

**MINI-PROJECT -REPORT**

**ON**

**Hotel Reservation System**

*NAME: Dev Taliyan*

*SECTION: 3B*

*UID: 24BCA10444*

*SUBJECT: Object Oriented Programming Language*

*SUBMITTED TO: Miss. Jyoti Rani*

1. **Abstract**

The Hotel Management System is a C++based application designed to streamline core hotel operations such as room booking, customer record management, editing and deleting guest information, and handling check-out procedures through a user-friendly, menu-driven console interface. Leveraging object-oriented programming principles, the system encapsulates all guest details, ensures data persistence via binary file handling, and provides robust methods for modifying and retrieving customer records. This project facilitates efficient administration of hotel services by automating routine tasks, enhancing data accuracy, and improving overall service quality for both staff and guests.

1. **Objectives**

. Automate hotel room booking and customer records for increased operational efficiency.

. Minimize manual errors by maintaining accurate, computerized data.

. Provide a simple and user-friendly menu system for hotel staff.

. Enable quick searching, editing, and deleting of customer information.​​

1. **Tools and Technologies Used**

| Component | Description |
| --- | --- |
| Programming Language | C++ |
| Compiler | GCC / Turbo C++ / Code: Blocks |
| Operating System | Windows / Linux |
| Concepts Used | Classes, Functions, Loops, Conditional Statements |

1. **System Design**

\* User enters a room number (instead of PIN in the ATM example).

\*The system checks if the room is booked or the customer exists.

\* If found, menu options are displayed:  
 a. Book a Room  
 b. Display All Booked Rooms  
 c. Display Customer Record  
 d. Edit Customer Record  
 e. Delete Customer Record  
 f. Check-Out  
 g. Exit

\*User chooses an option.

\*The corresponding function executes as per choice:

* + Booking a room records customer details to file.
  + Display options show bookings or customer info.
  + Edit allows updating customer data.
  + Delete removes customer record.
  + Check-Out deletes booking of a room.
* The system loops back to the main menu until exit is chosen

1. **Source Code (C++)**

#include <iostream>

#include <fstream>

#include <iomanip>

#include <string>

using namespace std;

class Hotel {

int room\_no;

string name;

string address;

string phone;

public:

void mainMenu();

void addCustomer();

void displayRooms();

void displayCustomer(int);

void editCustomer(int);

void deleteCustomer(int);

void checkOut(int);

};

void Hotel::mainMenu() {

int choice;

while (true) {

system("cls"); // clear screen (works on Windows)

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

cout << "\n\t\t HOTEL MANAGEMENT SYSTEM";

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

cout << "\n\n\t1. Book a Room";

cout << "\n\t2. Display All Booked Rooms";

cout << "\n\t3. Display Customer Record";

cout << "\n\t4. Edit Customer Record";

cout << "\n\t5. Delete Customer Record";

cout << "\n\t6. Check-Out";

cout << "\n\t7. Exit";

cout << "\n\n\tEnter your choice: ";

cin >> choice;

switch (choice) {

case 1: addCustomer(); break;

case 2: displayRooms(); break;

case 3: {

int r;

cout << "\nEnter Room No.: ";

cin >> r;

displayCustomer(r);

break;

}

case 4: {

int r;

cout << "\nEnter Room No.: ";

cin >> r;

editCustomer(r);

break;

}

case 5: {

int r;

cout << "\nEnter Room No.: ";

cin >> r;

deleteCustomer(r);

break;

}

case 6: {

int r;

cout << "\nEnter Room No.: ";

cin >> r;

checkOut(r);

break;

}

case 7:

cout << "\nExiting... Thank you!";

exit(0);

default:

cout << "\nInvalid choice. Try again!";

}

cout << "\n\nPress Enter to continue...";

cin.ignore();

cin.get();

}

}

void Hotel::addCustomer() {

ofstream fout("hotel.dat", ios::app | ios::binary);

cout << "\nEnter Room No.: ";

cin >> room\_no;

cout << "Enter Name: ";

cin.ignore();

getline(cin, name);

cout << "Enter Address: ";

getline(cin, address);

cout << "Enter Phone No.: ";

getline(cin, phone);

fout.write((char\*)this, sizeof(Hotel));

fout.close();

cout << "\nRoom Booked Successfully!";

}

void Hotel::displayRooms() {

ifstream fin("hotel.dat", ios::in | ios::binary);

if (!fin) {

cout << "\nNo records found!";

return;

}

cout << "\n\n\t\tList of Booked Rooms\n";

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

cout << setw(10) << "Room No" << setw(20) << "Name" << setw(25) << "Address" << setw(15) << "Phone\n";

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

while (fin.read((char\*)this, sizeof(Hotel))) {

cout << setw(10) << room\_no << setw(20) << name << setw(25) << address << setw(15) << phone << endl;

}

fin.close();

}

void Hotel::displayCustomer(int r) {

ifstream fin("hotel.dat", ios::in | ios::binary);

bool found = false;

while (fin.read((char\*)this, sizeof(Hotel))) {

if (room\_no == r) {

cout << "\nRoom No.: " << room\_no;

cout << "\nName: " << name;

cout << "\nAddress: " << address;

cout << "\nPhone: " << phone;

found = true;

break;

}

}

fin.close();

if (!found)

cout << "\nCustomer not found!";

}

void Hotel::editCustomer(int r) {

fstream file("hotel.dat", ios::in | ios::out | ios::binary);

bool found = false;

while (file.read((char\*)this, sizeof(Hotel))) {

if (room\_no == r) {

cout << "\nEnter New Name: ";

cin.ignore();

getline(cin, name);

cout << "Enter New Address: ";

getline(cin, address);

cout << "Enter New Phone: ";

getline(cin, phone);

int pos = -1 \* sizeof(Hotel);

file.seekp(pos, ios::cur);

file.write((char\*)this, sizeof(Hotel));

cout << "\nRecord Updated Successfully!";

found = true;

break;

}

}

file.close();

if (!found)

cout << "\nCustomer not found!";

}

void Hotel::deleteCustomer(int r) {

ifstream fin("hotel.dat", ios::in | ios::binary);

ofstream fout("temp.dat", ios::out | ios::binary);

bool found = false;

while (fin.read((char\*)this, sizeof(Hotel))) {

if (room\_no != r)

fout.write((char\*)this, sizeof(Hotel));

else

found = true;

}

fin.close();

fout.close();

remove("hotel.dat");

rename("temp.dat", "hotel.dat");

if (found)

cout << "\nRecord Deleted Successfully!";

else

cout << "\nCustomer not found!";

}

void Hotel::checkOut(int r) {

deleteCustomer(r);

cout << "\nCustomer Checked Out!";

}

int main() {

Hotel h;

h.mainMenu();

return 0;

}

1. **Sample Output**

=====================================

Welcome to Hotel System

=====================================

Enter Room No. to access your booking: 101

--- Hotel Menu ---

1. Book a Room

2. Display All Booked Rooms

3. Display Customer Record

4. Edit Customer Record

5. Delete Customer Record

6. Check-Out

7. Exit

Enter your choice: 3

Customer Details:

Room No.: 101

Name: John Doe

Address: 123 Street, City

Phone: 9876543210

--- Hotel Menu ---

1. Book a Room

2. Display All Booked Rooms

3. Display Customer Record

4. Edit Customer Record

5. Delete Customer Record

6. Check-Out

7. Exit

Enter your choice: 4

Enter New Name: John Smith

Enter New Address: 456 Avenue, City

Enter New Phone: 9123456780

Record Updated Successfully!

--- Hotel Menu ---

Enter your choice: 2

List of Booked Rooms:

Room No Name Address Phone

101 John Smith 456 Avenue, City 9123456780

--- Hotel Menu ---

Enter your choice: 6

Customer Checked Out!

--- Hotel Menu ---

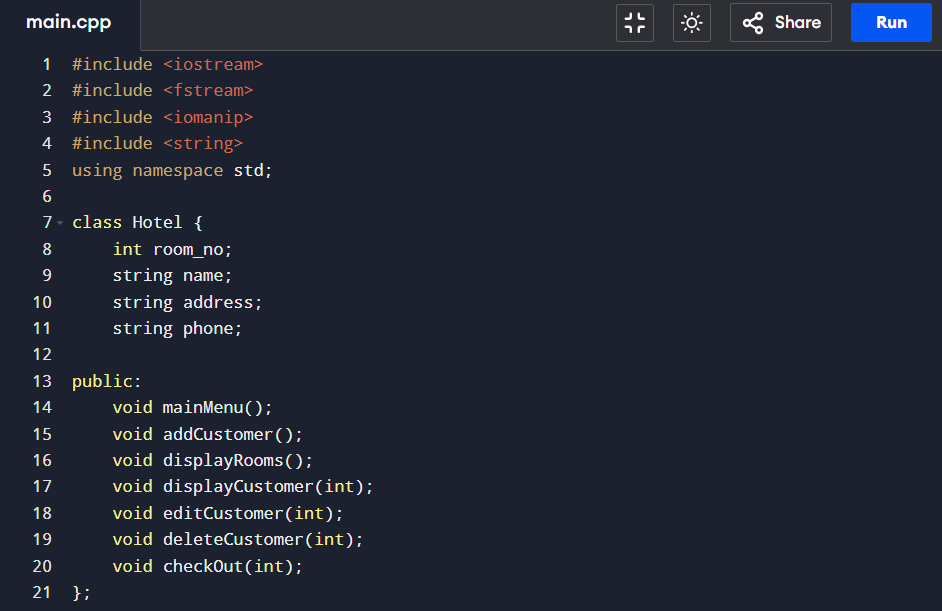
Enter your choice: 7

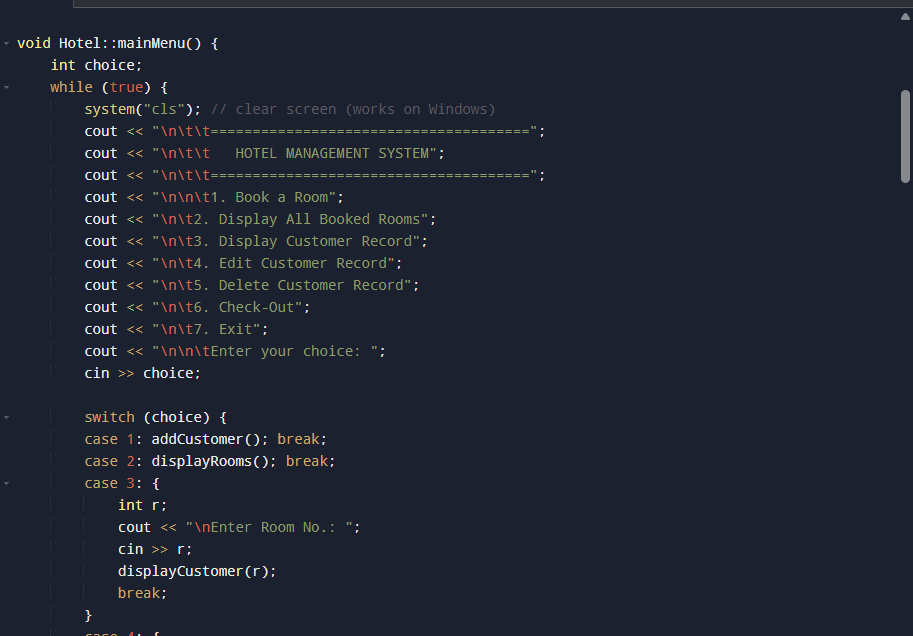
Thank you for using the Hotel Management System!

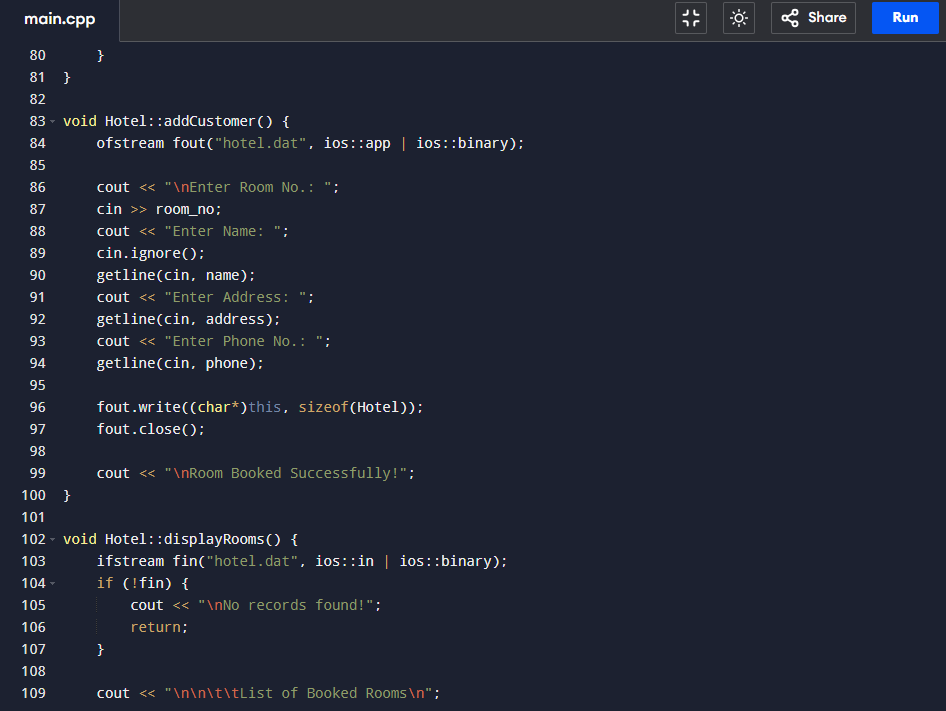
1. **Conclusion**

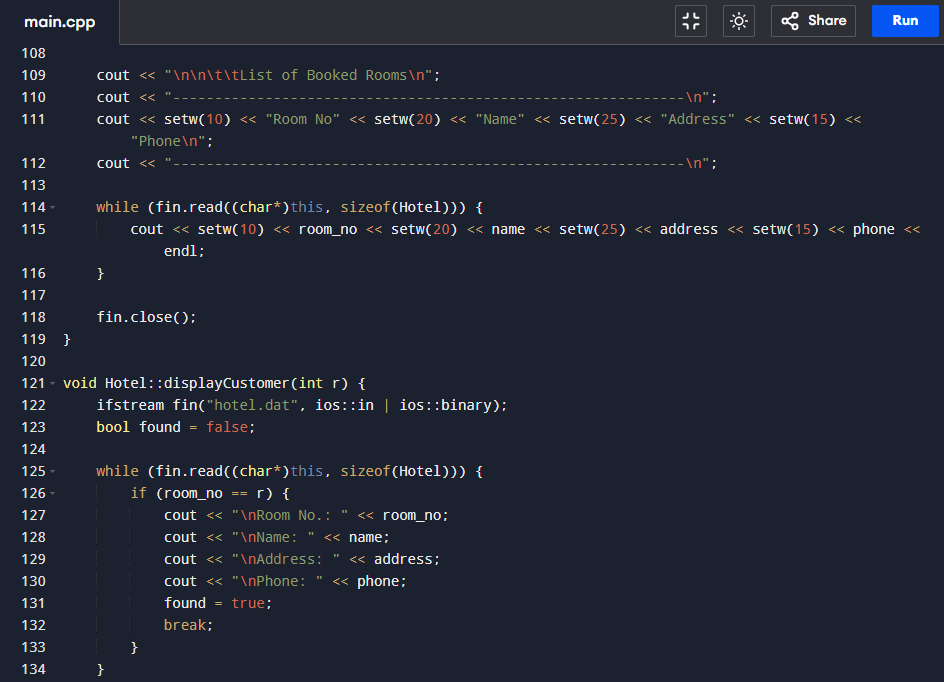
The Hotel Management System project successfully automates the core functions of hotel operations including room booking, customer record management, editing, deleting records, and checking out. By leveraging object-oriented programming and file handling in C++, the system improves operational efficiency, reduces manual errors, and ensures quick and reliable access to customer data. This computerized solution replaces conventional manual record keeping, saving time and enhancing the accuracy of information management. Overall, the system streamlines daily hotel management tasks and provides a foundation for further enhancements like multi-user support and graphical interfaces.

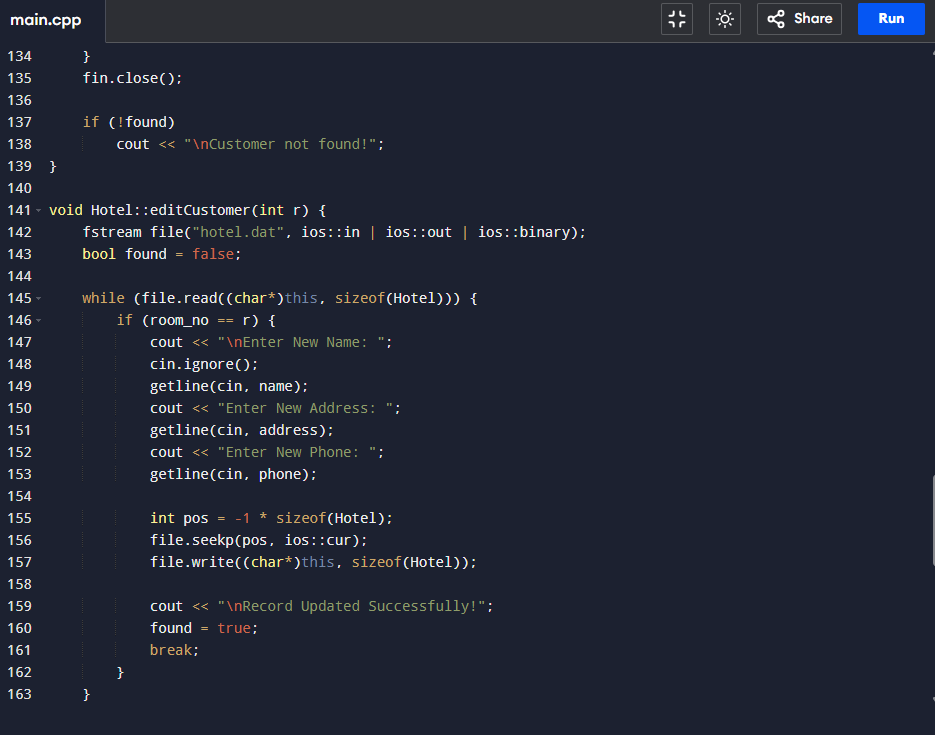
**9.Screenshot or Result**

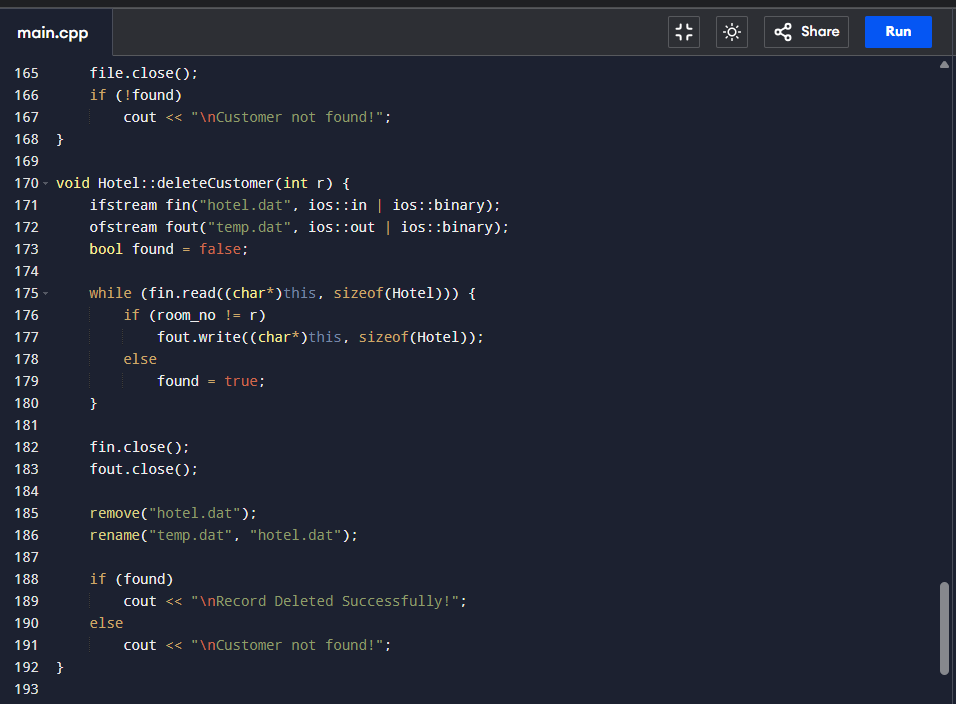
****

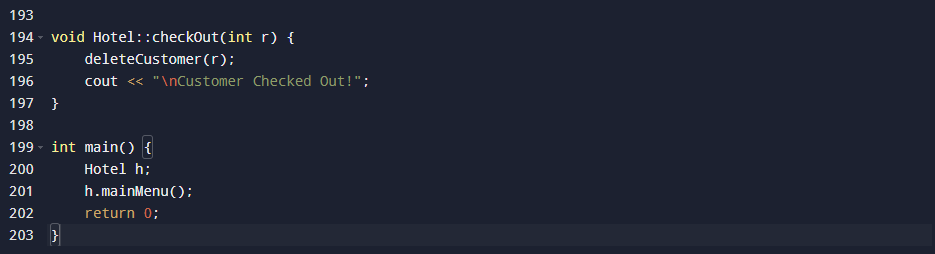
****

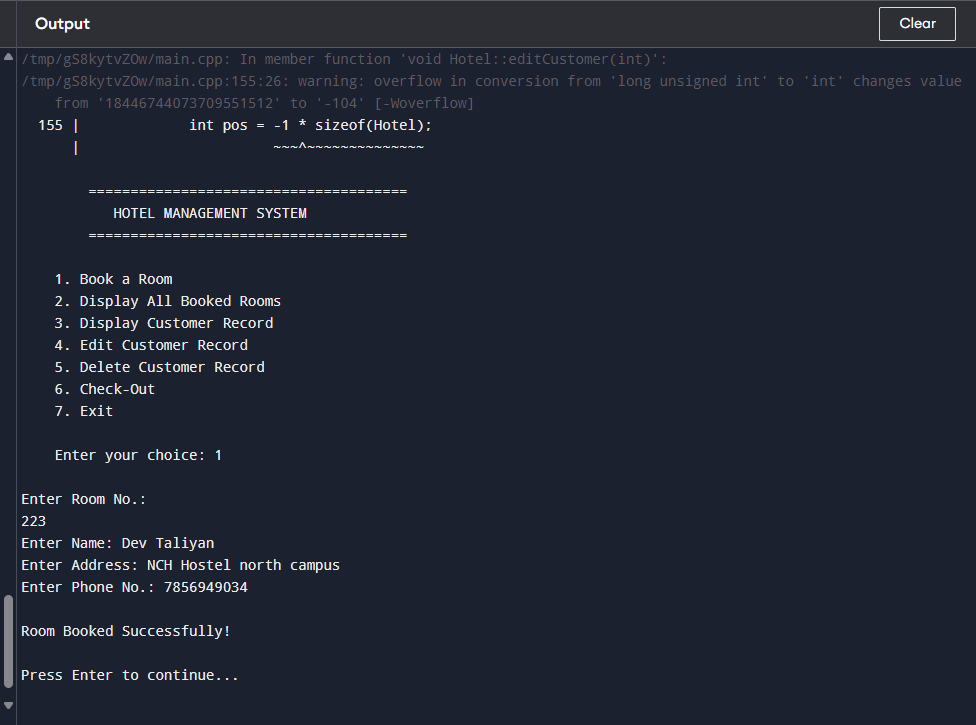
****

****

****

****

****

****