)

{

    for (int i =0;i < admins.size();i++)

    {

        if (admins[i].getUsername() == username &&
admins[i].getPassword() == password)

        {

            if(admins[i].isSuper()){

                ::superAdmin = true;

            } else {

                ::superAdmin = false;

            }

            return true;

        }

    }

    return false;
```

```
}



// Enter to Continue
void continueProcess()
{
    cin.ignore();
    cout << "\n\nPress Enter To Continue...";
    cin.get();
    system("cls");
}


// Login Screen
bool loginScreen(vector<Admin>& admins)
{
    string username,password;
    cout << "\n\nLogin To Panel\n==================\n";
    cout << "Username: ";
    cin >> username;
    cout << "Password: ";
    cin >> password;
    if(checkAdmin(admins,username,password))
```

```
    {
        cin.get();
        system("cls");
        return true;
    }
    else
    {
        return false;
    }
}



// Menu Screen
void menuScreen(vector<Admin>& admins,vector<Customer>&
customers,Hotel& hotel)
{
    // Temporary Variables
    int option;
    string id;
    int custType;
    int cust;
    Customer* tempCust;
```

```
    // Admin Panel UI
    cout << "Admin Panel\n=================\n";
    cout <<"Select An Option (1-6)\n\n";
    cout <<MAGENTA<< "[1] "<<RESET<< "All Rooms"<<endl;
    cout <<MAGENTA<< "[2] "<<RESET<< "All Customers"<<endl;
    cout <<MAGENTA<< "[3] "<<RESET<< "Room Details"<<endl;
    cout <<MAGENTA<< "[4] "<<RESET<< "Edit Room Price"<<endl;
    cout <<MAGENTA<< "[5] "<<RESET<< "Book A Room"<<endl;
    cout <<MAGENTA<< "[6] "<<RESET<< "Vacate A Room"<<endl;
    if(::superAdmin){
        cout <<MAGENTA<< "[7] "<<RESET<< "Add New
Admin"<<endl;
        cout <<MAGENTA<< "[8] "<<RESET<< "Logout"<<endl;
    } else {
        cout <<MAGENTA<< "[7] "<<RESET<< "Logout"<<endl;
    }
    cout << "\n> ";
    cin >> option;
    system("cls");
```

```
    // After Menu Selection

    switch(option)

    {

    case 1:

        // Check All Rooms

        cout << "All Rooms\n=================\nList of all
rooms in the hotel\n\n";

        hotel.viewAllRooms();

        continueProcess();

        break;

    case 2:

        // Check All Registered Customers

        cout << "All Customers\n=================\nList of
all registered customers in the hotel\n\n";

        viewAllCustomers(customers);

        continueProcess();

        break;

    case 3:

        // Check Room Details

        cout << "Room Details\n=================\nEnter a
Room Number below to get the room details\n\n";

        cout << "Room Number: ";
```

```
        cin >> id;

        cout << endl<< endl;

        cust = hotel.checkRoom(id);

        checkCustomer(customers,cust);

        continueProcess();

        break;


    case 4:

        // Edit Room Price

        cout << "Edit Room Price\n=================\nEnter a
Room Number below to edit it\'s details\n\n";

        cout << "Room No: ";

        cin >> id;

        hotel.editRoom(id);

        continueProcess();

        break;


    case 5:

        // Book A Room

        cout << "Book A Room\n=================\nEnter a Room
Number below to book a room\n\n";

        cout << "Room No:";

        cin >> id;
```

```
        if(!hotel.isRoomAvailable(id))

        {

            cout <<RED<< "Room " << id << " Is Not
Available!"<<RESET << endl;

            continueProcess();

            break;

        }

        cout << "Room Price ($): " << hotel.getRoomPrice(id)
<<endl;



        // Register A Customer or Use Existing

        cout << "\nCustomer Type:\n"<<MAGENTA<<
"[1]"<<RESET<<" Existing\n"<<MAGENTA<<"[2]"<<RESET<<" New
Customer\n\n> ";

        cin >> custType;

        if(custType == 1)

        {

            cout << "Customer ID:";

            cin >> cust;

            tempCust = checkCustomer(customers,cust);

            if(tempCust != nullptr)

            {

                hotel.bookRoom(id, *tempCust);

            }
```

```
            else
            {
                cout <<RED<< "\nInvalid Customer ID!"<<RESET;
            }
        }
        else if(custType == 2)
        {
            customers.push_back(Customer());
            hotel.bookRoom(id, customers.back());
        }
        else
        {
            cout <<RED<< "\nInvalid Choice!"<<RESET;
        }
        continueProcess();
        break;

    case 6:
        // Make A Room Available
        cout << "Vacant A Room\n=================\nEnter a
Room Number below to make a room available\n\n";
        cout << "Room No:";
```

```
        cin >> id;

        hotel.vacateRoom(id);

        continueProcess();

        break;

    case 7:

        if(::superAdmin){

            cout << "Add An Admin\n=================\nEnter a
login credentials of the new admin\n\n";

            admins.push_back(Admin());

            continueProcess();

        } else {

            ::loggedIn = false;

            system("cls");

            cout <<GREEN<< "Logged Out
Successfully!...\n\n"<<RESET;

        }

        break;

    case 8:

        if(::superAdmin){

            ::loggedIn = false;

            system("cls");

            cout <<GREEN<< "Logged Out
Successfully!...\n\n"<<RESET;
```

```
        } else {

            cout <<RED<< "Invalid Option! Try
again...\n\n"<<RESET;

            continueProcess();

        }

        break;

    default:

        cout <<RED<< "Invalid Option! Try
again...\n\n"<<RESET;

        continueProcess();

        break;

    }

}




int main()

{

    string tempStr1,tempStr2;

    int tempInt1,tempInt2;

    double tempDouble1;

    bool loggedIn = false;
```

```
    // Intro Screen

    cout << "======================================"<<endl;

    cout << "|| ZealTyro Hotel Management System ||"<<endl;

    cout <<
"======================================"<<endl<<endl;

    cout << "This Management System is developed as a project
of the course - CSC 284\n";

    cout << "Given by our honerable faculty -
["<<YELLOW<<"ASM Shakil Ahamed"<<RESET<<"]\n\n";

    cout << BOLD << "DESCRIPTION:\n-----------\n" << RESET;

    cout << "A complete solution to manage your hotel with
multiple admins and a super admin. Where only the super
admin"

         " can add new admins. All admins can manage the
rooms, book a room for the customers, check all rooms, "

         "edit price of the rooms, add new customers, and
vacate a room when customers leave! \n\n";

    cout << BOLD << "DEVELOPERS:\n-----------\n" << RESET;

    cout << "Students of International University of Business
Agriculture and Technology";

    cout << "\n[NAME:"<<BLUE<<" Afsana Meem"<<RESET<<",
ID:"<<BLUE<<" 23203063"<<RESET<<"]\t";

    cout << "[NAME:"<<BLUE<<" Shanjida Afrin"<<RESET<<",
ID:"<<BLUE<<" 23103290"<<RESET<<"]\t";

    cout << "\n[NAME:"<<BLUE<<" Sumayia Akter"<<RESET<<",
ID:"<<BLUE<<" 22303423"<<RESET<<"]\t";

    cout << "[NAME:"<<BLUE<<" Sumiaya Afrin"<<RESET<<",
ID:"<<BLUE<<" 23203130"<<RESET<<"]\t";
```

```
    cout << "\n[NAME:"<<BLUE<<" Md. Mahedi Zaman
Zaber"<<RESET<<", ID:"<<BLUE<<"
23203134"<<RESET<<"]\n"<<endl;

    cout << "\nYou are just one step away from having your
hotel management system!";

    cout << "\nPress Enter To Continue...";

    cin.get();

    system("cls");




    // First Time Hotel Setup

    cout << "First Time Setup\n=================\n";

    cout << "Before we can continue, let\'s setup your hotel
first. "

         "Please fill the form below:\n\n";

    cout << "Hotel Name:";

    getline(cin, tempStr1);

    cout << "Total Floors:";

    cin >> tempInt1;

    cout << "Rooms In Each Floors:";

    cin >> tempInt2;

    cout << "Default Room Cost ($):";

    cin >> tempDouble1;

    Hotel hotel(tempStr1,tempInt1,tempInt2,tempDouble1);
```

```
    vector<Customer> customers;

    cout <<GREEN<< "Hotel Information Saved!"<<RESET;




    // Super Admin Setup

    cout << "\n\nSuper Admin\n=================\n";

    cout << "Let\'s setup a super admin of your hotel. Only
super admins have the power to"

         " add more admins and have all admin powers\n\n";

    vector<Admin> admins;

    admins.push_back(Admin(true));

    cout << "\nSuper Admin created! Now continue to the panel
using your username "

         "and password to start managing your hotel";

    continueProcess();


    // Admin Panel

    while(true)

    {

        if(::loggedIn)

        {

            //Main Menu

            menuScreen(admins,customers,hotel);
```

```
        }
        else
        {
            // Login Screen
            if(loginScreen(admins))
            {
                ::loggedIn = true;
            }
            else
            {
                system("cls");
                cout <<RED<< "Invalid Iogin
Credentials!"<<RESET;

            }
        }
    }
    return 0;
}
```

8.3 References

List of all references used in the development and testing of the system.

- **C++ Documentation:**
  https://en.cppreference.com/w/