HOTEL RESERVATION SYSTEM

CASE STUDY REPORT



## Submitted by:

Ayush Kumar

## Submitted to:

Ms. Jyoti Rani

## Course:

Object-Oriented Programming (OOPs)

## University:

Chandigarh University

|  |
| --- |
| Table of Contents |
| 1. Introduction |
| 2. Objectives |
| 3. Tools and Technologies Used |
| 4. System Design and Structure |
| 5. Features |
| 6. Working of the System |
| 7. Code |
| 8. Code Explanation |
| 9. Advantages and Limitations |
| 10. Future Enhancements |
| 11. Conclusion |
|  |

# 1. Introduction

The Hotel Reservation System is a console-based project developed in C++ that simplifies the process of booking, managing, and checking out hotel rooms. The project demonstrates the application of Object-Oriented Programming (OOP) concepts such as classes, functions, file handling, and encapsulation. It is designed to automate manual tasks in hotels, improving efficiency and reducing human errors.

# 2. Objectives

The main objectives of the Hotel Reservation System are:  
• To automate the process of room booking and management.  
• To provide a user-friendly console interface for hotel staff.  
• To demonstrate file handling and object-oriented programming in C++.  
• To enable easy storage and retrieval of customer records.

# 3. Tools and Technologies Used

• Programming Language: C++  
• IDE: Visual Studio Code / Turbo C++ / Dev-C++  
• Concepts: Classes, File Handling, Loops, Switch Case  
• File Type: Binary file (hotel.dat)

# 4. System Design and Structure

The system is designed using a single class 'Hotel' that encapsulates customer data such as room number, name, address, and phone number. It includes various functions to perform operations such as adding, editing, deleting, and displaying customer records. File handling is used to store data persistently.

# 5. Features

1. Book a Room  
2. Display All Booked Rooms  
3. Display a Specific Customer Record  
4. Edit a Customer Record  
5. Delete a Customer Record  
6. Check-Out Function  
7. Exit Option

# 6. Working of the System

The Hotel Reservation System follows a menu-driven structure where users can choose operations such as booking, viewing, editing, or deleting records. Data is stored in a binary file named 'hotel.dat'. Each customer’s information is written to the file when a booking is made and can be accessed or modified later using their room number.

7. CODE

#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);

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;

}

# 8. Code Explanation

The project uses a class 'Hotel' that contains attributes for storing customer information and member functions for various operations. File handling in binary mode ensures efficient storage. The main() function creates an object of the Hotel class and repeatedly calls the main menu to perform actions. The code is modular and easy to maintain.

# 9. Advantages and Limitations

Advantages:  
• User-friendly and easy to understand.  
• Data stored permanently using file handling.  
• Demonstrates OOP and data management.  
  
Limitations:  
• No graphical interface.  
• No real-time database connection.  
• Single-user system.

# 10. Future Enhancements

• Develop a GUI version using Qt or web technologies.  
• Connect with an SQL database for secure data storage.  
• Add authentication for admin and staff.  
• Include billing and payment modules.  
• Provide cloud-based booking access.

# 11. Conclusion

The Hotel Reservation System project successfully demonstrates how object-oriented programming and file handling can be used to create a real-world application. It simplifies hotel management processes and showcases core programming skills. This project can be further enhanced with database integration and GUI for more practical use.