***Khulna University of Engineering & Technology***



## 

## *Department of Computer Science & Engineering*

## *Object Oriented Programming Laboratory Project*

***Project Title :*** Hotel Website with Three Interfaces - Customer,

Receptionist and Hotel Authority

***Course No :*** CSE 1206

***Course Title :*** Object Oriented Programming Laboratory

|  |  |
| --- | --- |
| Submitted To,  Mr. Repon Islam  Department of CSE  KUET  Mr. Safin Ahmmed  Department of CSE  KUET | Submitted by,  ***Name :*** Dipra Datta  ***Roll :*** 2107070  ***Year :*** 1st  ***Term :*** 2nd |

Project Title:

**Hotel Website with Three Interfaces - Customer Receptionist, and Hotel Authority**

1. Project Objectives:

* Customer Interface:
* User Registration and Login System
* Room Information
* Room Booking
* Check-Out Feature
* Navigation and Exit
* Receptionist Interface:
* Receptionist Registration and Login System
* Room Management
* Room Booking and Check-Out
* Navigation and Exit
* Hotel Authority Interface:
* Authority Registration System
* Room Management
* Room Administration
* Customer Information
* Navigation and Exit

2. Project Background:

This project was chosen as part of the educational experience to demonstrate object-oriented programming (OOP) concepts in C++. The choice was made freely, allowing for exploration of personal interests.

3. Project Scope:

The project's scope included the development of a hotel website with three interfaces, each catering to specific user roles (Customer, Receptionist, and Hotel Authority). It encompassed features related to user registration, room management, booking, and basic navigation.

4. Project Technologies:

The project was successfully implemented using basic C++ with a strong focus on applying object-oriented programming (OOP) principles. No external libraries or frameworks were utilized, emphasizing core programming skills.

5. Project Team:

The project was conceived, designed, and developed independently.

6. Implementation:

* + Customer Interface:
* In the securityInterphase class, a robust registration system was implemented to collect user details securely. User data, including username, password, and security information, was stored in a file for authentication and future access. The login system ensured secure user access to the system, requiring valid credentials.
* Password change functionality was implemented, allowing users to update their passwords securely. Additionally, a forgot password (reset password) system was in place, enhancing user convenience.
* The customerInfo class featured a read\_customer function, which captured customer details during check-in. This information was stored in a file for reference. The ShowAllCustomer function allowed for the retrieval and display of customer details, making it an essential component of the system.
* Input/output operators were overloaded to streamline the process of displaying customer information in the console, reading customer data from files, and storing customer data in files.
* The roomInfo class also featured input/output operator overloading to facilitate similar functionalities for room data.
  + Receptionist Interface:
* The roomMng class was designed to provide receptionists with essential functionalities for managing hotel rooms and customer interactions. It inherited from securityInterphase, ensuring a secure access system.
* Functions such as Allroom, Myrooms, AvailableRoom, findRoom, check\_in, and check\_out were implemented to empower receptionists to efficiently manage room bookings and customer check-ins/check-outs.
* Receptionists could access real-time information about room availability, enabling them to assist customers effectively. The system allowed for the assignment of rooms to customers during check-ins and facilitated the check-out process when guests departed.
  + Hotel Authority Interface:
* The roomOperation class extended the room management capabilities by incorporating functions like RemoveRoom, Addroom, and UpdateRoom. These functions allowed hotel authorities to modify room details, such as room numbers and prices, ensuring accurate and up-to-date room information.
* The HotelAuthority class served as the core interface for hotel authorities, bringing together functionalities from various classes. It inherited from securityInterphase, customerInfo, roomOperation, and Interphase, granting hotel authorities comprehensive control over the system.
* The firstAuthorityAccess function provided initial access to hotel authorities, ensuring a smooth onboarding process. The choices and operations functions offered a user-friendly interface for managing rooms and customer data.

7. Results:

* Customer Interface:
* The Customer Interface successfully implemented user registration, login, and room booking functionalities. Users could securely change and reset their passwords, enhancing the overall user experience.
* The customerInfo class effectively captured and displayed customer details, streamlining the check-in process and providing a means to review customer data.
* The code ensured data security and user-friendly input, resulting in a robust Customer Interface.
* Receptionist Interface:
* The Receptionist Interface achieved its objectives by enabling receptionists to manage room bookings, reservations, and interactions effectively.
* Room availability information was readily accessible, allowing receptionists to assist customers promptly.
* User-friendly input and secure data handling contributed to the success of this interface.
* Hotel Authority Interface:
* The Hotel Authority Interface empowered hotel authorities to manage room and customer data with ease. Functions such as room addition, removal, and updates were executed smoothly.
* The firstAuthorityAccess function provided a secure onboarding process for hotel authorities.
* Data management and operations were efficient and user-friendly, enhancing overall system control.

8. Discussion:

* The results reflect the successful implementation of a hotel management system with distinct user interfaces.
* Object-oriented programming (OOP) principles, including inheritance, encapsulation, and polymorphism, were effectively applied throughout the project, resulting in a modular and maintainable codebase.
* Challenges encountered during the project were addressed, with a focus on user input validation and security.
* Future development possibilities include enhancing the check-out functionality and improving user interfaces for better user experience.

9. Conclusion:

* In conclusion, the project has demonstrated the successful application of OOP concepts in C++ to develop a hotel management system with three distinct user interfaces.
* The project achieved its primary objectives and contributed to a deeper understanding of OOP principles in practical implementation.
* Overall, the project was a valuable educational experience, showcasing the potential of object-oriented programming in real-world applications.

10. Future Work:

* Future work may include enhancing the check-out functionality to provide a complete hotel management solution.
* User interface improvements and usability enhancements can be considered to further enhance the user experience.
* Exploration of additional security measures and data validation techniques could be pursued to strengthen the system.

11. References:

No external references were consulted for this project, as it primarily focused on implementing OOP principles in C++.

12. Appendices:

Here is the Code snippet-

#include <bits/stdc++.h>

#include<conio.h>

#include <iostream>

#include <thread>

#include <chrono>

using namespace std;

typedef long long ll;

#define cls system("cls");

#define pause system("pause");

bool isAdult=1;

void take(int &userInput)

{

bool isValidInput = false;

while (!isValidInput) {

if (cin >> userInput)

isValidInput = true;

else {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "\tInvalid input. Please enter a valid choice: ";

}

}

}

//Checking if the year is leap year or not

bool isLeapYear(int year) {

return ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0));

}

//Allright Checking if the provided date is okay or not

bool check\_date(string s, bool x)

{

if(s.size()!=10 || s[2]!='/' || s[5]!='/') return false;

string ss[3];

int ct=0;

for(int i=0;i<10;i++)

{

if(i==2 || i==5) ct++;

else if(s[i]<'0' || s[i]>'9') return false;

else ss[ct].push\_back(s[i]);

}

int mn[13];

for(int i=1,cc=1;i<=12;i++,cc++) mn[i]=(cc-i/8)%2?31:30;

int a=stoi(ss[0]), b=stoi(ss[1]), c=stoi(ss[2]);

mn[2]=28+isLeapYear(c);

if(x && (c<2023 || (c==2023 && b<9))) return false;

else if(b>12 || b<1 || a<1 || a>mn[b]) return false;

else if(!x && (c>2005 || (c==2005 && b<10)))

{

cout << "\tYou are under 18. Make sure that you are 18 and then register.\n";

isAdult=0;

return false;

}

return true;

}

//Allright Class to manage security Interface for username and password setting and sign-in

class securityInterface

{

protected:

string name, username, password, DOB, old\_pw, new\_pw,mob, un, pw, dob, nm, phn, stored\_username, address,ad,Interface,n;

int cnt, choice,s;

string file;

vector<string> userInfo;

securityInterface()

{

username = password = file = "";

cnt=0;

}

bool check(ifstream &security)

{

s=0;

security >> s >> un >> pw >> dob;

security.ignore();

getline(security,nm);

security >> phn;

if(file=="UsersSecurityFile.txt")

{

security.ignore();

getline(security,ad);

}

return s==1;

}

void display\_option()

{

cout <<"\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\t| " << Interface <<" Security page\t|\n\t|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n\t|\t(1) LOGIN\t\t|\n";

cout << "\t|\t(2) Change PASSWORD\t|" << "\n";

cout << "\t|\t(3) Forgot Your PASSWORD|" << "\n";

if(file=="UsersSecurityFile.txt") cout << "\t|\t(4) Register\t\t|\n";

cout << "\t|\t(0) EXIT.\t\t|\n\t|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

cout << "\tEnter your Choice: ";

}

bool accessed()

{

display\_option();

take(choice);

cls

switch (choice)

{

case 1:

return sign\_in();

break;

case 2:

change\_pw();

return accessed();

case 3:

forgot\_pw();

return accessed();

}

if(file=="UsersSecurityFile.txt" && choice==4)

{

setSecurity();

return accessed();

}

else return 0;

}

void wrong\_message()

{

cout << "\tPress 1 to try again.\n";

cout << "\tPress 2 to Back\n";

cout << "\tEnter your choice: ";

}

bool setSecurity() // to register the author first time

{

cout << "\t===============================================\n";

cout << "\t--------------------Sign Up--------------------\n";

cout << "\tEnter Your Username: ";

cin >> username;

ifstream sc(file);

while(check(sc)){

if (username == un)

{

sc.close();

cout << "\tThe username is already taken. Try again.....\n";

return setSecurity();

}

}

sc.close();

cin.ignore();

cout << "\tEnter Your Full Name: ";

getline(cin,name);

take\_dob:

cout << "\tEnter Your Date of Birth(dd/mm/yyyy): ";

cin >> DOB;

if(!check\_date(DOB,0))

{

if(!isAdult){

isAdult=1;

return 0;

}

cout << "\tInvalid Date\n\tPlease check the date or date format.\n";

goto take\_dob;

}

if(file=="UsersSecurityFile.txt")

{

cin.ignore();

cout << "\tEnter you full address: ";

getline(cin,address);

}

cout << "\tEnter Your Mobile NO: ";

cin >> mob;

take\_pw:

cout << "\tEnter Your Password: ";

cin >> password;

cls;

if (password.size() < 8)

{

cout << "\tYour password must be at least 8 characters long. Please try another.\n";

cout << "\t(1) Try again\n\t(2) Exit Program\n\tChoice: ";

cin >> n;

cls;

if (n == "1") goto take\_pw;

else return 0;

}

fstream security(file, ios::app);

security << 1 << endl

<< username << endl

<< password << endl << DOB << endl << name << endl << mob << endl;

if(file=="UsersSecurityFile.txt")

security << address << endl;

security.close();

cout << "\tRegistration completed.\n";

return 1;

}

// checking the entered password is correct or wrong

bool sign\_in()

{

ifstream sc(file);

if (!check(sc) && file!="UsersSecurityFile.txt")

{sc.close();cout << "\tHave not been registered yet.\n";return setSecurity();}

sc.close();

sign\_in:

password.clear();

cout << "\t===============================================\n";

cout << "\t--------------------SIGN IN--------------------\n";

cnt++;

cout << "\tEnter Your Username: ";

cin >> username;

cout << "\tEnter Your Password: ";

char ch;

do {

ch = \_getch();

if (ch != '\r') {

if (ch == '\b' && !password.empty()) { // Handle backspace

cout << "\b \b";

password.pop\_back();

}

else if (ch != '\b') { // Ignore backspace when password is empty

password.push\_back(ch);

cout << '\*';

}

}

} while (ch != '\r');

cls;

ifstream security(file);

while(check(security)){

if (username == un && password == pw)

{

security.close();

cout << "\tYou are successfully signed in.\n";

stored\_username=username;

cnt=0;

return 1;

}

}

security.close();

cout << "\tPlease check your username and password.\n";

if (cnt % 5 == 0)

{

cout << "\tYou have tried " << cnt << " times wrong attempt.\n";

pause;

cls;

for (int i = cnt \* 6 - 1; i >= 0; i--)

{

cout << "\tTry again in " << i << " seconds\n";

this\_thread::sleep\_for(chrono::seconds(1));

cls;

}

}

wrong\_message();

cin >> n;

cls;

if (n == "1") goto sign\_in;

else return 0;

}

bool change\_pw() // to change the password if everything is ok

{

change\_pw:

cout << "\tEnter your Username: ";

cin >> username;

cout << "\n\tEnter your old Password: ";

cin >> old\_pw;

cout << "\n\tEnter your new Password: ";

cin >> new\_pw;

system("cls");

bool flag=false;

ifstream sc(file);

while(check(sc)){

if (!flag && un == username && pw == old\_pw)

flag=true, pw=new\_pw;

userInfo.insert(userInfo.end(),{to\_string(s),un,pw,dob,nm,phn});

}

sc.close();

if(flag)

{

ofstream sc(file);

for(int i=0;i<userInfo.size();i++) sc << userInfo[i] << endl;

cout << "\n\tYour PASSWORD is successfully updated.\n\n";

pause;

}

else

{

cout << "\tSorry!! Please check your username and password\n";

wrong\_message();

cin >> n;

system("cls");

if (n == "1")

goto change\_pw;

}

cls;

userInfo.clear();

return flag;

}

bool forgot\_pw()

{

cout << "\tEnter your Username: ";

cin >> username; // taking username from user

take\_dob:

cout << "\tEnter Your Date of Birth(dd/mm/yyyy): ";

cin >> DOB;

if(!check\_date(DOB,0))

{

cout << "\tInvalid Date\n\tPlease check the date or date format.\n";

goto take\_dob;

}

system("cls");

bool flag=false;

ifstream security(file);

while(check(security)){

if (!flag && un == username && dob==DOB) // checking if the username is

{

cout << "\tType a new PASSWORD: ";

cin >> new\_pw;

pw=new\_pw;

flag= true;

}

userInfo.insert(userInfo.end(),{to\_string(s),un,pw,dob,nm,phn});

}

security.close();

if(flag)

{

ofstream sc(file);

for(int i=0;i<userInfo.size();i++) sc << userInfo[i] << endl;

sc.close();

cls;

cout << "\tYour password has been successfully reset.\n";

pause;

cls;

}

else

{

cout << "\tYou have entered Wrong Username of Wrong Date of Birth.\n";

wrong\_message();

cin >> n;

system("cls");

if (n == "1")

return forgot\_pw();

}

userInfo.clear();

return flag;

}

};

// Class to manage customer details

class customerInfo

{

protected:

string name, address, mob, idate, odate,id;

int advance, room\_no;

friend class roomMng;

friend class roomOperation;

customerInfo() {}

customerInfo(string name, string address, string mob, int room\_no, string idate, string odate, int advance, string id):

name(name), address(address), mob(mob), room\_no(room\_no), idate(idate), odate(odate), advance(advance), id(id){}

//Allright Function to read customer details from user input and store them in a file

void read\_customer()

{

cin.ignore();

cout << "\tEnter Customer Details: \n\n";

cout << "\tName: ";

getline(cin,name);

cout << "\tAddress: ";

getline(cin,address);

cout << "\tPhone: ";

cin >> mob;

in:

cout << "\n\tEnter Check-In Date(dd/mm/yyyy): ";

cin >> idate;

if(!check\_date(idate,1))

{

cout << "\tInvalid Date\n\tPlease check the date or date format.\n";

goto in;

}

out:

cout << "\n\tEnter Check-Out Date(dd/mm/yyyy): ";

cin >> odate;

if(!check\_date(odate,1))

{

cout << "\tInvalid Date\n\tPlease check the date or date format.\n";

goto out;

}

cout << "\n\tAdvance Payment: ";

cin >> advance;

cout << "\tBooking ID: ";

cin >> id;

ofstream cm("customers.txt", ios::app);

cm << name << "\n"

<< address << "\n"

<< mob << "\n" << idate << "\n"

<< odate << "\n" << advance << "\n"

<< id << "\n";

cm.close();

}

//Function to show all customers

void showAllCustomers()

{

cout << "\tAll Customers Info: \n\n";

ifstream cm("customers.txt");

customerInfo cc;

map<string,bool> customer;

while (cm >> cc)

{

string a=cc.id;

for(int i=0;i<8 && cc.id.size()>8;i++)

a.pop\_back();

if(customer[a]) continue;

cout << cc << endl;

customer[a]=1;

}

cm.close();

cm.close();

}

//Allright operator overloading to display customer details in console

friend ostream& operator <<(ostream &out, customerInfo &c)

{

out << "\tName: " << c.name << "\n\tAddress: " << c.address << "\n\tMobile No: " << c.mob << "\n\tRoom No: "

<< c.room\_no << "\n\tCheck-In Date(dd/mm/yyyy): " << c.idate << "\n\tCheck-Out Date(dd/mm/yyyy): "

<< c.odate << "\n\tAdvance Payement: " << c.advance << "\n\tBooking ID: " << c.id << "\n";

return out;

}

//operator overloading to take input from file

friend ifstream& operator >>(ifstream& in, customerInfo &c)

{

in >> c.room\_no;

in.ignore();

getline(in,c.name);

getline(in,c.address);

in >> c.mob >> c.idate >> c.odate >> c.advance >> c.id;

return in;

}

//Operator overloading to write in file

friend ofstream& operator <<(ofstream& out, customerInfo &c)

{

out << c.room\_no << endl << c.name << endl << c.address << endl << c.mob << endl << c.idate << endl << c.odate << endl << c.advance << endl << c.id << endl;

return out;

}

friend fstream& operator <<(fstream& out, customerInfo &c)

{

out << c.room\_no << endl << c.name << endl << c.address << endl << c.mob << endl << c.idate << endl << c.odate << endl << c.advance << endl << c.id << endl;

return out;

}

};

//Allright Class to manage room details

class roomInfo

{

protected:

int rent, room\_no, type, status;

// Friend classes and functions to access private members

friend class roomMng;

friend class roomOperation;

roomInfo():status(0) {}

// Friend Function to read room details from user input and store them in a file

friend void read\_room();

// operator overloading to display room details in console

friend ostream& operator <<(ostream &out, roomInfo &r)

{

out << "\n\tRoom No: " << r.room\_no << "\n";

out << "\tType: " << (r.type == 1 ? "Single" : (r.type == 2 ? "Double" : "Family")) << " room\n";

out << "\tStatus: " << (r.status == 0 ? "Available" : "Occupied") << "\n";

out << "\tRent: " << r.rent << "\n";

return out;

}

//Operator overloading to take input from file

friend ifstream& operator >>(ifstream& in, roomInfo &r)

{

in >> r.room\_no >> r.type >> r.status >> r.rent;

return in;

}

//Operator overloading to write in file

friend ofstream& operator <<(ofstream &out, roomInfo &r)

{

out << r.room\_no << endl << r.type << endl << r.status << endl << r.rent << endl;

return out;

}

};

// Function to read room details from user input and store them in a file

void read\_room()

{

ifstream room;

roomInfo rr;

again:

room.open("rooms.txt");

cout << "\tEnter Room No: ";

cin >> rr.room\_no;

roomInfo r;

while (room >> r)

{

if (r.room\_no == rr.room\_no)

{

cout << "\tThe room already exists.\n\tPlease try again........\n";

room.close();

goto again;

}

}

cout << "\tRoom Type (Single Room = 1, Double Room = 2, Family Room = 3): ";

cin >> rr.type;

cout << "\tRoom Rent: ";

cin >> rr.rent;

ofstream rooms;

rooms.open("rooms.txt", ios::app);

rooms << rr.room\_no << "\n"

<< rr.type << "\n"

<< rr.status << "\n"

<< rr.rent << "\n";

rooms.close();

}

// Class to manage room operations

class roomMng:virtual public securityInterface

{

protected:

string name, address, mob,cs,idate, odate;

int cnt, rm\_no, c, x,adv;

bool flag, isAuthor;

roomMng()

{

cnt = 0;

isAuthor=0;

}

//Allright Function to display details of all rooms

void AllRoom()

{

cout << "\t===============================================\n";

cout << "\t---------------LIST OF ALL ROOMS---------------\n";

ifstream rooms;

rooms.open("rooms.txt");

roomInfo r;

while (rooms >> r) cout << r;

rooms.close();

cout << "\n\t----------------END OF THE LIST----------------\n";

cout << "\t===============================================\n";

}

void MyRooms()

{

ifstream cm("customers.txt");

customerInfo cc;

map<int,bool> room;

while (cm >> cc)

{

if(cc.id.size()<=8) continue;

for(int i=0;i<8;i++)

cc.id.pop\_back();

if(cc.id==stored\_username)

room[cc.room\_no]=1;

}

cm.close();

if(room.empty()) cout << "You haven't booked any room yet.\n";

else

{

cout << "\tYou have booked the following rooms:\n\n";

roomInfo r;

ifstream rr("rooms.txt");

while(rr >> r)

if(room[r.room\_no]) cout << r << endl;

rr.close();

}

}

//Allright Function to display details of available rooms

void AvailableRoom()

{

cout << "\t===============================================\n";

cout << "\t----------LIST OF THE AVAILABLE ROOMS----------\n";

ifstream rooms;

rooms.open("rooms.txt");

roomInfo r;

bool c=0;

while (rooms >> r) if(!r.status)c=1, cout << r;

rooms.close();

if(!c) cout << "\tNo Room Available at the Moment.\n";

cout << "\n\t----------------END OF THE LIST----------------\n";

cout << "\t===============================================\n";

}

//Allright Function to find a specific room and perform check-in or check-out

void findRoom()

{

cout << "\t===============================================\n";

cout << "\t------------------FIND A ROOM------------------\n";

cout << "\tEnter Room No: ";

cin >> rm\_no;

ifstream rms("rooms.txt");

roomInfo r;

while (rms >> r)

{

if (r.room\_no == rm\_no)

{

cout << "\tThe room has been found.\n";

cout << "\tHere the details of the room:" << endl;

cout << r;

rms.close();

if (isAuthor && r.status)

{

cout << "\tCustomer Info: \n";

ifstream cm("customers.txt");

customerInfo cc;

while (cm >> cc)

{

if (cc.room\_no == r.room\_no)

{

cout << cc;

break;

}

}

cm.close();

cout << "\n\tDo you want to " << (r.status ? "check it out" : "occupy it") << "? (y/n): ";

cin >> cs;

if (cs != "y")

cout << "\tOkay!!Let's go to previous page.\n";

else

r.status ? check\_out() : check\_in();

return;

}

cout << "\t===============================================\n";

}

}

cout << "\t----------THE ROOM HAS NOT BEEN FOUND----------\n";

cout << "\t===============================================\n";

}

// Function to perform check-in for a room

void check\_in()

{

cout << "\t===============================================\n";

cout << "\t\t\tALL AVAILABLE ROOMS\n";

int cnt=0;

ifstream rooms;

rooms.open("rooms.txt");

roomInfo r;

while (rooms >> r)

if (!r.status)

cout << "\t\tRoom No: " << r.room\_no << "\n", cnt++;

rooms.close();

if (cnt == 0)

cout << "\tSorry!! There is no available room.\n";

cout << "\t===============================================\n";

if (cnt == 0)

return;

vector<roomInfo> info;

again:

cout << "\t(1) Show Details\n\t(2) Back\n\t(3) Continue....\n\tChoice: ";

cin >> n;

if(n=="2") return;

if (n == "1")

{

AvailableRoom();

cout << "\t===============================================\n";

goto again;

}

cls

cout << "\t===============================================\n";

cout << "\t--------------------CHECK IN-------------------\n";

rooms.open("rooms.txt");

info.clear();

cout << "\tEnter the Room No: ";

cin >> rm\_no;

flag = true;

while (rooms >> r)

{

if (r.room\_no == rm\_no)

{

if (flag = r.status)

break;

cout << "\n\tRoom to be Checked In: ";

cout << r;

r.status = 1;

}

info.push\_back(r);

}

rooms.close();

if (flag)

{

cout << "\tThe room does not found or the room is already occupied.\n";

rooms.close();

cout << "\t(1) Back\n\t(2) Continue\n\tChoice: ";

cin >> n;

if (n == "1")

return;

cout << "\tPlease Try again....\n";

goto again;

}

ofstream rm;

rm.open("rooms.txt");

for (ll i = 0; i < info.size(); i++)

rm << info[i];

if(!isAuthor)

{

ifstream sc(file);

while(check(sc)){

if (stored\_username == un)

{

sc.close();

in:

cout << "\n\tEnter Check-In Date(dd/mm/yyyy): ";

cin >> idate;

if(!check\_date(idate,1))

{

cout << "\tInvalid Date\n\tPlease check the date or date format.\n";

goto in;

}

out:

cout << "\n\tEnter Check-Out Date(dd/mm/yyyy): ";

cin >> odate;

if(!check\_date(odate,1))

{

cout << "\tInvalid Date\n\tPlease check the date or date format.\n";

goto out;

}

cout << "\tHow much do you want to pay in advance?\n\t";

cin >> adv;

string y=un, yy=idate;

yy.erase(yy.begin()+2);

yy.erase(yy.begin()+4);

y=y+yy;

customerInfo cus( nm, ad, phn, rm\_no, idate, odate, adv, y);

fstream cm("customers.txt", ios::app);

cm << cus;

cm.close();

return ;

}

}

sc.close();

}

ofstream cm("customers.txt", ios::app);

cm << rm\_no << "\n";

cm.close();

customerInfo cmm;

cmm.read\_customer();

cout << "\t-----------------CHECK IN DONE-----------------\n";

cout << "\t===============================================\n";

}

// Function to perform check-out for a room

void check\_out()

{

vector<customerInfo> cminfo;

vector<roomInfo> rminfo;

ifstream rooms;

cout << "\t===============================================\n";

cout << "\t-------------------CHECK OUT-------------------\n";

chk\_out:

cminfo.clear();

rminfo.clear();

cout << "\tEnter Room No: ";

cin >> rm\_no;

rooms.open("rooms.txt");

flag = true;

roomInfo r;

while (rooms >> r)

{

if (rm\_no == r.room\_no)

{

if (flag = (r.status == 0))

break;

cout << "\n\tRoom to be Checked Out:";

cout << r;

cout << "\tCustomer Info: \n";

ifstream cm("customers.txt");

customerInfo cc;

while (cm >> cc)

{

if (cc.room\_no == r.room\_no) cout << cc;

else cminfo.push\_back(cc);

}

cm.close();

r.status = 0;

}

rminfo.push\_back(r);

}

rooms.close();

if (flag)

{

cout << "\tThe Room does not Exist or Unoccupied\n";

cout << "\t(1) Back\n\t(2) Continue\n\tChoice: ";

cin >> n;

if (n == "1")

return;

cout << "\tPlease Try again....\n";

goto chk\_out;

}

cout << "\tAre you confirm to check out (y/n): ";

cin >> cs;

if (cs != "y")

{

cout << "\tOkay\n\t(1) Back to previous page\n\t(2) Continue\n\tChoice: ";

cin >> n;

if (n == "1")

return;

else

goto chk\_out;

}

ofstream cmr("customers.txt");

for (ll i = 0; i < cminfo.size(); i++) cmr << cminfo[i];

cmr.close();

ofstream rms("rooms.txt");

for (ll i = 0; i < rminfo.size(); i++) rms << rminfo[i];

rms.close();

cout << "\tPlease take the bill from the guest before his/her leaving.\n\n";

cout << "\t-----------------CHECK OUT DONE----------------\n";

cout << "\t===============================================\n";

}

};

class roomOperation : public roomMng

{

protected:

roomOperation()

{

isAuthor=1;

}

// Function to remove a room and related customer details

void RemoveRoom()

{

cout << "\t===============================================\n";

cout << "\t-----------------REMOVE A ROOM-----------------\n";

vector<roomInfo> info;

ifstream rms;

update\_again:

cout << "\tEnter Room No: ";

cin >> rm\_no;

rms.open("rooms.txt");

flag = true;

info.clear();

roomInfo r;

while (rms >> r)

{

if (r.room\_no == rm\_no)

{

if (flag = r.status)

break;

else

continue;

}

info.push\_back(r);

}

rms.close();

if (flag)

{

cout << "\tThe Room Doesn't Exist Or Already Occupied.\n\t(1) Back\n\t(2) Continue\n\tChoice: ";

cin >> n;

if (n == "1")

return;

cout << "\tPlease Try again........\n";

goto update\_again;

}

cout << "\tAre you sure to remove the room " << r.room\_no << "? (y/n): ";

cin >> cs;

if (cs != "y")

{

cout << "\tOkay\n\t(1) Back to previous page\n\t(2) Continue\n\tChoice: ";

cin >> n;

cls

if (n == "1")

return;

else

goto update\_again;

}

ofstream rmo("rooms.txt");

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

rmo << info[i];

rmo.close();

cout << "\t----THE ROOM HAS BEEN SUCCESSFULLY REMOVED-----\n";

cout << "\t===============================================\n";

}

//Allright Function to add a new room and related customer details

void Addroom()

{

cout << "\t===============================================\n";

cout << "\t--------------------ADD ROOM-------------------\n";

roomInfo r;

read\_room();

cout << "\t---------THE ROOM IS ADDED SUCCESSFULLY--------\n";

cout << "\t===============================================\n";

}

//Allright Function to update room details

void UpdateRoom()

{

cout << "\t===============================================\n";

cout << "\t----------------Update Room Info---------------\n";

vector<roomInfo> info;

vector<customerInfo> cminfo;

ifstream rms;

update\_again:

cout << "\tEnter Room No: ";

cin >> rm\_no;

rms.open("rooms.txt");

flag = true;

bool mark = false;

info.clear();

cminfo.clear();

roomInfo r;

while (rms >> r)

{

if (r.room\_no == rm\_no)

{

flag = false;

cout << "\t(1) Room Number\n\t(2) Room Type\n\t(3) Room Rent\n\tChoice: ";

cin >> n;

cout << "\tEnter New Room " << (n == "1" ? "Number: " : n == "2" ? "Type (Single Room = 1, Double Room = 2, Family Room = 3): ": "Rent: ");

cin >> x;

if (n == "1")

{

ifstream cm("customers.txt");

customerInfo cc;

while (cm >> cc)

{

if (cc.room\_no == r.room\_no)

cc.room\_no = x, mark = true;

cminfo.push\_back(cc);

// cminfo.insert(cminfo.end(), {to\_string(cc.room\_no), cc.name, cc.address, cc.mob, cc.idate, cc.odate, to\_string(cc.advance), to\_string(cc.id)});

}

cm.close();

}

n == "1" ? r.room\_no = x : n == "2" ? r.type = x

: r.rent = x;

}

info.push\_back(r);

}

rms.close();

if (flag)

{

cout << "\tThe Room Doesn't Exist.\n\t(1) Back\n\t(2) Continue\n\tChoice: ";

cin >> n;

if (n == "1")

return;

cout << "\tPlease Try again........\n";

goto update\_again;

}

ofstream rmo("rooms.txt");

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

rmo << info[i];

rmo.close();

if (mark)

{

ofstream cmm("customers.txt");

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

cmm << cminfo[i];

cmm.close();

}

cout << "\t THE INFORMATION HAS BEEN SUCCESSFULLY UPDATED.\n";

cout << "\t===============================================\n";

}

};

class Interphase

{

public:

virtual int choices()=0;

virtual bool operations()=0;

};

class Receptionist: virtual public securityInterface, public roomMng,public Interphase

{

protected:

// Function to display menu choices

int choices()

{

char c = 3;

cout << "\t============" << c << " Welcome to STAR Hotel " << c << "============\n\t|\t\t\t\t\t\t|\n";

cout << "\t|\t\t(1) Show all rooms\t\t|\n";

cout << "\t|\t\t(2) Show available rooms\t|\n";

cout << "\t|\t\t(3) Find a room\t\t\t|\n";

cout << "\t|\t\t(4) Check-in\t\t\t|\n";

cout << "\t|\t\t(5) Check-out\t\t\t|\n";

cout << "\t|\t\t(6) Go to Home Page\t\t|\n";

cout << "\t|\t\t(0) Exit\t\t\t|\n\t|\t\t\t\t\t\t|\n";

cout << "\t=================================================\n";

int a;

cout << "\t\t\tCHOICE: ";

cin >> a;

cls return a;

}

public:

Receptionist()

{

Interface="Receptionist";

file="ReceptionistSecurityFile.txt";

isAuthor=1;

}

bool operations()

{

if (!accessed())

return 1;

pause

main\_menu:

cls switch (choices())

{

case 1:

AllRoom();

pause goto main\_menu;

case 2:

AvailableRoom();

pause goto main\_menu;

case 3:

findRoom();

system("pause");

goto main\_menu;

case 4:

check\_in();

pause;

goto main\_menu;

case 5:

check\_out();

pause;

goto main\_menu;

case 6:

return 1;

default:

return 0;

break;

};

}

};

class HotelAuthority: virtual public securityInterface, public customerInfo,public roomOperation,public Interphase

{

protected:

int choices()

{

char c = 3;

cout << "\t============" << c << " Welcome to STAR Hotel " << c << "============\n\t|\t\t\t\t\t\t|\n";

cout << "\t|\t\t(1) Show all rooms\t\t|\n";

cout << "\t|\t\t(2) Show available rooms\t|\n";

cout << "\t|\t\t(3) Find a room\t\t\t|\n";

cout << "\t|\t\t(4) Update Room Info \t|\n";

cout << "\t|\t\t(5) Add Room\t\t\t|\n";

cout << "\t|\t\t(6) Remove a Room \t\t|\n";

cout << "\t|\t\t(7) Show All Customers \t|\n";

cout << "\t|\t\t(8) Go to Home Page\t\t|\n";

cout << "\t|\t\t(0) Exit\t\t\t|\n\t|\t\t\t\t\t\t|\n";

cout << "\t=================================================\n";

int a;

cout << "\t\t\tCHOICE: ";

cin >> a;

cls return a;

}

public:

HotelAuthority()

{

Interface="Authority";

file="AuthoritySecurityFile.txt";

}

bool firstAuthorityAccess()

{

ifstream sc(file);

if (check(sc)) {sc.close();return true;}

sc.close();

cout << "\tPlease Register as the Author of this Hotel.\n";

bool x=setSecurity();

if(x)

{

cout << "\tAuthority Registration Successfull.\n";

pause

cls

}

return x;

}

bool operations()

{

if (!accessed())

return 1;

pause

main\_menu:

cls switch (choices())

{

case 1:

AllRoom();

pause goto main\_menu;

case 2:

AvailableRoom();

pause goto main\_menu;

case 3:

findRoom();

system("pause");

goto main\_menu;

case 4:

UpdateRoom();

pause;

goto main\_menu;

case 5:

Addroom();

pause;

goto main\_menu;

case 6:

RemoveRoom();

pause;

goto main\_menu;

case 7:

showAllCustomers();

pause;

goto main\_menu;

case 8:

return 1;

default:

return 0;

break;

};

}

~HotelAuthority()

{

cout << "\t\t\t Have a Nice Day.\n";

cout << "\t------------------Thanks for Using---------------\n";

cout << "\t=================================================\n\n";

}

};

class Users : virtual public securityInterface, public roomMng,public Interphase

{

protected:

int choices()

{

char c = 3;

cout << "\t============" << c << " Welcome to STAR Hotel " << c << "============\n\t|\t\t\t\t\t\t|\n";

cout << "\t|\t\t(1) Show all rooms\t\t|\n";

cout << "\t|\t\t(2) Show available rooms\t|\n";

cout << "\t|\t\t(3) Your Rooms \t\t\t|\n";

cout << "\t|\t\t(4) Find a room\t\t\t|\n";

cout << "\t|\t\t(5) Check-in\t\t\t|\n";

cout << "\t|\t\t(6) Check-out\t\t\t|\n";

cout << "\t|\t\t(7) Go to Home Page\t\t|\n";

cout << "\t|\t\t(0) Exit\t\t\t|\n\t|\t\t\t\t\t\t|\n";

cout << "\t=================================================\n";

int a;

cout << "\t\t\tCHOICE: ";

cin >> a;

cls return a;

}

public:

Users()

{

Interface="Customers";

file="UsersSecurityFile.txt";

isAuthor=0;

}

bool operations()

{

if (!accessed())

return 1;

pause

main\_menu:

cls switch (choices())

{

case 1:

AllRoom();

pause goto main\_menu;

case 2:

AvailableRoom();

pause goto main\_menu;

case 3:

MyRooms();

system("pause");

goto main\_menu;

case 4:

findRoom();

pause;

goto main\_menu;

case 5:

check\_in();

pause;

goto main\_menu;

case 6:

cout << "Sorry!! This feature is not available at the moment.\n\tPlease contact with the receptionist." << endl;

pause;

goto main\_menu;

case 7:

return 1;

pause goto main\_menu;

default:

return 0;

break;

};

}

};

int Interfaces()

{

char c=3;

cout <<"\t" << c <<"Welcome to Star Hotel" << c << endl;

cout << "\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\t| Your Identity |\n\t|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

cout << "\t|(1) Customer |\n\t|(2) Receptionist |\n\t|(3) Hotel Authority |\n\t|(4) Exit Program |\n\t|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n\tEnter Your Choice: ";

int a;

cin >> a;

cls return a;

}

int main()

{

cout << "\033[44m";

cls;

const char\* white = "\033[1;37m";

cout << white << endl;

cls;

HotelAuthority h;

Receptionist r;

Users u;

Interphase \*Interface;

if (!h.firstAuthorityAccess())

{

cout << "\tSorry!! Something went wrong!!\n";

return 0;

}

int x;

home\_page:

x = Interfaces();

if(x<1 || x>3) return 0;

Interface = ((x == 1) ? (Interphase\*)&u : (x == 2) ? (Interphase\*)&r :(Interphase\*)&h);

if(!Interface)

{

cout << "\tSorry!! Something went wrong!!\n";

return 0;

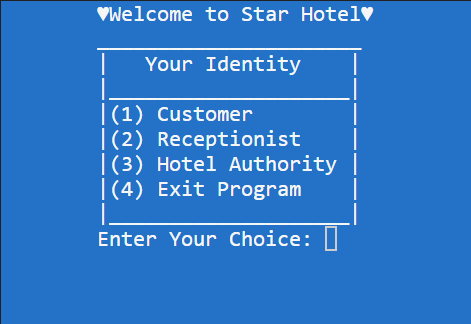
}

Interface->operations();

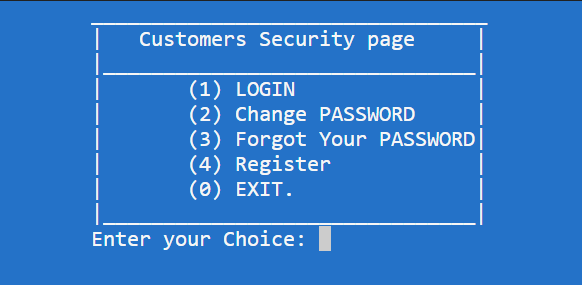
goto home\_page;

}

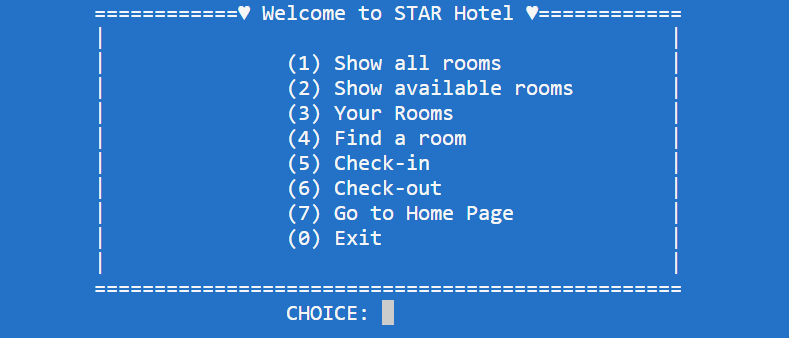
**Here is the homepage:**



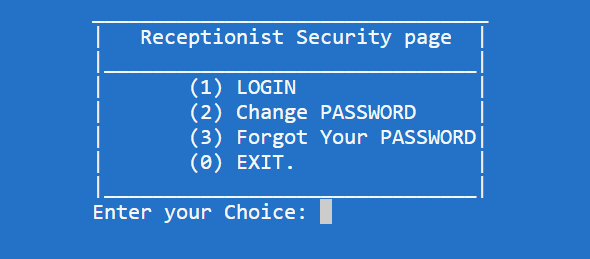
**Here is the Security page of Customer Interface:**



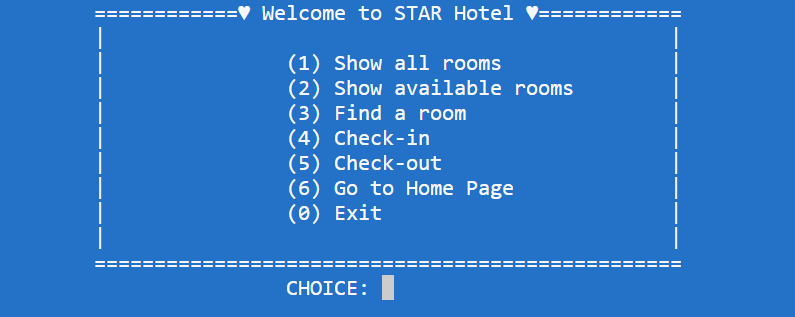
**Here is the Functionality of Customer Interface:**



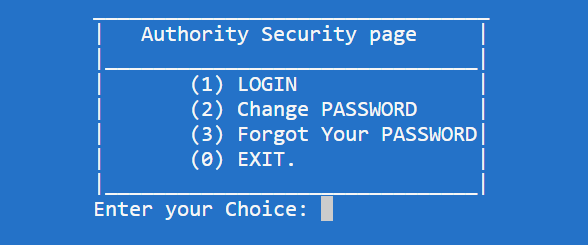
**Here is the Security Interface of Receptionist:**



**Here is the functionality of Receptionist Interface:**



**Here is the Security Interface of Hotel Authority Interface:**



**Here is the functionality of Hotel Authority Interface:**

