Summary of the Hotel Reservation System Code:

This C++ program simulates a basic Hotel Reservation System with the following features:

1**. Classes and Objects:**

- Customer Class: Represents customer details like name, address, phone number, booking ID, date of stay, and advance payment.

-Room Class: Represents rooms in the hotel, with attributes such as room number, room type (AC/Non-AC), comfort type (Standard/Luxury), size (Single/Double), daily rent, and booking status.

- Key Methods:
  - addRoom(int): Allows the admin to add a new room with specified attributes.
  - searchRoom(int): Searches for a room by room number and displays its availability.
  - displayRoom(Room): Displays the details of a room.

2. **HotelMgnt (Hotel Management) Class:**

- Inherits from the Room class and provides functions to manage customer interactions and reservations:
  - checkIn(): Allows a customer to check into a room, taking their details and updating the room status to "booked."
  - getAvailRoom(): Displays a list of all available rooms in the hotel.
  - searchCustomer(char*): Searches for a customer by name and displays their booking details.
  - checkOut(int): Handles the customer's checkout process by calculating the bill based on the number of days stayed and the room rent, then resets the room status to "available."
  - guestSummaryReport(): Provides a summary report of all guests currently staying in the hotel.

3. **Global Variables:**

- rooms[max]: An array that stores all rooms in the hotel (with a maximum capacity defined by max).
- count: Keeps track of the number of rooms added to the system.

4. **Main Menu:**

- The program presents a menu where the user can:
  1. Manage rooms (add or search for rooms).
  2. Check in a customer.
  3. View available rooms.
  4. Search for a customer by name.
  5. Check out a customer.
  6. View a guest summary report.
  7. Exit the program.

Key Features:

- Room management (add/search).
- Customer check-in and check-out functionality.
- Availability display for rooms.
- Guest summary reports.
- Basic billing system during checkout, calculating the total amount due based on room rent and stay duration.

This system offers a simple way for hotel staff to manage rooms and customer reservations efficiently.

2. Table of Contents

- Introduction…… 1

- Objectives……... 2-3

- Methodology/Implementation……4-8

-Code Structure……. 9-26

- Conclusion……... 27

- References…… 28

- Appendix…….. 29-32

3. Introduction

Introduction to the Hotel Reservation System Code

The Hotel Reservation System is designed to manage hotel room bookings, customer check-ins, and check-outs, while also providing key administrative functions. The primary goal of this system is to automate and streamline the hotel reservation process, making it easier for both hotel staff and customers to handle room reservations.

This system allows users to:

1. Manage Rooms: Admins can add new rooms to the system with specific attributes like room type, size, rent, and availability.

2. Check-In/Check-Out Guests: Customers can be checked into available rooms, and their details such as booking ID, name, contact information, and stay duration are recorded. Upon checkout, the system calculates the total bill based on the number of days stayed and the room rent.

3. Search for Available Rooms: The system provides a feature to check the availability of rooms, ensuring that bookings are made for only vacant rooms.

4. Customer Lookup: Hotel staff can search for guests based on their name and retrieve relevant booking information.

5. Generate Guest Reports: The system can generate summary reports showing all currently checked-in guests.

This program, written in C++, simulates the operations of a basic hotel management system, leveraging object-oriented programming principles to encapsulate customer and room data in classes. It simplifies daily hotel management tasks by automating room allocation, billing, and customer information tracking, leading to improved efficiency and reduced manual errors.

4. Objectives of the Hotel Reservation System Code

The primary objectives of this Hotel Reservation System are:

General Objective:

To develop a simple and efficient system that automates the management of hotel room bookings, customer check-ins, check-outs, and billing processes.

Specific Objectives:

1. Room Management:

Provide a functionality for hotel administrators to add and manage room details, including room type (AC/Non-AC), comfort level (Standard/Luxury), size, daily rent, and booking status.

2. Customer Reservation:

Allow customers to check into available rooms by entering necessary details such as name, address, phone number, and booking duration.

Ensure that rooms already booked by other customers are marked as unavailable.

3. Check-Out and Billing:

Automate the process of generating a bill based on the number of days stayed and the daily room rent.

Calculate the final payable amount by subtracting any advance payment made during check-in.

4. Room Availability:

Provide a feature to check and display all available rooms in real-time, ensuring efficient room allocation and preventing overbooking.

5. Customer Search:

Enable hotel staff to search for customers by name, retrieving their booking information, including room number and personal details.

6. Guest Summary Reports:

Generate summary reports of all currently checked-in guests, displaying important information like customer name, room number, address, and contact details.

7. Data Integrity:

Ensure that room data and customer information are consistently managed and updated throughout the system to maintain accuracy.

By achieving these objectives, the system reduces manual errors, enhances the customer experience, and improves overall operational efficiency in managing hotel reservations.

5. Methodology/Implementation of the Hotel Reservation System

The Hotel Reservation System is implemented using object-oriented programming in C++. The design and development of the system revolve around the core functionalities required to manage room bookings and customer operations efficiently. The system consists of several classes and methods that handle specific tasks such as room management, customer check-in, and check-out, as well as bill generation.

Here's a breakdown of the methodology and implementation:

1. **System Design**:

The system is designed using the following object-oriented principles:

Encapsulation: Data related to customers and rooms is encapsulated within the `Customer` and `Room` classes, respectively.

Inheritance: The `HotelMgnt` class inherits from the `Room` class to reuse and extend room management functionality.

Modularity: The system is broken down into modules (methods) that handle specific tasks such as adding rooms, checking in customers, and generating reports.

2. **Key Classes**:

Customer Class: This class holds the details of the customers, such as name, address, phone number, booking ID, date of stay (from and to), and advance payment. These attributes are associated with each booking in the system.

Room Class: The `Room` class manages the room details, including:

- Room type (AC/Non-AC), size (Single/Double), rent, and status (booked or available).

- Key Methods:

- `addRoom(int rno)`: Allows the administrator to add rooms to the system by setting the room number and details like room type, size, and rent.

- `searchRoom(int rno)`: Searches for a room by its room number and displays its availability and details.

- `displayRoom(Room)`: Displays the specific details of a room, such as its rent, type, and availability status.

- HotelMgnt Class: This class manages customer-related activities such as check-in, check-out, and generating guest reports. It inherits the room management functions from the `Room` class.

- Key Methods:

- `checkIn()`: Facilitates the process of checking in a customer. It collects customer details like name, address, booking duration, and advance payment, and marks the room as "booked."

- `checkOut(int roomNum)`: Handles the checkout process, calculating the final bill based on the number of days stayed and the room rent, minus the advance payment.

- `getAvailRoom()`: Displays a list of all available rooms in the hotel.

- `guestSummaryReport()`: Generates a summary report of all currently checked-in guests, showing their name, room number, and contact details.

- `searchCustomer(char *pname)`: Searches for a customer by name and retrieves their booking information.

3**. Data Flow and Operations**:

The system operates based on user inputs through a menu-driven interface. The key operations are:

Adding a Room: Admins can add a room by specifying the room number, type (AC or Non-AC), comfort level (Standard or Luxury), and rent. The room is then marked as available until it is booked.

Checking in a Customer: When a customer checks into a room, the system prompts the user to input the customer’s details (name, address, phone number, etc.). The room's status is updated to "booked."

Room Search and Availability: Users can search for a room by room number to check its availability. The system also allows for listing all available rooms for easy booking.

Checking out a Customer: The system calculates the total bill based on the number of days the customer stayed and the room's rent. It then subtracts any advance payment and updates the room’s status back to "available."

Guest Report Generation: A summary report of all currently checked-in guests can be generated, showing their personal details and room information.

4. **Data Structures Used**:

Room Array: The program uses an array (`rooms[max]`) to store all room objects. This allows efficient access and management of room data. The global variable `count` keeps track of how many rooms are added to the system.

String Operations: Functions like `strcmp()` are used for customer name comparisons during search operations.

5. **Algorithms**:

Search Algorithm: Linear search is employed to find rooms by their room number and customers by their name. For each search operation, the system iterates through the array of rooms and checks for matches.

Billing Algorithm: The billing process calculates the total payable amount using the formula:

$$total\ bill = number\ of\ days * room\ rent - advance\ payment$$

6. **User Interface**:

The system uses a command-line interface (CLI) where users are presented with a menu. They can select from options such as adding rooms, checking in, checking out, searching for rooms, and generating reports. Each operation prompts the user for necessary inputs (e.g., room number, customer name) and displays appropriate output.

7. **Error Handling**:

The system includes basic error handling through conditional checks. For example:

- If a room is already booked, the system prevents duplicate bookings.

- If no rooms are added, the system notifies the user to add rooms before proceeding with check-ins or check-outs.

- Incorrect menu options prompt users to re-enter valid choices.

8. **Implementation Flow**:

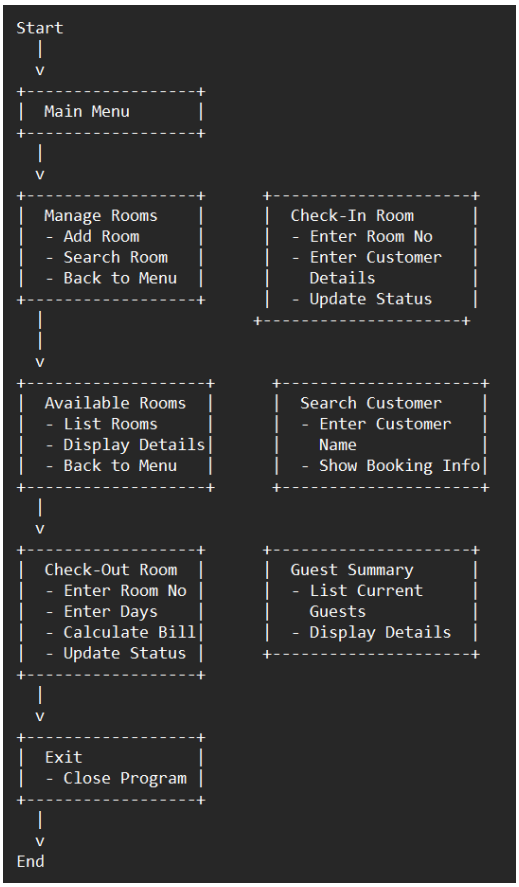
1. Initialization: The system initializes with no rooms. Admins must first add rooms.

2. Room Management: Rooms are added and managed through the `manageRooms()` function.

3. Booking: Customers can be checked into available rooms, and their details are stored.

4. Check-Out and Billing: Bills are generated based on the number of days stayed, and rooms are freed up after checkout.

5. Reports and Searches: Users can view guest reports and search for specific customers.



Conclusion:

This methodical approach ensures that the hotel reservation system is both functional and easy to use. The system efficiently manages room bookings, customer information, and billing operations, helping hotel staff streamline daily operations and minimize errors

6. Code Structure

```
#include<iostream>

#include<string.h>


#define max 100

using namespace std;


//Class Customer

class Customer

{

public:

char name[100];

char address[100];

char phone[12];

char from_date[20];

char to_date[20];

float payment_advance;

int booking_id;

};



class Room

{
```

```
public:
char type;
char stype;
char ac;
int roomNumber;
int rent;
int status;

class Customer cust;
class Room addRoom(int);
void searchRoom(int);
void deleteRoom(int);
void displayRoom(Room);
};

//Global Declarations
class Room rooms[max];
int count=0;


Room Room::addRoom(int rno)
{
class Room room;
```

```
room.roomNumber=rno;
cout<<"\nType AC/Non-AC (A/N) : ";
cin>>room.ac;
cout<<"\nType Comfort (S/N) : ";
cin>>room.type;
cout<<"\nType Size (B/S) : ";
cin>>room.stype;
cout<<"\nDaily Rent : ";
cin>>room.rent;
room.status=0;

cout<<"\n Room Added Successfully!";
return room;
}



void Room::searchRoom(int rno)
{
int i,found=0;
for(i=0;i<count;i++)
{
if(rooms[i].roomNumber==rno)
{
```

```
found=1;
break;
}
}
if(found==1)
{
cout<<"Room Details\n";
if(rooms[i].status==1)
{
cout<<"\nRoom is Reserved";
}
else
{
cout<<"\nRoom is available";
}
displayRoom(rooms[i]);
}
else
{
cout<<"\nRoom not found";
}
}
```

```
void Room::displayRoom(Room tempRoom)
{
cout<<"\nRoom Number: \t"<<tempRoom.roomNumber;
cout<<"\nType AC/Non-AC (A/N) "<<tempRoom.ac;
cout<<"\nType Comfort (S/N) "<<tempRoom.type;
cout<<"\nType Size (B/S) "<<tempRoom.stype;
cout<<"\nRent: "<<tempRoom.rent;
}

//hotel management class
class HotelMgnt:protected Room
{
public:
void checkIn();
void getAvailRoom();
void searchCustomer(char *);
void checkOut(int);
void guestSummaryReport();
};

void HotelMgnt::guestSummaryReport(){
```

```
if(count==0){
        cout<<"\n No Guest in Hotel !!";
}
for(int i=0;i<count;i++)
{
if(rooms[i].status==1)
{
cout<<"\n Customer First Name : "<<rooms[i].cust.name;
cout<<"\n Room Number : "<<rooms[i].roomNumber;
cout<<"\n Address (only city) : "<<rooms[i].cust.address;
cout<<"\n Phone : "<<rooms[i].cust.phone;
cout<<"\n---------------------------------------";
}

}

}

//hotel management reservation of room
void HotelMgnt::checkIn()
{
int i,found=0,rno;
```

```
class Room room;
cout<<"\nEnter Room number : ";
cin>>rno;
for(i=0;i<count;i++)
{
if(rooms[i].roomNumber==rno)
{
found=1;
break;
}
}
if(found==1)
{
if(rooms[i].status==1)
{
cout<<"\nRoom is already Booked";
return;
}

cout<<"\nEnter booking id: ";
cin>>rooms[i].cust.booking_id;

cout<<"\nEnter Customer Name (First Name): ";
```

```
cin>>rooms[i].cust.name;

cout<<"\nEnter Address (only city): ";
cin>>rooms[i].cust.address;

cout<<"\nEnter Phone: ";
cin>>rooms[i].cust.phone;

cout<<"\nEnter From Date: ";
cin>>rooms[i].cust.from_date;

cout<<"\nEnter to  Date: ";
cin>>rooms[i].cust.to_date;


cout<<"\nEnter Advance Payment: ";
cin>>rooms[i].cust.payment_advance;

rooms[i].status=1;

cout<<"\n Customer Checked-in Successfully..";
}
}
```

```
//hotel management shows available rooms
void HotelMgnt::getAvailRoom()
{
int i,found=0;
for(i=0;i<count;i++)
{
if(rooms[i].status==0)
{
displayRoom(rooms[i]);
cout<<"\n\nPress enter for next room";
found=1;
}
}
if(found==0)
{
cout<<"\nAll rooms are reserved";
}
}
```

```
//hotel management shows all persons that have booked room
```

```
void HotelMgnt::searchCustomer(char *pname)
{
int i,found=0;
for(i=0;i<count;i++)
{
if(rooms[i].status==1 && strcmp(rooms[i].cust.name,pname)==0)
{
cout<<"\nCustomer Name: "<<rooms[i].cust.name;
cout<<"\nRoom Number: "<<rooms[i].roomNumber;

cout<<"\n\nPress enter for next record\n";
found=1;
}
}
if(found==0)
{
cout<<"\nPerson not found.\n";
}
}

//hotel managemt generates the bill of the expenses
void HotelMgnt::checkOut(int roomNum)
```

```
{
int i,found=0,days,rno;
float billAmount=0;
for(i=0;i<count;i++)
{
if(rooms[i].status==1 && rooms[i].roomNumber==roomNum)
{
//rno = rooms[i].roomNumber;
found=1;
//getch();
break;
}
}
if(found==1)
{
cout<<"\nEnter Number of Days:\t";
cin>>days;
billAmount=days * rooms[i].rent;

cout<<"\n\t######## CheckOut Details ########\n";
cout<<"\nCustomer Name : "<<rooms[i].cust.name;
cout<<"\nRoom Number : "<<rooms[i].roomNumber;
cout<<"\nAddress : "<<rooms[i].cust.address;
```

```
cout<<"\nPhone : "<<rooms[i].cust.phone;

cout<<"\nTotal Amount Due : "<<billAmount<<" /";

cout<<"\nAdvance Paid: "<<rooms[i].cust.payment_advance<<" /";

cout<<"\n*** Total Payable: "<<billAmount-rooms[i].cust.payment_advance<<"/ only";


rooms[i].status=0;

}

}



//managing rooms (adding and searching available rooms)

void manageRooms()

{

class Room room;

int opt,rno,i,flag=0;

char ch;

do

{

cout<<"\n### Manage Rooms ###";

cout<<"\n1. Add Room";

cout<<"\n2. Search Room";

cout<<"\n3. Back to Main Menu";

cout<<"\n\nEnter Option: ";
```

```
cin>>opt;



//switch statement
switch(opt)
{
case 1:
cout<<"\nEnter Room Number: ";
cin>>rno;
i=0;
for(i=0;i<count;i++)
{
if(rooms[i].roomNumber==rno)
{
flag=1;
}
}
if(flag==1)
{
cout<<"\nRoom Number is Present.\nPlease enter unique Number";
flag=0;
}
else
```

```
{
rooms[count]=room.addRoom(rno);
count++;
}
break;
case 2:
cout<<"\nEnter room number: ";
cin>>rno;
room.searchRoom(rno);
break;
case 3:
//nothing to do
break;
default:
cout<<"\nPlease Enter correct option";
break;
}
}while(opt!=3);
}
using namespace std;
int main()
{
class HotelMgnt hm;
```

```
int i,j,opt,rno;
char ch;
char pname[100];



do
{
cout<<"######## Hotel Management #########\n";
cout<<"\n1. Manage Rooms";
cout<<"\n2. Check-In Room";
cout<<"\n3. Available Rooms";
cout<<"\n4. Search Customer";
cout<<"\n5. Check-Out Room";
cout<<"\n6. Guest Summary Report";
cout<<"\n7. Exit";
cout<<"\n\nEnter Option: ";
cin>>opt;
switch(opt)
{
case 1:
manageRooms();
break;
case 2:
```

```
if(count==0)
{
cout<<"\nRooms data is not available.\nPlease add the rooms first.";
}
else
hm.checkIn();
break;
case 3:
if(count==0)
{
cout<<"\nRooms data is not available.\nPlease add the rooms first.";
}
else
hm.getAvailRoom();
break;
case 4:
if(count==0)
{
cout<<"\nRooms are not available.\nPlease add the rooms first.";
}
else
{
cout<<"Enter Customer Name: ";
```

```
cin>>pname;
hm.searchCustomer(pname);
}
break;
case 5:
if(count==0)
{
cout<<"\nRooms are not available.\nPlease add the rooms first.";
}
else
{
cout<<"Enter Room Number : ";
cin>>rno;
hm.checkOut(rno);
}
break;
case 6:
hm.guestSummaryReport();
break;
case 7:
cout<<"\nTHANK YOU! FOR USING SOFTWARE\n";
break;
default:
```

```
cout<<"\nPlease Enter correct option";
break;
}
}while(opt!=7);


}
```

6. **Conclusion of the Hotel Reservation System Code**

The Hotel Reservation System implemented in C++ successfully automates the core functions of hotel management, such as room allocation, customer check-in/check-out, billing, and report generation. By leveraging object-oriented programming principles, the system ensures that customer and room data are efficiently organized and managed, reducing manual errors and improving the overall efficiency of hotel operations.

Key benefits of this system include:

Automated Room Management: The system allows for the easy addition of rooms, searching for availability, and displaying room details.

Streamlined Customer Check-In/Check-Out: With the customer booking process digitized, staff can easily record customer details, check availability, and ensure seamless bookings.

Accurate Billing: The system generates a final bill based on the customer's stay duration, ensuring accuracy in payments and minimizing human errors.

Real-Time Room Availability: Staff can quickly check which rooms are available and assign them to customers without conflicts.

Efficient Guest Reporting: The system provides summary reports of all guests currently staying at the hotel, enabling staff to monitor guest activities.

In conclusion, this code offers a reliable and user-friendly solution to manage a hotel's daily operations. It improves overall service efficiency, enhances customer experience, and reduces the workload for hotel staff, making it an effective tool for small to medium-sized hotels.

7. Reference

C++ Reference Documentation: A comprehensive resource for C++ standard library functions and language features.

ISO/IEC Standard for C++: The official standard document outlining the specifications for the C++ programming language.

C++ Programming :Principles and Practice using C++ by Bjarne Stroustrup: A detailed book by the creator of C++ that provides in depth coverage of C++ programming concepts and practices.

8. Appendix

The appendix provides detailed information about the code structure, data input/output examples, class definitions, and error handling mechanisms for the Hotel Reservation System implemented in C++.

A. Code Structure Overview

Classes and Methods:

Customer Class: Stores details related to customers.

Attributes: name, address, phone, from_date, to_date, payment_advance, booking_id.

Room Class: Manages room details and operations.

Attributes: type, stype, ac, roomNumber, rent, status.

Methods:

addRoom(int rno): Adds a new room with the given room number and details.

searchRoom(int rno): Searches for a room by room number and displays its details.

displayRoom(Room tempRoom): Displays the details of a specific room.

HotelMgnt Class: Manages hotel operations related to customers and rooms.

Methods:

checkIn(): Handles customer check-in, updating room status and recording customer details.

checkOut(int roomNum): Processes check-out, calculates the bill, and updates room status.

getAvailRoom(): Lists all available rooms.

searchCustomer(char *pname): Searches for customers by name and displays their booking information.

guestSummaryReport(): Generates and displays a summary report of all checked-in guests.

Global Variables:

Room rooms[max]: Array to store room objects, where max is defined as 100.

int count: Keeps track of the number of rooms currently in the system.

Input/Output Example:

```
######## Hotel Management ##########

1. Manage Rooms
2. Check-In Room
3. Available Rooms
4. Search Customer
5. Check-Out Room
6. Guest Summary Report
7. Exit

Enter Option: 1

### Manage Rooms ###
1. Add Room
2. Search Room
3. Back to Main Menu

Enter Option: 1

Enter Room Number: 101

Type AC/Non-AC (A/N) : A

Type Comfort (S/N) : S

Type Size (B/S) : B

Daily Rent : 5000

 Room Added Successfully!
### Manage Rooms ###
1. Add Room
2. Search Room
3. Back to Main Menu

Enter Option: 2

Enter room number: 101
Room Details

Room is available
Room Number:    101
Type AC/Non-AC (A/N) A
Type Comfort (S/N) S
Type Size (B/S) B
Rent: 5000
### Manage Rooms ###
1. Add Room
2. Search Room
3. Back to Main Menu
```

D. Error Handling Mechanisms

Duplicate Room Number Check:

Before adding a new room, the system checks if the room number already exists. If it does, the user is informed and prompted to enter a unique room number.

Room Availability Check:

When checking in a customer, the system verifies if the room is already booked. If the room is unavailable, the system notifies the user.

Input Validation:

The system ensures that all user inputs are valid, such as ensuring numerical values for room rent and advance payment, and valid date formats for check-in and check-out dates.