Visvesvaraya Technological University
Belagavi-590 018, Karnataka

A Mini Project Report on

"Resort Database Management"

Mini Project Report submitted in partial fulfilment of the requirement for the DBMS Laboratory with Mini Project [18CSL58]

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by
Karthik Y [1JT18CS027]
Prajwal A R [1JT18CS043]

Department of Computer Science and Engineering
Jyothy Institute of Technology
Tataguni, Bengaluru-560082

Jyothy Institute of Technology
Tataguni, Bengaluru-560082
Department of Computer Science and Engineering

CERTIFICATE

Certified that the mini project work entitled **“Resort Database Management”** carried out by **Karthik Y [1JT18CS027] and Prajwal A R [1JT18CS043],** bonafide students of Jyothy Institute of Technology, in partial fulfilment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** department of the **Visvesvaraya Technological University, Belagavi** during academic the year **2019-2020**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Mrs. Nikitha S	**Dr. Prabhanjan S**
Guide, Asst. Professor	Professor & HOD
Dept. of CSE	Dept. of CSE

External Viva Examiner   Signature with Date :

1.
2.

ACKNOWLEDGEMENT

Firstly, we are very grateful to this esteemed institution **“Jyothy Institute of Technology**” for providing us an opportunity to complete our project.

We express our sincere thanks to our **Principal Dr. Gopalakrishna K** for providing us with adequate facilities to undertake this project.

We would like to thank **Dr. Prabhanjan S, Professor and Head of Computer Science** and Engineering Department for providing for his valuable support.

We would like to thank our guide **Mrs.Nikitha S, Assistant Professor** for her keen interest and guidance in preparing this work.

Finally, we would thank all our friends who have helped us directly or indirectly in this project.

Karthik Y [1JT18CS027]
Prajwal A R [1JT18CS043]

ABSTRACT

The main objective of this Resort Management System is to develop and implement a data management system for resort to help the management in storing information about the guests staying in rooms, Food orders, Staff details and their salary information which must be necessarily taken care of. The Proposed System has a user-friendly environment and has all the details about the business under study like the room and food services and maintenance. This System will make the Authority people job easier and faster rather than having these details stored manually. The Proposed system is Secure and only the Intended User can login and use the service. For this Application SQL is used as backend system which is a database containing all the necessary information. For the User Interface Java Programming is used to provide an interactive environment.

Table of Contents

CHAPTER 1 INTRODUCTION

1. INTRODUCTION

1.1 Introduction to DBMS

A database is simply an organized collection of related data, typically stored on disk, and accessible by possibly many concurrent users. Databases are generally separated into application areas. For example, one database may contain Human Resource (employee and payroll) data; another may contain sales data; another may contain accounting data; and so on. Databases are managed by a DBMS.
The choice of a database product is often influenced by factors such as:

- the computing platform (i.e., hardware, operating system)
- the volume of data to be managed
- the number of transactions required per second
- existing applications or interfaces that an organization may have
- support for heterogeneous and/or distributed computing
- cost
- vendor support

1.2 Introduction to SQL

SQL (pronounced as seqeul), which is an abbreviation for Structured Query Language, is a language to request data from a database, to add, update, or remove data within a database, or to manipulate the metadata of the database.

SQL is a declarative language in which the expected result or operation is given without the specific details about how to accomplish the task. The steps required to execute SQL statements are handled transparently by the SQL database. Sometimes SQL is characterized as non-procedural because procedural languages generally require the details of the operations to be specified, such as opening and closing tables, loading and searching indexes, or flushing buffers and writing data to file systems.
Therefore, SQL is considered to be designed at a higher conceptual level of operation than procedural languages because the lower level logical and physical operations aren't specified and are determined by the SQL engine or server process that executes it.

1.3 Introduction to Resort Database Management

The world in the 21st century is growing up in the technology in every field such as education, medicine, transport etc. The use of technology makes the world so faster and easier than the early world and it releases the world from manual usage in every field.

In the early days, the manual usage causes many mistakes by the user and administrative. Using manual properties in the fields was not comfortable for the consumers because it was slower than technical usages, caused wastages of the consumers' time and contained many formalities in usage.

This Project of using technology in Resorts helps the management in storing the details of the Guests staying in rooms, food ordered by them and details of the Staff serving the Guests. A computerized application is created for the above fields. The Application is developed based on the requirements and to make the work easier.

1.4 Scope and importance of work

The scope of the project is clear to give a simple and attractive application to simplify the work as well as to reduce the efforts while doing it offline or so called traditional methods.

In this Application we are able to store the complete business details of the Resort, Information about the Guests, Food Orders, Staff are Stored.

The Resort Database includes four entities namely room, guest, food, and staff. The room table has 3 fields - room number, type of room (AC/ NON AC) and price of the room. The guest table has fields like guest ID, name, age, phone number, address, and dates of arrival and departure. Food table contains food code, type, name and price. The staff table has details such as name, id, age, phone number and salary.

The database also contains three relations namely stays in, orders and service. The stays in records data of guests' room bookings like the room number and the number of days that the room is booked for. The orders relation is used to track the food orders made by guests. It includes data like order number, food item ordered, number of plates ordered, breakfast/lunch/dinner tag, and the staff member servicing the order to the guest. Finally, the service relation keeps track of which staff member has provided what service to which room. It has data like service number, the staff id, and the name of service offered.

CHAPTER 2 DESIGN

Theory of ER Diagram

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. There are three basic components of an entity relationship diagram:

Entities

The basic object that the ER model represents is an entity, which is a thing in the real world with an independent existence. An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course). They are usually represented by a rectangle.

Attribute

Each entity has attributes-- the particular properties that describe it. A particular entity will have a value for each of its attributes. The attribute values that describe each entity become a major part of the data stored in the database.

Relationships

They are relationships between and among those entities.

The three main cardinalities are:

1. A one-to-one relationship (1:1). For example, if each customer in a database is associated with one mailing address.
2. A one-to-many relationship (1:M). For example, a single customer might place an order for multiple products. The customer is associated with multiple entities, but all those entities have a single connection back to the same customer.
3. A many-to-many relationship (M:N). For example, at a company where all call center agents work with multiple customers, each agent is associated with multiple customers, and multiple customers might also be associated with multiple agents.

ER Diagram

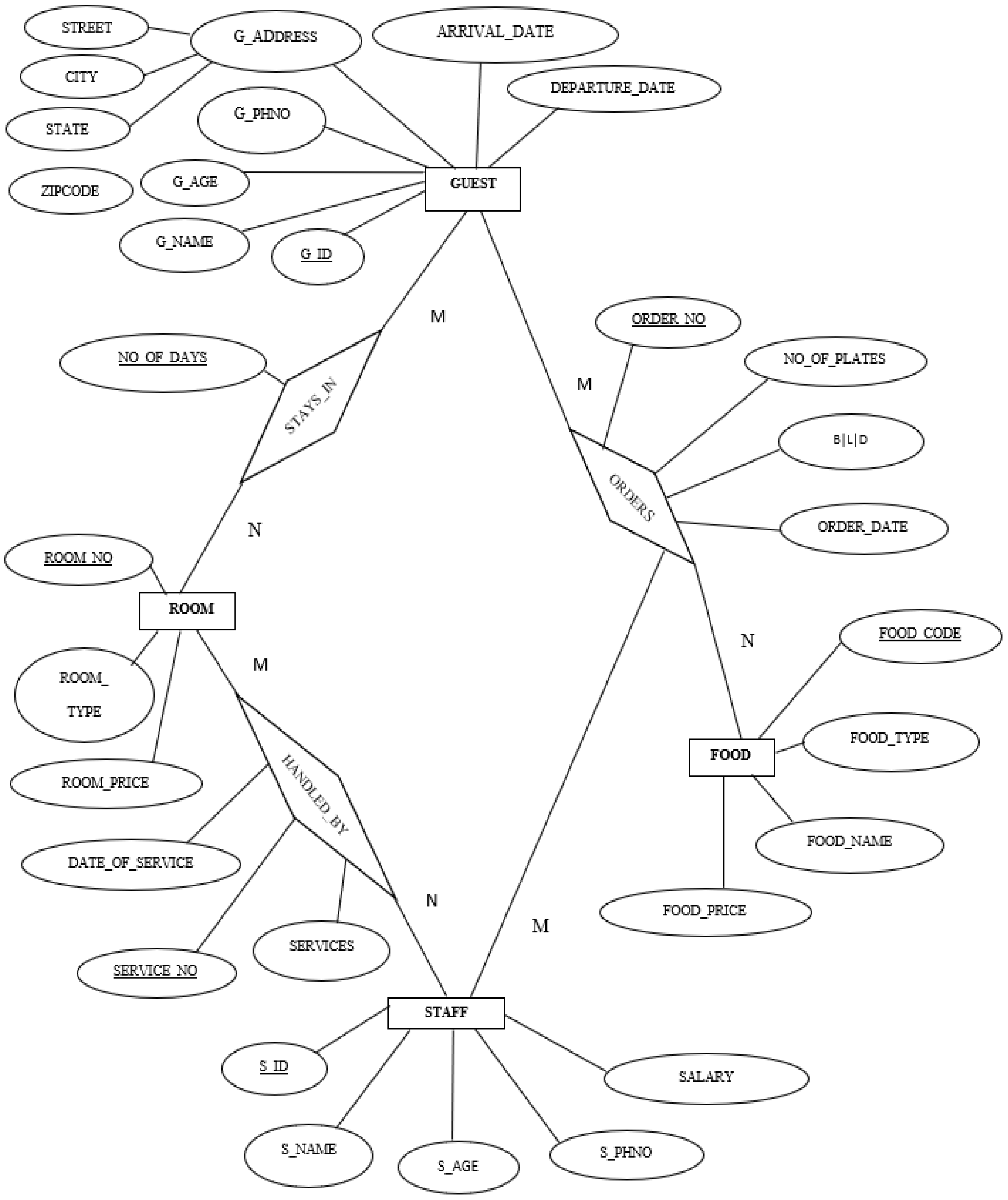


Figure 2.1 : ER Diagram

Schema Diagram

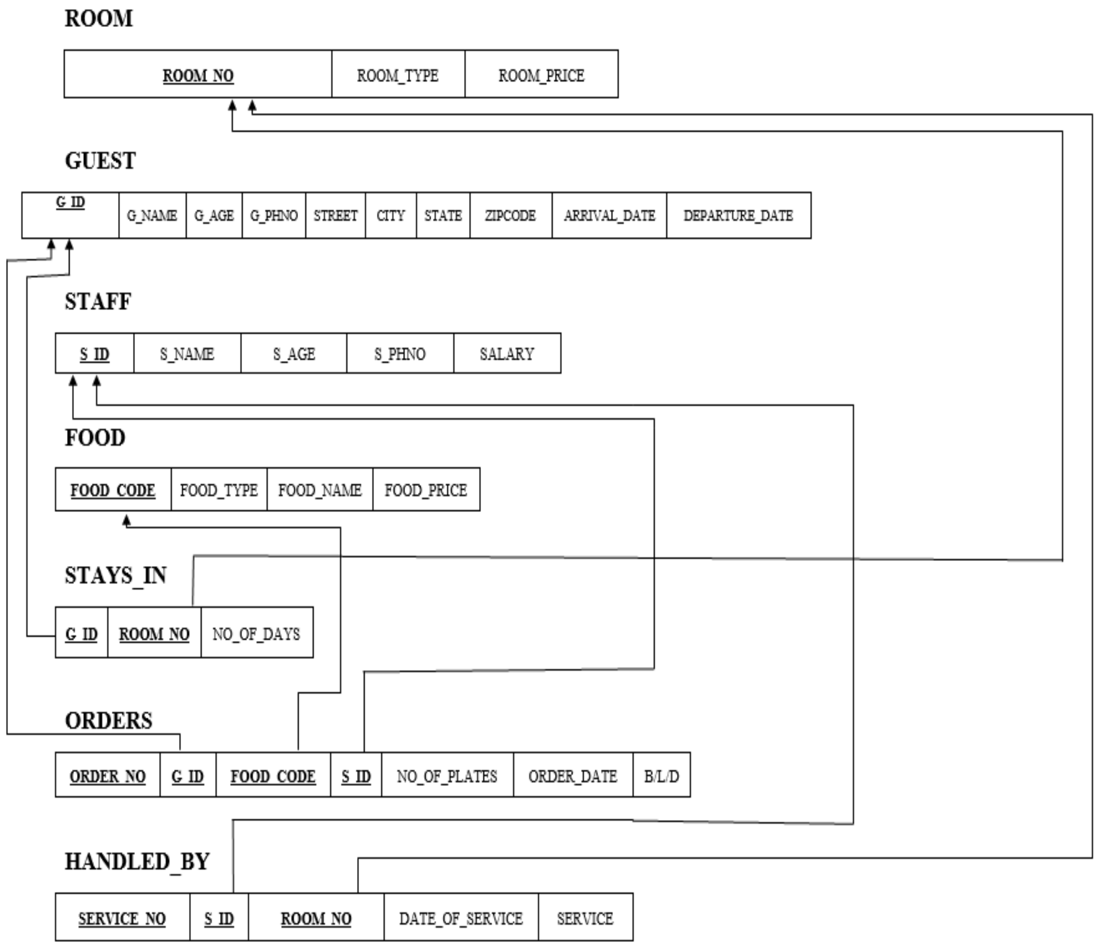


Figure 2.2: Schema Diagram

List of Tables

1. ROOM: To maintain details of the rooms, like their price, type(AC/NON AC).
2. GUEST: Details of the guests.
3. STAFF: Details of the staff.
4. FOOD: Food items menu, their price, type(Indian/Mexican).
5. STAYS_IN: To maintain data on which guest is staying in which room.
6. ORDERS: Order details of each guests.
7. HANDLED_BY: Services provided to rooms.

CHAPTER 3
IMPLEMENTATION

Create table commands:

```sql
create table room(room_no int auto_increment, room_type varchar(20),room_price
decimal(10,2), primary key(room_no));
alter table room auto_increment=100;

create table guest(g_id int auto_increment,g_name varchar(30),g_age int,g_phno varchar(10),
street varchar(20), city varchar(20), state varchar(20), zipcode varchar(10),arrival_date
datetime,departure_date datetime ,primary key(g_id));

create table staff(s_id int auto_increment, s_name varchar(30),s_age int,s_phno varchar(10),
salary decimal(10,2), primary key(s_id));
alter table staff auto_increment=1000;

create table food(food_code int auto_increment, food_name varchar(30),food_type
varchar(30),food_price decimal(10,2), primary key(food_code));
alter table food auto_increment=50;

create table stays_in(g_id int,room_no int, no_of_days int, primary key(g_id,room_no),
foreign key(g_id) references guest(g_id) on delete cascade, foreign key(room_no) references
room(room_no) on delete cascade on update cascade);

create table orders(order_no int auto_increment,g_id int,food_code int,no_of_plates int,s_id
int, order_date datetime,bld char(1), primary key(order_no,g_id,food_code,s_id), foreign
key(g_id) references guest(g_id) on delete cascade, foreign key(food_code) references
food(food_code) on delete cascade on update cascade, foreign key(s_id) references staff(s_id)
on delete cascade on update cascade);

create table handled_by(service_no int auto_increment,s_id int,room_no int,service_date
datetime, service varchar(30),primary key(service_no,s_id,room_no), foreign key(s_id)
references staff(s_id) on delete cascade on update cascade, foreign key(room_no) references
room(room_no) on delete cascade on update cascade);
```

Insertion tables values

Insertion of room table

```
insert into room values(‘AC’,2000);
insert into room values(‘NON-AC’,1000);
insert into room values(‘NON-AC’,1000);
insert into room values(‘AC’,2000);
insert into room values(‘AC’,2000);
insert into room values(‘NON-AC’,1000);
insert into room values(‘AC’;3000);
insert into room values(‘NON-AC’,2000);
insert into room values(‘AC’,1000),
insert into room values(‘AC’,2000);
```

Insertion of guests table

```
insert into guest values
(‘Dev’,19,383717,’Kslayout’,’Bangalore’,’Karnataka’,’560078’,’2020-11-17
14:00:00’,’2020-11-19 14:00:00’);
insert into guest values
(‘Finch’,34,383718,’Shrirangapatana’,’Mysore’,’Karnataka’,’530089’,’2020-11-19
09:15:00’,’2020-11-20 09:15:00’);
insert into guest values
(‘Virat’,32,383719,’Uttarahalli’,’Bangalore’,’Karnataka’,’560078’,’2020-10-18
10:30:00’,’2020-10-20 10:30:00’);
insert into guest values
(‘AB Devilliers’,36,383720,’Jubliehills’,’Hyderabad’,’Andhrapradesh’,’606817’,’2020-11-
18 15:00:00’,’2020-11-19 15:00:00’);
insert into guest
values(‘Parthiv’,35,383721,’Jayanagar’,’Bangalore’,’Karnataka’,’560033’,’2020-11-18
08:45:00’,’2020-11-21 08:45:00’);
insert into guest values
```

(‘Moenali’,37,383722,’Kanchi’,’Chennai’,’Tamilnadu’,’376550’,’2020-11-19 20:00:00’,’2020-11-24 20:00:00’);
insert into guest values
(‘Shivam’,29,383723,’Nagamangala’,’Mandya’,’Karnataka’,’514915’,’2020-11-16 13:15:00’,’2020-11-20 13:15:00’);
insert into guest values
(‘Morris’,29,383724,’Shabarimala’,’Kottayam’,’Kerala’,’707188’,’2020-11-18 10:00:00’,’2020-11-22 10:00:00’);

Insertion of staff table

insert into staff values(‘Rohit’,30,328836,15000.0);
insert into staff values(‘Pollard’,25,338332,20000.0);
insert into staff values(‘Hardik’,40,377898,25000.0);
insert into staff values(‘Malinga’,20,325898,30000.0);
insert into staff values(‘Krunal’,45,397728,45000.0);
insert into staff values(‘Bumrah’,30,367328,20000.0);
insert into staff values(‘surya’,25,344849,15000.0);
insert into staff values(‘Deepak’,30,367328,20000.0);

Insertion of food table

insert into food values(‘American’,’Burger’,50);
insert into food values(‘American’,’Fries’,120);
insert into food values(‘Chinese’,’Noodles’,75);
insert into food values(‘Chinese’,’Gobi manchurian’,300);
insert into food values(‘Indian’,’Idly’,40);
insert into food values(‘Indian’,’Dosa’,60);
insert into food values(‘Indian’,’South Indian meals’,250);
insert into food values(‘Indian’,’Pongal’,120);
insert into food values(‘Indian’,’Pongal’,70);
insert into food values(‘Italian’,’Spaghettie’,100);
insert into food values(‘Italian’,’Pizza’,150);
insert into food values(‘Mexican’,’Tacos’,80);

Insertion of stays_in table

insert into stays_in values(1,100,2);

insert into stays_in values(2,101,1);

insert into stays_in values(3,102,2);

insert into stays_in values(3,108,2);

insert into stays_in values(4,103,1);

insert into stays_in values(5,104,3);

insert into stays_in values(6,105,5);

insert into stays_in values(7,106,4);

insert into stays_in values(8,107,4);

Insertion of orders table

insert into orders values(1,56,1,1003,’2020-11-17 14:15:00’,’l’);

insert into orders values(1,53,1,1006,’2020-11-17 21:30:01’,’d’);

insert into orders values(1,54,1,1000,’2020-11-18 09:15:00’,’b’);

insert into orders values(1,56,1,1003,’2020-11-18 14:15:00’,’l’);

insert into orders values(1,52,1,1006,’2020-11-18 21:30:01’,’d’);

insert into orders values(3,60,5,1006,’2020-11-18 22:50:06’,’l’);

insert into orders values(3,56,3,1002,’2020-11-18 14:15:45’,’d’);

insert into orders values(4,60,1,1004,’2020-11-18 09:00:00’,’l’);

insert into orders values(4,61,1,1004,’2020-11-18 09:00:00’,’d’);

insert into orders values(5,53,1,1005,’2020-11-18 08:45:00’,’b’);

insert into orders values(5,56,2,1005,’2020-11-18 14:45:00’,’l’);

insert into orders values(5,52,1,1005,’2020-11-18 21:45:00’,’d’);

insert into orders values(7,54,3,1007,’2020-11-18 15:50:56’,’l’);

insert into orders values(7,58,2,1000,’2020-11-18 20:50:56’,’d’);

insert into orders values(8,60,2,1001,’2020-11-18 11:50:44’,’b’);

insert into orders values(8,56,2,1007,’2020-11-18 15:45:05’,’l’);

insert into orders values(8,54,2,1006,’2020-11-18 21:00:08’,’d’);

insert into orders values(1,50,1,1000,’2020-11-19 09:15:00’,’b’);

insert into orders values(2,55,2,1001,’2020-11-19 10:10:00’,’b’);

insert into orders values(2,52,2,1001,’2020-11-19 20:50:06’,’d’);

```
insert into orders values(3,56,2,1002,’2020-11-19 14:15:45’,’l’);
insert into orders values(3,57,3,1002,’2020-11-19 23:15:45’,’d’);
insert into orders values(4,50,1,1004,’2020-11-19 09:00:00’,’b’);
insert into orders values(5,51,1,1005,’2020-11-19 08:45:00’,’b’);
insert into orders values(5,60,1,1005,’2020-11-19 14:45:00’,’l’);
insert into orders values(5,61,2,1005,’2020-11-19 21:45:00’,’d’);
insert into orders values(6,59,1,1006,’2020-11-19 13:50:06’,’b’);
insert into orders values(6,58,1,1006,’2020-11-19 13:50:06’,’l’);
insert into orders values(6,56,2,1006,’2020-11-19 20:50:06’,’d’);
insert into orders values(7,57,1,1002,’2020-11-19 09:50:56’,’b’);
insert into orders values(7,58,1,1003,’2020-11-19 15:50:56’,’l’);
insert into orders values(8,53,2,1001,’2020-11-19 11:50:00’,’b’);
insert into orders values(8,56,2,1002,’2020-11-19 15:45:15’,’l’);
insert into orders values(8,52,2,1002,’2020-11-19 21:00:00’,’d’);
```

Insertion of handled_by table

```
insert into handled_by values(1000,100,’2020-11-19 11:00:00’,’cleaning’);
insert into handled_by values(1001,101,’2020-11-19 11:00:00’,’cleaning’);
insert into handled_by values(1002,102,’2020-11-19 10:30:00’,’cleaning’);
insert into handled_by values(1003,102,’2020-11-19 14:45:00’,’amenity’);
insert into handled_by values(1003,103,’2020-11-19 11:00:00’,’cleaning’);
insert into handled_by values(1004,104,’2020-11-19 11:00:00’,’cleaning’);
insert into handled_by values(1004,104,’2020-11-19 18:00:00’,’toiletries’);
insert into handled_by values(1005,105,’2020-11-19 18:00:00’,’cleaning’);
insert into handled_by values(1006,106,’2020-11-19 20:00:00’,’cleaning’);
insert into handled_by values(1007,107,’2020-11-19 15:00:00’,’cleaning’);
insert into handled_by values(1004,107,’2020-11-15 21:00:00’,’toiletries’);
insert into handled_by values(1001,108,’2020-11-19 14:00:00’,’maintenance’);
insert into handled_by values(1002,108,’2020-11-19 16:00:00’,’cleaning’);
```

GUI implementation (sample)

CheckAvailability.java

Code:

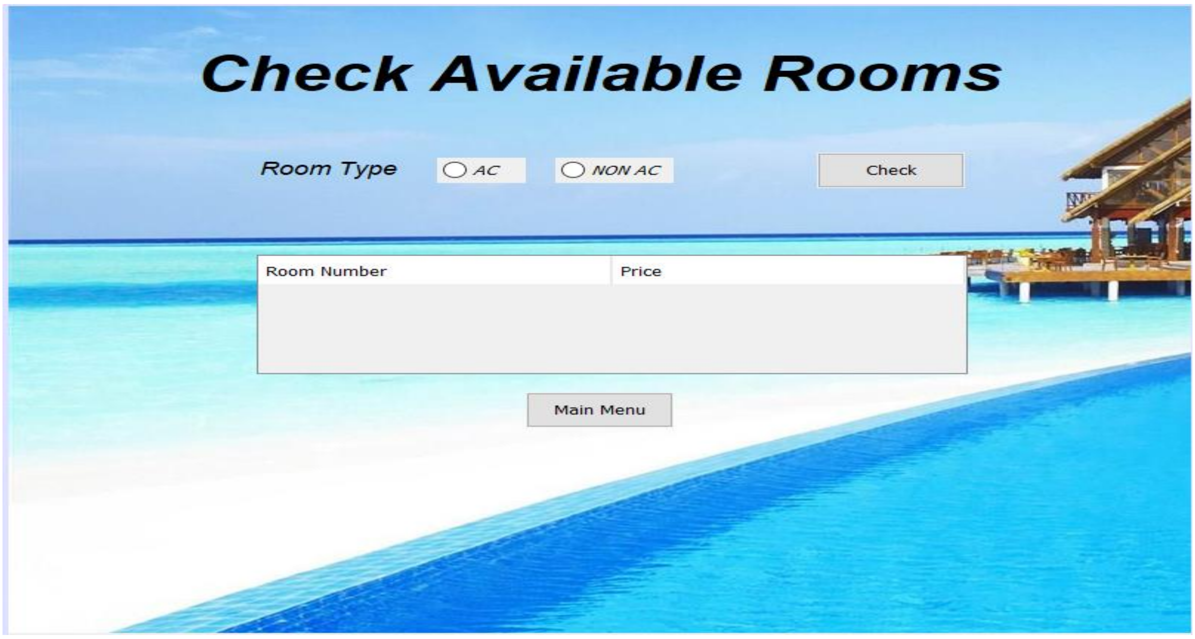
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package guestops;
import misc.MainMenu;
import java.sql.*;
import javax.swing.JoptionPane;
import javax.swing.table.DefaultTableModel;
/**
 *
 * @author Anantha
 */
public class CheckAvailability extends javax.swing.Jframe {

    /**
     * Creates new form checkavail
     */
    public CheckAvailability() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
```

```
// <editor-fold defaultstate=”collapsed” desc=”Generated Code”>
private void initComponents() {

    buttonGroup1 = new javax.swing.ButtonGroup();
    jLabel1 = new javax.swing.Jlabel();
    jLabel2 = new javax.swing.Jlabel();
    jRadioButton1 = new javax.swing.JradioButton();
    jRadioButton2 = new javax.swing.JradioButton();
    jButton1 = new javax.swing.Jbutton();
    jScrollPane1 = new javax.swing.JscrollPane();
    jTable1 = new javax.swing.Jtable();
    jButton2 = new javax.swing.Jbutton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setMinimumSize(new java.awt.Dimension(800, 600));
    setResizable(false);

    jLabel1.setFont(new java.awt.Font(“Arial”, 1, 48)); // NOI18N
    jLabel1.setText(“Check Available Rooms”);

    jLabel2.setFont(new java.awt.Font(“Arial”, 0, 12)); // NOI18N
    jLabel2.setText(“Room Type”);

    buttonGroup1.add(jRadioButton1);
    jRadioButton1.setText(“AC”);

    buttonGroup1.add(jRadioButton2);
    jRadioButton2.setText(“NON AC”);

    jButton1.setText(“Check”);
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });

    jTable1.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {

          },
        new String [] {
            “Room Number”, “Price”
        }
    ) {
        21oolean[] canEdit = new 21oolean [] {
            false, false
        };

        public 21oolean isCellEditable(int rowIndex, int columnIndex) {
            return canEdit [columnIndex];
        }
    });
    jScrollPane1.setViewportView(jTable1);

    jButton2.setText(“Main Menu”);
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(127, 127, 127)
                    .addComponent(jLabel1))
                .addGroup(layout.createSequentialGroup()
                    .addGap(177, 177, 177)
                    .addComponent(jLabel2)
                    .addGap(53, 53, 53)
                    .addComponent(jRadioButton1)
                    .addGap(18, 18, 18)
                    .addComponent(jRadioButton2)
                    .addGap(118, 118, 118)
                    .addComponent(jButton1,      javax.swing.GroupLayout.PREFERRED_SIZE,
100,javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap(130, Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
            .addGap(0, 0, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                    .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE,
481, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(151, 151, 151))
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                    .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE,
100, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(350, 350, 350))))
      );
```

```
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(39, 39, 39)
                .addComponent(jLabel1)
                .addGap(50, 50, 50)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jRadioButton1)
                    .addComponent(jRadioButton2)
                    .addComponent(jButton1,          javax.swing.GroupLayout.PREFERRED_SIZE,
35,javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(66, 66, 66)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
118, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE,
35, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(196, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        new MainMenu().setVisible(true);
        this.setVisible(false);
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        try
        {

            Class.forName("com.mysql.jdbc.Driver");
            Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/dbms_mini_pro",   "root",
"root");
                String sql = "select * from room where room_type=? And room_no not in (select
room_no from stays_in)";
                PreparedStatement p = con.prepareStatement(sql);
                if(jRadioButton1.isSelected()) {
                    p.setString(1,"AC");
                }
                else if(jRadioButton2.isSelected()) {
                    p.setString(1,"NON-AC");
                }
                else {
                    JoptionPane.showMessageDialog(this,"Select a room type please");
                     return;
                }
                ResultSet rs = p.executeQuery();
                DefaultTableModel t = (DefaultTableModel)jTable1.getModel();
                t.setRowCount(0);
                while(rs.next()) {
Object o[]={rs.getString("room_no"), rs.getString("room_price")};
                    t.addRow(o);
                }
                if(t.getRowCount() == 0) {
                    JoptionPane.showMessageDialog(this,"Oops! Looks like there aren't any rooms
available. Please try again later.");
                }

        }
        catch(ClassNotFoundException | SQLException e)
        {
            JoptionPane.showMessageDialog(null, e);
            }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
*For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for(javax.swing.UIManager.LookAndFeelInfo
info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(CheckAvailability.class.getName()).log(java.util.loggin
g.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(CheckAvailability.class.getName()).log(java.util.loggin
g.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
```

```
java.util.logging.Logger.getLogger(CheckAvailability.class.getName()).log(java.util.loggin
g.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(CheckAvailability.class.getName()).log(java.util.loggin
g.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new CheckAvailability().setVisible(true);
            }
        });
    }

    // Variables declaration – do not modify
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.Jbutton jButton1;
    private javax.swing.Jbutton jButton2;
    private javax.swing.Jlabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jTable1;
    // End of variables declaration
  }
```

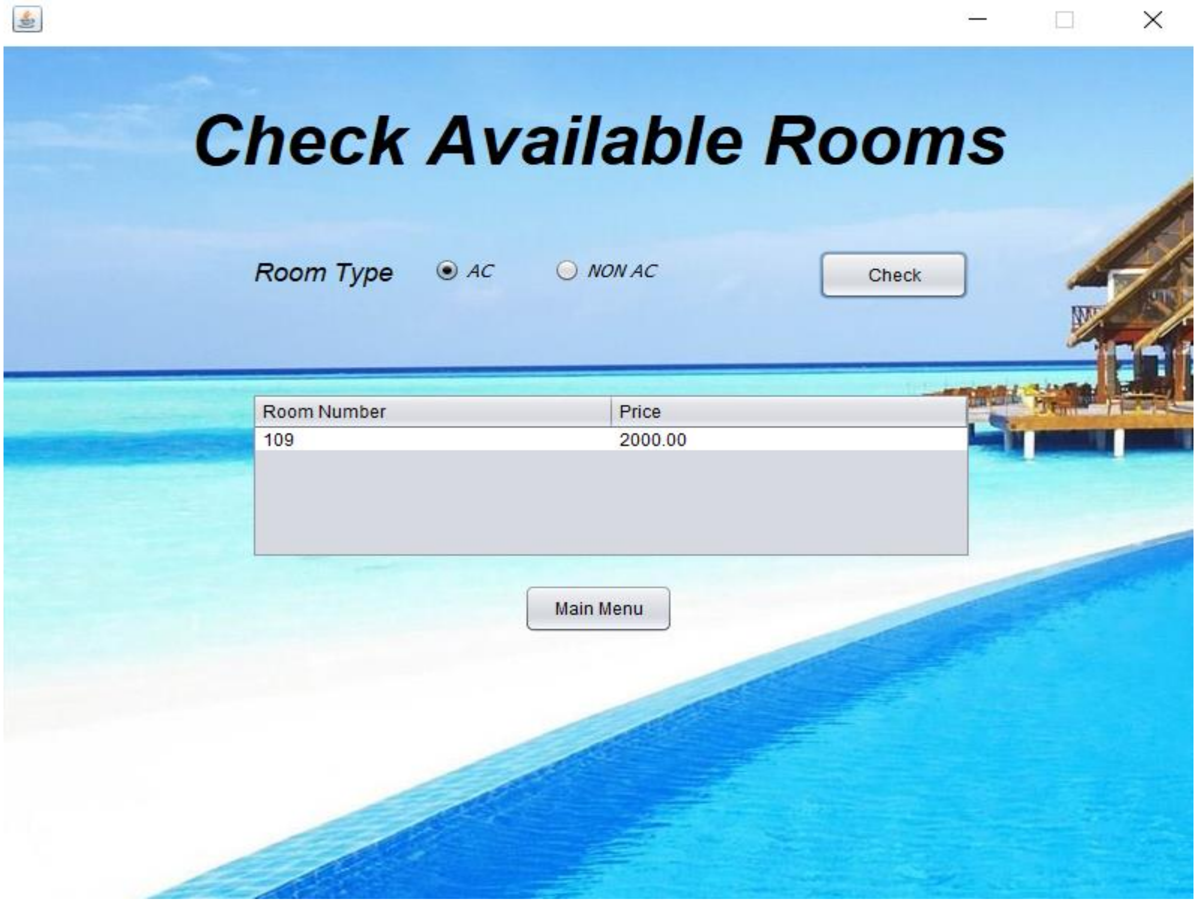
Design preview:



Room Number	Price

Figure 3.1: Sample design of an output jframe

Output:



Room Number	Price
109	2000.00

Figure 3.2: Output of the check available rooms jframe on clicking the check button and selecting the AC radio button

Summary:

GUI (Graphical User Interface) implementation involves providing an aesthetic and more user convenient interface to an existing application. It is helpful in combining requirements from the user and data elements from the backend and shaping them into application objects that complete the functionality.

Here, Netbeans IDE is used to integrate the MySQL backend elements with the essence of java in the form of Java DataBase Connectivity (JDBC) to capture the requirements in a more user-friendly manner than the MySQL command line client.

Each element of a single view of the application is viewed and contained within a swing container called java frame or jframe for short. The jframe encapsulates the structure and code altogether of the particular instance. It contains elements called swing controls which include labels, buttons, text fields, text areas, checklists, lists, radio buttons, etc. These options prove very helpful to the end user to fulfil his requirement.

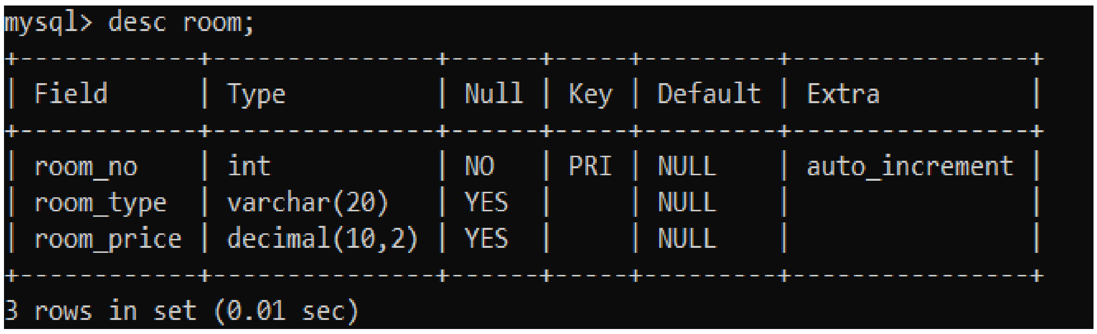
The elements for a given frame can be inserted using the drag and drop feature which gives a fresh and developer friendly approach to the developers to run through his development. He can code every element individually and lessen the burden of having to implement the same manually.

Overall GUI implementation is an aesthetic access to the end user and the front end stream of development for the developer.

CHAPTER 4
RESULTS AND SNAPSHOTS

Description of Table:

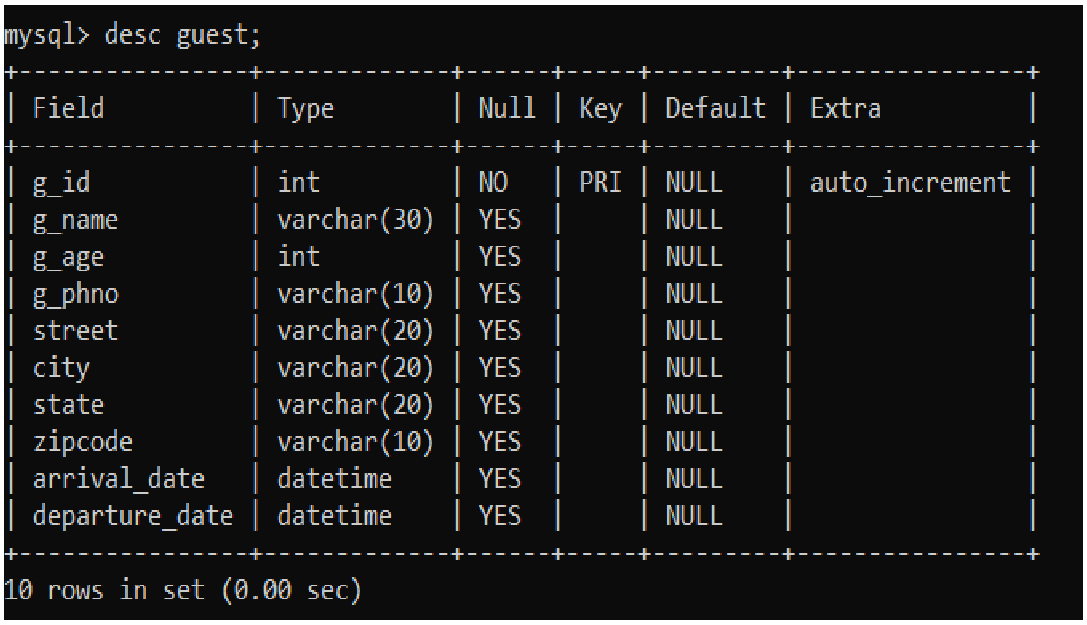
1. room table



```
mysql> desc room;
+------------+---------------+------+-----+---------+----------------+
| Field      | Type          | Null | Key | Default | Extra          |
+------------+---------------+------+-----+---------+----------------+
| room_no    | int           | NO   | PRI | NULL    | auto_increment |
| room_type  | varchar(20)   | YES  |     | NULL    |                |
| room_price | decimal(10,2) | YES  |     | NULL    |                |
+------------+---------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

Figure 4.1 : description of room table

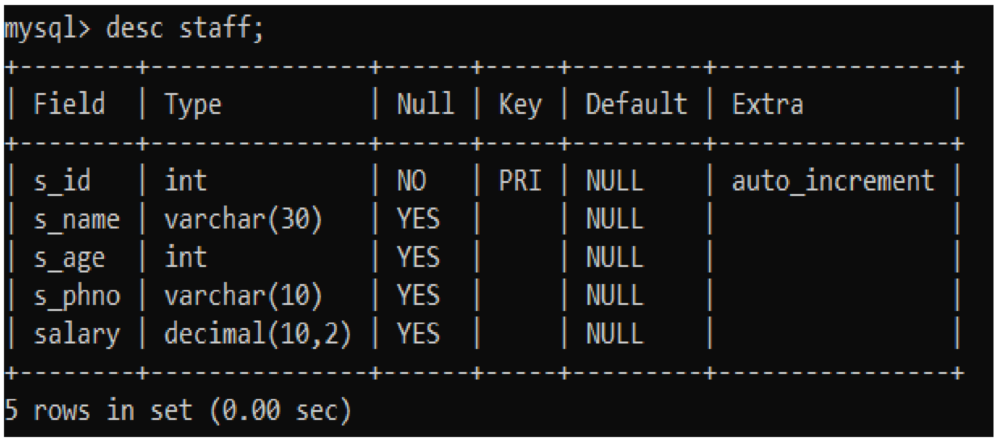
2. guest table



```
mysql> desc guest;
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| g_id           | int         | NO   | PRI | NULL    | auto_increment |
| g_name         | varchar(30) | YES  |     | NULL    |                |
| g_age          | int         | YES  |     | NULL    |                |
| g_phno         | varchar(10) | YES  |     | NULL    |                |
| street         | varchar(20) | YES  |     | NULL    |                |
| city           | varchar(20) | YES  |     | NULL    |                |
| state          | varchar(20) | YES  |     | NULL    |                |
| zipcode        | varchar(10) | YES  |     | NULL    |                |
| arrival_date   | datetime    | YES  |     | NULL    |                |
| departure_date | datetime    | YES  |     | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
10 rows in set (0.00 sec)
```

Figure 4.2 : description of guest table

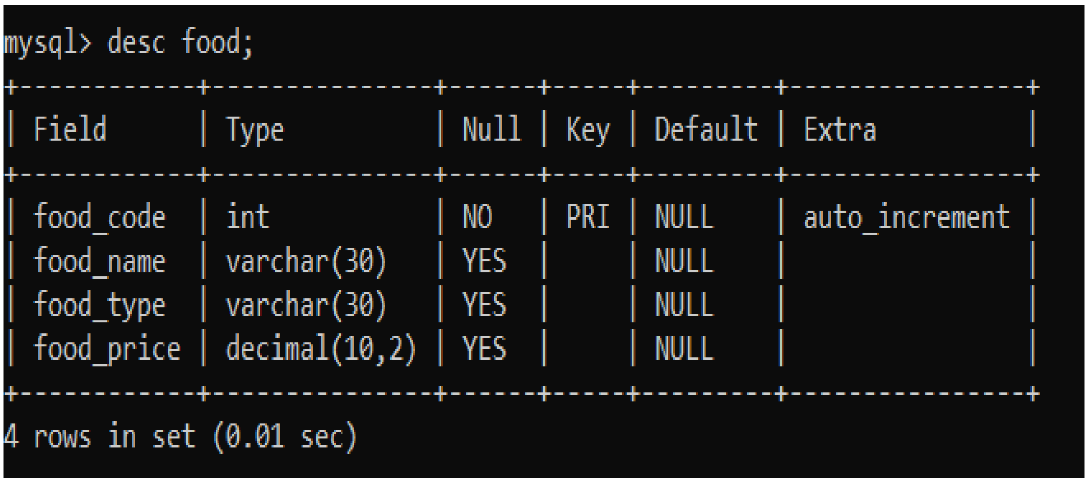
3. staff table



```
mysql> desc staff;
+--------+---------------+------+-----+---------+----------------+
| Field  | Type          | Null | Key | Default | Extra          |
+--------+---------------+------+-----+---------+----------------+
| s_id   | int           | NO   | PRI | NULL    | auto_increment |
| s_name | varchar(30)   | YES  |     | NULL    |                |
| s_age  | int           | YES  |     | NULL    |                |
| s_phno | varchar(10)   | YES  |     | NULL    |                |
| salary | decimal(10,2) | YES  |     | NULL    |                |
+--------+---------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Figure 4.3 : description of staff table

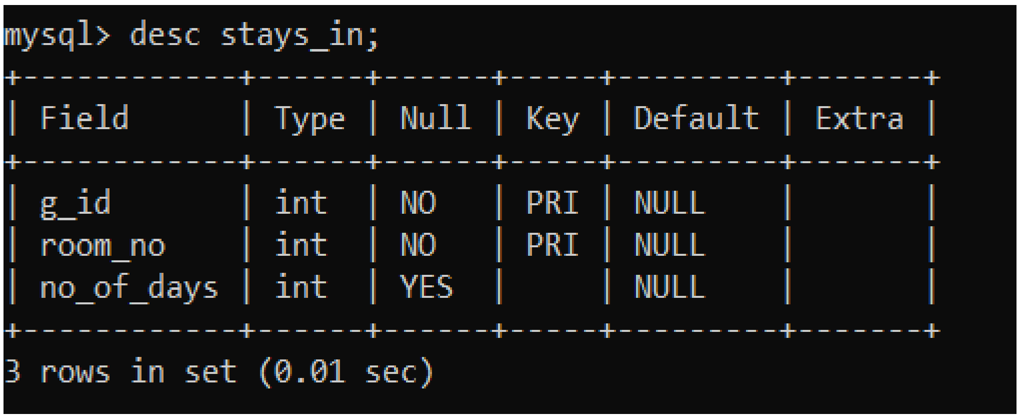
4. food table



```
mysql> desc food;
+------------+---------------+------+-----+---------+----------------+
| Field      | Type          | Null | Key | Default | Extra          |
+------------+---------------+------+-----+---------+----------------+
| food_code  | int           | NO   | PRI | NULL    | auto_increment |
| food_name  | varchar(30)   | YES  |     | NULL    |                |
| food_type  | varchar(30)   | YES  |     | NULL    |                |
| food_price | decimal(10,2) | YES  |     | NULL    |                |
+------------+---------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

Figure 4.4 : description of food table

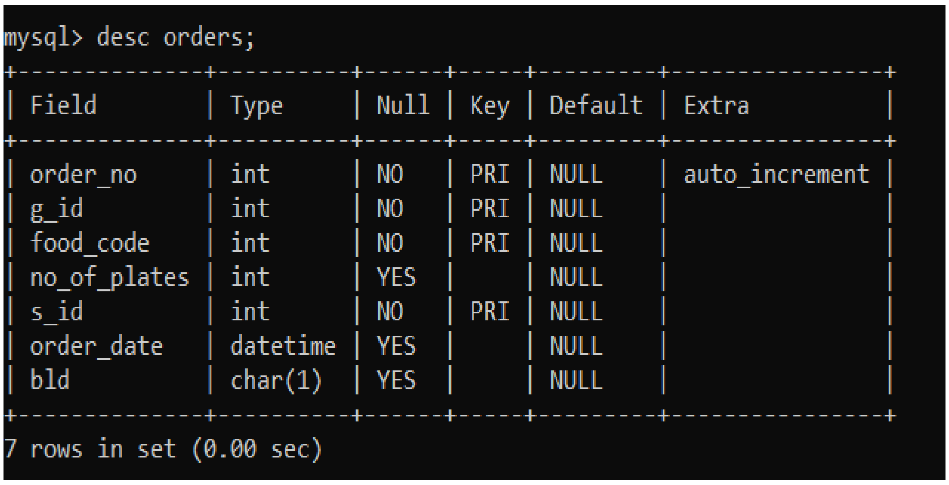
5. stays_in table



```
mysql> desc stays_in;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| g_id       | int  | NO   | PRI | NULL    |       |
| room_no    | int  | NO   | PRI | NULL    |       |
| no_of_days | int  | YES  |     | NULL    |       |
+------------+------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

Figure 4.5 : description of stays_in table

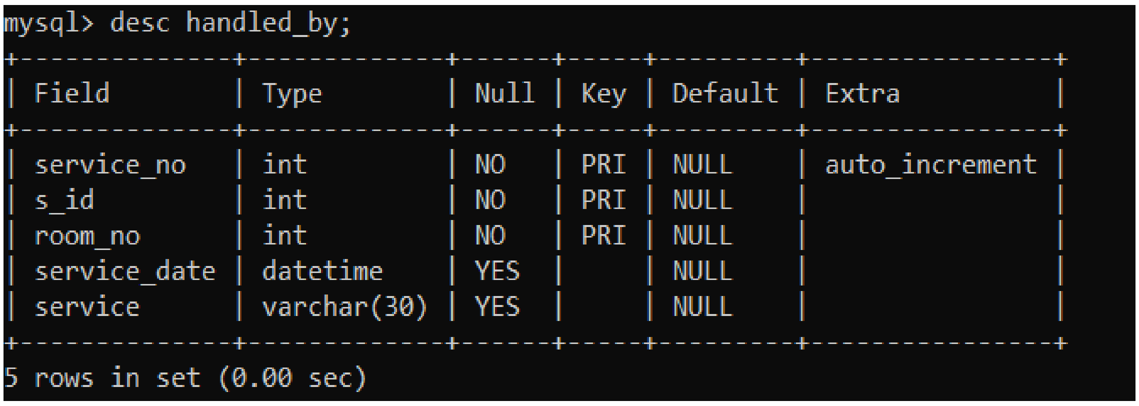
6. orders table



```
mysql> desc orders;
+-------------+----------+------+-----+---------+----------------+
| Field       | Type     | Null | Key | Default | Extra          |
+-------------+----------+------+-----+---------+----------------+
| order_no    | int      | NO   | PRI | NULL    | auto_increment |
| g_id        | int      | NO   | PRI | NULL    |                |
| food_code   | int      | NO   | PRI | NULL    |                |
| no_of_plates| int      | YES  |     | NULL    |                |
| s_id        | int      | NO   | PRI | NULL    |                |
| order_date  | datetime | YES  |     | NULL    |                |
| bld         | char(1)  | YES  |     | NULL    |                |
+-------------+----------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

Figure 4.6 : description of orders table

7. handled_by table

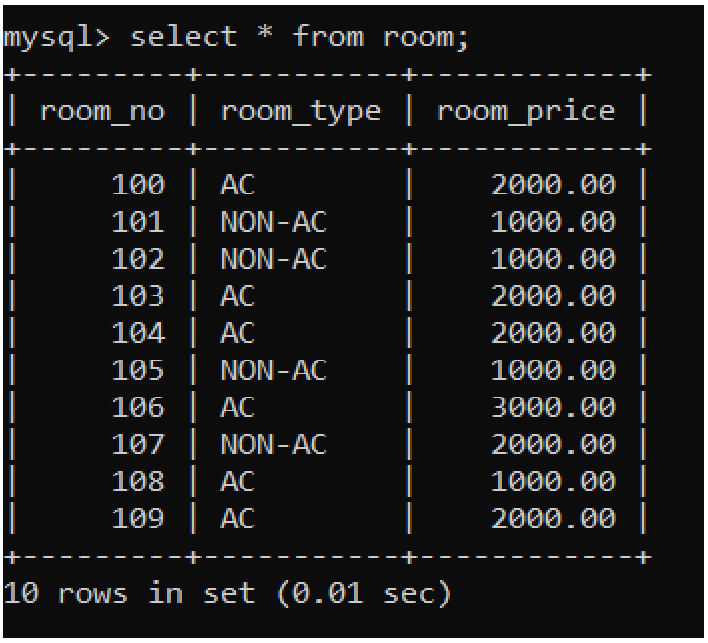


```
mysql> desc handled_by;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| service_no   | int         | NO   | PRI | NULL    | auto_increment |
| s_id         | int         | NO   | PRI | NULL    |                |
| room_no      | int         | NO   | PRI | NULL    |                |
| service_date | datetime    | YES  |     | NULL    |                |
| service      | varchar(30) | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Figure 4.7 : description of handled_by table

Displaying the contents of the table:

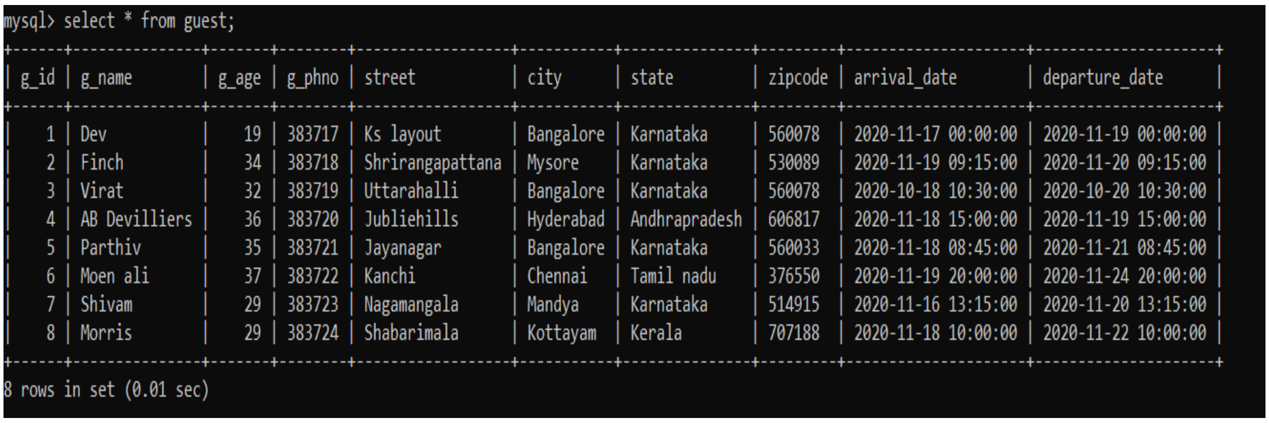
1. room table



```
mysql> select * from room;
+---------+-----------+------------+
| room_no | room_type | room_price |
+---------+-----------+------------+
|     100 | AC        |    2000.00 |
|     101 | NON-AC    |    1000.00 |
|     102 | NON-AC    |    1000.00 |
|     103 | AC        |    2000.00 |
|     104 | AC        |    2000.00 |
|     105 | NON-AC    |    1000.00 |
|     106 | AC        |    3000.00 |
|     107 | NON-AC    |    2000.00 |
|     108 | AC        |    1000.00 |
|     109 | AC        |    2000.00 |
+---------+-----------+------------+
10 rows in set (0.01 sec)
```

Figure 4.8 : contents of room table

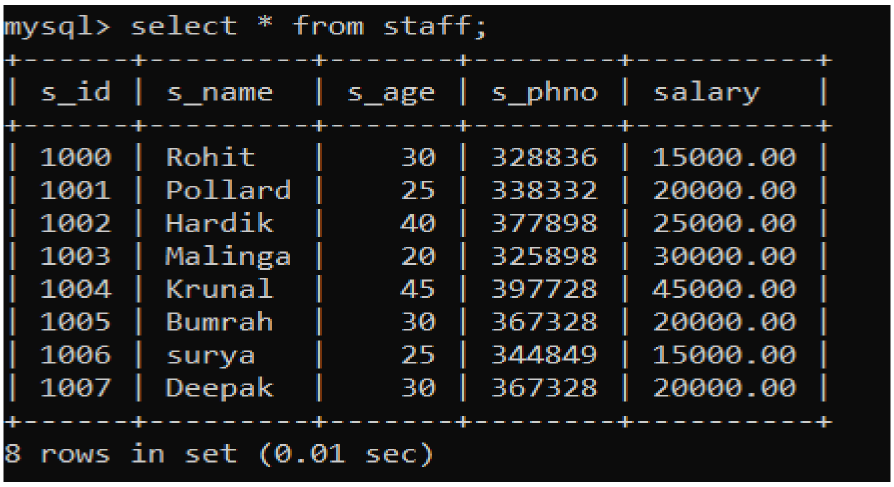
2. guest table



```
mysql> select * from guest;
+------+---------------+-------+--------+------------------+-----------+---------------+---------+---------------------+---------------------+
| g_id | g_name        | g_age | g_phno | street           | city      | state         | zipcode | arrival_date        | departure_date      |
+------+---------------+-------+--------+------------------+-----------+---------------+---------+---------------------+---------------------+
|    1 | Dev           |    19 | 383717 | Ks layout        | Bangalore | Karnataka     | 560078  | 2020-11-17 00:00:00 | 2020-11-19 00:00:00 |
|    2 | Finch         |    34 | 383718 | Shrirangapattana | Mysore    | Karnataka     | 530089  | 2020-11-19 09:15:00 | 2020-11-20 09:15:00 |
|    3 | Virat         |    32 | 383719 | Uttarahalli      | Bangalore | Karnataka     | 560078  | 2020-10-18 10:30:00 | 2020-10-20 10:30:00 |
|    4 | AB Devilliers |    36 | 383720 | Jubliehills      | Hyderabad | Andhrapradesh | 606817  | 2020-11-18 15:00:00 | 2020-11-19 15:00:00 |
|    5 | Parthiv       |    35 | 383721 | Jayanagar        | Bangalore | Karnataka     | 560033  | 2020-11-18 08:45:00 | 2020-11-21 08:45:00 |
|    6 | Moen ali      |    37 | 383722 | Kanchi           | Chennai   | Tamil nadu    | 376550  | 2020-11-19 20:00:00 | 2020-11-24 20:00:00 |
|    7 | Shivam        |    29 | 383723 | Nagamangala      | Mandya    | Karnataka     | 514915  | 2020-11-16 13:15:00 | 2020-11-20 13:15:00 |
|    8 | Morris        |    29 | 383724 | Shabarimala      | Kottayam  | Kerala        | 707188  | 2020-11-18 10:00:00 | 2020-11-22 10:00:00 |
+------+---------------+-------+--------+------------------+-----------+---------------+---------+---------------------+---------------------+
8 rows in set (0.01 sec)
```

Figure 4.9 : contents of guest table

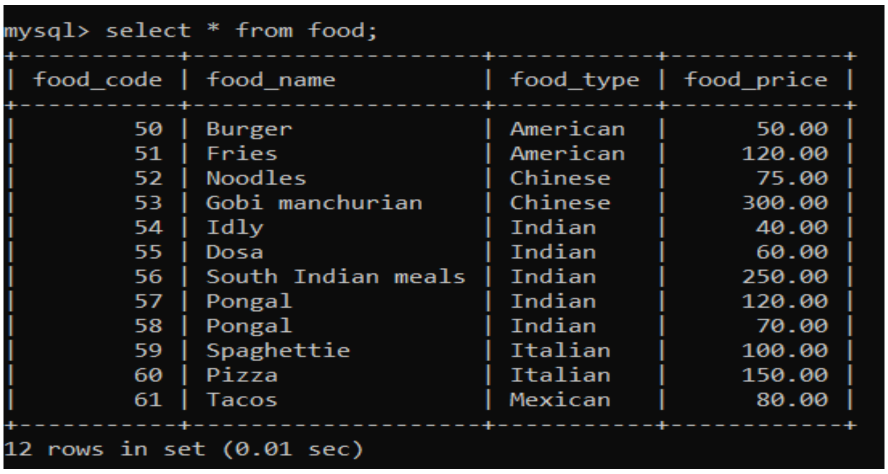
3. staff table



```
mysql> select * from staff;
+------+---------+-------+--------+----------+
| s_id | s_name  | s_age | s_phno | salary   |
+------+---------+-------+--------+----------+
| 1000 | Rohit   |    30 | 328836 | 15000.00 |
| 1001 | Pollard |    25 | 338332 | 20000.00 |
| 1002 | Hardik  |    40 | 377898 | 25000.00 |
| 1003 | Malinga |    20 | 325898 | 30000.00 |
| 1004 | Krunal  |    45 | 397728 | 45000.00 |
| 1005 | Bumrah  |    30 | 367328 | 20000.00 |
| 1006 | surya   |    25 | 344849 | 15000.00 |
| 1007 | Deepak  |    30 | 367328 | 20000.00 |
+------+---------+-------+--------+----------+
8 rows in set (0.01 sec)
```

Figure 4.10 : contents of staff table

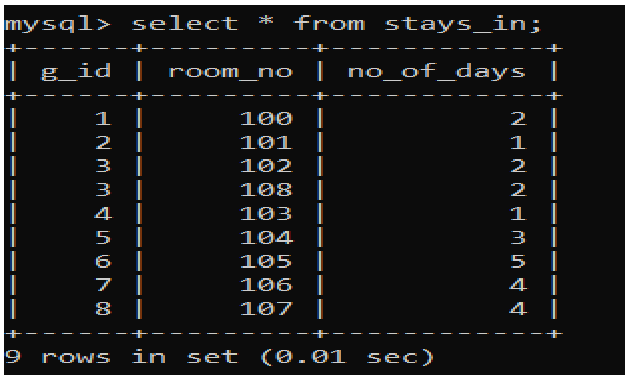
4. food table



```
mysql> select * from food;
+-----------+--------------------+-----------+------------+
| food_code | food_name          | food_type | food_price |
+-----------+--------------------+-----------+------------+
|        50 | Burger             | American  |      50.00 |
|        51 | Fries              | American  |     120.00 |
|        52 | Noodles            | Chinese   |      75.00 |
|        53 | Gobi manchurian    | Chinese   |     300.00 |
|        54 | Idly               | Indian    |      40.00 |
|        55 | Dosa               | Indian    |      60.00 |
|        56 | South Indian meals | Indian    |     250.00 |
|        57 | Pongal             | Indian    |     120.00 |
|        58 | Pongal             | Indian    |      70.00 |
|        59 | Spaghettie         | Italian   |     100.00 |
|        60 | Pizza              | Italian   |     150.00 |
|        61 | Tacos              | Mexican   |      80.00 |
+-----------+--------------------+-----------+------------+
12 rows in set (0.01 sec)
```

Figure 4.11 : contents of food table

5. stays_in table



```
mysql> select * from stays_in;
+------+---------+------------+
| g_id | room_no | no_of_days |
+------+---------+------------+
|    1 |     100 |          2 |
|    2 |     101 |          1 |
|    3 |     102 |          2 |
|    3 |     108 |          2 |
|    4 |     103 |          1 |
|    5 |     104 |          3 |
|    6 |     105 |          5 |
|    7 |     106 |          4 |
|    8 |     107 |          4 |
+------+---------+------------+
9 rows in set (0.01 sec)
```

Figure 4.12 : contents of stays_in table

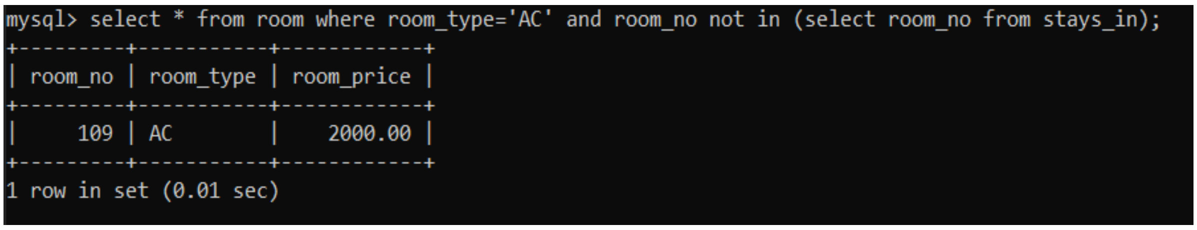
Queries:

1. Check available rooms (AC or NONAC)

i. AC

i. **Command:** select * from room where room_type='AC' and room_no not in (select room_no from stays_in);

ii. **Output:**



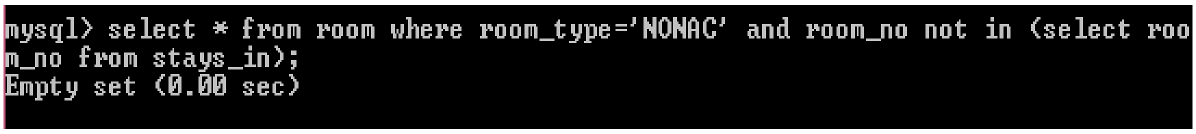
```
mysql> select * from room where room_type='AC' and room_no not in (select room_no from stays_in);
+---------+-----------+------------+
| room_no | room_type | room_price |
+---------+-----------+------------+
|     109 | AC        |    2000.00 |
+---------+-----------+------------+
1 row in set (0.01 sec)
```

Figure 4.13 : query to check available AC rooms

ii. NONAC

i. **Command:** select * from room where room_type='NONAC' and room_no not in (select room_no from stays_in)

ii. **Output:**



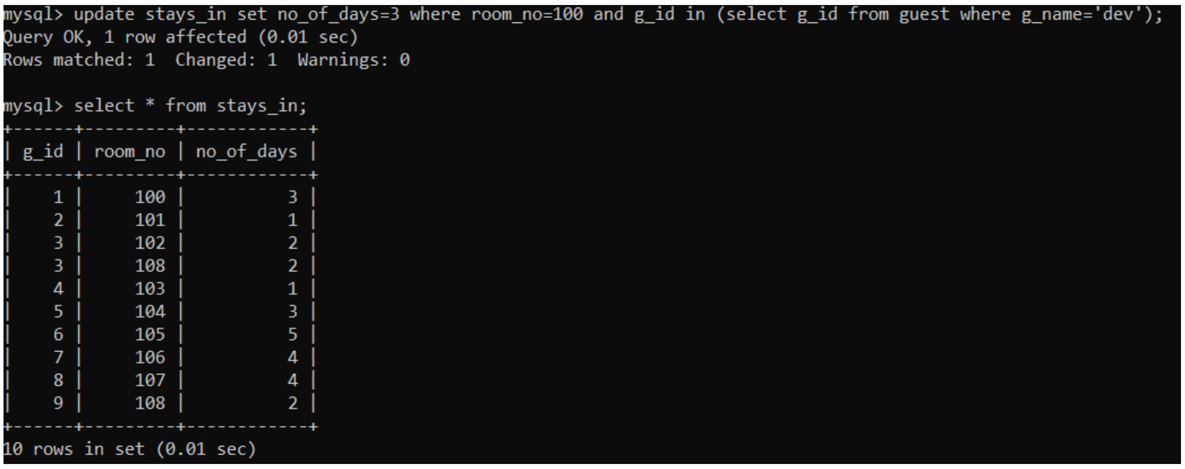
```
mysql> select * from room where room_type='NONAC' and room_no not in (select roo
m_no from stays_in);
Empty set (0.00 sec)
```

Figure 4.14 : query to check available NON AC rooms

2. a. Update Booking

i. **Command:** update stays_in set no_of_days=? where room_no=? and g_id in (select g_id from guest where g_name=?);

ii. **Output:**



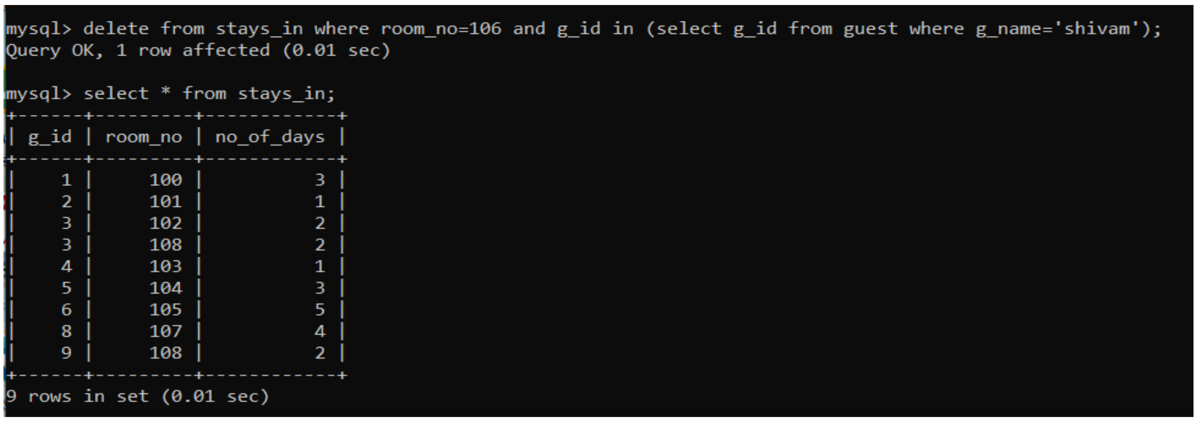
```
mysql> update stays_in set no_of_days=3 where room_no=100 and g_id in (select g_id from guest where g_name='dev');
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from stays_in;
+------+---------+------------+
| g_id | room_no | no_of_days |
+------+---------+------------+
|    1 |     100 |          3 |
|    2 |     101 |          1 |
|    3 |     102 |          2 |
|    3 |     108 |          2 |
|    4 |     103 |          1 |
|    5 |     104 |          3 |
|    6 |     105 |          5 |
|    7 |     106 |          4 |
|    8 |     107 |          4 |
|    9 |     108 |          2 |
+------+---------+------------+
10 rows in set (0.01 sec)
```

Figure 4.15 : query to update an existing room booking

b. Cancel Booking

i. **Command:** delete from stays_in where room_no=? and g_id in (select g_id from guest where g_name=?);

ii. **Output:**



```
mysql> delete from stays_in where room_no=106 and g_id in (select g_id from guest where g_name='shivam');
Query OK, 1 row affected (0.01 sec)

mysql> select * from stays_in;
+------+---------+------------+
| g_id | room_no | no_of_days |
+------+---------+------------+
|    1 |     100 |          3 |
|    2 |     101 |          1 |
|    3 |     102 |          2 |
|    3 |     108 |          2 |
|    4 |     103 |          1 |
|    5 |     104 |          3 |
|    6 |     105 |          5 |
|    8 |     107 |          4 |
|    9 |     108 |          2 |
+------+---------+------------+
9 rows in set (0.01 sec)
```

Figure 4.16 : query to cancel an existing booking

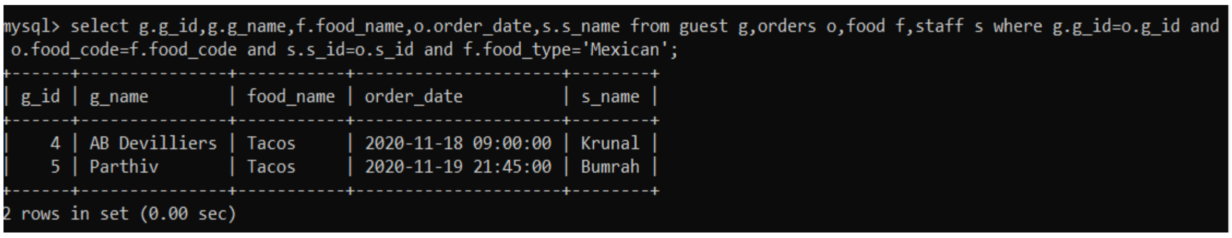
3. Display the guest name who has ordered :

- **Indian Food/American/Chinese/Italian/Mexican**

i. **Command:**

select g.g_id,g.g_name,f.food_name,o.order_date,s.s_name from guest g,orders o,food f,staff s where g.g_id=o.g_id and o.food_code=f.food_code and s.s_id=o.s_id and f.food_type='???';

ii. **Output:**



```
mysql> select g.g_id,g.g_name,f.food_name,o.order_date,s.s_name from guest g,orders o,food f,staff s where g.g_id=o.g_id and
 o.food_code=f.food_code and s.s_id=o.s_id and f.food_type='Mexican';
+------+--------------+-----------+---------------------+--------+
| g_id | g_name       | food_name | order_date          | s_name |
+------+--------------+-----------+---------------------+--------+
|    4 | AB Devilliers | Tacos     | 2020-11-18 09:00:00 | Krunal |
|    5 | Parthiv      | Tacos     | 2020-11-19 21:45:00 | Bumrah |
+------+--------------+-----------+---------------------+--------+
2 rows in set (0.00 sec)
```

Figure 4.17 : query to display order details by the item ordered

4. Generate Foodbill:

i. Procedure:

delimiter //

create procedure foodbill(in guest_id varchar(50))

begin

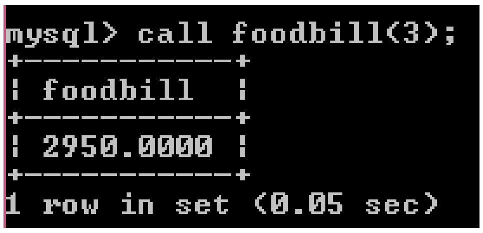
select (1.25 * sum(food_price * no_of_plates)) as foodbill from food f, orders o where f.food_code = o.food_code and g_id=guest_id;

end //

delimiter ;

ii. Command: call foodbill(?);

iii. Output



```
mysql> call foodbill(3);
+-----------+
| foodbill  |
+-----------+
| 2950.0000 |
+-----------+
1 row in set (0.05 sec)
```

Figure 4.18 : query to compute foodbill

5. Generate Roombill:

i. Procedure:

delimiter //

create procedure roombill(in guest_id varchar(50))

begin

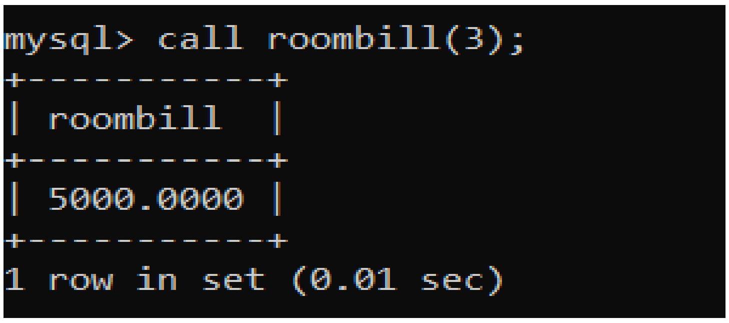
select (1.25 * sum(room_price * no_of_days)) as roombill from room r, stays_in s where r.room_no = s.room_no and g_id=guest_id;

end //

delimiter ;

ii. Command: call roombill(?);

iii. Output:



```
mysql> call roombill(3);
+-----------+
| roombill  |
+-----------+
| 5000.0000 |
+-----------+
1 row in set (0.01 sec)
```

Figure 4.19 : query to compute roombill

FRONT END FRAMES

Login Page

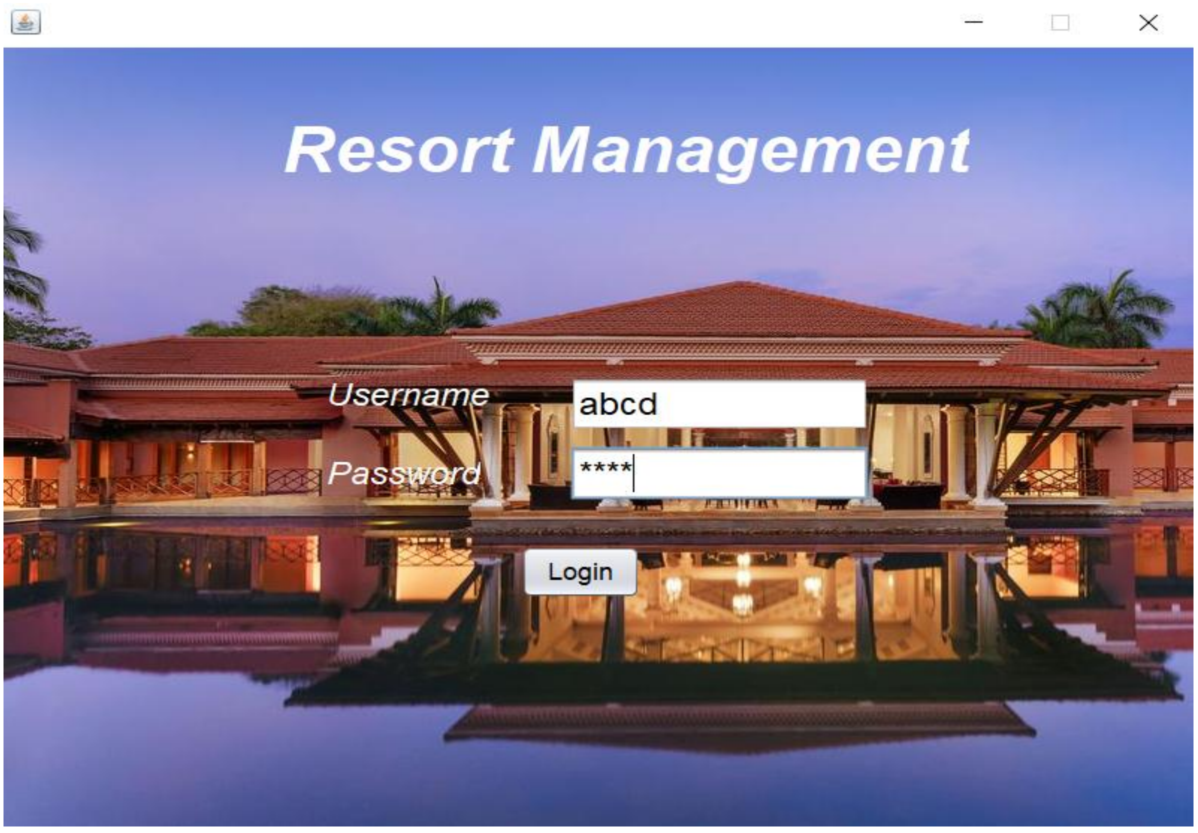


Figure 4.20 : Login frame

Home Page:

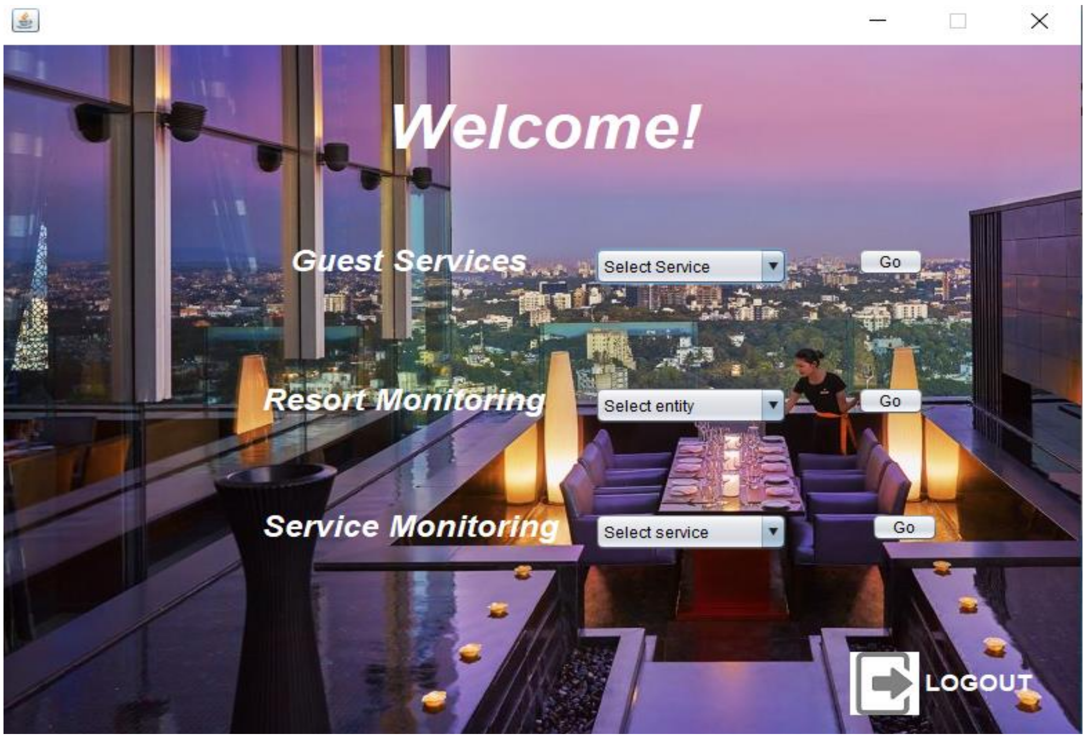


Figure 4.21 : Main Menu frame

Guest Services

Check_in

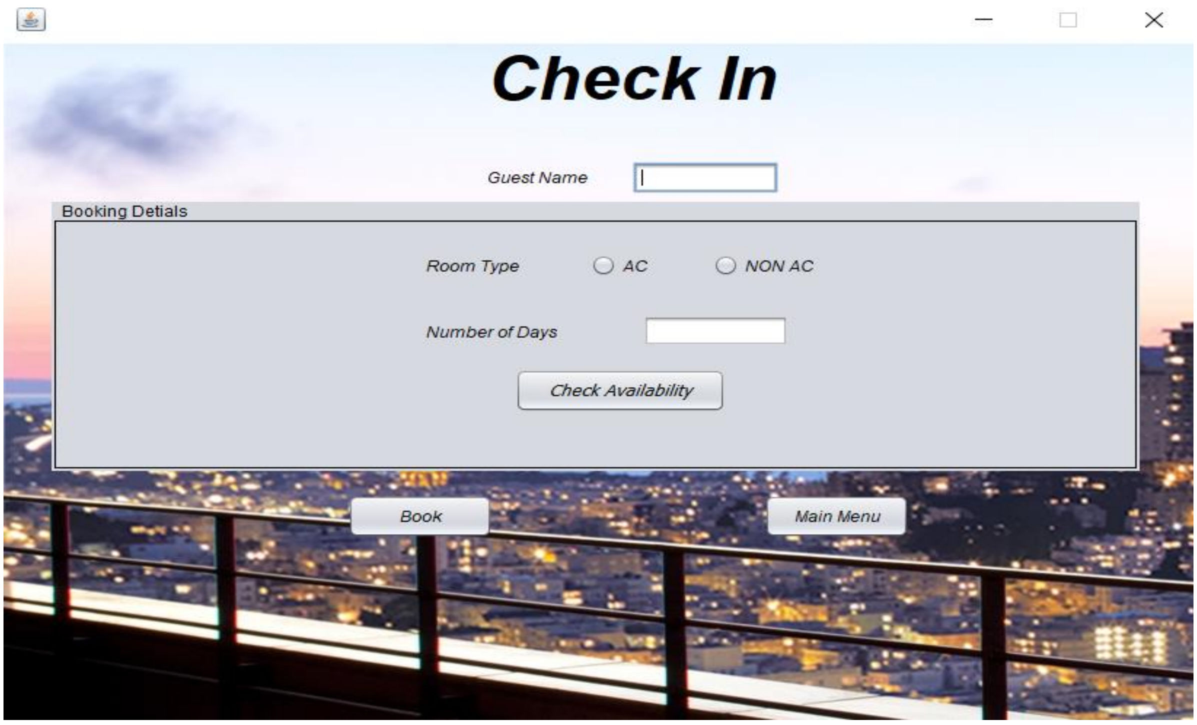


Figure 4.22 : Check in frame

Check_availiability

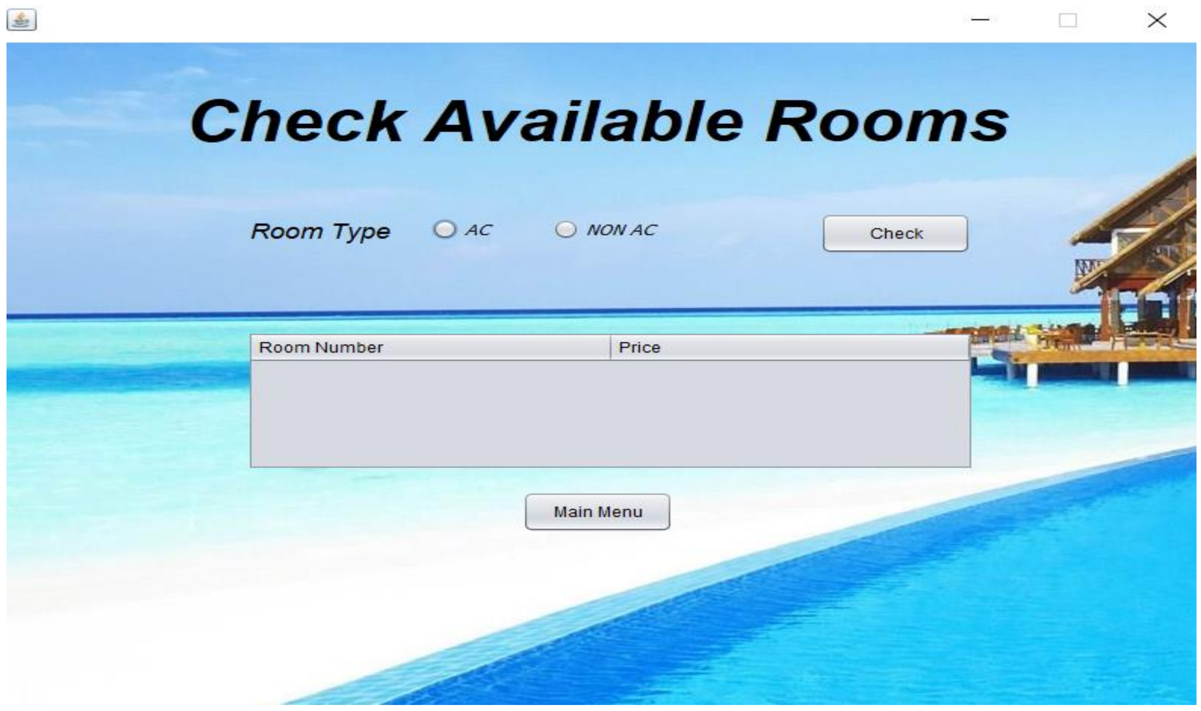


Figure 4.23 : Check out frame

Update booking

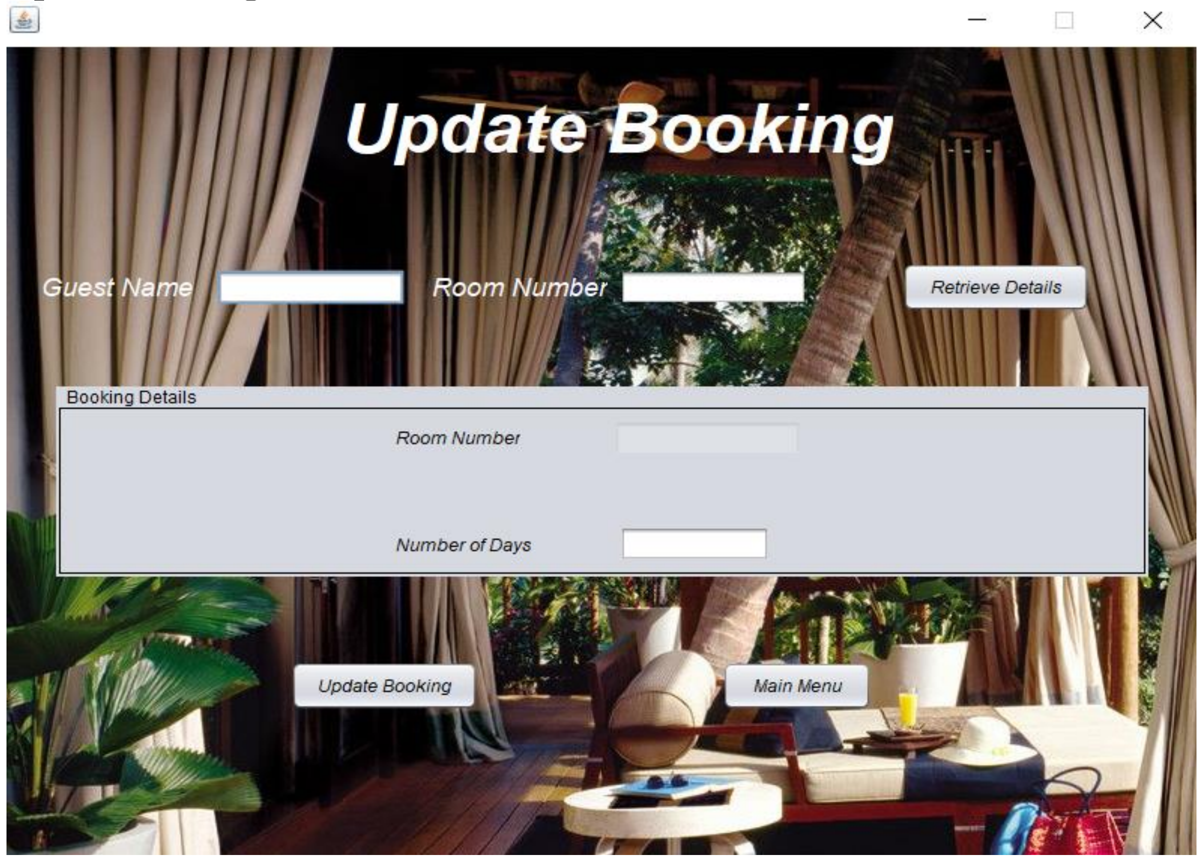


Figure 4.24 : update booking frame

Cancel booking

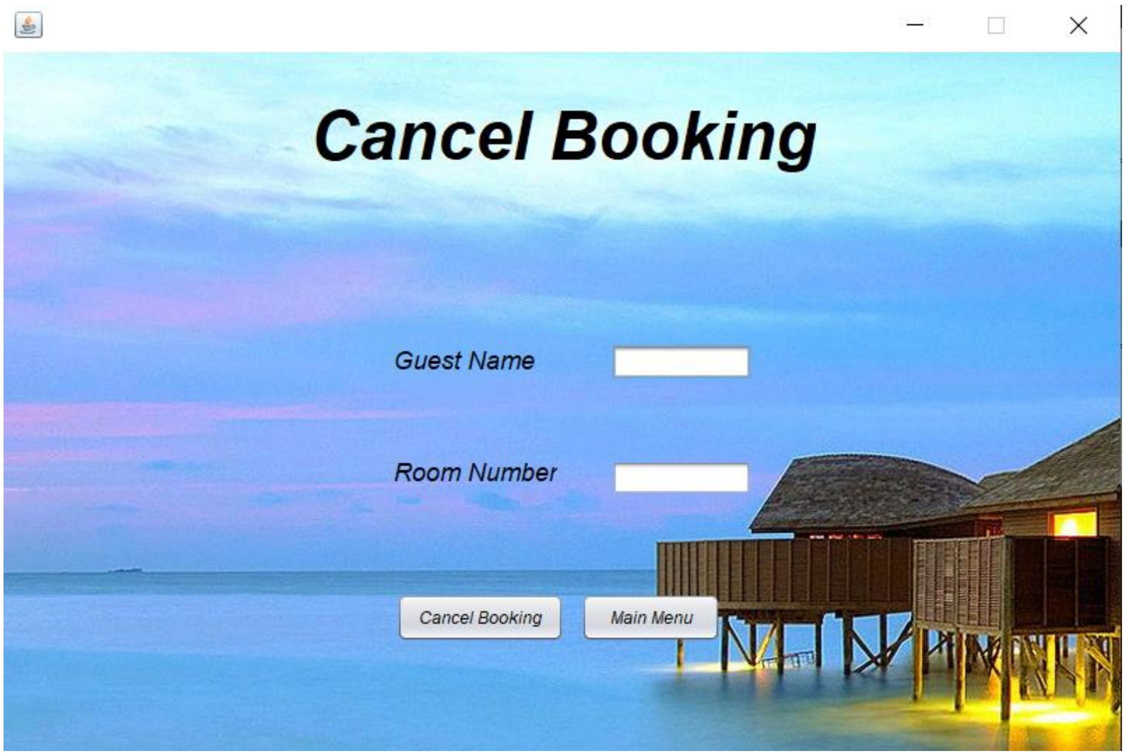
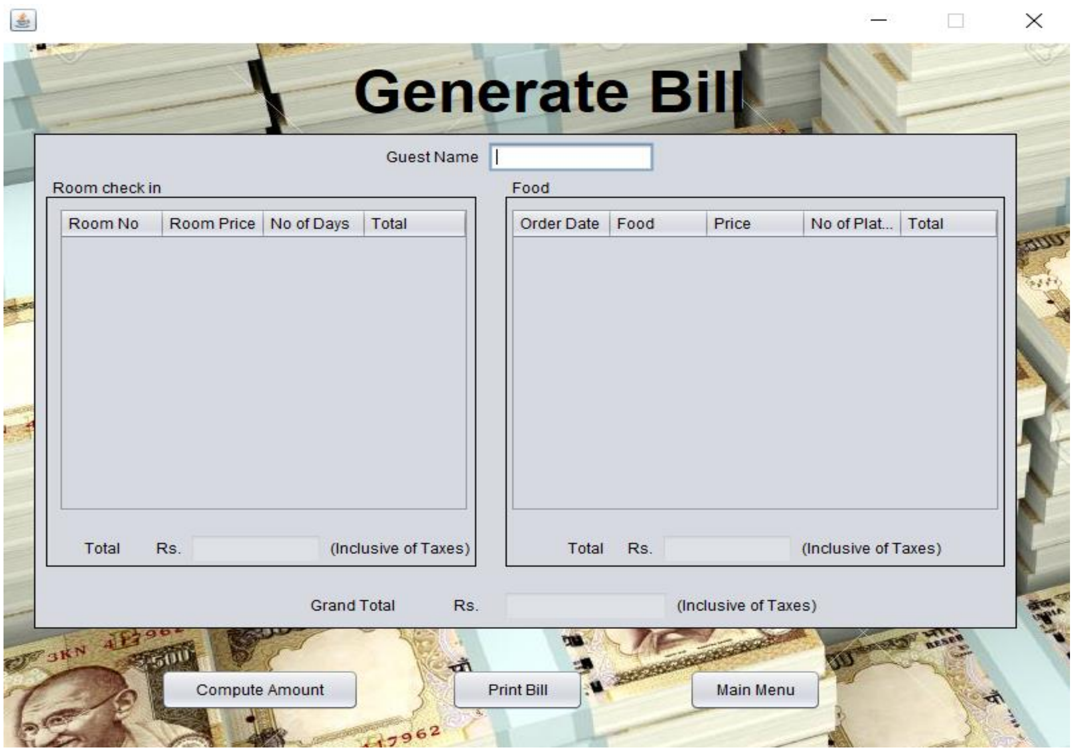


Figure 4.25 : cancel booking frame

Generate_bill



Room No	Room Price	No of Days	Total

Total Rs. (Inclusive of Taxes)

Order Date	Food	Price	No of Plat...	Total

Total Rs. (Inclusive of Taxes)

Grand Total Rs. (Inclusive of Taxes)

Figure 4.26 : generate bill frame

Check_out

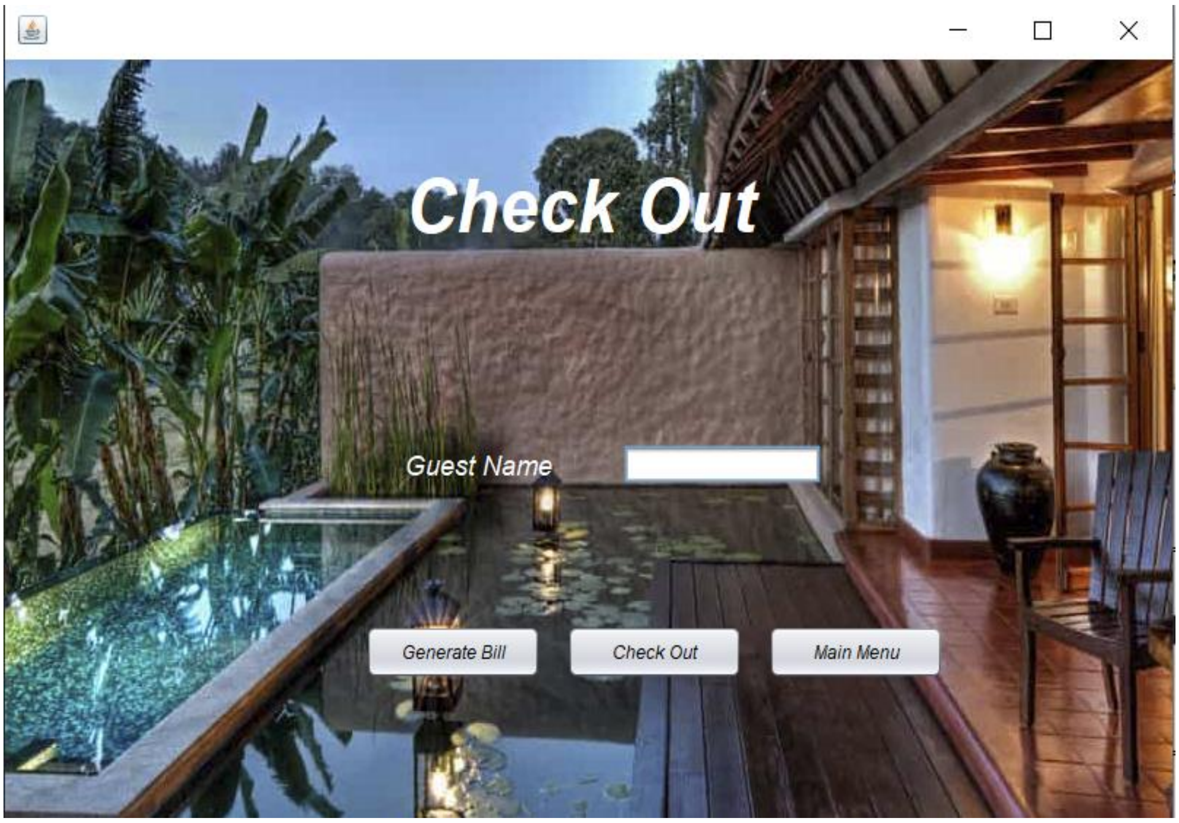
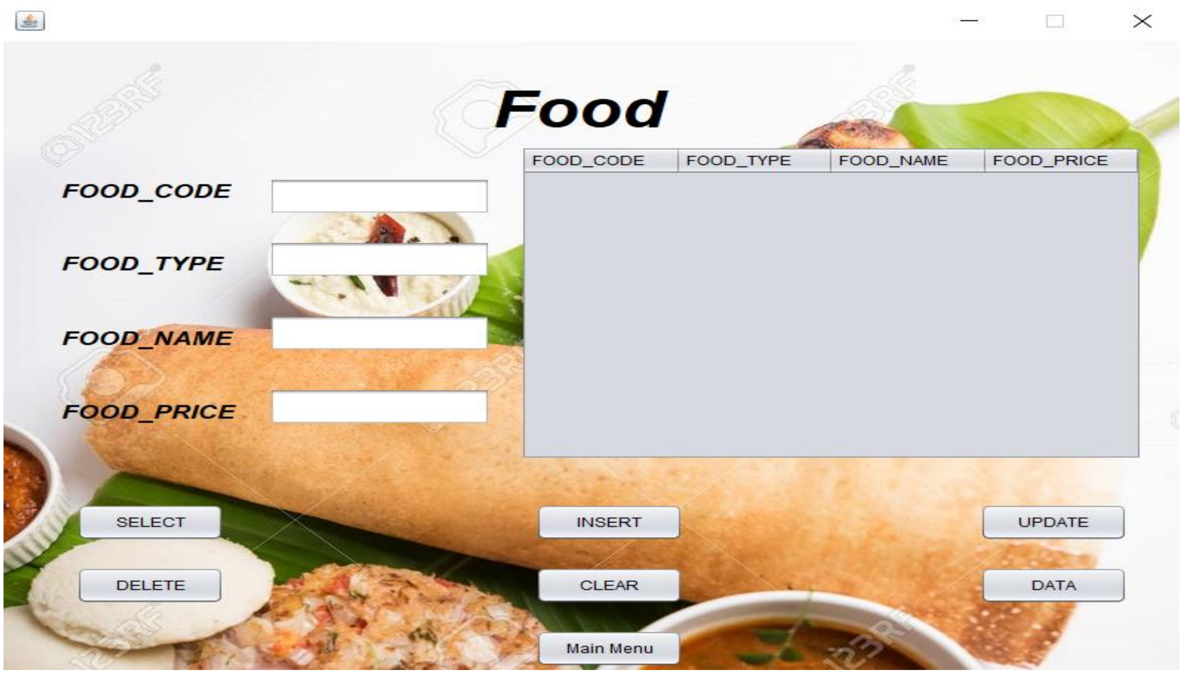


Figure 4.27 : check out frame

Resort Monitoring

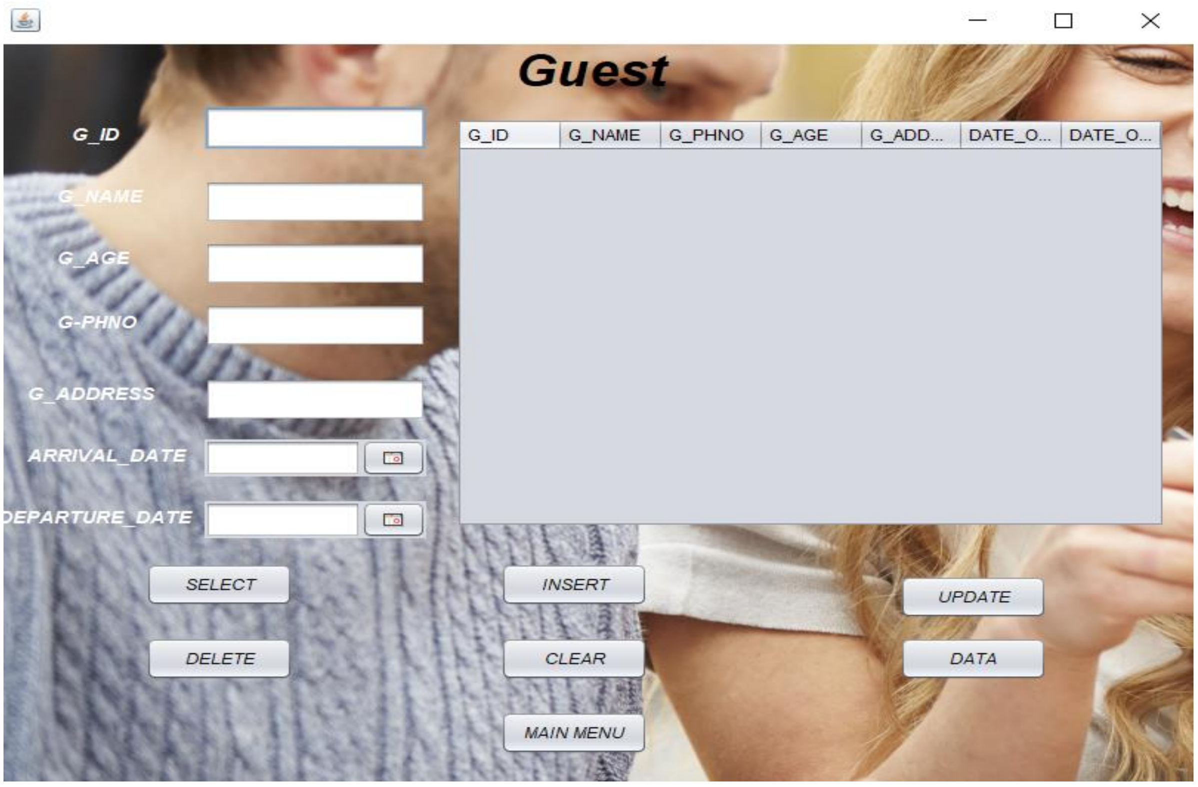
Food



FOOD_CODE	FOOD_TYPE	FOOD_NAME	FOOD_PRICE

Figure 4.28 : food entity frame

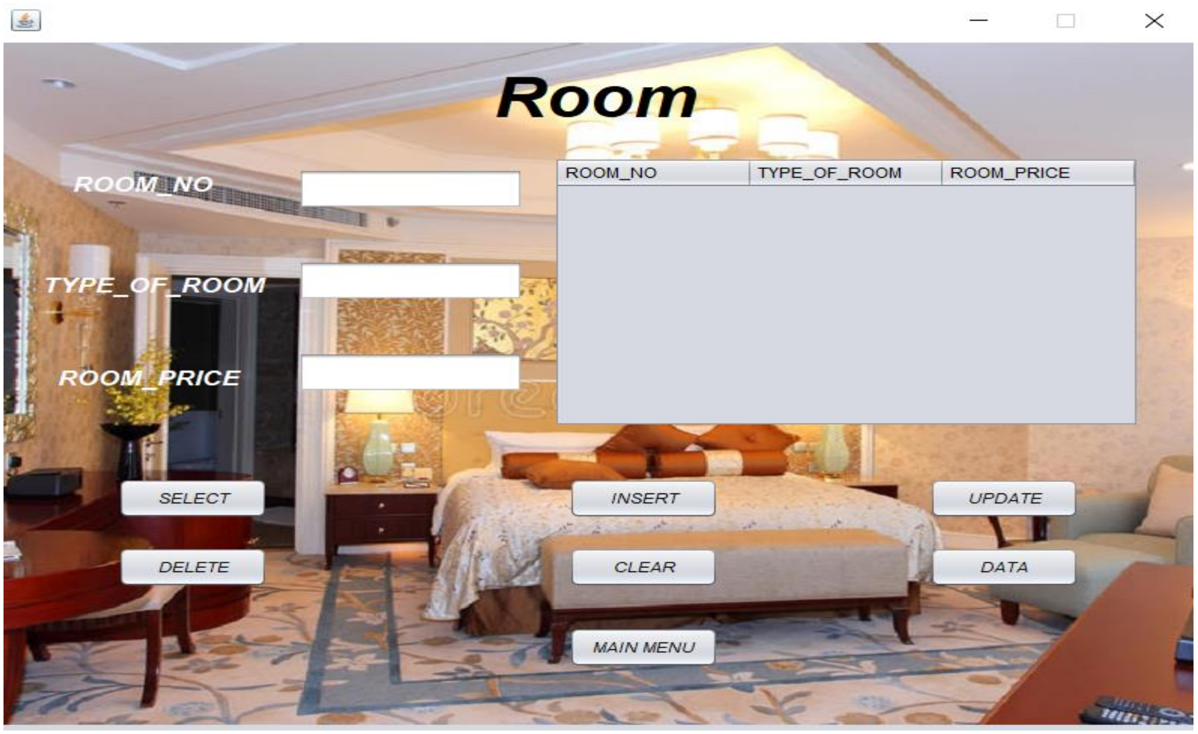
Guests



G_ID	G_NAME	G_PHNO	G_AGE	G_ADD...	DATE_O...	DATE_O...

Figure 4.29 : guest entity frame

Room



ROOM_NO	TYPE_OF_ROOM	ROOM_PRICE

Figure 4.30 : room entity frame

Staff

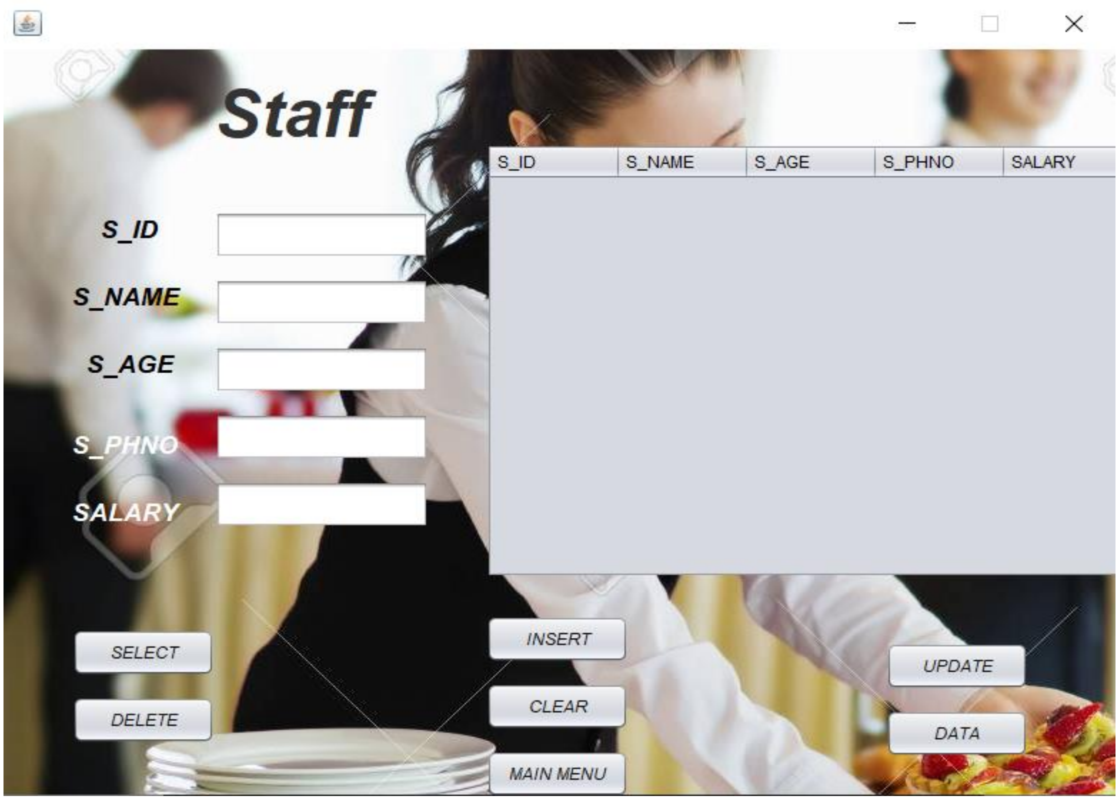


Figure 4.31 : staff entity frame

Service Monitoring

Food_orders

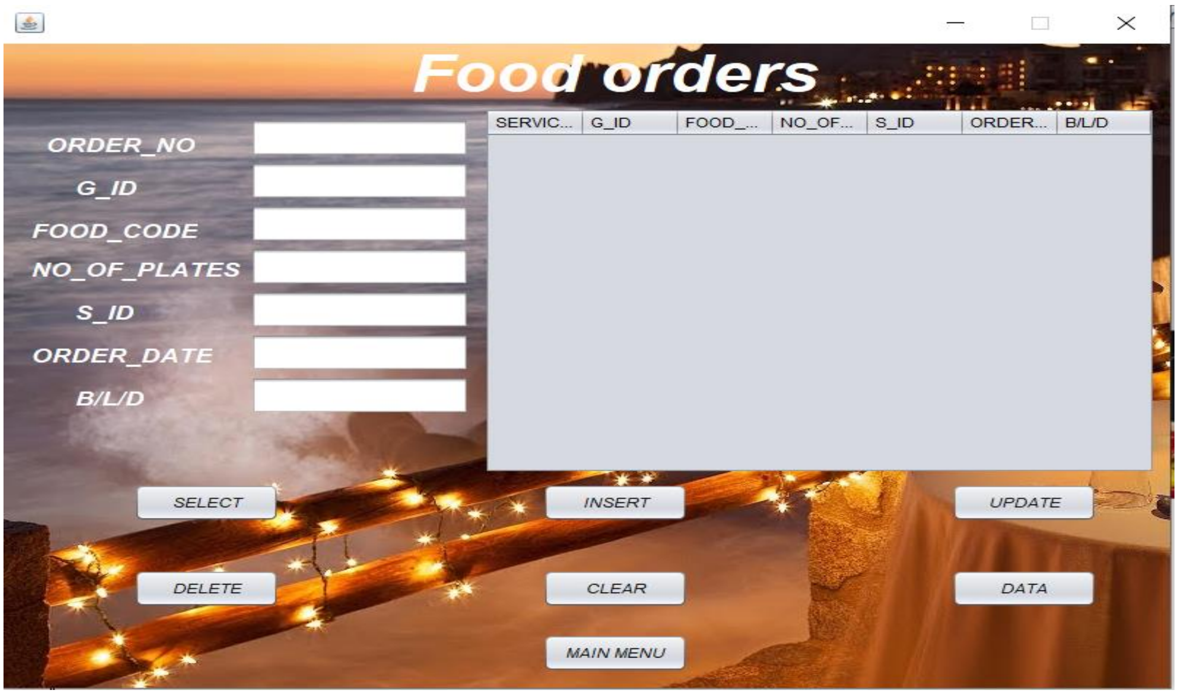


Figure 4.32 : food orders monitoring frame

Room_service

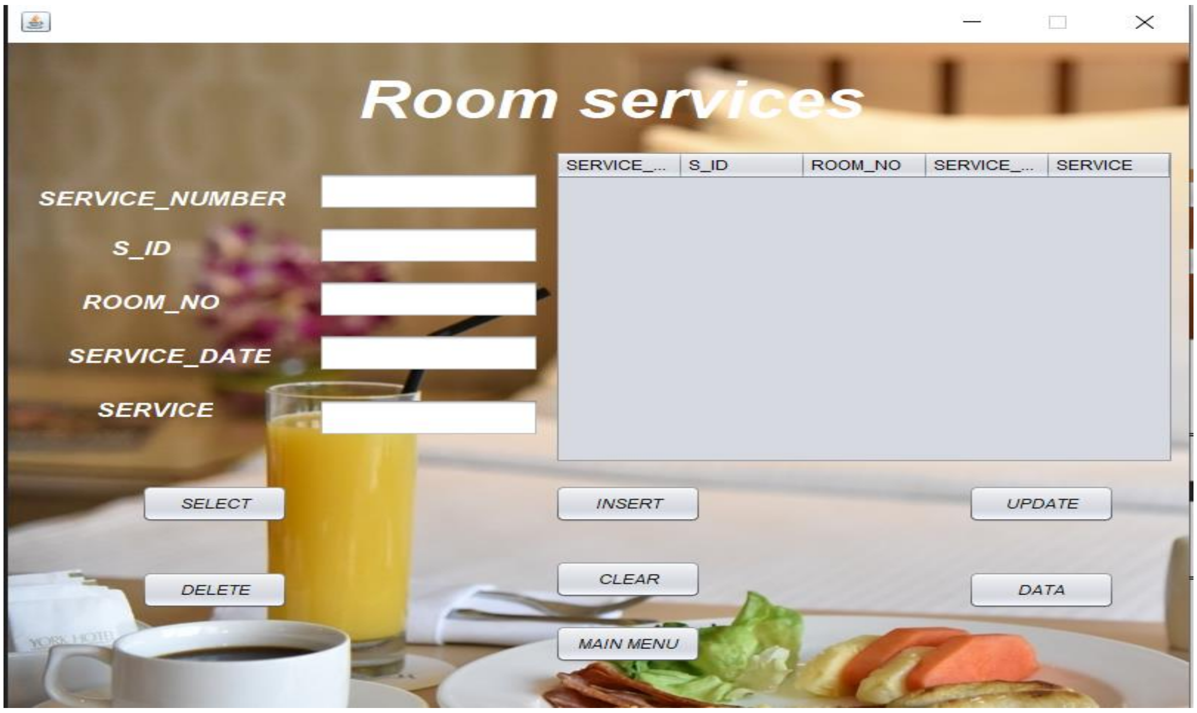


Figure 4.33 : room services monitoring frame

Food_type

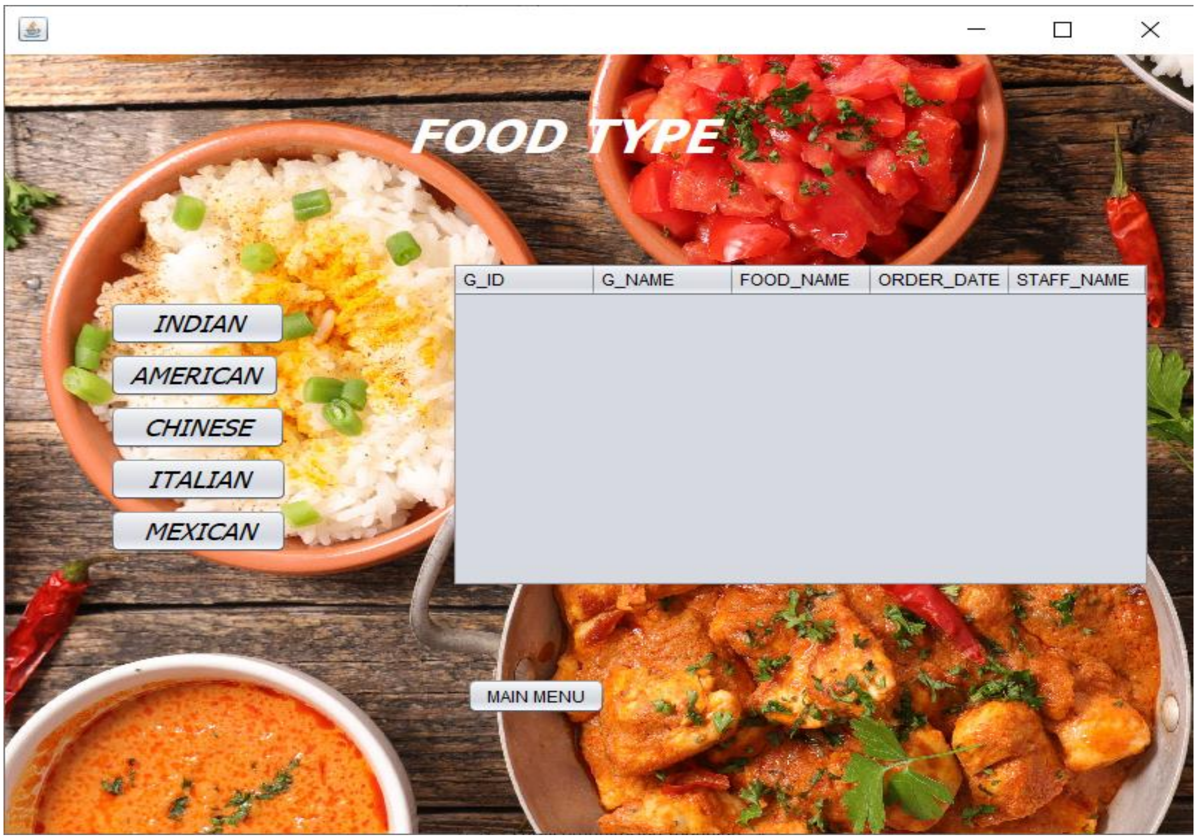
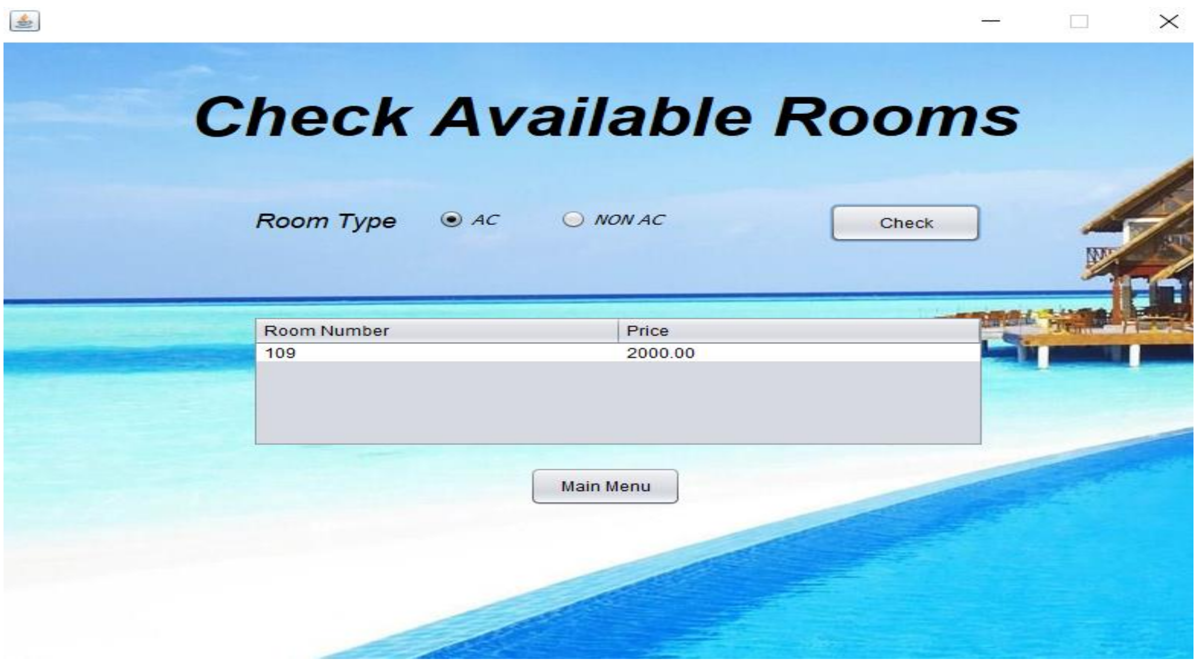


Figure 4.34 : food orders by type monitoring system frame

Queries

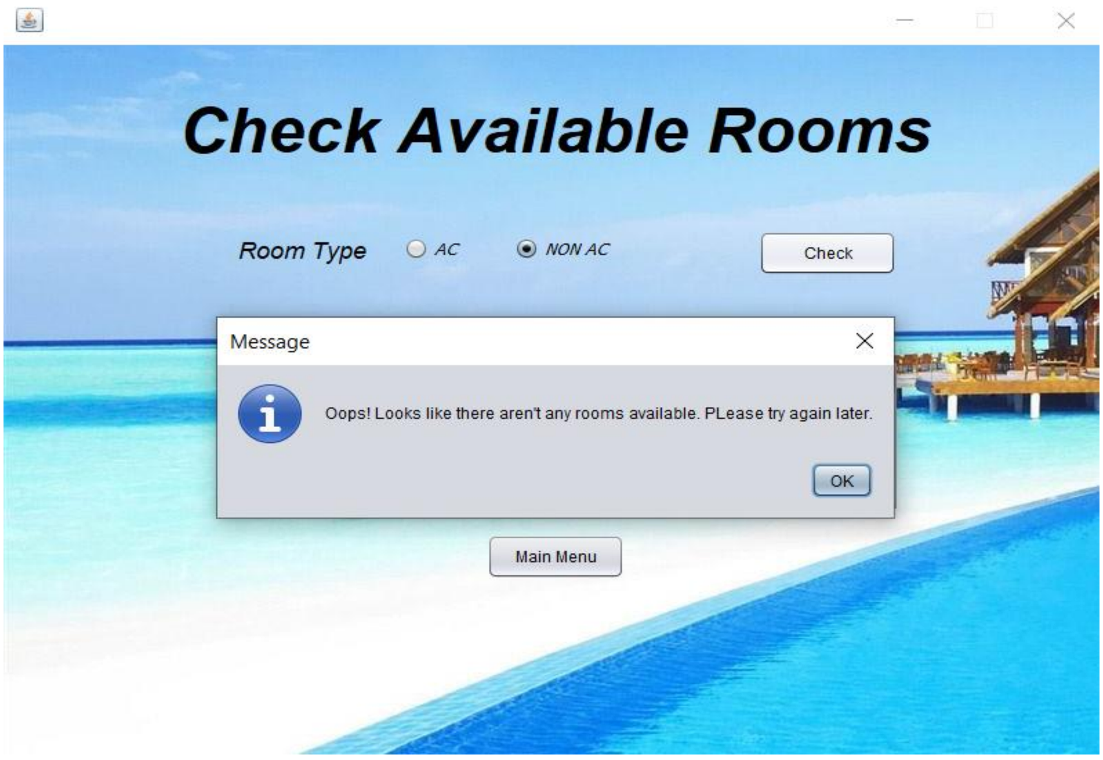
1. **Check available rooms (AC or NONAC)**
    a. **AC**



Room Number	Price
109	2000.00

Figure 4.35 : check available rooms (AC) output frame

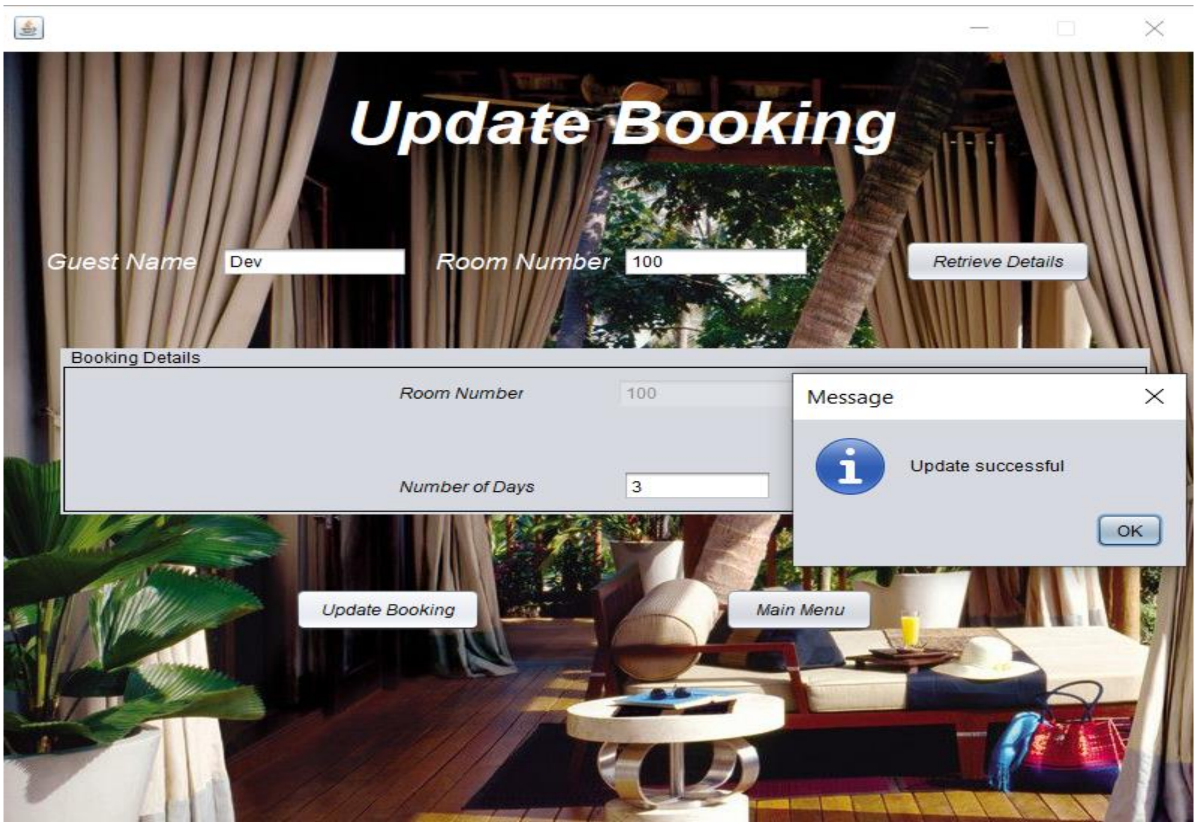
    b. **NON AC**



Oops! Looks like there aren't any rooms available. PLease try again later.

Figure 4.36 : check available rooms(NONAC) output frame

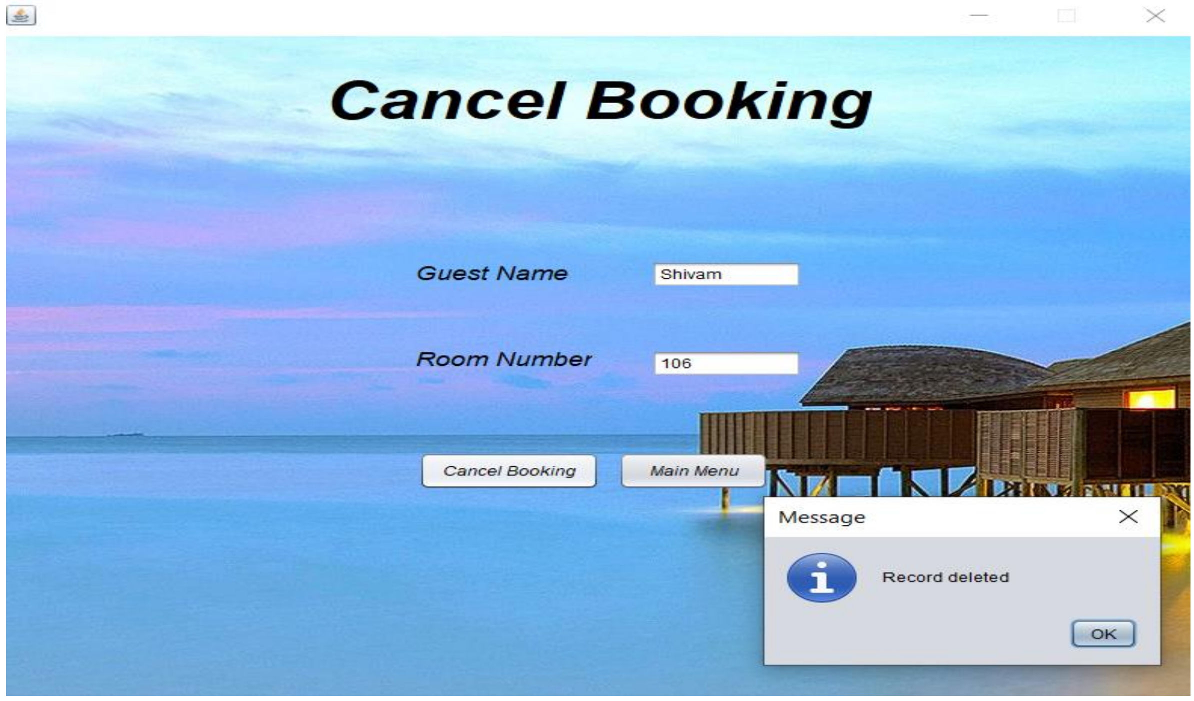
2. a. Update Booking for a room



Guest Name: Dev
Room Number: 100
Room Number: 100
Number of Days: 3
Update successful

Figure 4.37 : update booking output frame

b. Cancel Booking for a room

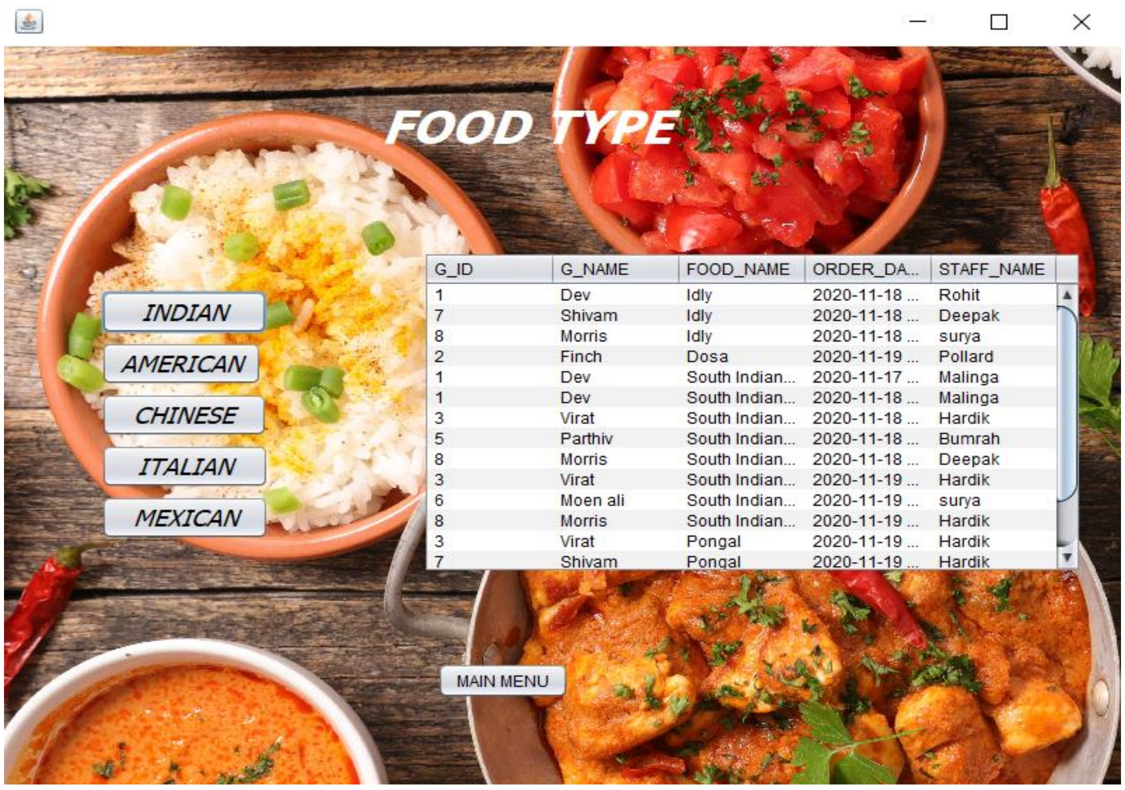


Guest Name: Shivam
Room Number: 106
Record deleted

Figure 4.38 : cancel booking output frame

3. **Display the guest name who has ordered :**
    iv. **Indian Food/American/Chinese/Italian/Mexican**

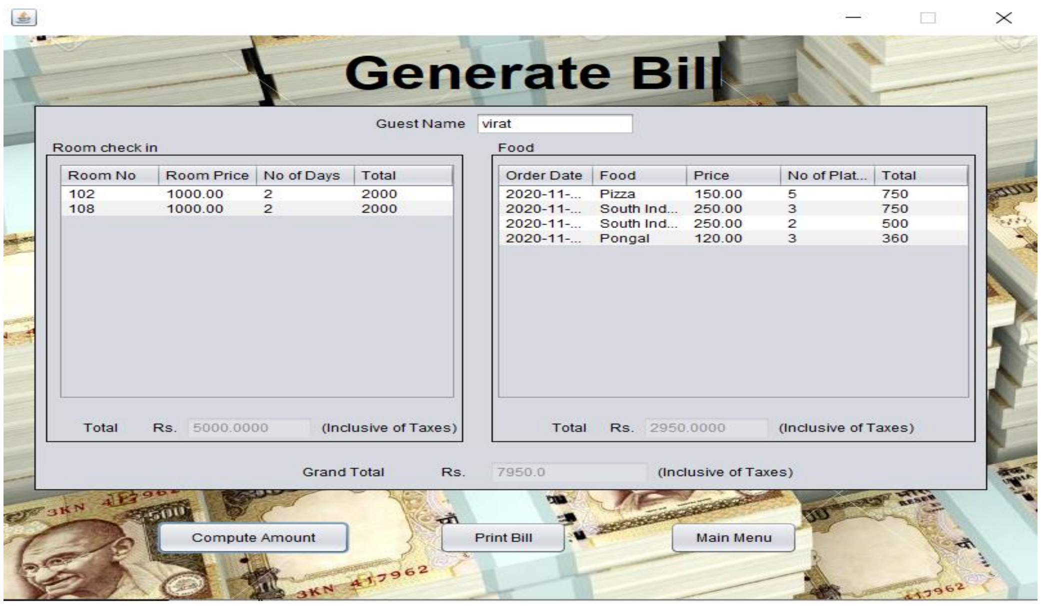
Indian:



G_ID	G_NAME	FOOD_NAME	ORDER_DA...	STAFF_NAME
1	Dev	Idly	2020-11-18 ...	Rohit
7	Shivam	Idly	2020-11-18 ...	Deepak
8	Morris	Idly	2020-11-18 ...	surya
2	Finch	Dosa	2020-11-19 ...	Pollard
1	Dev	South Indian...	2020-11-17 ...	Malinga
1	Dev	South Indian...	2020-11-18 ...	Malinga
3	Virat	South Indian...	2020-11-18 ...	Hardik
5	Parthiv	South Indian...	2020-11-18 ...	Bumrah
8	Morris	South Indian...	2020-11-18 ...	Deepak
3	Virat	South Indian...	2020-11-19 ...	Hardik
6	Moen ali	South Indian...	2020-11-19 ...	surya
8	Morris	South Indian...	2020-11-19 ...	Hardik
3	Virat	Pongal	2020-11-19 ...	Hardik
7	Shivam	Pongal	2020-11-19 ...	Hardik

Figure 4.39 : food order monitoring by food type output frame

4. **Generating room bill**
5. **Generating food bill and computing total**



Guest Name: virat

Room check in

Room No	Room Price	No of Days	Total
102	1000.00	2	2000
108	1000.00	2	2000

Total Rs. 5000.0000 (Inclusive of Taxes)

Food

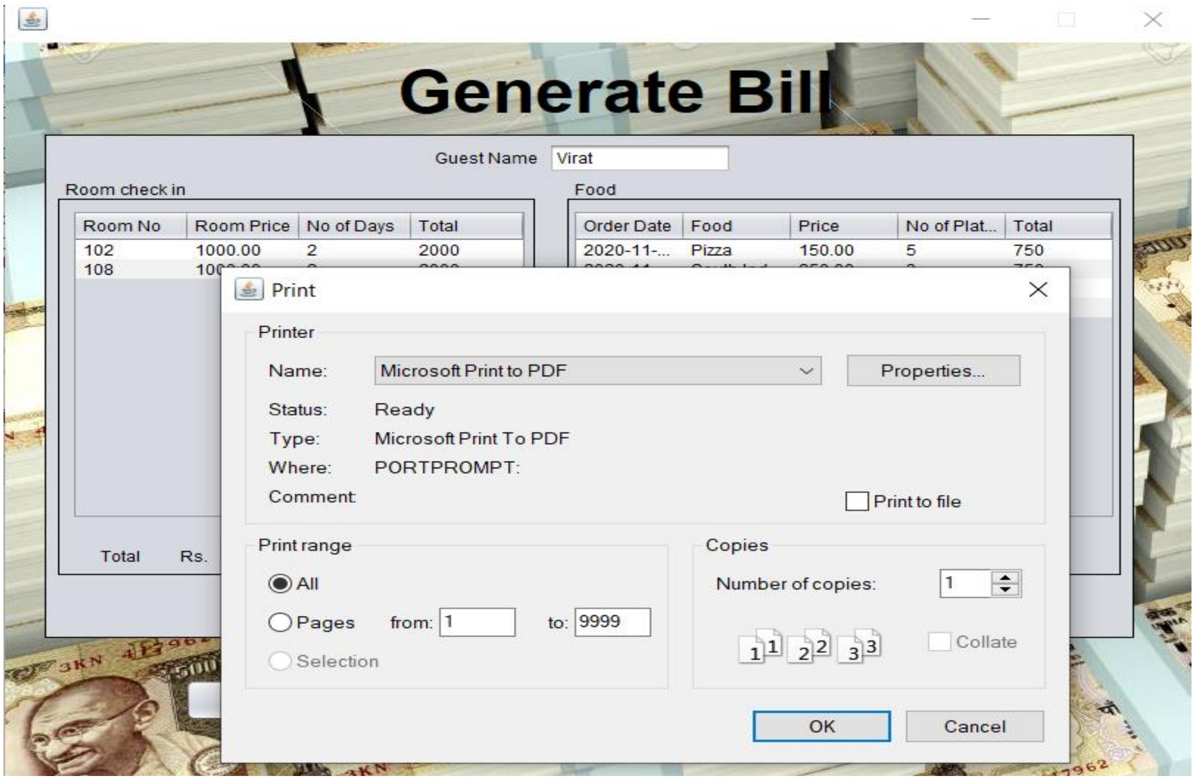
Order Date	Food	Price	No of Plat...	Total
2020-11-...	Pizza	150.00	5	750
2020-11-...	South Ind...	250.00	3	750
2020-11-...	South Ind...	250.00	2	500
2020-11-...	Pongal	120.00	3	360

Total Rs. 2950.0000 (Inclusive of Taxes)

Grand Total Rs. 7950.0 (Inclusive of Taxes)

Figure 4.40 : generate foodbill and roombill output frame

- An extra feature of our application is that we generate a pdf of bill which is printable.



Guest Name: Virat

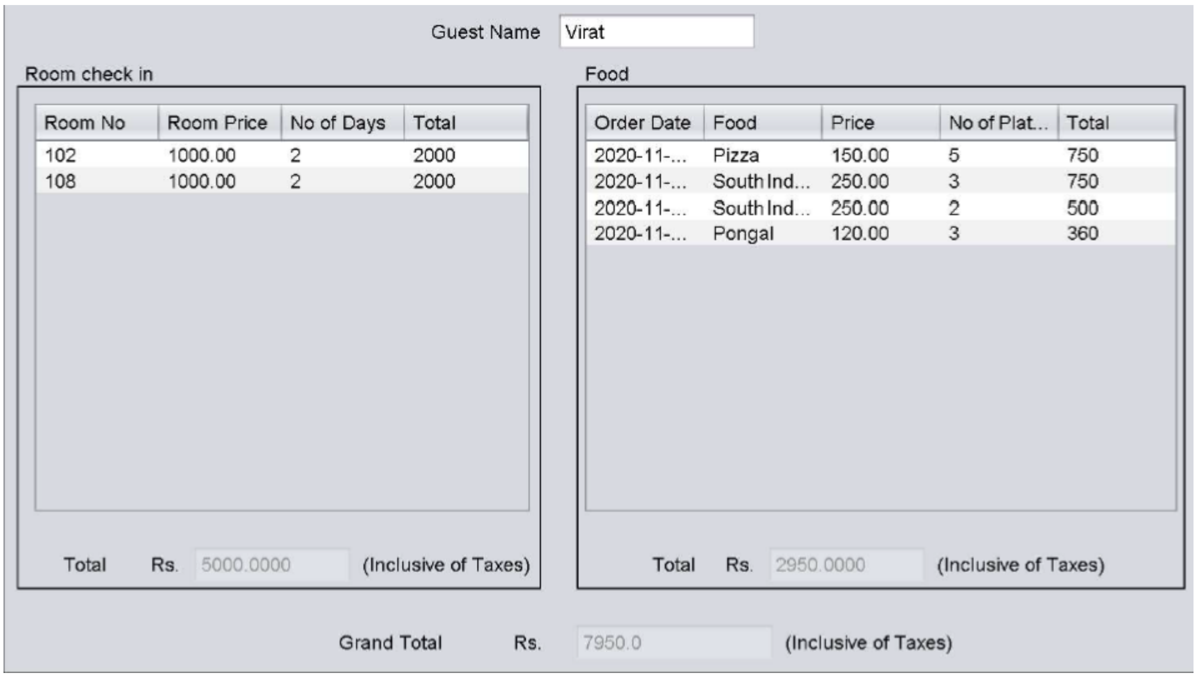
Printer
Name: Microsoft Print to PDF
Status: Ready
Type: Microsoft Print To PDF
Where: PORTPROMPT:
Comment:
Print to file: ☐

Print range: All [✓]; Pages from: 1 to: 9999; Selection

Copies: Number of copies: 1; Collate ☐

Figure 4.41 : window download pdf of generated bill

Receipt:



Guest Name: Virat

Room check in

Room No	Room Price	No of Days	Total
102	1000.00	2	2000
108	1000.00	2	2000

Total Rs. 5000.0000 (Inclusive of Taxes)

Food

Order Date	Food	Price	No of Plat...	Total
2020-11-...	Pizza	150.00	5	750
2020-11-...	South Ind...	250.00	3	750
2020-11-...	South Ind...	250.00	2	500
2020-11-...	Pongal	120.00	3	360

Total Rs. 2950.0000 (Inclusive of Taxes)

Grand Total Rs. 7950.0 (Inclusive of Taxes)

Figure 4.42 : sample receipt

CONCLUSION

We have successfully implemented RESORT DATABASE MANAGEMENT which helps us in simulating of tasks performed by the reception in a resort. Tables are used to display all the components of an entity or relationship so that user can see all the components of a particular type in one shot. One can just select the component and modify or remove the component. We have successfully used various functionalities of JAVA and SQL and created the fully functional database management system for resort.

Features:

1. Clean separation of various components to facilitate easy modification and revision.
2. Facilitates easy modification since all the data is maintained in a separate file.
3. Clean structure and maintenance of data manipulation operations. All the data required for different operations is kept in a separate file.
4. Quick and easy saving and loading of database file.

REFERENCES

- MySQL cheat sheet: https://www.mysqltutorial.org/mysql-cheat-sheet.aspx
- Netbeans IDE documentation: https://netbeans.org/kb/index.html
- Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
- Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill
- W3Schools: https://www.w3schools.com/sql/