B Tech Computer Science and Engineering (Data Science)

-311 Program Semester – II

**Academic Year 2023-24**

**Object Oriented Programming and Design**

Mini Project

# Hotel Management System

Developed by

**L003 Shivam Batra**

**L008 Harsh Gulwani**

**L013 Soham Joshi**

**L021 Manav Puria**

Faculty In charge

Dr. Dhirendra Mishra

Dr. Priteegandha Naik

**SVKM's NMIMS**

## 1. Abstract

This project explores how C++ programming can be used to make a secured hotel management system. To keep up with the modern technology and be informed about all the activities going on in the hotel, it has become crucial for managers to have a digital hotel management system which holds and maintains all important information needed to run the hotel. This project leverages core functionalities of object oriented programming the project uses these functionalities to help the manager in reservation management, guest check-in/check-out procedures, room allocation, restaurant and billing. Guest security holds the highest importance in the system, therefore, techniques like data hiding, encapsulation were implemented. The project developed a user-friendly interface is developed to streamline the interaction between the manager and the system. This project aimed at developing a program to make the daily tasks of managers storing information , managing data and fetching information from the database.

## Introduction

As the manager of a hotel, they must oversee a wide range of activities while also protecting sensitive guest data. The sheer volume of operations in a hotel, along with the requirement for painstaking attention to detail, requires new approaches to maximize efficiency and expedite management processes.

Recognizing these problems, our project aims to transform hotel administration by delivering a comprehensive solution that consolidates all key data and processes into a single, easily accessible platform. This platform acts as a centralized hub, allowing the management to efficiently monitor and handle many areas of the hotel's operations even when they are not physically present on-site at all times.

Key aspects of our project include:

- Centralized Data Management: All guest information, reservations, financial transactions, are kept in a secure central database. This guarantees that critical data is readily available and can be controlled efficiently through a single interface.

- Real-Time Monitoring: The platform allows for real-time monitoring of various hotel activities such as room occupancy, housekeeping status, guest requests, and maintenance chores. This allows the manager to keep aware and respond quickly to any difficulties that may emerge.

By leveraging technology to centralize data and streamline management processes, our project empowers hotel managers to achieve maximum efficiency, enhance guest satisfaction, and maintain the highest standards of service and security.

## Scope of the Project -Societal need of this project

Our project prioritizes the assurance of data security, employing robust measures such as double-step verification to safeguard sensitive guest information from potential breaches. In an age where data privacy is paramount, this commitment to security provides peace of mind to both hoteliers and their clientele.

- Labor Cost Reduction: By simplifying and automating key managerial operations, the initiative helps the hotel minimize its dependency on manual labor, potentially saving money.

- Time Efficiency: The project improves time efficiency by centralizing data and operations, allowing managers to access information and complete tasks more quickly and effectively. This time efficiency not only improves the hotel's operations, but it also helps to increase general productivity and economic efficiency in the hospitality business.

- Data Security Assurance: The project uses strong security methods such as double-step verification to protect sensitive guest data from unwanted access or breaches. [1] This assurance of data protection is critical.

- Simplification of Management Tasks: Managing a hotel entail handling a slew of complex tasks, which may be overwhelming, especially on a large scale. The project simplifies the management process by storing all necessary data in a single, easily accessible location, making it more controllable and less prone to errors. This simplicity benefits not just hotel managers, but also contributes to the general efficiency and reliability of hotel operations, consequently improving quality.

## Organization of project

### Storyline

Beginning your day with the familiar routine of logging into the hotel management system, you ensure smooth operations by verifying your credentials. Once authenticated, you're greeted with options to navigate through the various modules. Opting to start with the reception module, you seamlessly guide guests through the check-in or check-out process, meticulously updating databases with each interaction. Transitioning to the restaurant module, you effortlessly retrieve menus, take orders, and manage billing, ensuring guests enjoy a delightful dining experience. As the day progresses, you deftly navigate between tasks, adeptly addressing guest needs and ensuring efficient operations.

## Roadmap

The project aims to explore how Object-Oriented Programming (OOP) principles can be effectively employed to develop a robust and secure hotel management system.
In alignment with this overarching objective, the project is divided into four key sections:

1. Understanding Object-Oriented Programming (OOP) Principles in C++:
    - Explored C++ scope, functions, classes, and their interactions relevant to a hotel management system.
    - Studied Inheritance, Polymorphism, Encapsulation, and Abstraction in the context of C++ programming.
    - Research common design patterns and best practices in OOP for efficient and secure software development.
    - Development of a research question and a problem statement.
2. Developing a Storyline:
    - Defined the requirements and functionalities of the hotel management system.
    - Creation of a narrative to guide the development process, considering user roles and system interactions.
    - Identified potential security threats and vulnerabilities within the storyline to address during system design and implementation.
3. Designing UML Diagrams:
    - Developed Unified Modeling Language (UML) diagrams to visualize the structure and behavior of the hotel management system.
    - Created class diagrams to represent the relationships and interactions between different classes and objects.
    - Designed sequence diagrams to illustrate the flow of events and interactions between system components during various operations.
4. Writing the Code:
    - Implemented the hotel management system using C++ based on the previously defined storyline and UML diagrams.
    - Followed OOP principles and design patterns identified during research to ensure efficiency and security.
    - Incorporated appropriate data structures, error handling mechanisms, and security measures to mitigate potential risks.
    - Test the developed system thoroughly to ensure functionality, reliability, and security compliance.

## RESEARCH QUESTION

How can Object-Oriented Programming (OOP) principles be used to enhance the efficiency and security of hotel management systems ?

What specific design patterns and practices are most effective in developing a robust and secure OOP-based hotel management system?

### Research objective

To investigate how Object-Oriented Programming (OOP) principles can be leveraged to improve the efficiency and security of hotel management systems in the context of evolving digital threats and customer security concerns. The project attempts to discover the most effective design patterns and techniques for designing a strong and safe OOP-based hotel management system, taking into account potential vulnerabilities and risk mitigation strategies against hackers and other security threat

## OVERALL BLOCK DIAGRAM OF ALL FUNCTIONALITIES

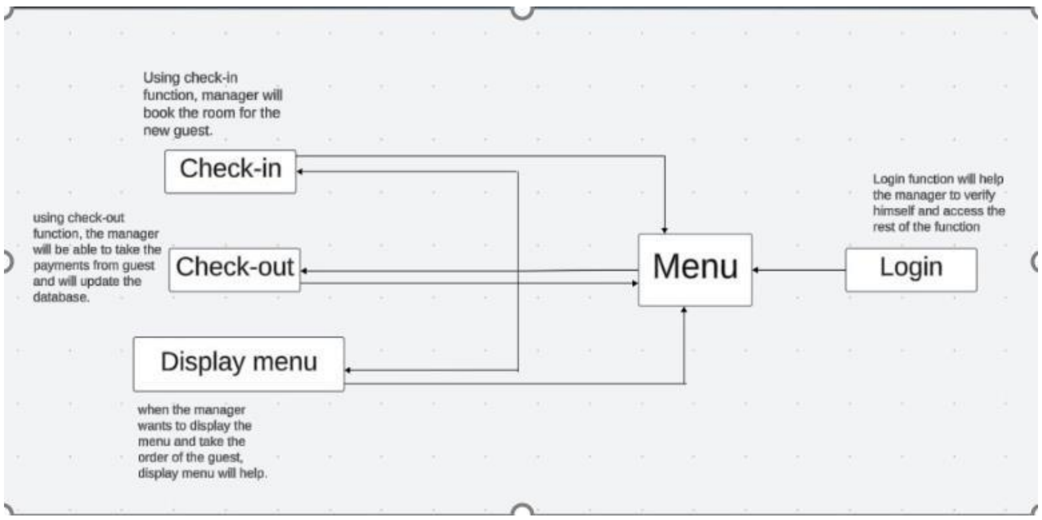

Using check-in function, manager will book the room for the new guest.
Check-in
using check-out function, the manager will be able to take the payments from guest and will update the database.
Check-out
Menu
Login function will help the manager to verify himself and access the rest of the function
Login
Display menu
when the manager wants to display the menu and take the order of the guest, display menu will help.

*Figure 1 Block diagram of the functions*

A block diagram is a visual representation of a system that uses blocks (rectangles) to represent the system's functions. These blocks are then connected by lines, which show how the different parts interact with each other. The above diagram shows the basic 3 functions that will be displayed in a menu format to the user (manager) after they successfully login into the system.

## PROBLEM STATEMENT

In the hospitality industry, maintaining a hotel effectively while protecting sensitive guest data presents considerable hurdles. Traditional hotel management systems frequently lack the sophistication and resilience required to efficiently streamline operations and protect data. The lack of a unified platform for managing multiple parts of hotel operations leads to inefficiencies, higher labor costs, and security breaches [1]. Furthermore, manual processes and fragmented data storage make it difficult for managers to obtain vital information quickly and make informed decisions. Thus, there is a compelling need to create an effective and safe hotel management system based on Object-Oriented Programming (OOP) concepts. This system should incorporate OOP concepts to optimize resource consumption, improve data security, and serve as a consolidated platform for hotel operations administration. Addressing these difficulties will result in increased efficiency, cost savings, and guest-satisfaction in the hotel industry.

## LIST OF CLASSES AND ITS MEMBERS PLANNED

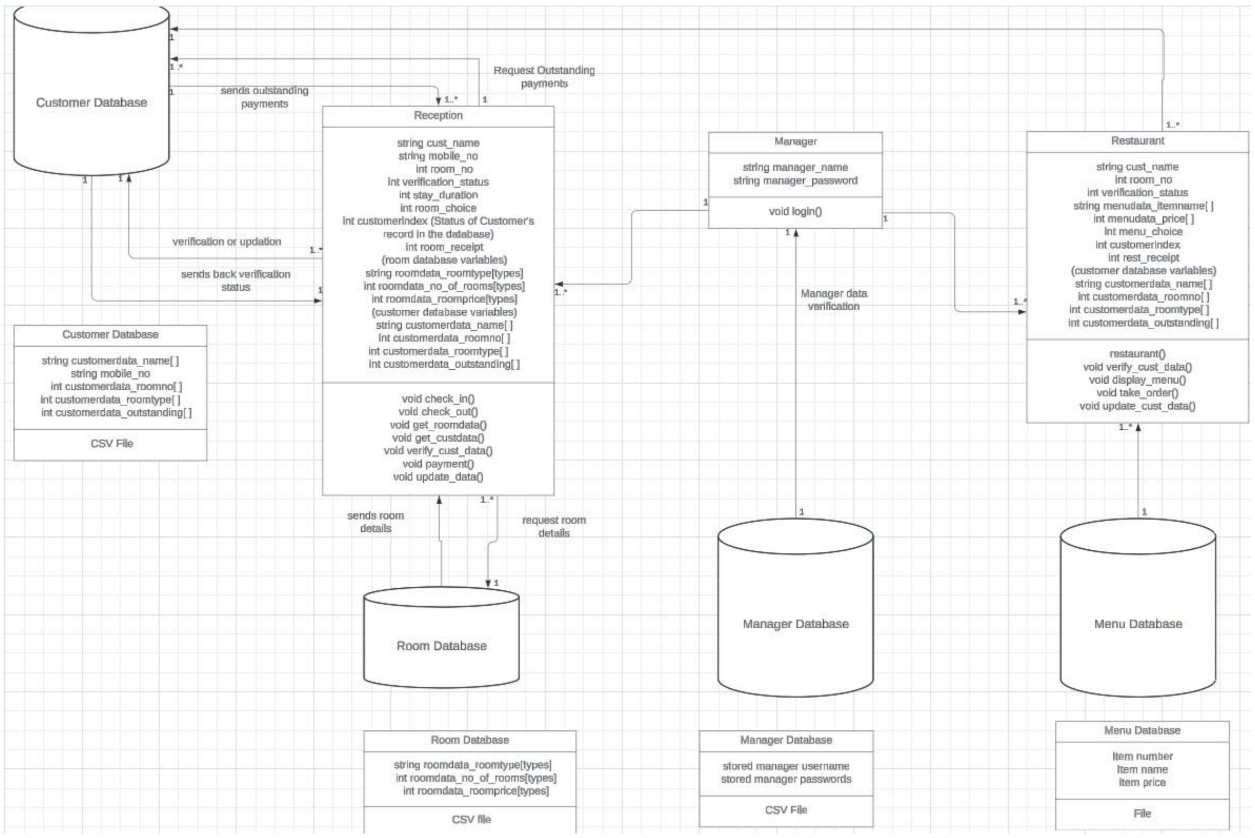


*Figure 2 Class Diagram*

Class 1: Manager

- Data Members
    1. manager_name : string
    2. manager_password : string
- Members Functions
    1. login()

Class 2: Reception

- Data Members
    1. cust_name : string
    2. mobile_no : string
    3. verification_status (flag) : int
    4. customer_index : int
    5. room_type[ ] :string
    6. room_price[ ] : double
    7. no_of_room[ ] : int
    8. room_choice : int

9. room_no : int
10. stay_duration : int
11. room_receipt : double

- Members Functions
    1. check-in()
    2. check-out()
    3. get_roomdata()
    4. get_custdata()
    5. verify_cust_data()
    6. update_data()
    7. payment()

Class 3: Restaurant

- Data Members
    1. cust_name : string
    2. room_no : int
    3. menu_choice : int
    4. rest_receipt : double
- Members Functions
    1. Restaurant() constructor
    2. verify_cust_data()
    3. display_menu()
    4. take_order()
    5. update_cust_data()

Room database:

- string roomdata_roomtype[types]
  int roomdata_no_of_rooms[types]
  int roomdata_roomprice[types]

Customer database variables:

- string customerdata_name[ ]
  string mobile_no
  int customerdata_roomno[ ]
  int customerdata_roomtype[ ]
  int customerdata_outstanding[ ]

## USE CASE DIAGRAM

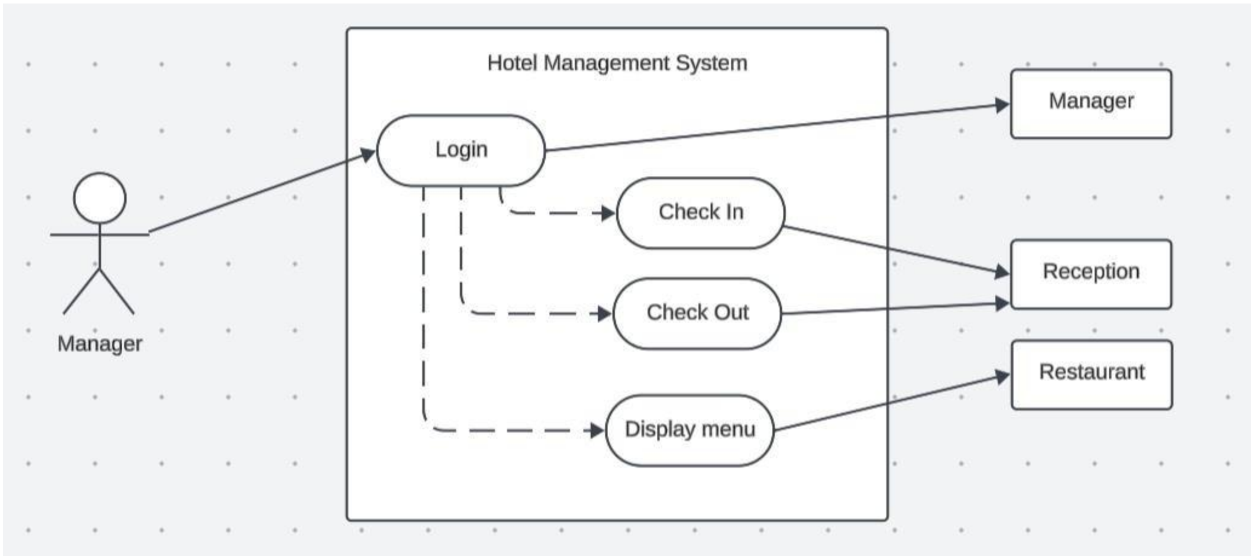


*Figure 3Use case diagram*

Use Case Diagram is one of the major UML diagrams for the project. It states and signifies the actors and how they interact with the classes, according to the task they want to perform. It gives us the Overview of how the actor will be accessing the different classes to carry out various tasks and functions.

- In Fig above, the program has only one actor, i.e., the manager.
- There are 4 Functions namely – Login, Check-in, Check-out, and Display menu.
- Manager has to login to proceed furthur in the code, for which Manager class is used.
- After that they can select check-in or check-out, according to the customer, through the Reception class.
- They can also choose to display menu to the customer, if the customer wants to eat, through the Restaurant class.

•

## CLASS DIAGRAM

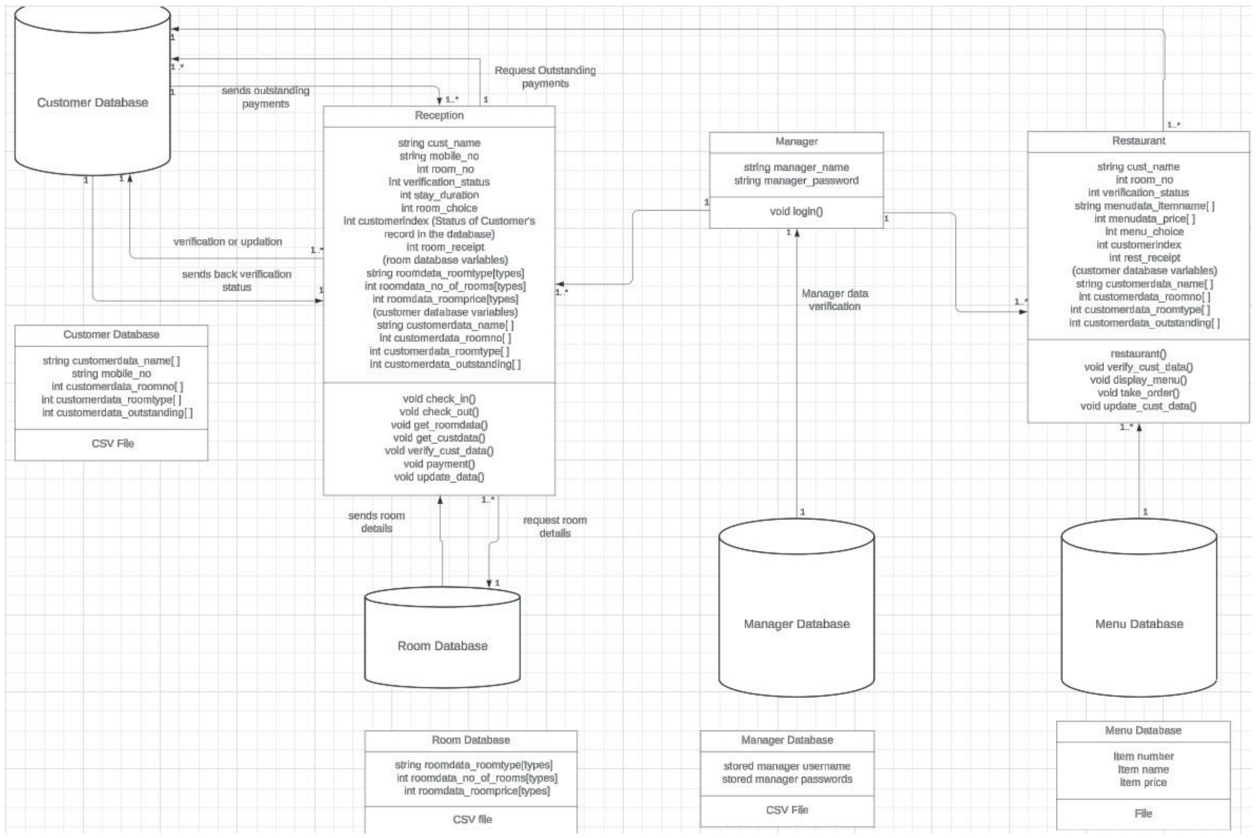


*Figure 4Class diagram*

Class Diagram is a crucial UML diagram which enables us to state and identify what functionalities the specific classes will be carrying out using which data members. It serves as a base diagram for sequence diagram; the flow is indicated using the very member functions mentioned in the Class Diagram.
We have 3 Classes namely – Manager, Reception, Restaurant – each having their own data members and member functions.

1. In Manager Class
    a. Manager logins into the system using login() function; manager enters username and password which is then verified using the manager database
    b. Manager gets outstanding payment from customer database and takes payment from the customer using payment()
2. In Reception Class
    a. Firstly, manager asks customer if they want to check-in() or check-out()
    b. In both check-in and check-out functions, the customer datais entered (name, mobile no, room no) is verified using verify_cust_data()
    c. Then the types of room (room_type) and respective prices (room_price) are displayed using get_roomdata()

   d. Customer room choice (room_choice) and duration of stay (stay_duration) is taken and data is updated in customer and room database using updata_data().

3. In Restaurant Class
   a. Customer name (cust_name) and room number (room_no) is verified from customer database using verify_cust_data()
   b. Restaurant Menu is retrieved from menu database and displayed using display_menu()
   c. Order is taken using menu_choice, and then delivered and the total bill (rest_receipt) is added to outstanding payments in customer database.

## ACTIVITY DIAGRAMS

Activity Diagrams are diagrammatical representation of algorithm; it gives clear idea of the flow of the code according to the choices made by the user (manager). It includes all the parallel processes ,conditional statements, decisions, their consequences and flow of data according to it.

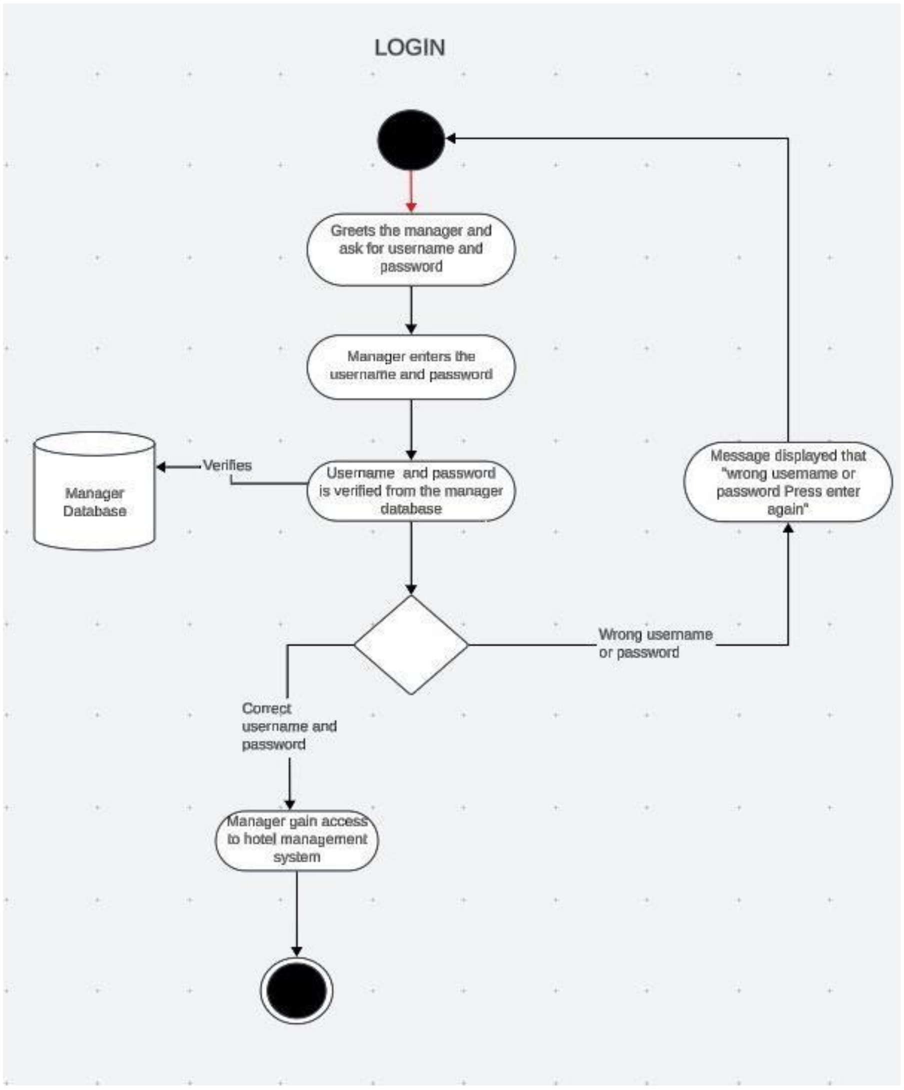

LOGIN
Greets the manager and ask for username and password
Manager enters the username and password
Username and password is verified from the manager database
Verifies
Manager Database
Message displayed that "wrong username or password Press enter again"
Wrong username or password
Correct username and password
Manager gain access to hotel management system

*Figure 5 activity diagram for login*

Above activity diagram illustrates the login process. The manager is asked to enter username and password for verification with the data in Manager Database.

- If verified, the manager is granted access to other functions and class.
- If not verified, the manager is given another 4 tries to verify himself with the system. After that, the program/system is automatically exited.

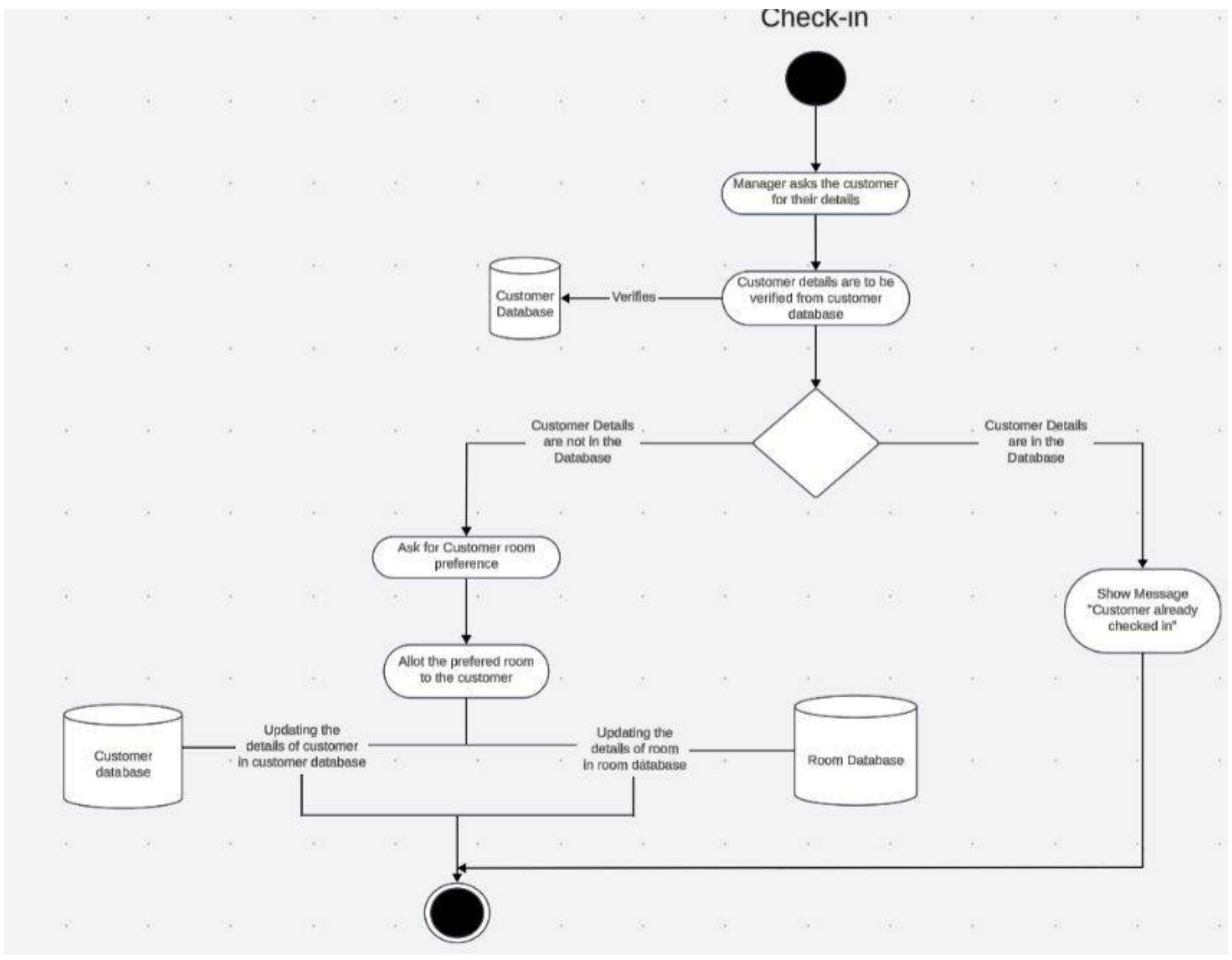

Check-in
Manager asks the customer for their details
Customer details are to be verified from customer database
Verifies
Customer Database
Customer Details are not in the Database
Customer Details are in the Database
Ask for Customer room preference
Show Message "Customer already checked in"
Allot the prefered room to the customer
Customer database
Updating the details of customer in customer database
Updating the details of room in room database
Room Database

*Figure 6 Reception Check in*

In the above activity diagram. We have displayed the check-in process. Customer details are entered and verified with the system.

- If customer isn't in the Customer database, then they are asked their choice of room after presenting the type of room available along with their charge per night. Room is allotted to the customer according to their choice and data is updated in customer and room database.
- If customer is in the database, an error message is displayed and customer is again given the choice between check-in and check-out.

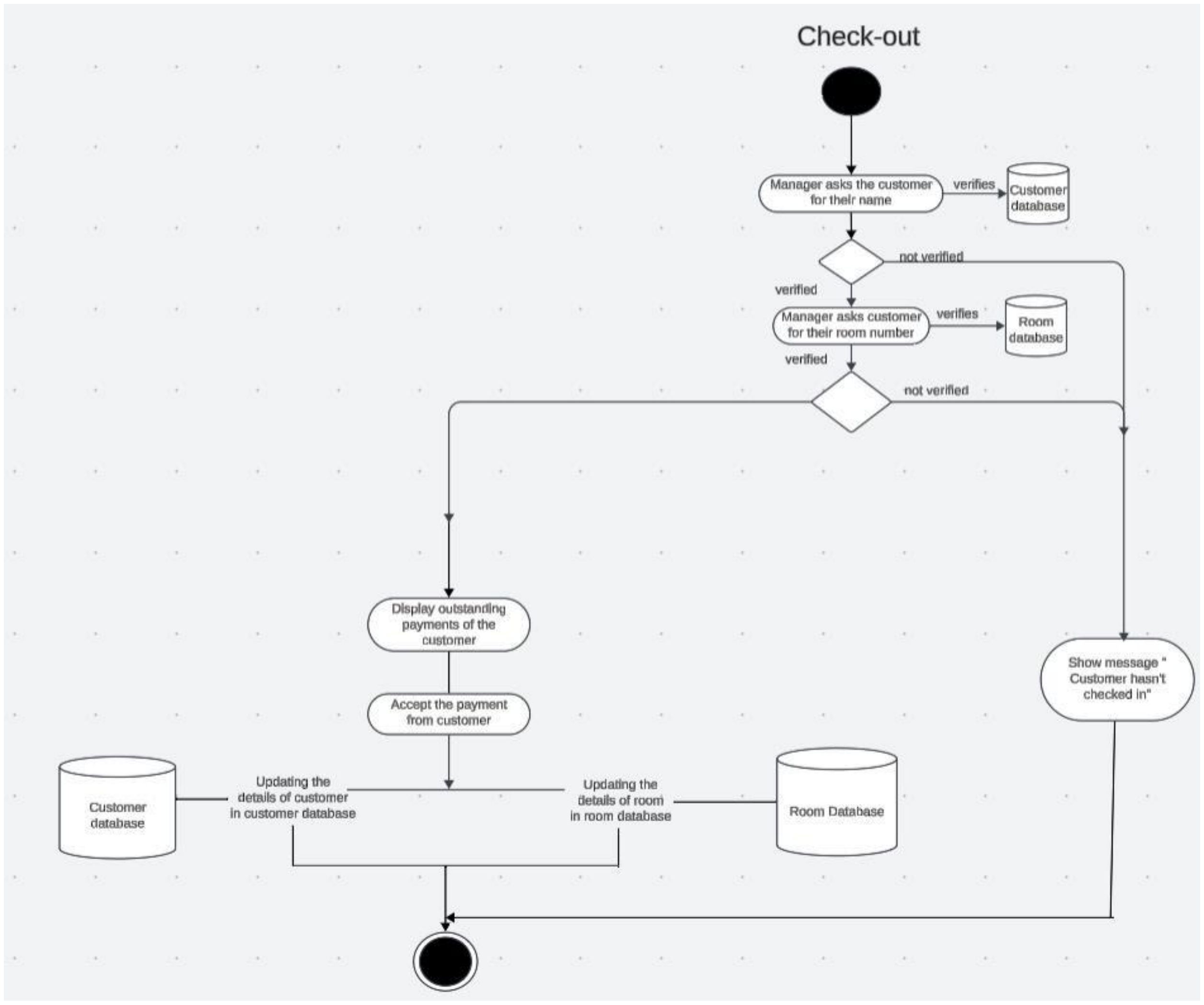

Check-out
Manager asks the customer for their name
verifies
Customer database
not verified
verified
Manager asks customer for their room number
verifies
Room database
verified
not verified
Display outstanding payments of the customer
Accept the payment from customer
Show message " Customer hasn't checked in"
Customer database
Updating the details of customer in customer database
Updating the details of room in room database
Room Database

*Figure 7Reception checkout*

In the above activity diagram. We have displayed the check-out process. Customer details (name, mobile no, room no) are entered and verified with the system.

- If customer is in the Customer database, then they are displayed the total outstanding payments and the payment is then received from them. Subsequently, Customer and Room database are updated.
- If customer isn't in the database, an error message is displayed and customer is again given the choice between check-in and check-out.

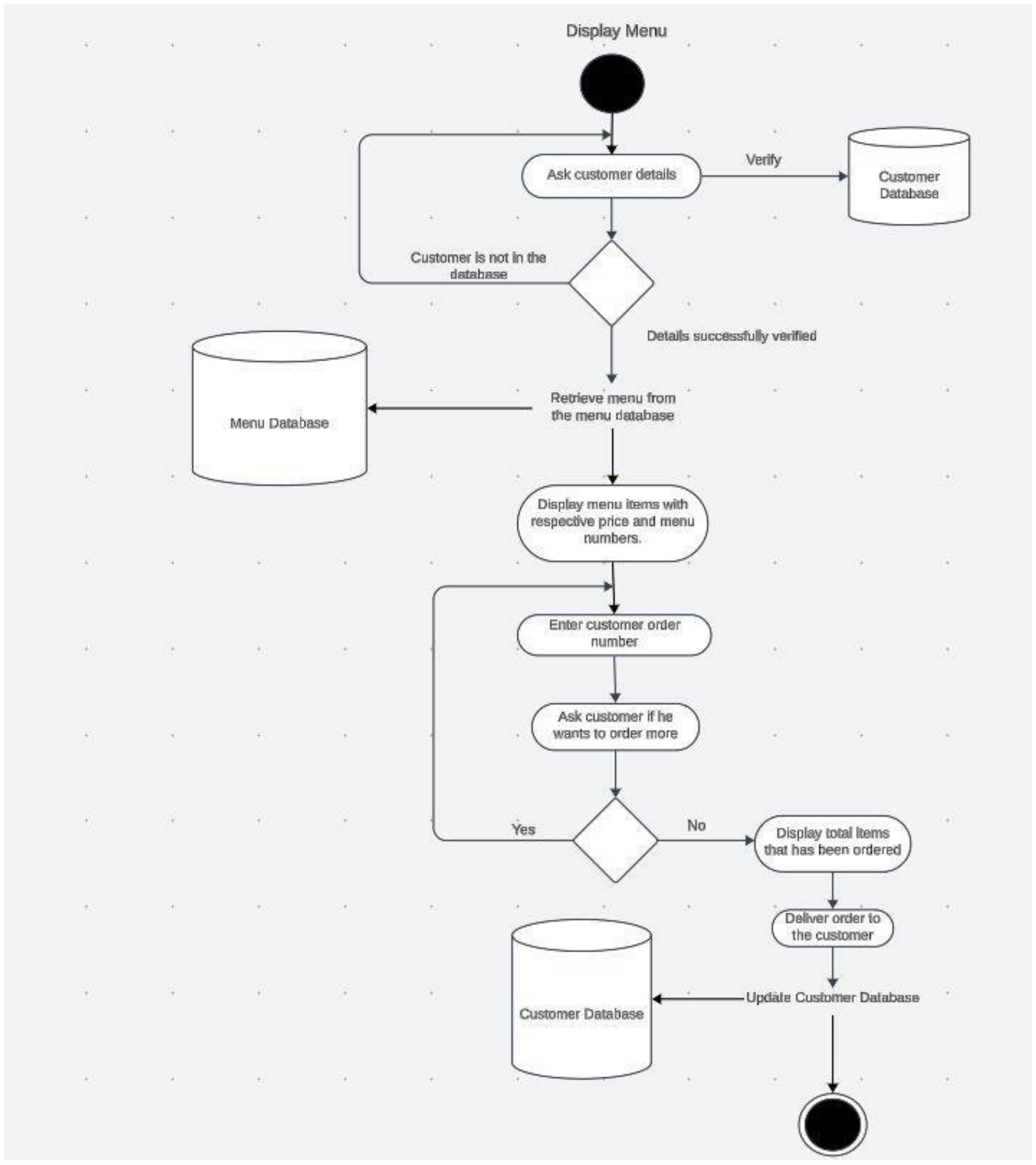

Display Menu
Ask customer details
Verify
Customer Database
Customer is not in the database
Details successfully verified
Menu Database
Retrieve menu from the menu database
Display menu items with respective price and menu numbers.
Enter customer order number
Ask customer if he wants to order more
Yes
No
Display total items that has been ordered
Deliver order to the customer
Customer Database
Update Customer Database

*Figure 8Restraunt activity diagram*

The above activity diagram shows the course of restaurant class, when the customer wants to order food. Firstly, customer details are taken and verified with the customer database. After that Customer is Presented with a wide variety of items to choose from, along with their prices. Customer can order as many items as it wants, and only after their order is complete, the order is delivered, and the total bill of the food is added to the outstanding payments in Customer Database.

## SEQUENCE DIAGRAM

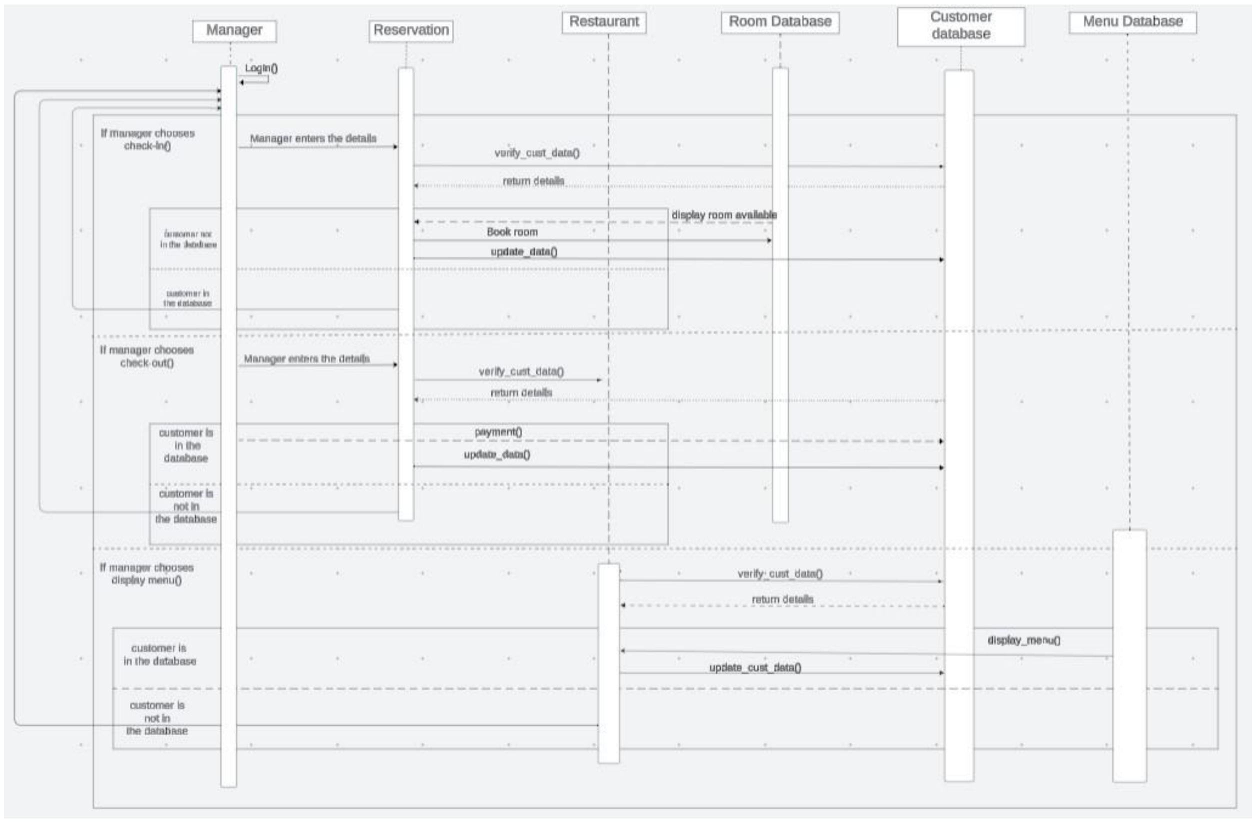


*Figure 9 Sequence diagram of the project*

A sequence diagram is a type of interaction diagram used in software engineering and system modeling to illustrate the interactions between different parts of a system or between a system and its users. It depicts the processes and objects involved in the interaction, along with the messages exchanged between them, arranged in a time sequence.

## OBJECT ORIENTED PROGRAMMING TO BE IMPLEMENTED IN CODE WITH JUSTIFICATION

### Concepts used

In the context of developing a hotel management system, several key OOPD concepts are critical in ensuring the efficiency and security of the software. This section provides a justification for the implementation of these OOP principles within the codebase:

1. Namespace: It is used in the code "using namespace std". It allows us to use the names defined in the std namespace without having to use std::. The std namespace is the standard library namespace and it contains most of the basic functions.
2. Class: It is the backbone of Object-Oriented Programming. It is user-defined data along with data members and member functions. It is used to hold data and functions to be called by the object. [2]
3. Object: It is the instance of a class. It is used to call the member functions of a class. Without this most of the program will not run. [2]
4. Abstraction: Abstraction is achieved by defining classes and only showing what is required to the user. In the case of the project all the code and the major chunk of processing is hidden and the user only sees what is required for the program to run. [3]
5. File Handling: File Handling is the major part of the project as it is used to store the data for login, notes, and passwords permanently on the system which can be accessible whenever the user logs in. [4]
6. Encapsulation: It is demonstrated by using the classes such as 'RESERVATION MODULE' and 'Manager' where the information of the customers and login details of the manager are secured. This provides controlled access of the details of the classes. [5]
7. Destructors: It is a good habit to apply a destructor. A destructor is a special member function of a class that is called when an object of that class goes out of scope or when the delete operator is used to explicitly delete an obj. [6]
8. Friend class: Friend classes provide a mechanism for granting privileged access to certain classes, enabling efficient collaboration and implementation of specific functionalities within a software system.

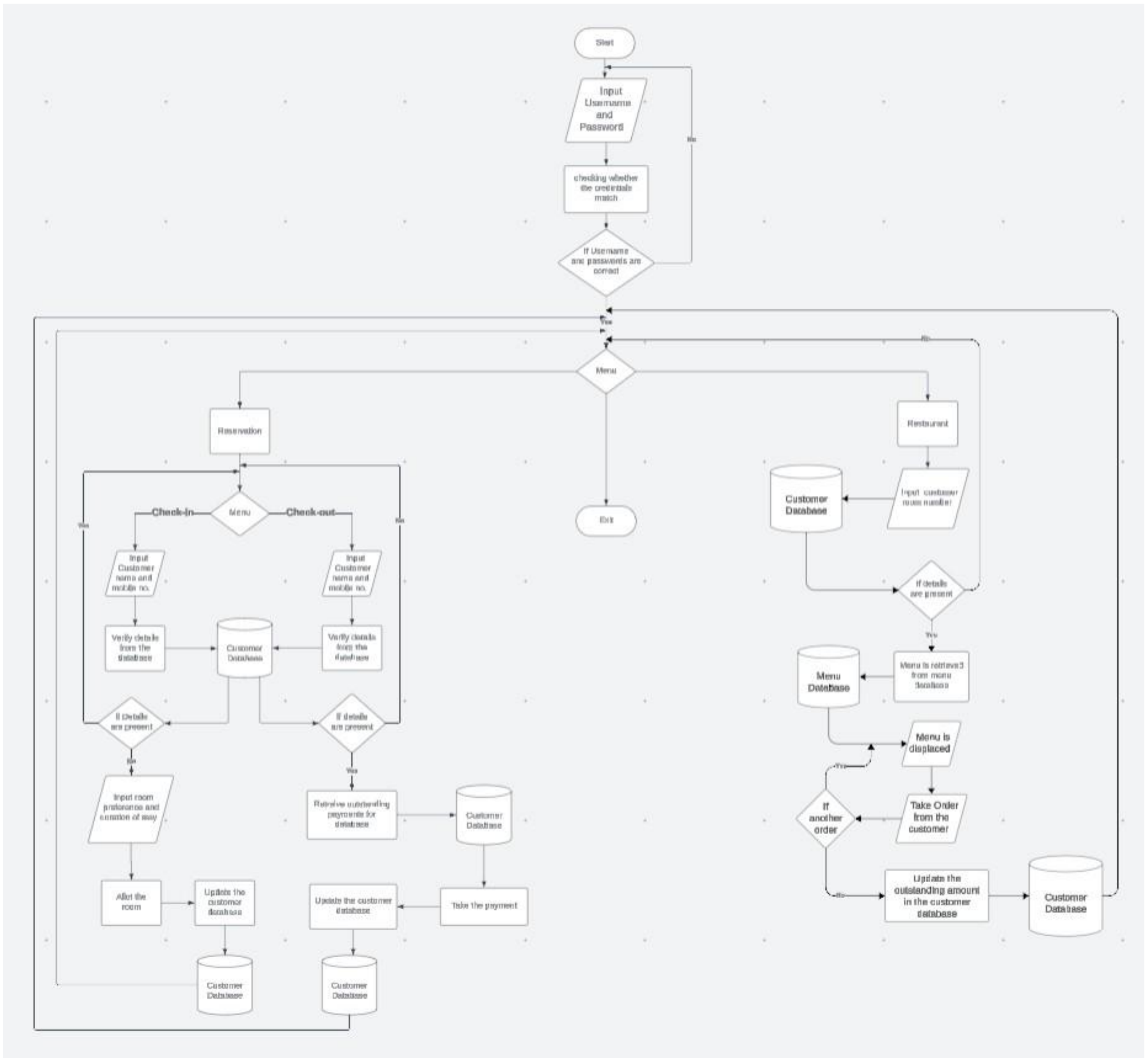
## ALGORITHM

1. START
2. Welcomes the user(manager) and displays login screen.
3. The manager enters the username and password for his/her verification.
4. If the username and password are correct then go to step 5 and if either of the username or password is incorrect then go to step 2.
5. The user is given a choice to select any of the following module
   - Reception module (Go to step 6)
   - Restaurant module (Go to step 21)
   - Exit Program (Go to step 29)
6. Give option check-in or check-out. If check-in, go to step 7. Else go to step 14
7. Ask customer for their details. (name)
8. Check if customer details are in the database:
   - If yes, then go to step 9
   - If no, then go to step 6
9. Ask the customer for their details such as mobile no.
10. Display room brochure
11. Enter customer choice and check availability.
    - If available, go to step 12
    - If not available, go to step 10
12. Ask customer for their details. (duration,)
13. Allot the room
14. Room recipt display
15. Update data room and customer
16. Ask customer for their details. (name)
    - If correct go to step17
    - If not then go to step16
17. Ask customer for their details (room no)
    - If correct, go step 18
    - If not go to step 17
18. Display his/her outstanding payments
19. Update Customer Database after taking payment.
20. Wish them good day!
    - (go to step 5)

21. User verifies the customer room number and full name from the database
    - If the details are right, go to step 22
    - If the details are wrong go step 21
22. Menu is displayed, after retrieving it from Menu Database
23. User takes the customer's order and enters it in the program
24. User is asked for ‘another order’ or if ‘order is over’.
25. If another order:

```
        • Go to step 192
        • Else if order is over:
        • Go to step 26
26. The bill is stored as outstanding payment in customer database.
27. Update customer data base
28. Restaurant Module is exited (Go to step 5)
29. Print an Exit program statement
30. END
```

## FLOWCHART


Start
Input Username and Password
checking whether the credentials match
If Username and passwords are correct
No
Yes
Menu
Reservation
Restaurant
Exit
Menu
Check-in
Check-out
Input Customer name and mobile no.
Input Customer name and mobile no.
Verify details from the database
Customer Database
Verify details from the database
If Details are present
If details are present
Yes
No
Input room preference and duration of stay
Retrieve outstanding payments for database
Customer Database
Allot the room
Update the customer database
Update the customer database
Take the payment
Customer Database
Customer Database
Customer Database
Input customer room number
If details are present
Menu Database
Menu is retrieved from menu database
Menu is displaced
If another order
Take Order from the customer
Update the outstanding amount in the customer database
Customer Database

A flowchart is a visual representation of a process or algorithm that uses a series of symbols and connecting arrows to show the steps involved and the decision points that need to be made.

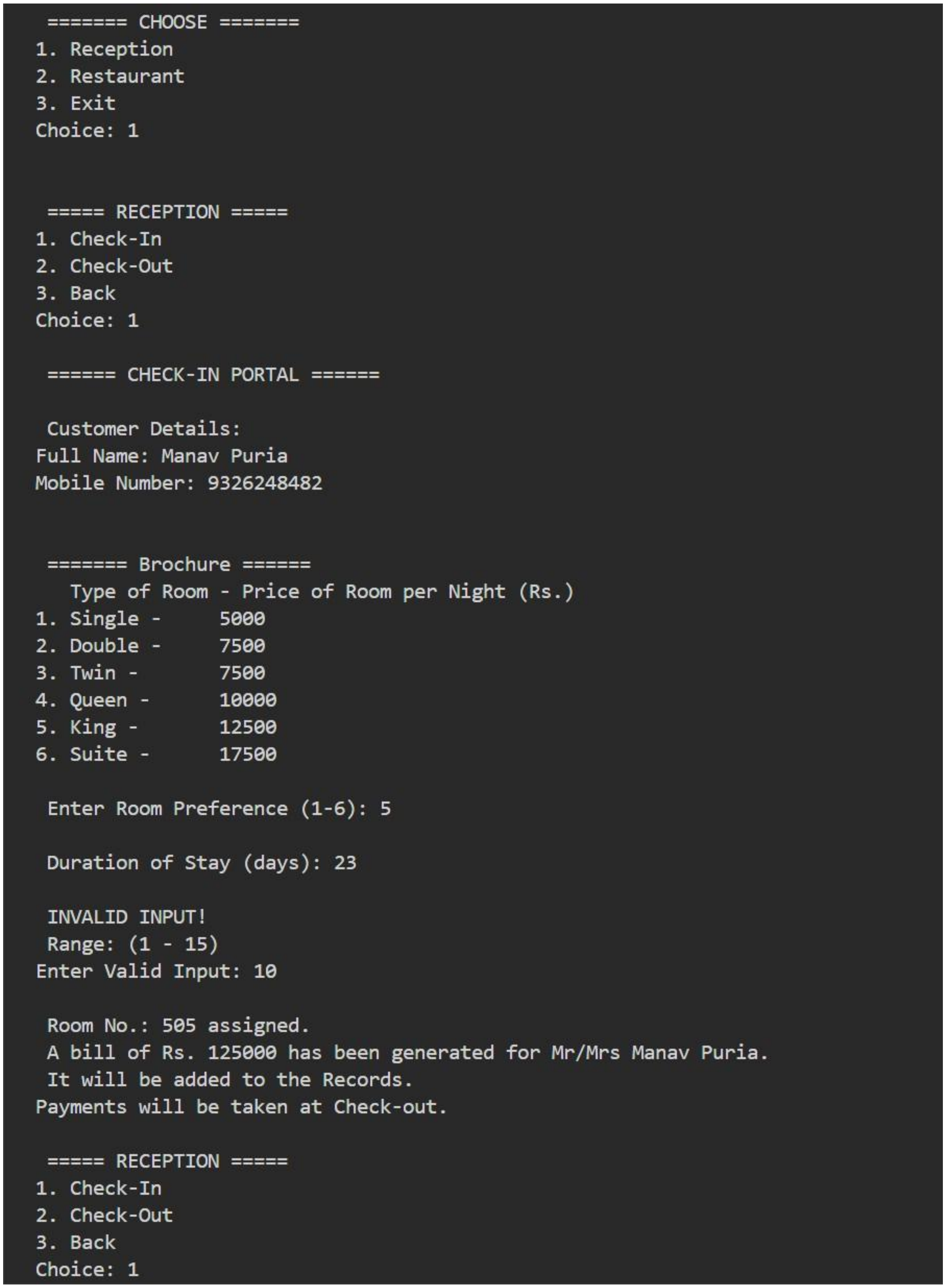
## CODE WRITTEN IN C++

```
#include<iostream>
#include<string>
#include<fstream>
#include<sstream>
using namespace std;
//manager database: name password
//room database: room_type, room per night charge, No of Rooms
//menu database: item number, item name, item price
//customer database: cust name, cust room, cust o/s payment
template <typename T>
T valid(T low, T high, T input){
    while(input<low || input>high){
        cout<<"\n INVALID INPUT! ";
              cout<<"\n Range: ("<<low<<" - " <<high<< ")\nEnter Valid Input: ";
        cin>>input;
    }
    return input;
}
class manager{
    string manager_name, manager_password;
    public:
        void login()
        {
            int i;
            string username, password;
            cout<<"\n ====== LOGIN ====== \n Username: ";
            cin>>manager_name;
            cout<<"Password: ";
            cin>>manager_password;
            ifstream ifile;
            ifile.open("Manager_Database.txt");
            if(ifile.fail()){
                cout<<"\n Login Failed due to file error. Please check the file and run the code
again! ";
                exit(1);
            }
            else{
                ifile>>username>>password;
            }
            i = 5;
            while(i != 0){
                if(manager_name==username && manager_password== password){
                    cout<<"\n LOGIN SUCCESSFUL! \n Welcome Manager!";
                    break;
                }
                else{
                    i--;
                    if (i==0){
                        cout<<"\n UNABLE TO LOGIN. Session Ended.";
                        exit(1);
                    }
                    cout<<"\n Username or password incorrect. Please try again! \n "<<i<<"
tries remaining \n Username: ";
                    cin>>manager_name;
                    cout<<"Password: ";
                    cin>>manager_password;
                }
        }
    }
};
const int types = 6; //types of rooms
class reception{
    string cust_name, mobile_no;
    int room_no, verification_status, stay_duration, room_choice, customerindex; //Status
of Customer's record in the database
    int room_receipt;
    //room database variables
    string roomdata_roomtype[types];
    int roomdata_no_of_rooms[types] = {0};
    int roomdata_roomprice[types]= {0};
    //customer database variables
    string customerdata_name[51];
    int customerdata_roomno[51] = {0}, customerdata_roomtype[51];
    int customerdata_outstanding[51] = {0};
    public:
        void check_in(){
            verification_status = 0;
            cout<<"\n ====== CHECK-IN PORTAL ====== \n";
            cout<<"\n Customer Details: ";
            cout<<"\nFull Name: ";
            cin.ignore(); //to clear any whitespace from the console.
            getline(cin, cust_name);
            verify_cust_data(1); //verifies if the customer has already checked-in or not.
            if (verification_status == 1){
                cout<<"\n Customer has already checked-in!! ";
                return;
            }
            cout<<"Mobile Number: ";
            getline(cin, mobile_no);
            RoomBrochure: //Destination for the Brochure
            cout<<" \n \n ======= Brochure ====== ";
            get_roomdata();
            cout<<"\n   Type of Room - Price of Room per Night (Rs.) "<<endl;
            for(int i = 0; i < types; i++){
                cout<<i+1<<". "<<roomdata_roomtype[i]<<" -
\t"<<roomdata_roomprice[i]<<endl;
            }
            cout<<"\n Enter Room Preference (1-6): ";
            cin>>room_choice;
            room_choice = valid(1, 6, room_choice);
            room_choice = room_choice - 1;
            if (roomdata_no_of_rooms[room_choice] == 0){
                int choice;
                cout<<"\n Room unavailable at the moment! \n1. Choose another Type of
Room \n2. Exit \n Choice: ";
                cin>>choice;
                choice = valid(1,2,choice);
                if(choice == 1){
                    goto RoomBrochure;
                }
                else{
                    return;
                }
            }
            cout<<"\n Duration of Stay (days): ";
            cin>>stay_duration;
            stay_duration = valid(1, 15, stay_duration);
            room_no = ((room_choice+1)*100)+roomdata_no_of_rooms[room_choice];
//generates room number
            cout<<"\n Room No.: "<<room_no<<" assigned.";
            roomdata_no_of_rooms[room_choice] = roomdata_no_of_rooms[room_choice] -
1; //updates the number of rooms available
            room_receipt = stay_duration * roomdata_roomprice[room_choice]; //generates
bill
            cout<<"\n A bill of Rs. "<<room_receipt<<" has been generated for Mr/Mrs
"<<cust_name<<". \n It will be added to the Records. \nPayments will be taken at Check-
out.";
            update_data(1); //room data replaced
            update_data(3); //customer data appended
        }
        void get_roomdata(){
            string line = "", tempstring = "";
            int n = 0;
            ifstream room;
            room.open("Room_Database.csv");
            if(room.fail()){
                cout<<"\n Login Failed due to file error. Please check the file and run the code 
again! ";
                exit(1);
            }
            else{
                while( getline(room, line)){
                    stringstream Inputstring(line);

                    getline(Inputstring, roomdata_roomtype[n], ',');
                    getline(Inputstring, tempstring, ',');
                    roomdata_roomprice[n] = atoi(tempstring.c_str());
                    tempstring = "";
                    getline(Inputstring, tempstring, ',');
                    roomdata_no_of_rooms[n] = atoi(tempstring.c_str());
                    n++;
                }
            }
            room.close();
        }
        void get_custdata(){
            string line = "", tempstring = "";
            int n = 0;
            ifstream customer;
            customer.open("Customer_Database.csv");
            if(customer.fail()){
                cout<<"\n Login Failed due to file error. Please check the file and run the code 
again! ";
                exit(1);
            }
            else{
                while( getline(customer, line)){
                    stringstream Inputstring(line);

                    getline(Inputstring, customerdata_name[n], ',');
                    getline(Inputstring, tempstring, ',');
                    customerdata_roomno[n] = atoi(tempstring.c_str());
                    tempstring = "";
                    getline(Inputstring, tempstring, ',');
                    customerdata_outstanding[n] = atoi(tempstring.c_str());
                    tempstring = "";
                    getline(Inputstring, tempstring, ',');
                    customerdata_roomtype[n] = atoi(tempstring.c_str());
                    n++;
                }
            }
            customer.close();
        }
        void check_out(){
            cout<<"\n ====== CHECK-OUT PORTAL ====== \n";
            cout<<"\n Customer Details: ";
            cout<<"\nFull Name: ";
            cin.ignore(); //to clear any whitespace from the console.
            getline(cin, cust_name);
            verify_cust_data(1); //verifies if the customer has checked-in or not.
            if (verification_status == 0){
                cout<<"\n Customer not found in Records! ";
                return;
            }
            cout<<"\nRoom No.: ";
            cin>>room_no;
            verify_cust_data(2);
            if(verification_status == 0){
                cout<<"\n Incorrect Room No. Try Again. ";
                return;
            }
            cout<<"\n Outstanding Payment: Rs. "<<room_receipt;
            //re-direct to payment function
            cout<<"\n Updating Records.";
            //get_roomdata(); //for some reason getting stuck here
            roomdata_no_of_rooms[0] = roomdata_no_of_rooms[0] + 1; //updating Room
Database
            update_data(1);
            update_data(2);

        }
        void verify_cust_data(int verification_type){
            get_custdata();
            verification_status = 0;
            if(verification_type == 1){ //check-in verification
                for(int i = 0; i<51; i++){
                    if(customerdata_name[i] == cust_name){
                        verification_status = 1; return;
                    }
                }
            }
            if(verification_type == 2){ //check-out verification
                for(int i = 0; i<51; i++){
                    if(customerdata_name[i] == cust_name && customerdata_roomno[i] ==
room_no){
                        verification_status = 1; customerindex = i;
                        room_choice = customerdata_roomtype[i]; //the room type is stored for
updating room records
                        return;
                    }
                }
            }
        }
        void update_data(int update_type) {
            if (update_type == 1) {
                // Update room database
                ofstream room("Room_Database.csv");
                if (!room) {
                    cout << "Error: Failed to open Room_Database.csv for writing!" << endl;
                    exit(1);
                }
                for (int i = 0; i < types; i++) {
                    room << roomdata_roomtype[i] << ", " << roomdata_roomprice[i] << ", "
<< roomdata_no_of_rooms[i] << "\n";
                }
                room.close();
            } else if (update_type == 3) {
                // Append to customer database
                ofstream customer("Customer_Database.csv", ios::app);
                if (!customer) {
                    cout << "Error: Failed to open Customer_Database.csv for writing!" << endl;
                    exit(1);
                }
                customer << cust_name << ", " << room_no << ", " << room_receipt << ", " <<
room_choice << ", " << mobile_no << "\n";
                customer.close();
            } else if (update_type == 2) {
                // Remove customer from customer database
                ifstream infile("Customer_Database.csv");
                ofstream outfile("temp.csv");
                if (!infile || !outfile) {
                    cout << "Error: Failed to open files for processing!" << endl;
                    exit(1);
                }

                string line;
                int count = 0;
                while (getline(infile, line)) {
                    count++;
                    if (count != customerindex) {
                        outfile << line << endl;
                    }
                }

                infile.close();
                outfile.close();

                // Remove original file and rename temp file
                remove("Customer_Database.csv");
                if (rename("temp.csv", "Customer_Database.csv") != 0) {
                    cout << "Error: Failed to rename temp.csv to Customer_Database.csv!" <<
endl;
                    exit(1);
                }
            }
        }
        void payment();
};
class restaurant{
    string cust_name;
    int room_no, verification_status = 0;
    string menudata_itemname[54];
    int menudata_price[54];
    int menu_choice, customerindex;
    int rest_receipt = 0;
    //customer database variables
    string customerdata_name[51];
    int customerdata_roomno[51] = {0}, customerdata_roomtype[51];
    int customerdata_outstanding[51] = {0};
    public:
        restaurant(){
            string line = "", tempstring = "";
            int n = 0;
            ifstream customer;
            customer.open("Customer_Database.csv");
            if(customer.fail()){
                cout<<"\n Login Failed due to file error. Please check the file and run the code
again! ";
                exit(1);
            }
            else{
                while(getline(customer, line)){
                    stringstream Inputstring(line);

                    getline(Inputstring, customerdata_name[n], ',');
                    getline(Inputstring, tempstring, ',');
                    customerdata_roomno[n] = atoi(tempstring.c_str());
                    tempstring = "";
                    getline(Inputstring, tempstring, ',');
                    customerdata_outstanding[n] = atoi(tempstring.c_str());
                    tempstring = "";
                    getline(Inputstring, tempstring, ',');
                    customerdata_roomtype[n] = atoi(tempstring.c_str());
                    n++;
                }
            }
            customer.close();
        }
        void display_menu(){
            string line = "", tempstring = ""; int n = 0;
            ifstream menu;
            menu.open("Restaurant_Database.csv");
            if(menu.fail()){
                cout<<"\n Login Failed due to file error. Please check the file and run the code
again! ";
                exit(1);
            }
            else{
                while( getline(menu, line)){
                    stringstream Inputstring(line);

                    getline(Inputstring, menudata_itemname[n], ',');
                    getline(Inputstring, tempstring, ',');
                    menudata_price[n] = atoi(tempstring.c_str());
                    n++;
                }
            }
            menu.close();
            cout<<"\n ====== MENU ====== \n";
            cout<<"Item No. | Item Name | Price\n";
            for(int i = 0; i < 54; i++){
                cout<<i+1<<". \t"<<menudata_itemname[i]<<" \t -
"<<menudata_price[i]<<endl;
            }
        }
        void verify_cust_data(){
            for(int i = 0; i<51; i++){
                    if(customerdata_name[i] == cust_name && customerdata_roomno[i] ==
room_no){
                    verification_status = 1; customerindex = i;
                    return;
                    }
        }
        }
```

```
        void take_order(){
            cout<<"====== RESTAURANT ======";
            cout<<"\n \nCustomer Full Name: ";
            cin.ignore(); //to clear any whitespace from the console.
            getline(cin, cust_name);
            cout<<"\n Enter Room No.: ";
            cin>>room_no;
            verification_status = 1; //bypassing
            if (verification_status == 0){ //not working
                cout<<"\n Customer name or Room No. Incorrect. Please Try Again.";
                return;
            }
            display_menu();
            rest_receipt = 0;
            do{
                cout<<"\n Order (item no.): ";
                cin>>menu_choice;
                menu_choice = valid(1, 54, menu_choice);
                rest_receipt = rest_receipt + menudata_price[menu_choice];
                cout<<"\n Press: \nAny number: Another Order | 0: Proceed to Billing ";
                cin>>menu_choice;
            }while (menu_choice != 0);
        cout<<"\n Restaurant Bill: Rs."<<rest_receipt;
        cout<<"\n Will be added to the Records and final payment will be recieved at
Check-Out.";
        update_cust_data();
        }
        void update_cust_data(){
            ifstream infile("Customer_Database.csv");
            ofstream outfile("temp.csv");
            if (!infile || !outfile) {
                cout << "Error: Failed to open files for processing!" << endl;
                exit(1);
            }

            string line;
            while (getline(infile, line)) {
                string name, roomno_str, outstanding_str, roomtype_str;
                stringstream ss(line);
                getline(ss, name, ',');
                getline(ss, roomno_str, ',');
                getline(ss, outstanding_str, ',');
                getline(ss, roomtype_str, ',');

                // Convert room number and outstanding payment to integers
                int roomno = stoi(roomno_str);
                int outstanding = stoi(outstanding_str);
                int roomtype = stoi(roomtype_str);

                // Update outstanding payment for the specified customer
                if (name == cust_name && roomno == room_no) {
                    outstanding += rest_receipt; // Add restaurant bill to outstanding payment
                }

                // Write the updated record to the temporary file
                outfile << name << ", " << roomno << ", " << outstanding << ", " << roomtype
<< "\n";
            }

            infile.close();
            outfile.close();

            // Replace original file with the temporary file
            remove("Customer_Database.csv");
            if (rename("temp.csv", "Customer_Database.csv") != 0) {
                cout << "Error: Failed to rename temp.csv to Customer_Database.csv!" <<
endl;
                exit(1);
            }
        }
};
int main()
{
    int main_choice, reception_choice;
    cout<<"\n ======== ______________HOTEL's MANAGER PORTAL
========"<<endl;
    manager m;
    restaurant rest;
    reception r;
    m.login();
    main_menu:
    cout<<"\n \n \n ======= CHOOSE ======= \n1. Reception \n2. Restaurant \n3. Exit
\nChoice: ";
    cin>>main_choice;
    switch (main_choice)
    {
        case 1:
            reception_menu:
            cout<<"\n \n ===== RECEPTION ===== \n1. Check-In \n2. Check-Out \n3.
Back \nChoice: ";
            cin>>reception_choice;
            switch (reception_choice)
```

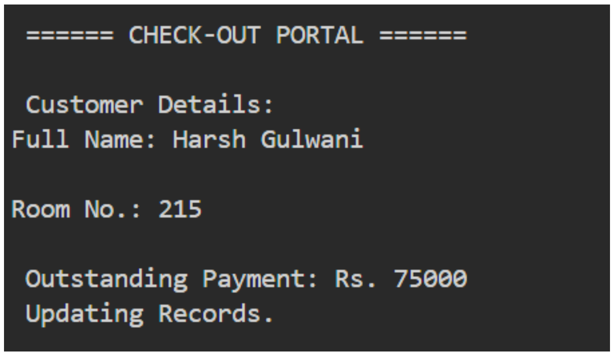
```
          {
            case 1:
              r.check_in();
              goto reception_menu;
            case 2:
              r.check_out();
              goto reception_menu;
            case 3:
              cout<<"\n Exiting Reception...";
              goto main_menu;
            default:
              cout<<"\n INVALID INPUT! \n";
              goto reception_menu;
          }
          goto main_menu;
        case 2:
          rest.take_order();
          goto main_menu;
        case 3:
          cout<<"\n ========== EXITING CODE ==========";
          exit(1);
        default:
          cout<<"\n INVALID INPUT! \n";
          goto main_menu;
      }
      return 0;
    }
```

## OUTPUT SCREENSHOT OF ALL FUNCTIONS



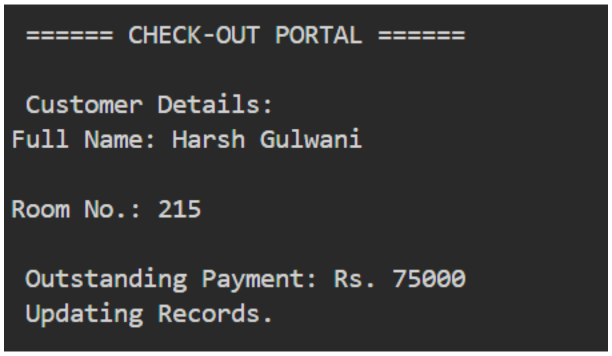
```
 ======= CHOOSE =======
1. Reception
2. Restaurant
3. Exit
Choice: 1


 ===== RECEPTION =====
1. Check-In
2. Check-Out
3. Back
Choice: 1

 ====== CHECK-IN PORTAL ======

 Customer Details:
Full Name: Manav Puria
Mobile Number: 9326248482


 ======= Brochure ======
   Type of Room - Price of Room per Night (Rs.)
1. Single -     5000
2. Double -     7500
3. Twin -       7500
4. Queen -      10000
5. King -       12500
6. Suite -      17500

 Enter Room Preference (1-6): 5

 Duration of Stay (days): 23

 INVALID INPUT!
 Range: (1 - 15)
Enter Valid Input: 10

 Room No.: 505 assigned.
 A bill of Rs. 125000 has been generated for Mr/Mrs Manav Puria.
 It will be added to the Records.
Payments will be taken at Check-out.

 ===== RECEPTION =====
1. Check-In
2. Check-Out
3. Back
Choice: 1
```

*Figure 10check in*



```
 ====== CHECK-OUT PORTAL ======

 Customer Details:
Full Name: Harsh Gulwani

Room No.: 215

 Outstanding Payment: Rs. 75000
 Updating Records.
```

*Figure 11check in*



```
 ====== CHECK-OUT PORTAL ======

 Customer Details:
Full Name: Harsh Gulwani

Room No.: 215

 Outstanding Payment: Rs. 75000
 Updating Records.
```

*Figure 12check out*

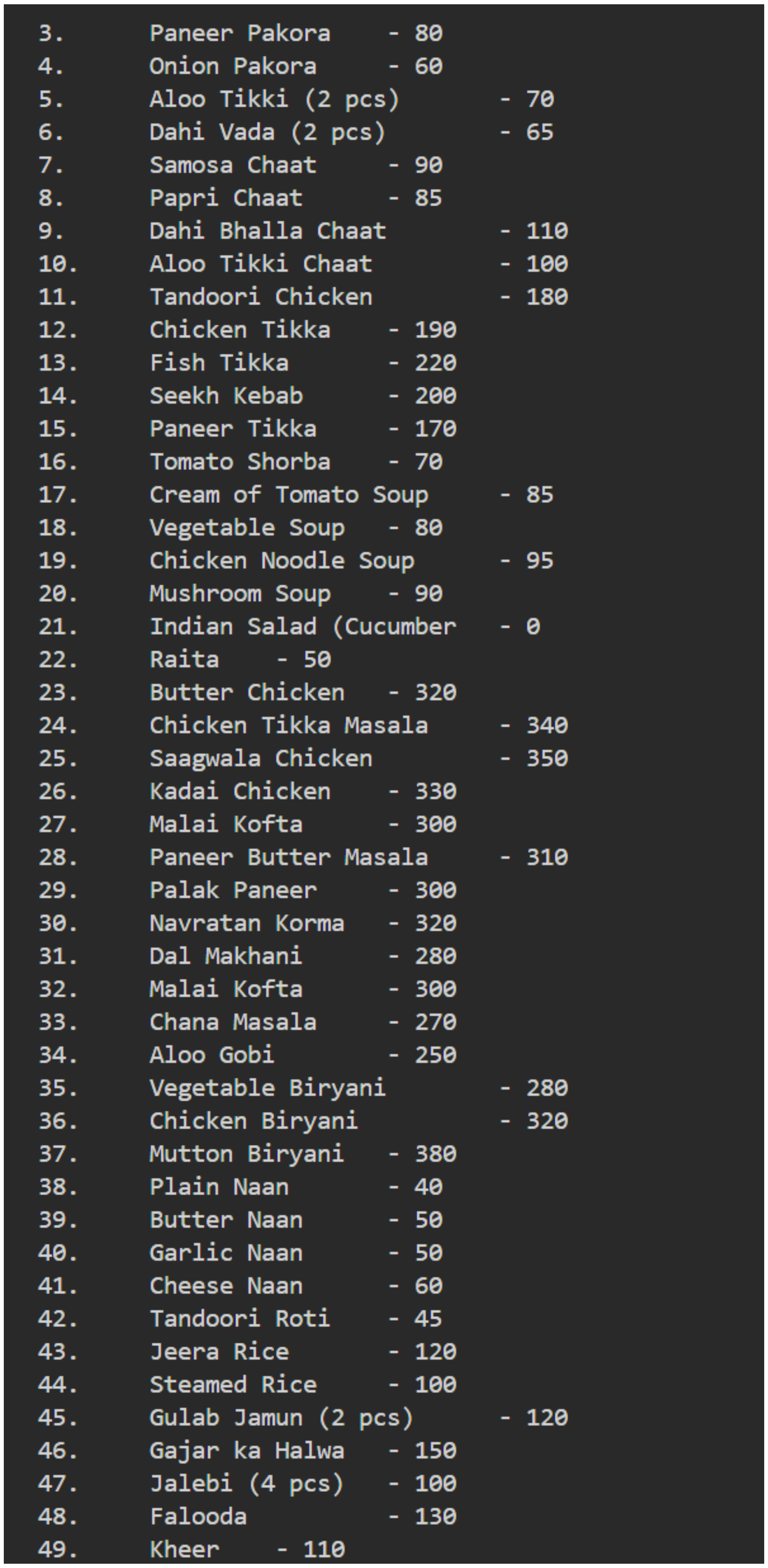
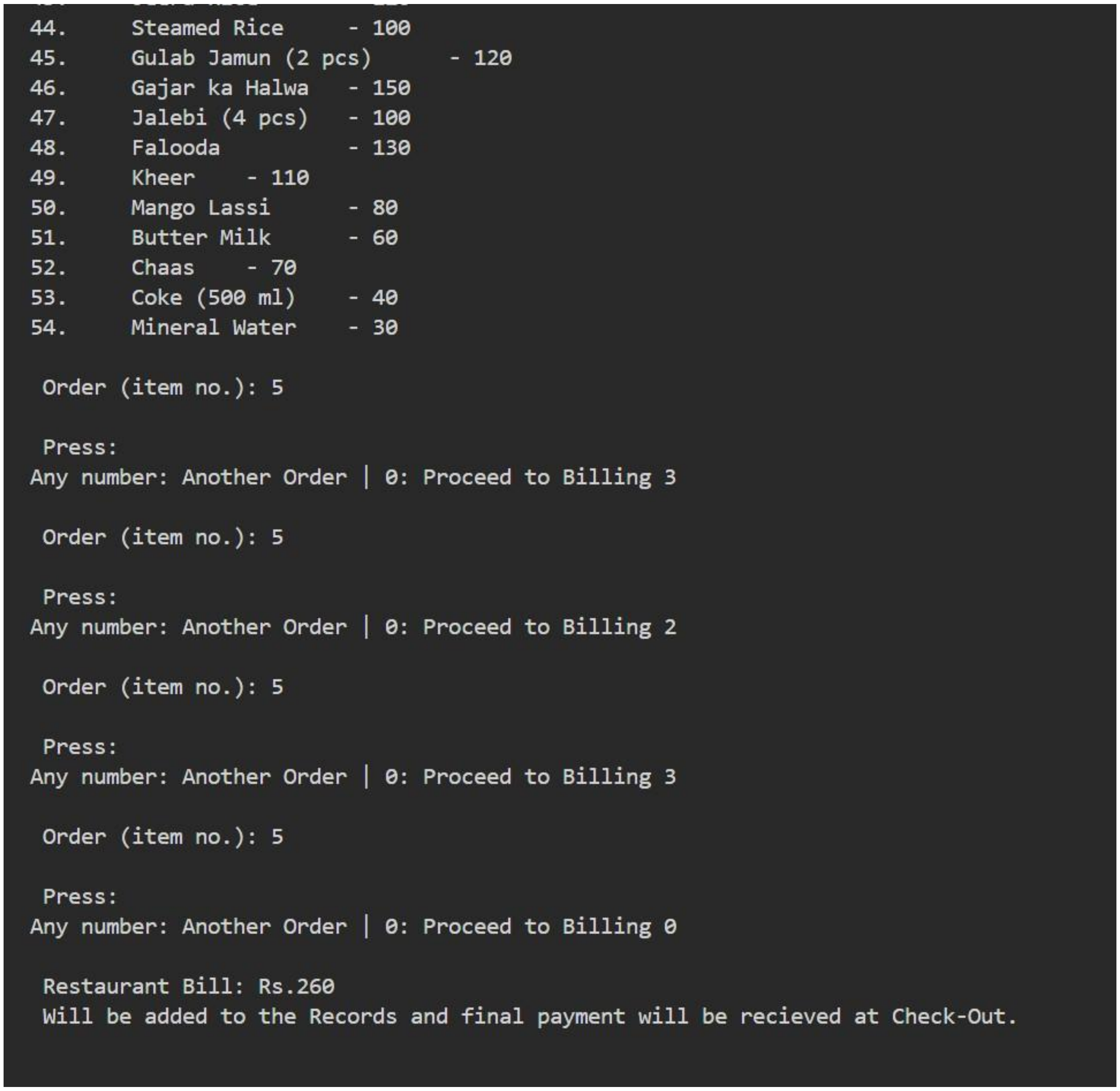
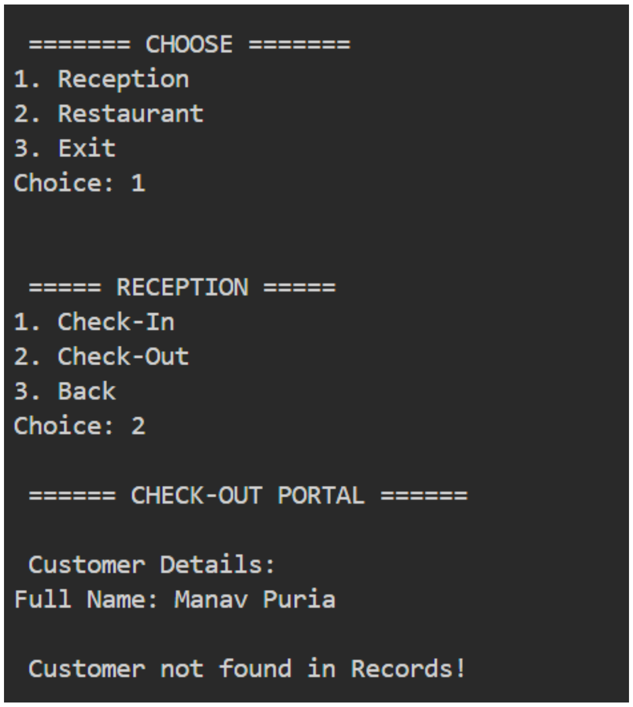

```
3.      Paneer Pakora    - 80
4.      Onion Pakora     - 60
5.      Aloo Tikki (2 pcs)       - 70
6.      Dahi Vada (2 pcs)        - 65
7.      Samosa Chaat     - 90
8.      Papri Chaat      - 85
9.      Dahi Bhalla Chaat        - 110
10.     Aloo Tikki Chaat         - 100
11.     Tandoori Chicken         - 180
12.     Chicken Tikka    - 190
13.     Fish Tikka       - 220
14.     Seekh Kebab      - 200
15.     Paneer Tikka     - 170
16.     Tomato Shorba    - 70
17.     Cream of Tomato Soup     - 85
18.     Vegetable Soup   - 80
19.     Chicken Noodle Soup      - 95
20.     Mushroom Soup    - 90
21.     Indian Salad (Cucumber   - 0
22.     Raita    - 50
23.     Butter Chicken   - 320
24.     Chicken Tikka Masala     - 340
25.     Saagwala Chicken         - 350
26.     Kadai Chicken    - 330
27.     Malai Kofta      - 300
28.     Paneer Butter Masala     - 310
29.     Palak Paneer     - 300
30.     Navratan Korma   - 320
31.     Dal Makhani      - 280
32.     Malai Kofta      - 300
33.     Chana Masala     - 270
34.     Aloo Gobi        - 250
35.     Vegetable Biryani        - 280
36.     Chicken Biryani          - 320
37.     Mutton Biryani   - 380
38.     Plain Naan       - 40
39.     Butter Naan      - 50
40.     Garlic Naan      - 50
41.     Cheese Naan      - 60
42.     Tandoori Roti    - 45
43.     Jeera Rice       - 120
44.     Steamed Rice     - 100
45.     Gulab Jamun (2 pcs)      - 120
46.     Gajar ka Halwa   - 150
47.     Jalebi (4 pcs)   - 100
48.     Falooda          - 130
49.     Kheer    - 110
```

*Figure 13Restraunt module*



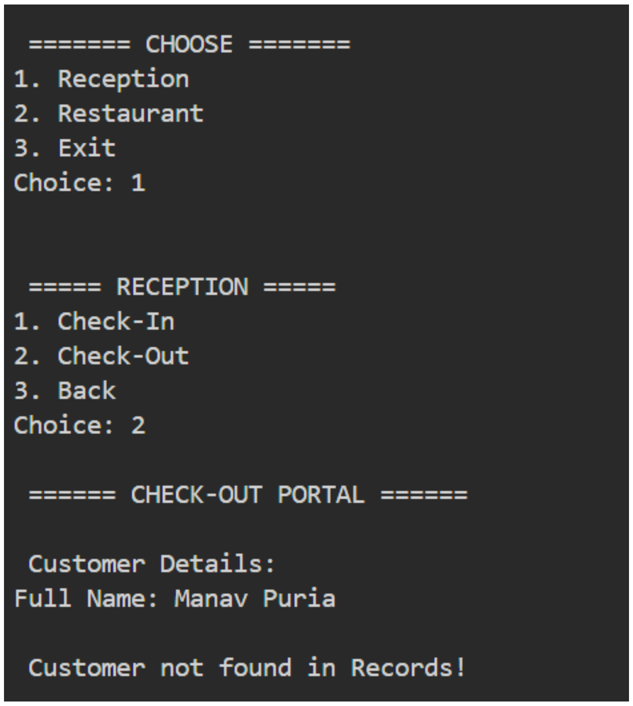
```
44.     Steamed Rice     - 100
45.     Gulab Jamun (2 pcs)      - 120
46.     Gajar ka Halwa   - 150
47.     Jalebi (4 pcs)   - 100
48.     Falooda          - 130
49.     Kheer    - 110
50.     Mango Lassi      - 80
51.     Butter Milk      - 60
52.     Chaas    - 70
53.     Coke (500 ml)    - 40
54.     Mineral Water    - 30

 Order (item no.): 5

 Press:
Any number: Another Order | 0: Proceed to Billing 3

 Order (item no.): 5

 Press:
Any number: Another Order | 0: Proceed to Billing 2

 Order (item no.): 5

 Press:
Any number: Another Order | 0: Proceed to Billing 3

 Order (item no.): 5

 Press:
Any number: Another Order | 0: Proceed to Billing 0

 Restaurant Bill: Rs.260
 Will be added to the Records and final payment will be recieved at Check-Out.
```

*Figure 14Restraunt taking ordder*



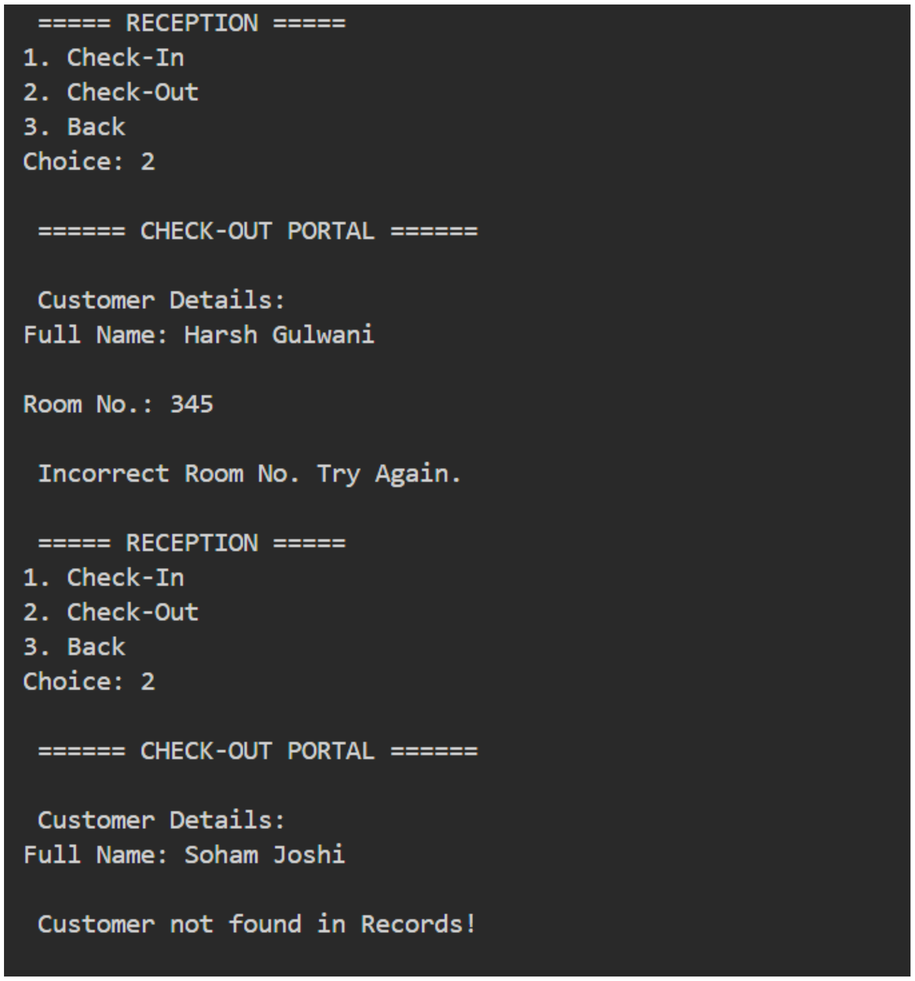
```
 ======= CHOOSE =======
1. Reception
2. Restaurant
3. Exit
Choice: 1


 ===== RECEPTION =====
1. Check-In
2. Check-Out
3. Back
Choice: 2

 ====== CHECK-OUT PORTAL ======

 Customer Details:
Full Name: Manav Puria

 Customer not found in Records!
```

*Figure 15exit*



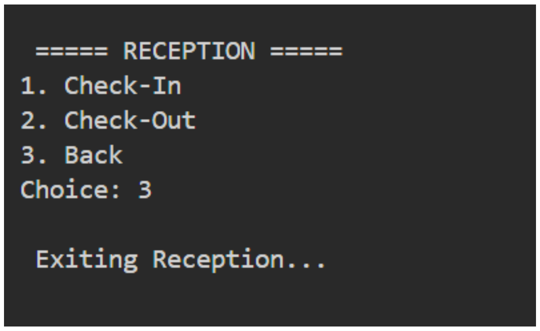
```
 ======= CHOOSE =======
1. Reception
2. Restaurant
3. Exit
Choice: 1


 ===== RECEPTION =====
1. Check-In
2. Check-Out
3. Back
Choice: 2

 ====== CHECK-OUT PORTAL ======

 Customer Details:
Full Name: Manav Puria

 Customer not found in Records!
```

*Figure 16Check-Out after Restaurant (Clears Customer Records)*



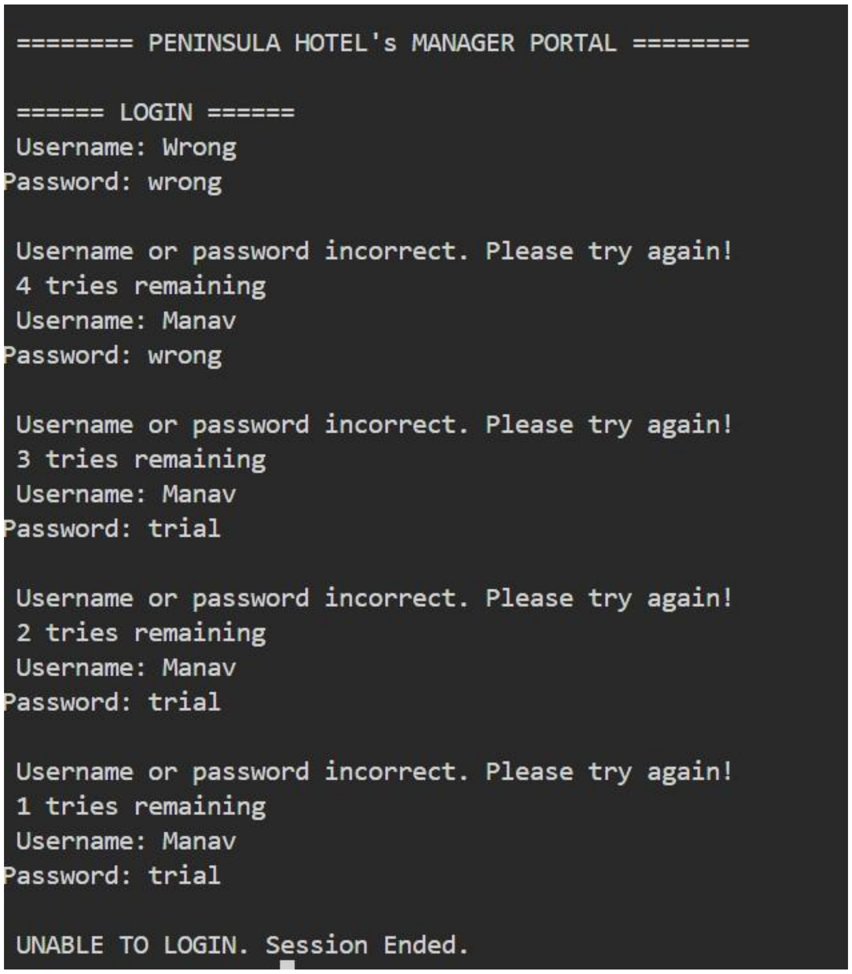
```
 ===== RECEPTION =====
1. Check-In
2. Check-Out
3. Back
Choice: 2

 ====== CHECK-OUT PORTAL ======

 Customer Details:
Full Name: Harsh Gulwani

Room No.: 345

 Incorrect Room No. Try Again.

 ===== RECEPTION =====
1. Check-In
2. Check-Out
3. Back
Choice: 2

 ====== CHECK-OUT PORTAL ======

 Customer Details:
Full Name: Soham Joshi

 Customer not found in Records!
```

*Figure 17checkout failed*



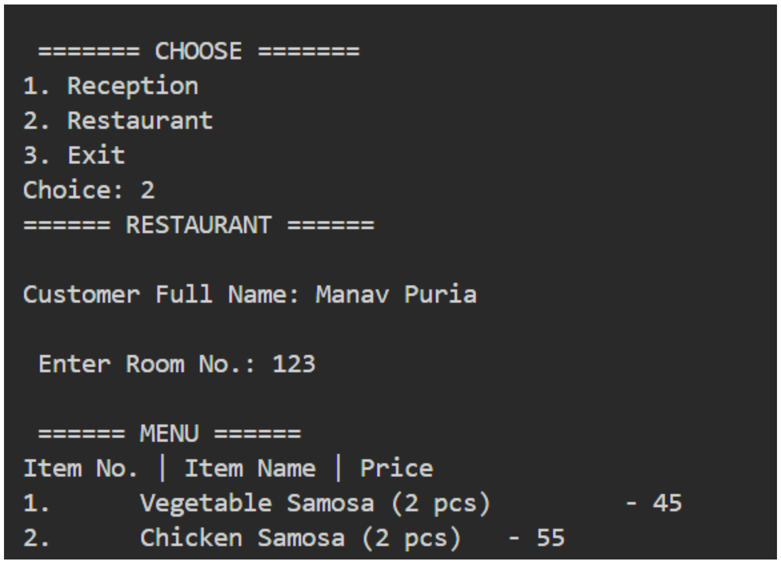
```
 ===== RECEPTION =====
1. Check-In
2. Check-Out
3. Back
Choice: 3

 Exiting Reception...
```

*Figure 18exit reception*



```
======== PENINSULA HOTEL's MANAGER PORTAL ========

 ====== LOGIN ======
 Username: Wrong
Password: wrong

 Username or password incorrect. Please try again!
 4 tries remaining
 Username: Manav
Password: wrong

 Username or password incorrect. Please try again!
 3 tries remaining
 Username: Manav
Password: trial

 Username or password incorrect. Please try again!
 2 tries remaining
 Username: Manav
Password: trial

 Username or password incorrect. Please try again!
 1 tries remaining
 Username: Manav
Password: trial

 UNABLE TO LOGIN. Session Ended.
```

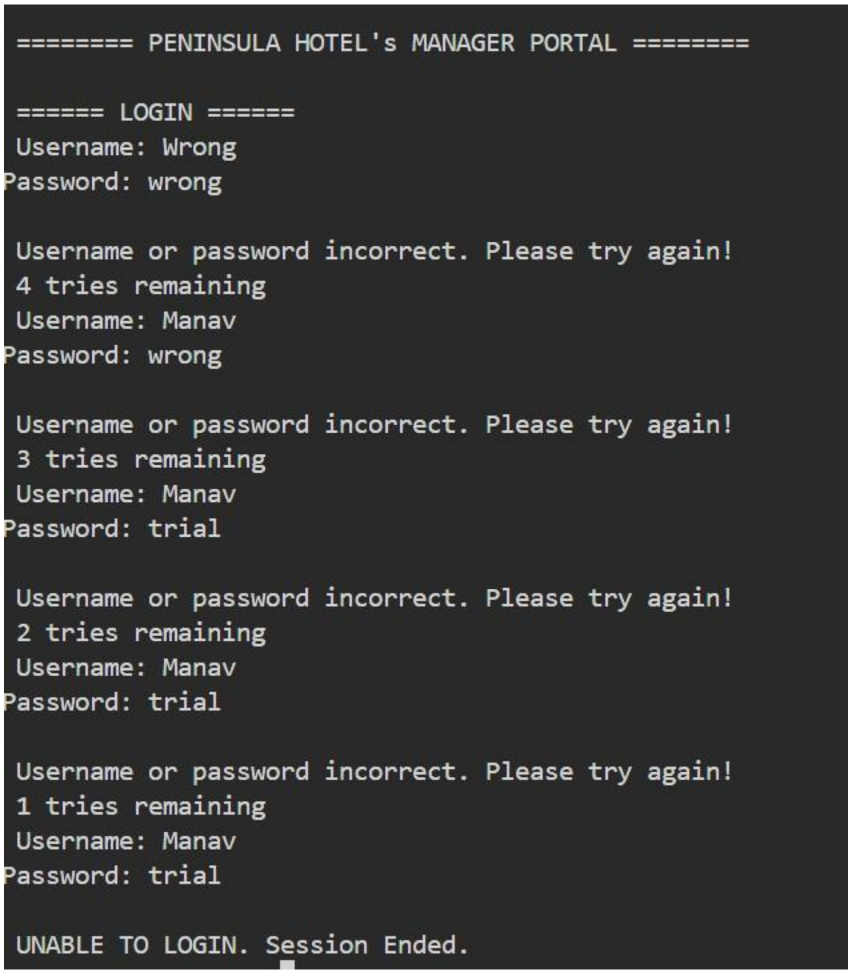
*Figure 19login failed*



```
 ======= CHOOSE =======
1. Reception
2. Restaurant
3. Exit
Choice: 2
====== RESTAURANT ======

Customer Full Name: Manav Puria

 Enter Room No.: 123

 ====== MENU ======
Item No. | Item Name | Price
1.      Vegetable Samosa (2 pcs)         - 45
2.      Chicken Samosa (2 pcs)   - 55
```

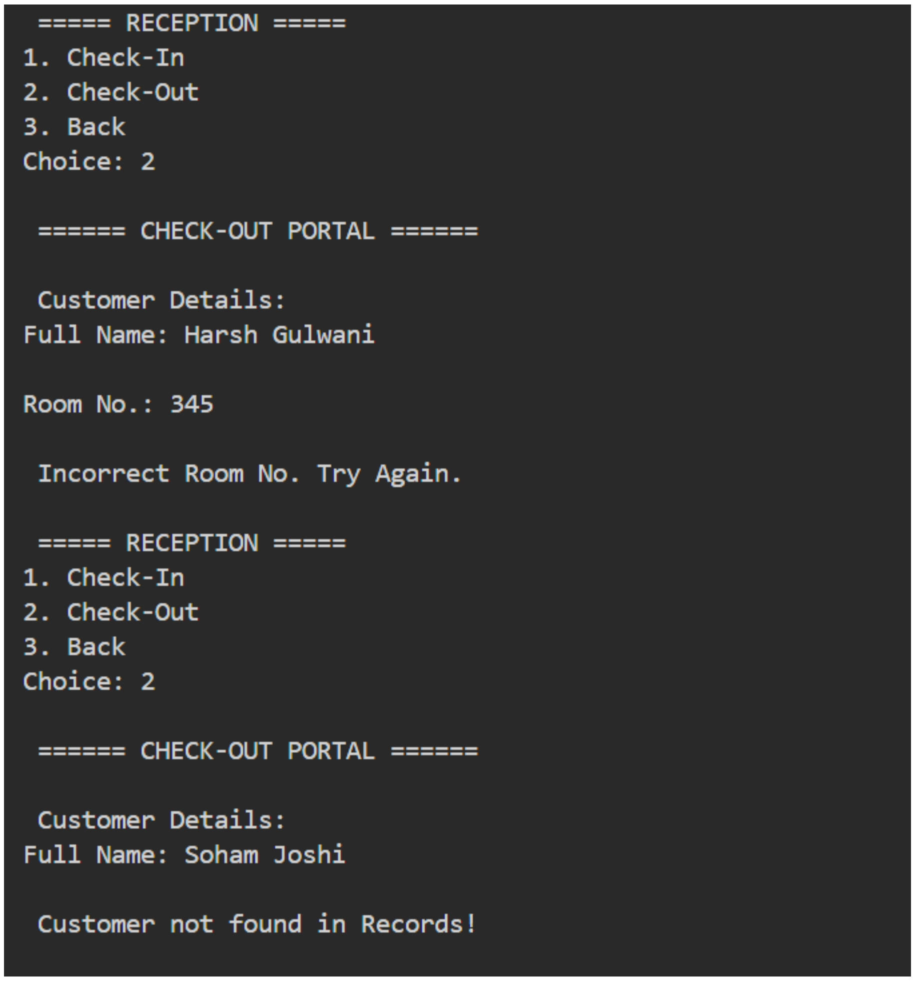
*Figure 20Restraunt error*

## LIST OF ERRORS AND ITS RESOLUTIONS



```
======== PENINSULA HOTEL's MANAGER PORTAL ========

====== LOGIN ======
Username: Wrong
Password: wrong

Username or password incorrect. Please try again!
4 tries remaining
Username: Manav
Password: wrong

Username or password incorrect. Please try again!
3 tries remaining
Username: Manav
Password: trial

Username or password incorrect. Please try again!
2 tries remaining
Username: Manav
Password: trial

Username or password incorrect. Please try again!
1 tries remaining
Username: Manav
Password: trial

UNABLE TO LOGIN. Session Ended.
```

*Figure 21login failed*



```
 ===== RECEPTION =====
1. Check-In
2. Check-Out
3. Back
Choice: 2

 ====== CHECK-OUT PORTAL ======

 Customer Details:
Full Name: Harsh Gulwani

Room No.: 345

 Incorrect Room No. Try Again.

 ===== RECEPTION =====
1. Check-In
2. Check-Out
3. Back
Choice: 2

 ====== CHECK-OUT PORTAL ======

 Customer Details:
Full Name: Soham Joshi

 Customer not found in Records!
```

*Figure 22checkout failed*

## SCOPE OF IMPROVEMENTS

Enhanced Automation: While the project already incorporates automation for key managerial operations, further refinement and expansion of automated processes can lead to greater labor cost reductions and operational efficiency. This could involve integrating advanced artificial intelligence (AI) and machine learning algorithms to automate additional tasks such as inventory management, scheduling, and personalized guest services.

Optimized User Experience: Improving the user interface and experience of the management platform can significantly enhance its usability and adoption rates among hotel staff. Conducting user research and incorporating feedback from stakeholders can help identify pain points and areas for improvement, leading to a more intuitive and user-friendly interface.

Advanced Analytics: Leveraging data analytics capabilities can provide valuable insights into hotel operations and guest preferences, enabling managers to make data-driven decisions that optimize resource allocation, pricing strategies, and marketing efforts. Implementing advanced analytics tools and predictive modeling techniques can unlock new opportunities for revenue generation and customer satisfaction.

## LIST OF REFERENCES/LIST OF RESEARCH PAPERS, BOOKS, WHITE PAPERS REFERRED

### Bibliography

[1] TIMESOFINDIA, "THE TIMES OF INIDA," 24 NOVEMBER 2023. [Online]. Available: https://timesofindia.indiatimes.com/gadgets-news/taj-hotel-data-breach-what-the-company-has-to-say-ransom-demanded-conditions-set-by-hackers/articleshow/105461155.cms. [Accessed 4TH APRIL 2024].

[2] T. C. L. a. R. Laganiere, "class and object," in *Object-Oriented Software Engineering*, New Delhi, Tata Mc Graw-Hill, 2004, pp. 27-29.

[3] T. C. L. a. R. Laganiere, "Abstraction," in *Object-Oriented Software Engineering*, New Delhi, Tata Mc Graw-Hill, 2004, pp. 27-29.

[4] E. Balagurusamy, "file," in *Object Oriented Programming With C++*, noida, Tata Mc Graw-Hill, 2008, p. 334.

[5] E. Balagurusamy, "Encapsulation," in *Object Oriented Programming With C++*, Noida, Tata Mcgraw, 2008, pp. 8-9.

[6] E. Balagurusamy, "destructor," in *Object Oriented Programming With C++*, noida, Tata Mc Graw-Hill, 2008, p. 144.