HOTEL MANAGEMENT PROJECT

PROJECT REPORT

Submitted by

Vijay Makkad [RA2211003010686]
Shubham Khatri [RA2211003010714]

Under the Guidance of

Dr .M. KANDAN

Assistant Professor, Department of Computing Technologies

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2023

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this Project Report titled **"Hotel Management System"** is the bonafide work done by **Vijay Makkad [RA2211003010686], Shubham Khatri [RA2211003010714]** who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE
Dr.M.Kandan
OODP – Course Faculty
Assistant Professor
Department of Computing Technologies
SRMIST

SIGNATURE
Dr.M.Pushpalatha
Professor & Head
Department of Computing Technologies
School of Computing
SRMIST

TABLE OF CONTENTS

1. PROBLEM STATEMENTS

- Difficulty in managing hotel rooms: The absence of the Room class could make it hard to keep track of which rooms are available and which are not. This could cause issues for hotel staff when they need to assign guests to rooms, and there may be errors in double-booking rooms or booking rooms that are already occupied.

- Lack of customer information: The Customer class is missing, which means that customer information such as name, address, and phone number cannot be stored. This could cause problems with guest check-in and check-out processes, as well as accounting and billing processes.

- Inability to calculate the duration of a stay: The dates class contains functions that calculate the number of days between two dates, which could be crucial for billing and payment purposes. Without these functions, hotel staff would need to manually calculate the duration of each guest's stay.

- No reservation system: There is no system for reserving rooms in the code. This means that customers would need to call or visit the hotel in person to inquire about room availability and book a room. A reservation system could make the process more efficient for both the hotel staff and the guests.

- Lack of reporting: The absence of the guestSummaryReport function in the HotelMgnt class means that there is no easy way to generate reports on guest occupancy, room availability, and other important metrics. This could make it challenging to manage the hotel effectively and make data-driven decisions.

1.1.1 Existing System

- In the existing system, the person who wants to book a room has to visit the hotel for booking hotel rooms, and enquiry.
- The existing system is manual system.
- The hotel management has to keep records of rooms manually.

Disadvantages:

- It is a time consuming process.
- There is no surety of availability of rooms.
- Paper work results in need of lot of space to keep the data.
- Lack of security.
- Chances of human errors.

1.2 OBJECTIVE OF THE PROJECT

- To enable online booking via the internet.
- To enable automated data entry methods.
- Ensure efficient and reliable communication within the hotel.
- Avoid data entry errors by use of input masks.
- Enable easy authorized modification of data.
- Enforce security measures to avoid unauthorized access to guest records.
- Enable fast and easy retrieval of guest records and data for fast reference activities

2 MODULES OF PROJECT

1. **noOfLeapYearDays(int date[])**: This function takes an array of three integers representing the date and calculates the number of leap years that have passed until the year of the date.It calculates the year of the date and checks how many years have passed until that year by dividing it by 4, 400, and 100. The sum of the results of the division is the number of leap years.It returns an integer value representing the number of leap years.
2. **noOfMonthsDays(int date[])**: This function takes an array of three integers representing the date and calculates the number of days in the months that have passed until the month of the date.It initializes a variable c to 0 and runs a loop from 0 to the month of the date minus 1. In each iteration, it adds the number of days in the current month to c.It returns an integer value representing the number of days.

3. **numberOfDays(int date1[], int date2[])**: This function takes two arrays of three integers representing two different dates and calculates the number of days between them.It calculates the number of days passed until the first date by multiplying the year of the date by 365, adding the number of days in the months that have passed until the month of the date, adding the day of the date, and adding the number of leap years that have passed until the year of the date.It calculates the number of days passed until the second date in the same way as the first date.It subtracts the number of days passed until the first date from the number of days passed until the second date and stores the result in a member variable a.

4. **input**(): This function prompts the user to enter two dates, calls the numberOfDays function to calculate the number of days between them, and stores the result in the member variable a.

5. **addRoom**(int rno): This function takes an integer value representing the room number and adds a new room to the rooms array with the specified number and other details entered by the user. It creates a new Room object and sets its properties using user input.It returns the new Room object.

6. **searchRoom**(int rno):This function takes an integer value representing the room number and searches for the room in the rooms array.It runs a loop over the rooms array and checks if the room with the specified number exists.If the room is found, it displays its details using the displayRoom function. Otherwise, it displays an error message.

7. **displayRoom**(Room tempRoom):This function takes a Room object and displays its details, including the room number, AC or non-AC type, size, and rent.

8. **checkIn**():This function handles the check-in process for a customer by assigning an available room to them and updating the status of the room to "reserved".It prompts the user to enter the customer details and searches for an available room using the getAvailRoom function.If a room is found, it assigns the room to the customer and updates the room's status to 1.

9. **getAvailRoom**(): This function is used to display the details of all available rooms in the hotel. It iterates through the rooms array and displays the details of each room that is available (i.e. status = 0).

10. **searchCustomer**(char*): This function is used to search for a customer by name. It takes a character array (string) as an argument which is the name of the customer to search for. It iterates through the rooms array and checks if the cust.name matches the search name. If it finds a match, it displays the details of the customer and the room.

11. **checkOut**(int): This function is used to check out a customer from a room. It takes an integer argument which is the room number of the customer to check out. It iterates through the rooms array and checks if the roomNumber matches the given argument. If it finds a match, it sets the status of the room to 0 (i.e. available) and displays the bill amount to be paid by the customer. It also prompts the user to enter the payment amount and calculates the change to be returned.

3. UML DIAGRAMS

3.1 SOFTWARE REQUIREMENTS/ HARDWARE REQUIREMENTS

3.1.1 Software Requirements

Operating System : windows 10

Frontend Languages : C++

3.1.2 Hardware Requirements

Processor : Standard processor with a speed of 1.6 GHz

RAM : 256 MB RAM or more

Hard Disk : 20 GB or more

Monitor : Standard colour monitor

3.2 UML DIAGRAMS

a. Usecase Diagram

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse an actor is a person, organization, or external system that plays a role in one or more interactions with your system.

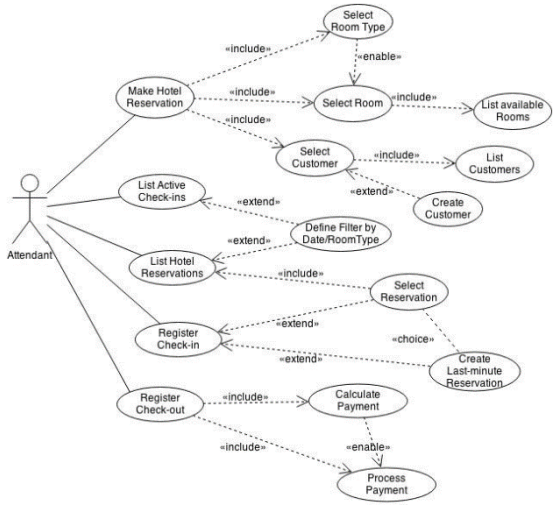


Fig 3.2.1 Usecase Diagram

b. Class Diagram

A class diagram describes the static structure of the symbols in your new system. It is a graphic presentation of the static view that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships. Classes are arranged in hierarchies sharing common structure and behaviour, and are associated with other classes

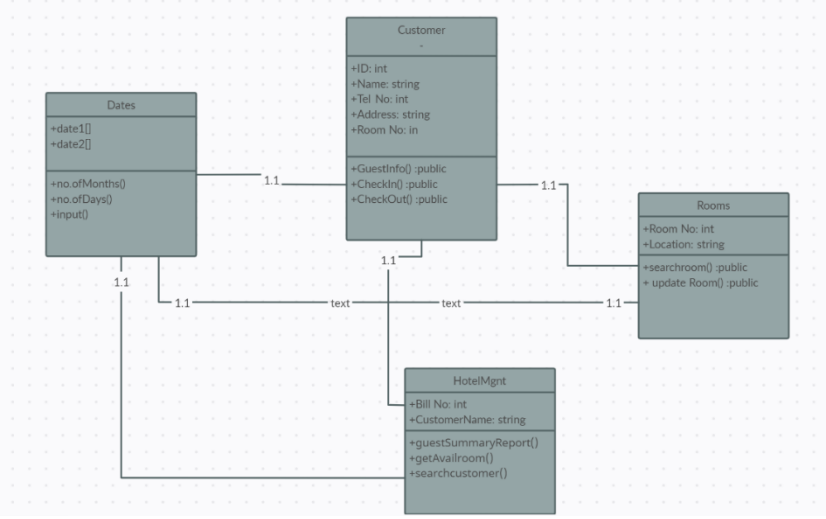


Fig 3.2.2 Class Diagram

BEHAVIOUR DIAGRAM

Interaction Diagram (Sequence Diagram and Collaboration Diagram)

c. Sequence Diagram

A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. Graphically, a sequence diagram is a table that shows objects arranged along x-axis and messages, ordered in increasing time, along the y-axis.

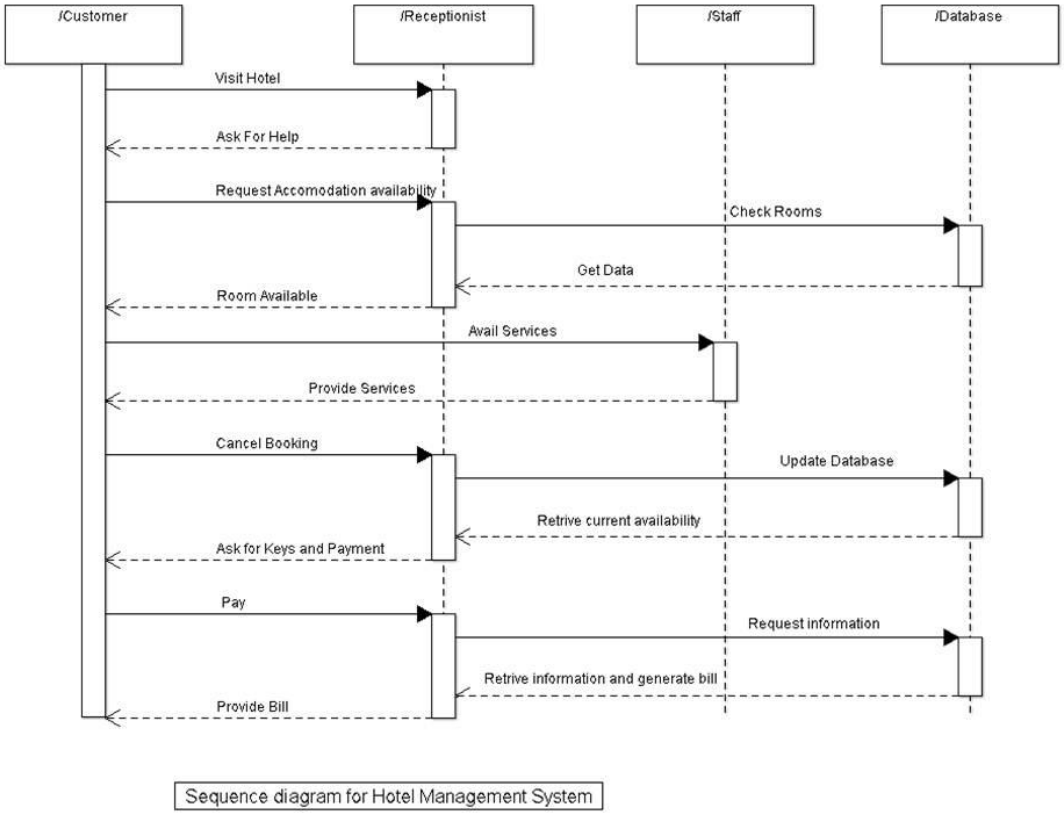


Fig 3.2.3 Sequence Diagram

d. Collaboration Diagram

Collaboration diagrams are used to show the relationships between objects in the system. Both sequence diagrams and collaboration diagrams show the same information, but they are different. Because it is based on object-oriented programming, it does not show message flow, but rather the architecture of the objects that reside in the system. Objects are made up of several properties. Several objects in the system are connected to each other. A collaboration diagram, also called a communication diagram, serves to represent the architecture of the objects in the system.

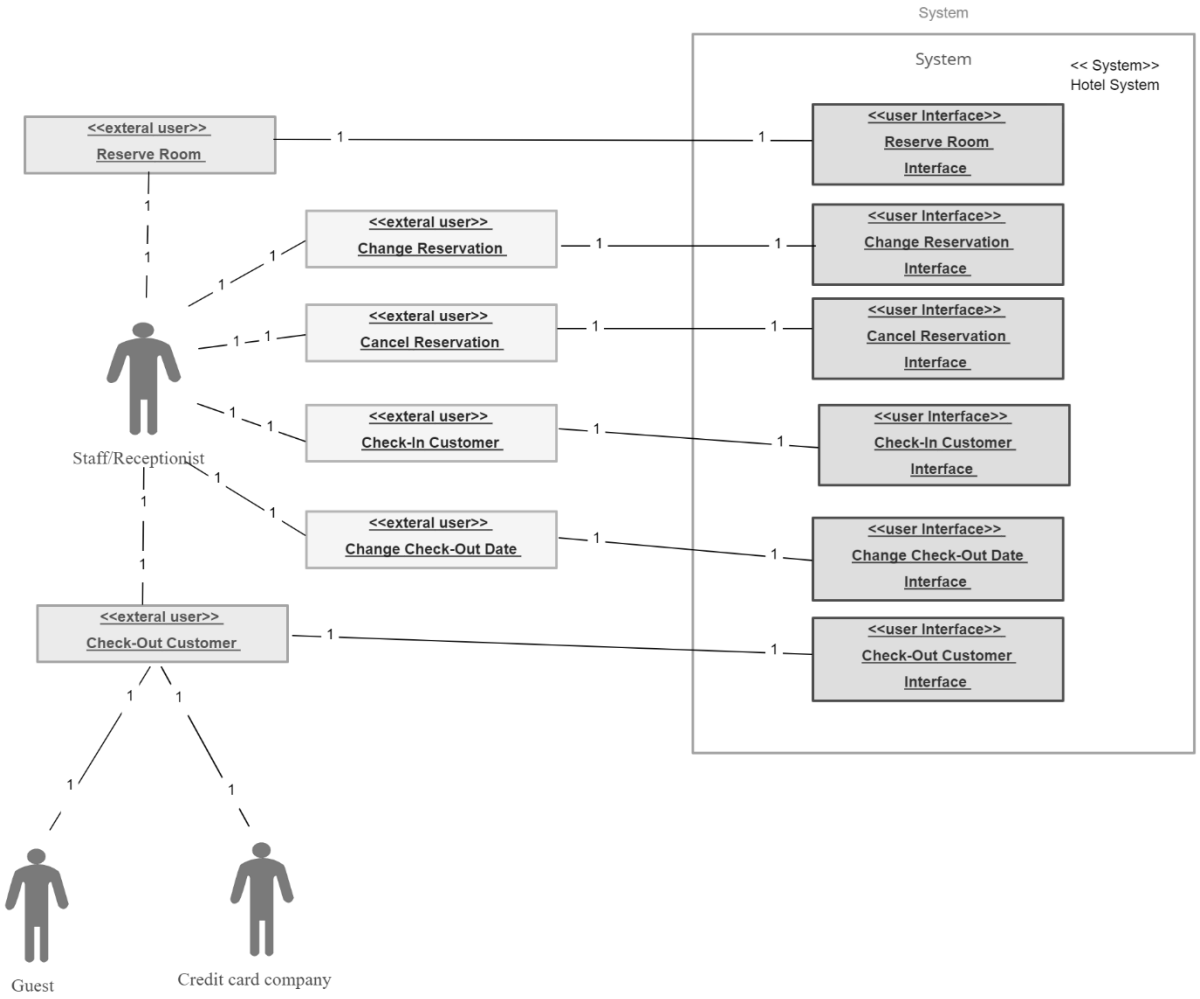


Fig 3.2.4 Collaboration Diagram

e. State Diagram

A state diagram, also known as a state machine diagram or state diagram, is a Unified Modelling Language (UML) representation of the states an object can reach and the transitions between those states. In this context, a state defines a stage of evolution or behaviour of an object. It is a unit of code that represents a particular entity or entities within a program.

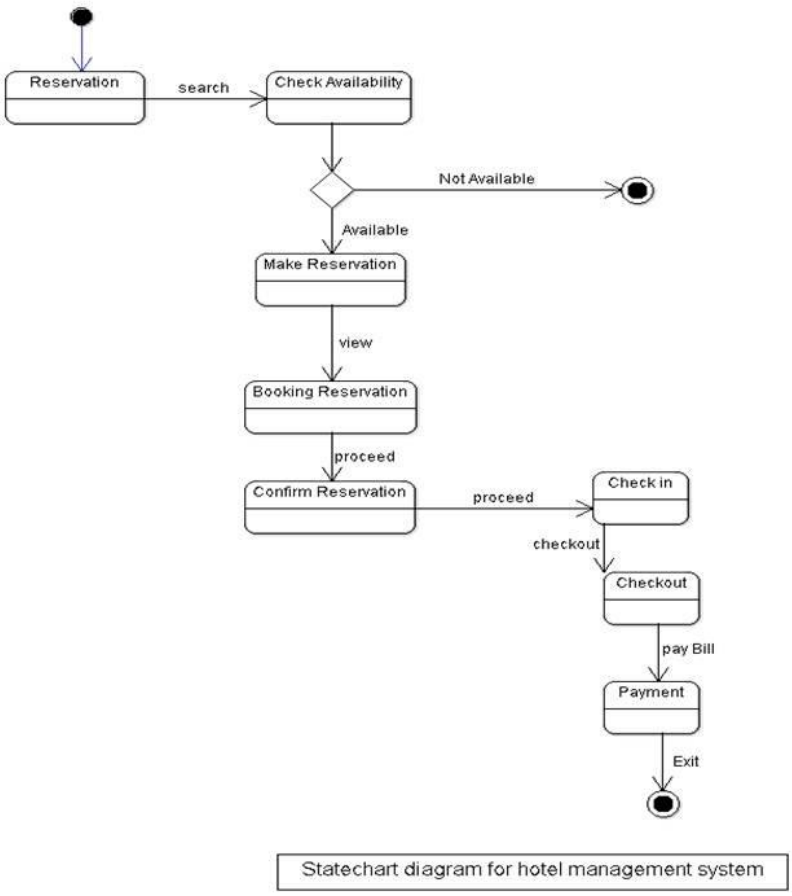


Fig 3.2.5 State Diagram

f. Activity Diagram

An activity diagram shows the flow from activity to activity. An activity is an ongoing non-atomic execution within a state machine. Activities ultimately result in some action, which is made up of executable atomic computations that result in a change in state of the system or the return of a value. Activity diagrams commonly contain Activity states and action states Transitions Objects Like all other diagrams, activity diagrams may contain notes and constrains.

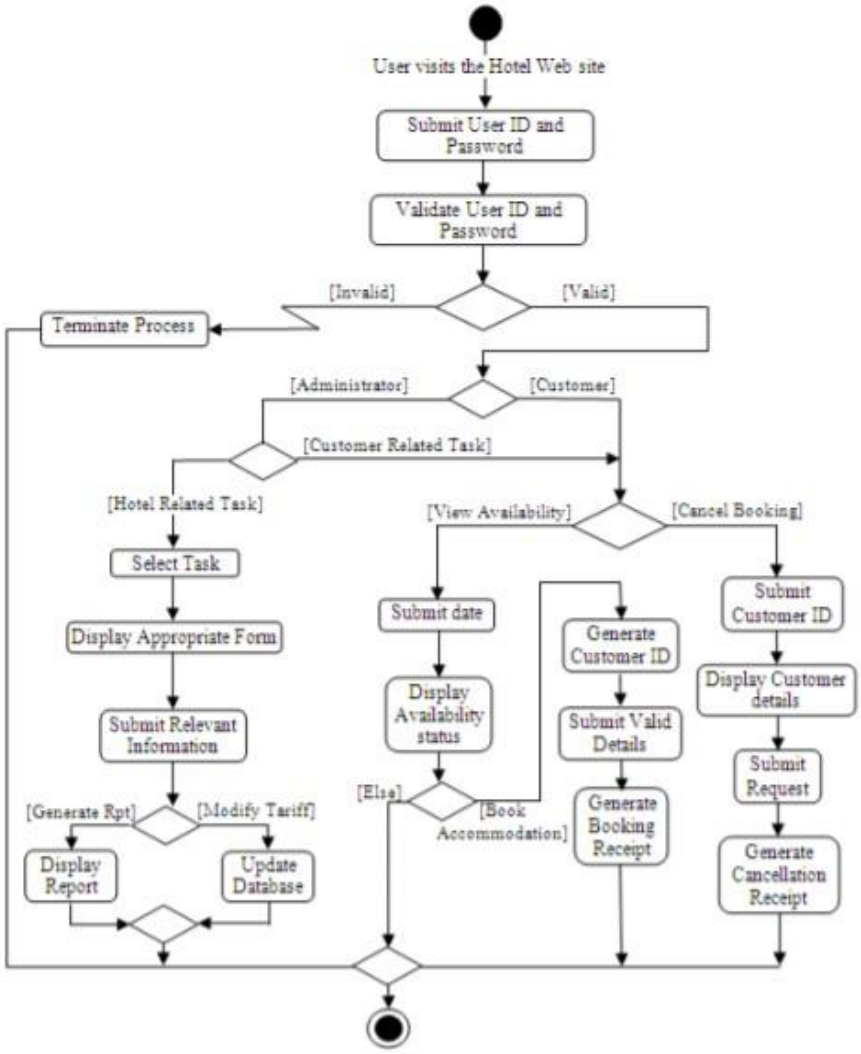


Fig 3.2.6 Activity Diagram

IMPLEMENTATION DIAGRAM

h. Component Diagram

The purpose of a component diagram is to show the relationships between the various components in your system. In UML 2.0, the term "component" refers to a module of a class that represents either an independent system or a subsystem whose function is to cooperate with the rest of the system.

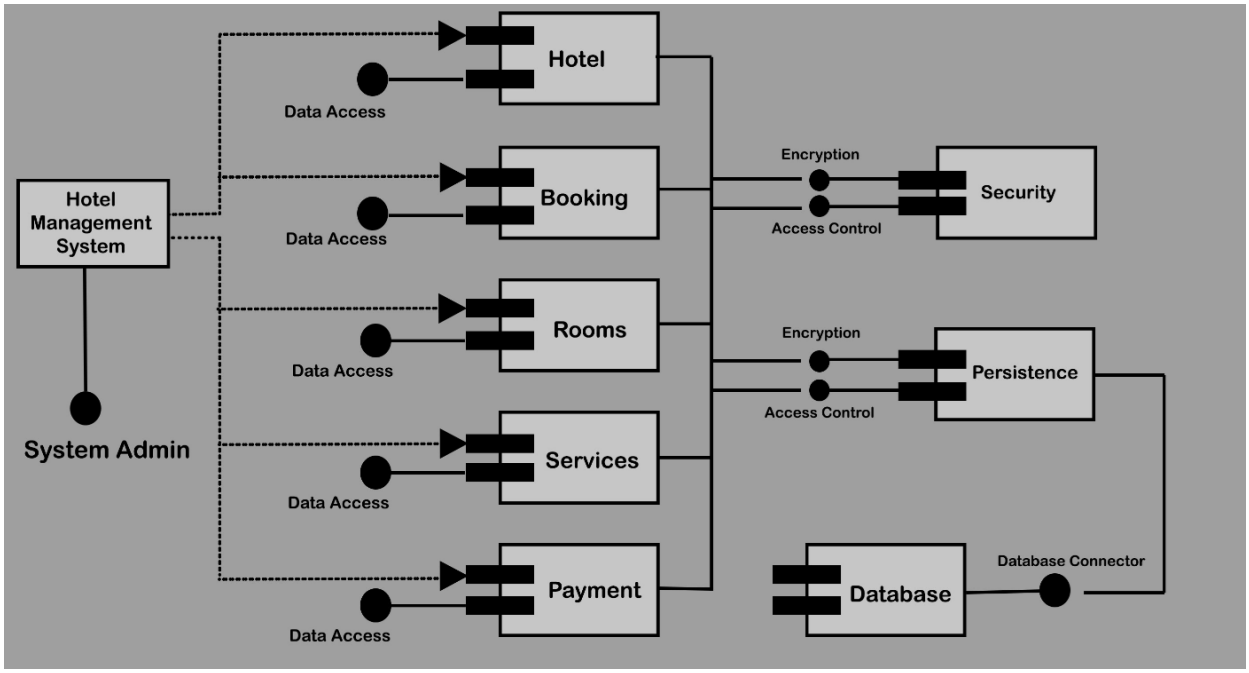


Fig 3.2.7 Component Diagram

i. Deployment Diagram

A deployment diagram is a type of UML diagram that shows the execution architecture of a system, including nodes such as hardware and software execution environments and the middleware that connects them. Deployment diagrams are typically used to visualize the physical hardware and software of a system.

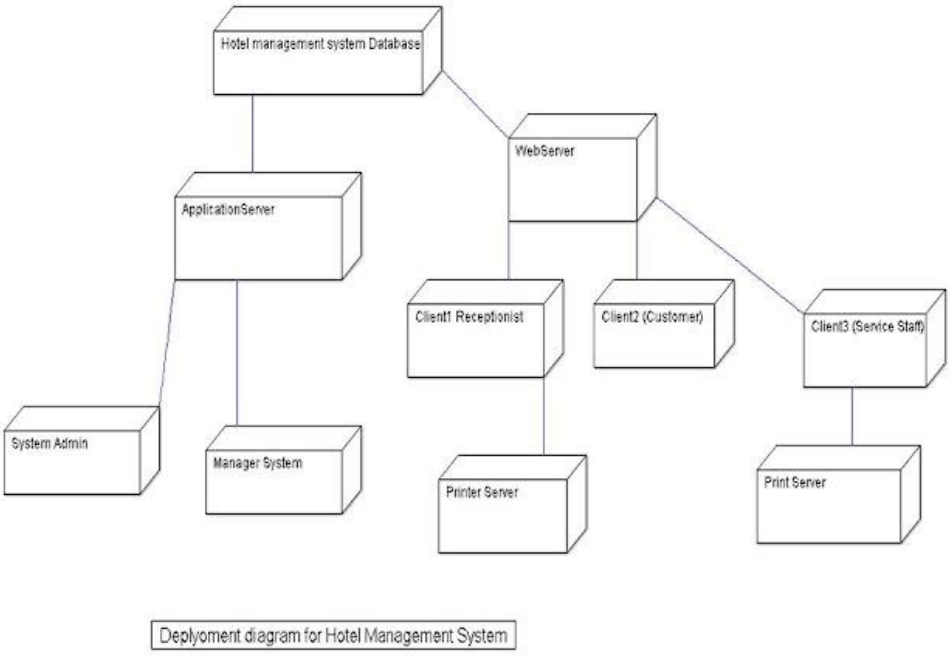


Fig 3.2.8 Deployment Diagram

4. CODE/OUTPUT SCREENSHOTS

4.1 SOURCE CODE

```
#include <iostream>
#include <string.h>
#include <conio.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);
};

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 Size (Delux(D)/Super Delux(S)) : ";
    cin >> room.stype;
    cout << "\nDaily Rent : ";
    cin >> room.rent;
    room.status = 0;

    cout << "\n Room Added Successfully!";
    getch();
    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]);
        getch();
    }
    else
    {
        cout << "\nRoom not found";
        getch();
    }
}

void Room::displayRoom(Room tempRoom)
```

```
{
   cout << "\nRoom Number: \t" << tempRoom.roomNumber;
   cout << "\nType AC/Non-AC (A/N) " << tempRoom.ac;
   cout << "\nType Size (D/SD) " << tempRoom.stype;
   cout << "\nRent: " << tempRoom.rent;
}

// hotel management class
class HotelMgnt : protected Room, public dates
{
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---------------------------------------";
      }
   }

   getch();
}

// 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";
            getch();
            return;
        }

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

        rooms[i].status = 1;

        cout << "\n Customer Checked-in Successfully..";
        getch();
    }
}

// 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;
            getch();
        }
    }
    if (found == 0)
    {
        cout << "\nAll rooms are reserved";
        getch();
    }
}

void HotelMgnt::searchCustomer(char *pname)
{
    int i, found = 0;
    for (i = 0; i < count; i++)
    {
        if (rooms[i].status == 1 && stricmp(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";
            found = 1;
            getch();
        }
    }
    if (found == 0)
    {
        cout << "\nPerson not found.";
        getch();
    }
}

// bill

void HotelMgnt::checkOut(int roomNum)
{
    int i, found = 0, days = a, rno;
    float billAmount = 0;
    for (i = 0; i < count; i++)
    {
```

```
        if (rooms[i].status == 1 && rooms[i].roomNumber == roomNum)
        {
            found = 1;
            // getch();
            break;
        }
    }
    if (found == 1)
    {
        cout << "\n Number of Days:\t";
        cout << 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;
    }
    getch();
}
// managing rooms
void manageRooms()
{
    class Room room;
    int opt, rno, i, flag = 0;
    char ch;
    do
    {
        system("cls");
        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;
               getch();
            }
            else
            {
               rooms[count] = room.addRoom(rno);
               count++;
            }
            break;
         case 2:
            cout << "\nEnter room number: ";
            cin >> rno;
            room.searchRoom(rno);
            break;
         case 3:
            break;
         default:
            cout << "\nPlease Enter correct option";
            break;
         }
   } while (opt != 3);
}
int main()
{
   class HotelMgnt hm;
   int i, j, opt, rno;
   char ch;
   char pname[100];

   system("cls");

   do
   {
```

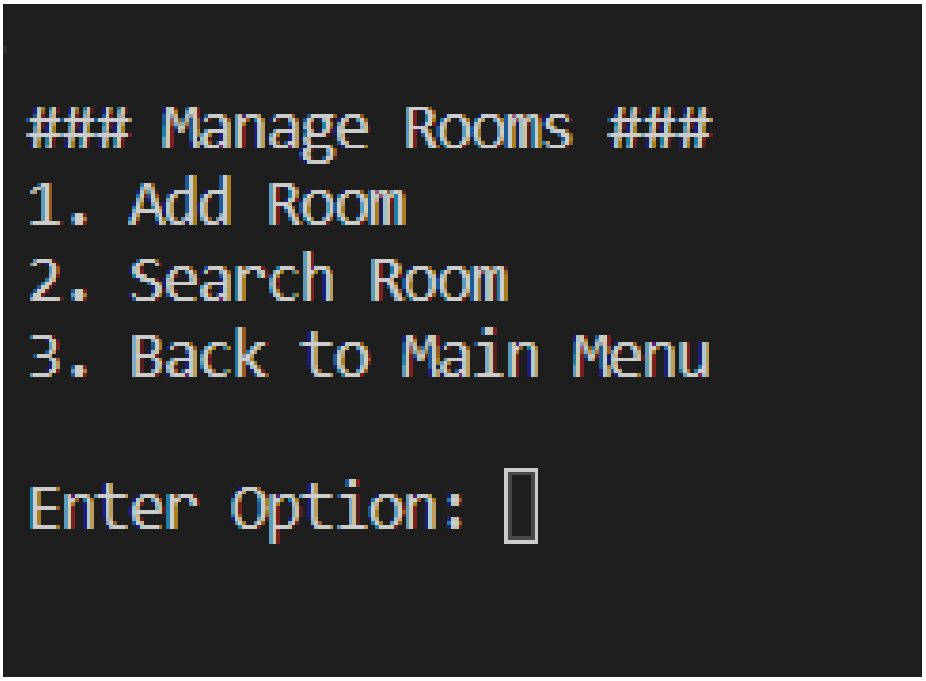
```
system("cls");
cout << "******* VS HOTELS *******\n";
cout << "\n1. Manage Rooms";
cout << "\n2. Check-In Room";
cout << "\n3. Available Rooms";
cout << "\n4. Search Customer";
cout << "\n5. Check-Out Room";
cout << "\n6. Guest Summary Report";
cout << "\n7. Exit";
cout << "\n\nEnter Option: ";
cin >> opt;
switch (opt)
{
case 1:
   manageRooms();
   break;
case 2:
   if (count == 0)
   {
      cout << "\nRooms data is not available.\nPlease add the rooms first.";
      getch();
   }
   else
      hm.checkIn();
   break;
case 3:
   if (count == 0)
   {
      cout << "\nRooms data is not available.\nPlease add the rooms first.";
      getch();
   }
   else
      hm.getAvailRoom();
   break;
case 4:
   if (count == 0)
   {
      cout << "\nRooms are not available.\nPlease add the rooms first.";
      getch();
   }
   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.";
                getch();
            }
            else
            {
                cout << "Enter Room Number : ";
                cin >> rno;
                hm.checkOut(rno);
            }
            break;
        case 6:
            hm.guestSummaryReport();
            break;
        case 7:
            cout << "\nTHANK YOU! FOR USING VS SOFTWARE";
            break;
        default:
            cout << "\nPlease Enter correct option";
            break;
        }
    } while (opt != 7);

    getch();
}
```

4.2 SCREENSHOTS

b.1 Registration Page

b.2 Managing Rooms



b.2 Adding Rooms:

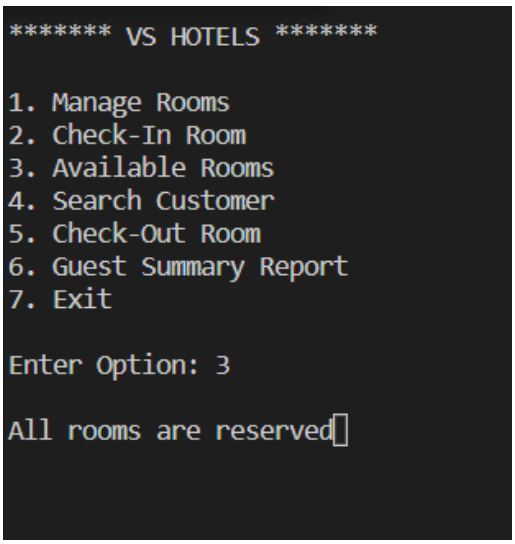
```
### Manage Rooms ###
1. Add Room
2. Search Room
3. Back to Main Menu

Enter Option: 2

Enter room number: 1003
Room Details

Room is available
Room Number:    1003
Type AC/Non-AC (A/N) A
Type Size (D/SD) D
Rent: 5000
```

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

Enter Option: 1

Enter Room Number: 1003

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

Type Size (Delux(D)/Super Delux(S)) : D

Daily Rent : 5000

 Room Added Successfully!
```

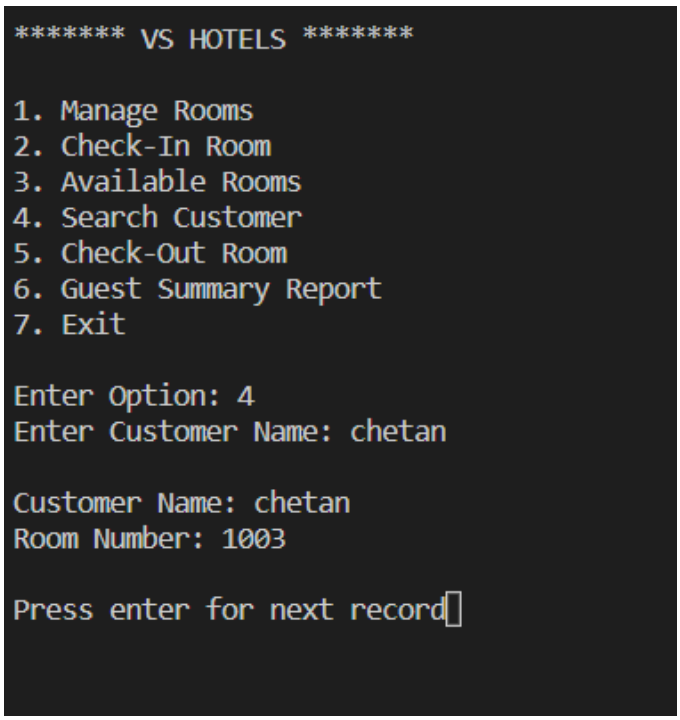
b.3 Check-In:

```
******* VS HOTELS *******

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: 2

Enter Room number : 1003

 booking id: 1234

Enter Customer Name (First Name): chetan

Enter Address (only city): chennai

Enter Phone: 9972074314
Check-in date:
Enter the Date: 01
Enter the month: 05
Enter the year: 2023
Check-out date:
Enter the Date: 27
Enter the month: 05
Enter the year: 2023

Enter Advance Payment: 4000

 Customer Checked-in Successfully..
```

b.4 Available Rooms:



b.5 Search Customers:



b.5 Check-Out:

```
******* VS HOTELS *******

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: 5
Enter Room Number : 1003

 Number of Days:        26
        ######## CheckOut Details ########

Customer Name : chetan
Room Number : 1003
Address : chennai
Phone : 9972074314
Total Amount Due : 130000 /
Advance Paid: 4000 /
*** Total Payable: 126000/ only
```

b.6 Exit:

```
******* VS HOTELS *******

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: 7

THANK YOU! FOR USING VS SOFTWARE
```

5. RESULTS AND CONCLUSION

5.1. RESULTS

The code is defining three classes: dates, Customer, and Room, and one subclass HotelMgnt that inherits from Room and dates.

The dates class contains functions for calculating the number of days between two dates. It prompts the user to input two dates and calculates the number of days between them using the numberOfDays function.

The Customer class defines attributes for a customer, including their name, address, phone number, check-in and check-out dates, payment advance, and booking ID.

The Room class defines attributes for a room, including its type, size, AC/non-AC status, room number, rent, and status (reserved or not). It also contains functions for adding a room, searching for a room, deleting a room, and displaying the details of a room.

The HotelMgnt class inherits from the Room and dates classes and defines functions for checking in a guest, getting available rooms, searching for a customer, checking out a customer, and generating a guest summary report.

5.1.1 Results Comparison

Current manual systems use paperwork and direct human verbal communication by word of mouth

run a hotel. This delays communication within the hotel.

Reservations can be made by phone or at the hotel reservation office. The guests Personal data such as name, age, nationality and length of stay will be entered during the booking process

The reservation office will order the room to be ready prior to the check-in date. Documents are manually sent to the entry department for editing guest documents File. On the important day, the file will be sent to the reception. When guests check in It also indicates if you want room service with your assigned room key.

The receptionist gives the guest file to the accountant at the next table. guest here You pay for your room and meals. Guest files are updated daily based on your costs cost. Accounting department prepares daily invoices and sends them to guests

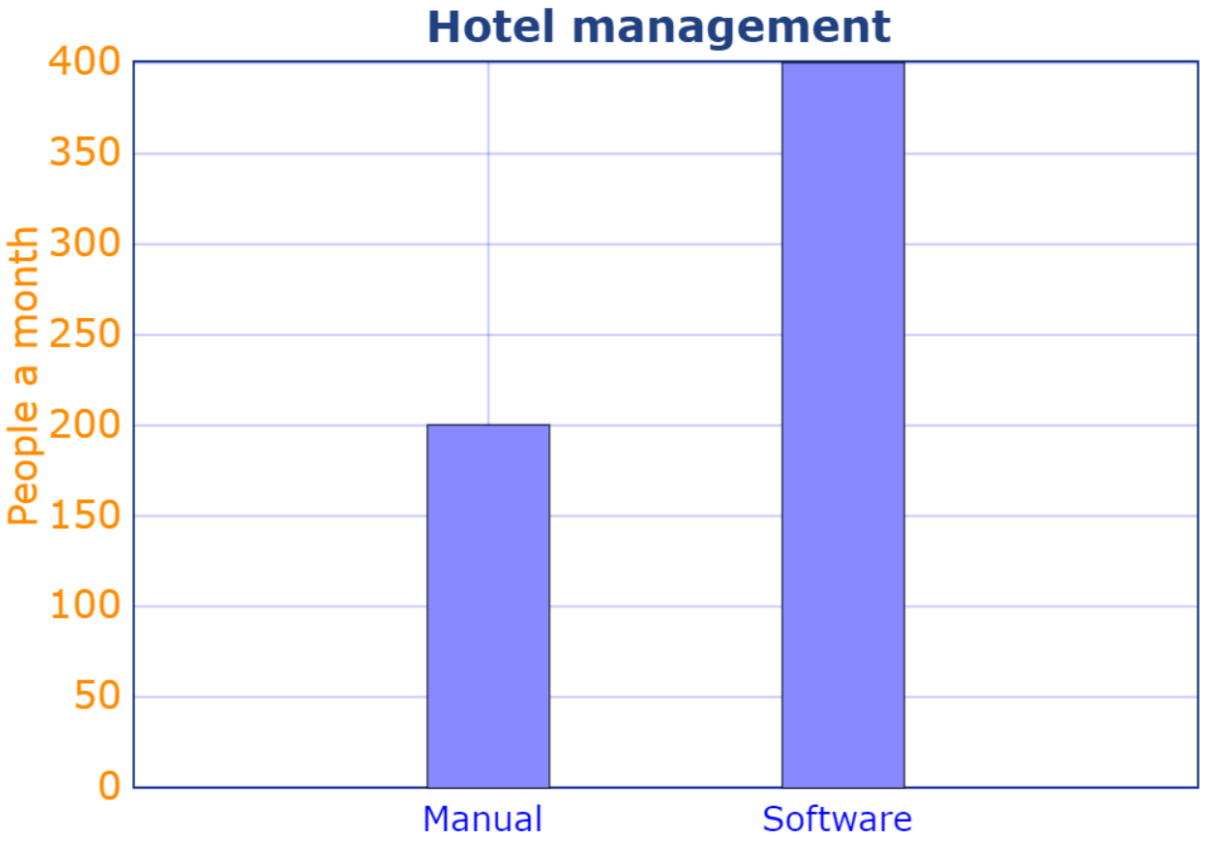
Their room at dusk by the maid. Guest pays at cash register with receipt Accountant before departure.

When a guest checks out, an expense report is created one day before her checkout date. Upon check-out, guests will receive a statement at the cash register.

Invoice balance, if any.

DISADVANTAGES OF MANUAL SYSTEM:

1. Guest files can easily get lost or mix up with other guest file documents.
2. Files occupy a large storage space .
3. Unnecessary duplication of data.
4. Files are prone to theft unauthorized modification due to low data security levels and standards.
5. Due to easy access to guest data by unauthorized users, guest data is extremely unconfident.
6. Retrieval of guest records is extremely difficult.
7. Data entry procedure is prone to errors. Guest records are extremely difficult to modify since modification generates dirty and unpresentable reports.



5.1.2 Application

This is a phase in which the system analyst did an evaluation of the changeover method that should be used to switch from present manual system to the developed computerized system. After a close analysis the analyst came up with parallel changeover method as the most appropriate for the system. Parallel method is whereby the computerized system will run concurrently with the manual system before discarding the manual system. Although expensive the changeover method will prove to be the most efficient because:

- Parallel changeover provides time for one the database administrator to update all the guest files before a total changeover to the new system.
- It's possible to troubleshoot any errors arising from loading process without affecting the hotel's transactions as the manual system will still be in place to carry out the hotel activities smoothly.
- Provides time for employees to learn and adapt to the new system.
- Lowers the risk to the management in case of a technical hitch or breakdown as the manual system will still be in place as the analyst fixes the technical hitch.

5.2 CONCLUSION

Computers have clear advantages over manual systems. Computerized systems are more reliable, efficient and faster. At the end of the project, we can say that computers play a very important role in the development of enterprises. All daily reports generated by the system must be reviewed by an officer to ensure that all transactions have been made on the proper accounts and settled against new vouchers. Computers do the most work in the least amount of time. It is used everywhere and offers comfort and fit for everyone. The company's main goal is to provide its guests with maximum facilities and comfort. To achieve this goal, we need to provide other modern computing facilities.

5.2.1 FUTURE ENHANCEMENT

There are several potential future enhancements that could be made to a hotel management system, including:

1. Integration with smart devices: As smart devices become more prevalent, it may be useful for a hotel management system to integrate with these devices. For example, guests could use their smartphones to check in and access their room, control the lights and temperature, or order room service.
2. Personalization: As hotels try to differentiate themselves from their competitors, personalization will become increasingly important. A hotel management system could collect data about guests' preferences and use that data to tailor their experience. For example, the system could suggest activities or restaurants based on a guest's past behavior or offer a customized welcome message when they arrive.
3. Artificial intelligence: AI could be used to optimize various aspects of the hotel management system, from inventory management to pricing to customer service. AI could also be used to predict demand and adjust pricing and inventory accordingly.
4. Mobile app: A mobile app could be developed that would allow guests to easily manage their reservations, check in and out, and access hotel amenities. The app could also provide personalized recommendations and promotions.
5. Sustainability features: As consumers become more environmentally conscious, hotels may want to incorporate sustainability features into their management system. For example, the system could track energy usage and suggest ways to reduce it, or encourage guests to reuse towels and linens.
6. Virtual reality: Virtual reality technology could be used to create virtual tours of hotel rooms and amenities, allowing potential guests to experience the hotel before they book. This could also be used to showcase special events or promotions.

6. REFERENCES

http://www.w3schools.com/html/html_intro.asp.

https://www.w3schools.com/php/default.asp.

https://www.w3schools.com/sql/default.asp.

Fundamentals of software engineering by Rajib mall, PHIlearning.

Web development and application development by Ivan Byross BPB publications.