Database Management System

PROJECT TITLE :

RESORT MANAGEMENT SYSTEM

TEAM : (G – section)
RAKESH C (PES2UG22CS431)
PUNEETH BU (PES2UG22CS419)

CHAPTER-1

INTRODUCTION

1.1 OBJECTIVES:

- The main objective of the project is to design and develop a user friendly-system Easy to use and an efficient computerized system.
- To develop an accurate and flexible system, for booking and cancelling the rooms and other needs present in the resort
- To study the functioning of Resort management System.
- To make a software fast in processing, with good user interface.
- To make software with good user interface so that user can change it and it should be used for a long time without error and maintenance.
- Computerization can be helpful as a means of saving time and money for the customer by allowing them do booking in just call .
- To provide better Graphical User Interface (GUI).
- In order to Provides Security ,used login and password method to secure the information about the customer details.
- Improving arrangements for customer coordination.
- Reducing paperwork.

1.2 LIMITATIONS:

- Time consumption in data entry as the records are to be manually entered by admin.

CHAPTER-2 **Resort Management System Case Study**

This case study focuses on designing and implementing a **Resort Management System** using Python, MySQL Workbench, and Tkinter. The system is intended to handle core management functions such as booking, cancellation, and viewing room availability.And all this is maintained by the admin to add rooms and any new features available in the Resort.

CHAPTER 3 3. DATABASE DESIGN

3.1 SOFTWARE REQUIREMENTS SPECIFICATION

3.1.2 **System Requirements:**

1. Software Requirements:

- **Python 3.x**: For developing backend logic.
- **MySQL Workbench**: For database design and management.
- **Tkinter**: For building the graphical user interface (GUI).

2. Functional Requirements:

- **Booking Management**:
  - Add new room bookings.
  - Check room availability before booking.
  - Generate booking confirmation and save booking details to the database.
- **Cancellation Management**:
  - Allow users to cancel existing bookings through the admin coordination.
  - Update room availability upon cancellation by the admin.
- **Customer Management**:
  - Maintain customer records.
- **Room Management**:
  - Manage room details, including type, price, and availability status.

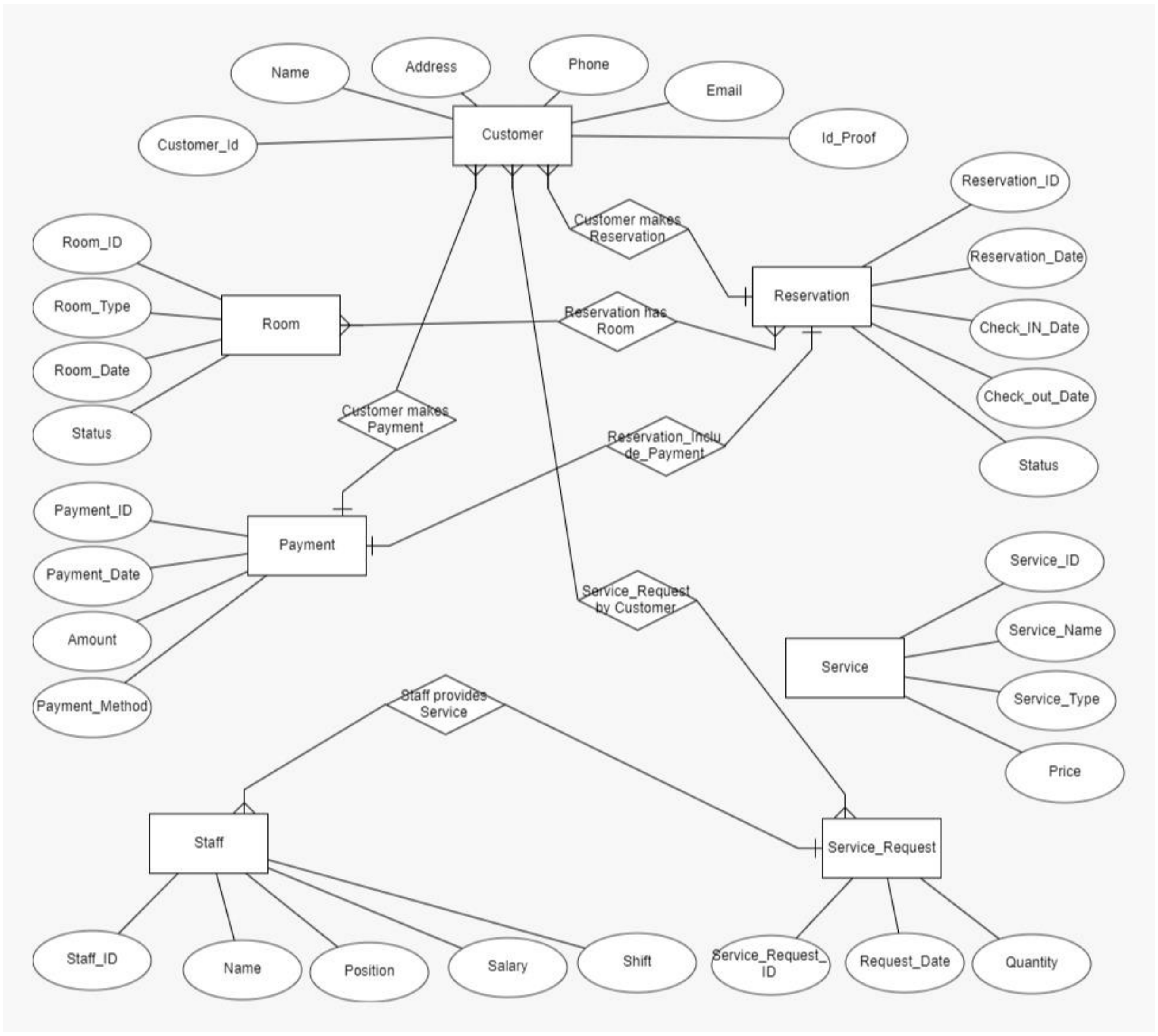
Minimum Hardware Requirements:

1. **Processor**:
   - **Intel Core i3** or equivalent (or higher)
   - Dual-core processor for handling basic tasks smoothly
2. **Memory (RAM)**:
   - **4 GB RAM** (sufficient for running Python scripts, MySQL Workbench, and Tkinter GUI without lag)
3. **Storage**:
   - **256 GB HDD** or **128 GB SSD** (for faster read/write speeds)
   - Enough space to store MySQL database files and project-related resources.
4. **Graphics**:
   - Integrated graphics are sufficient since the project mainly involves text-based and GUI-based applications.
5. **Operating System**:

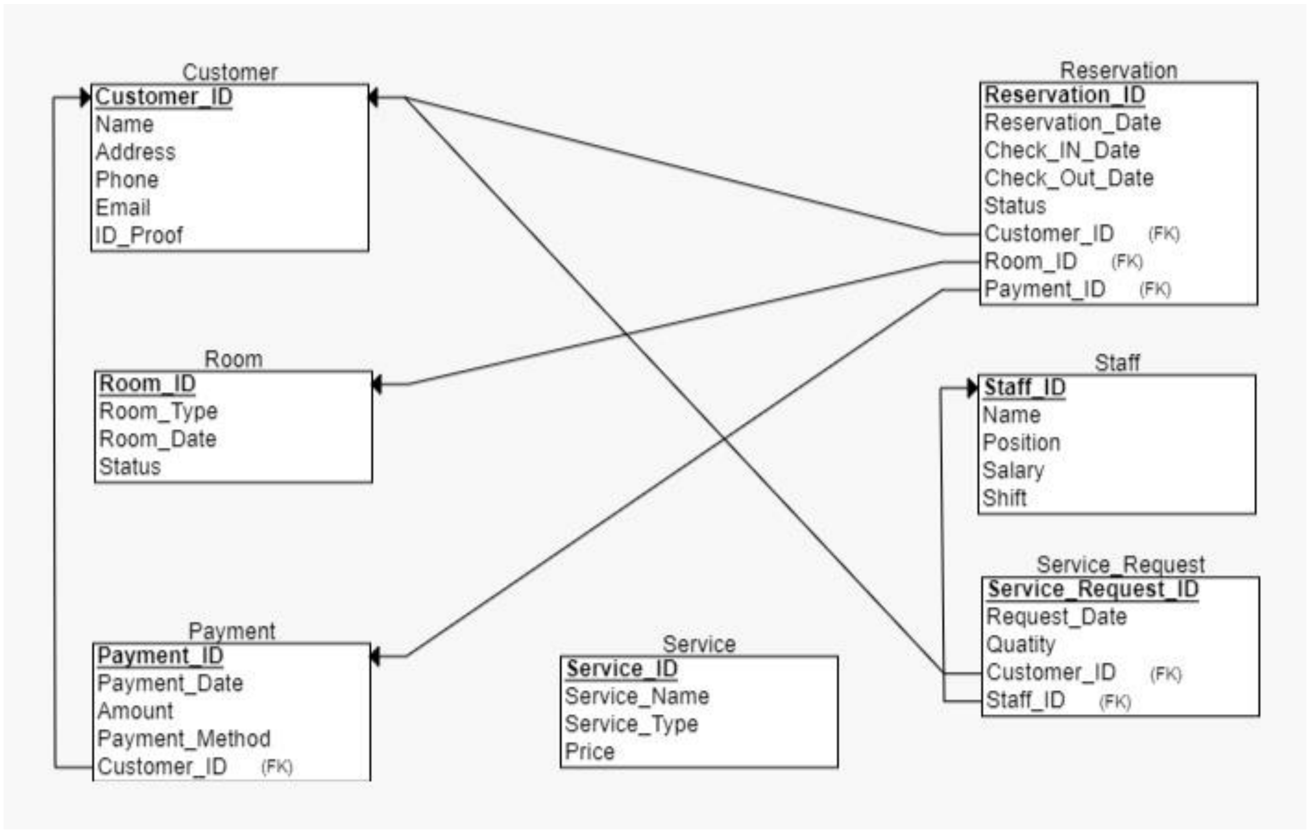
- **Windows 10/11**, **macOS**, or a **Linux distribution** (e.g., Ubuntu) that supports Python, MySQL, and Tkinter.

3.2 CONCEPTUAL DESIGN:

3.2.1 E-R DIAGRAM:



3.2.2 SCHEMA DIAGRAM:



Resort Management System Implementation Guide

1. System Setup and Configuration

- **Database Installation**:
  - Set up MySQL Workbench and configure a new database called resort_management where all tables and data related to customer management, room booking, and cancellations will be stored.
- **Python Environment Setup**:
  - Ensure Python is installed with necessary libraries for database connections (e.g., mysql-connector-python) and GUI development (Tkinter).
  - Configure environment variables and install dependencies as required.

---

2. Database Design

Define tables for core entities:

- **Customers Table**: Stores customer information, including personal details like name, email, and phone number.
- **Rooms Table**: Contains information on room types, prices, and availability status, allowing quick lookups for available rooms.
- **Bookings Table**: Records each booking's details, including the customer ID, room ID, check-in/check-out dates, and total cost.

- **Cancellations Table**: Stores records of canceled bookings, linking back to the Bookings table for tracking purposes.

3. Backend Logic Development

Customer Management:

- Create functions to add new customers to the system, search for existing customer details, and retrieve customer information for booking.

Room Availability:

- Implement functionality to check for room availability based on room type and the selected dates.
- Ensure a status update in the Rooms table when a room is booked or becomes available after a cancellation.

Booking Management:

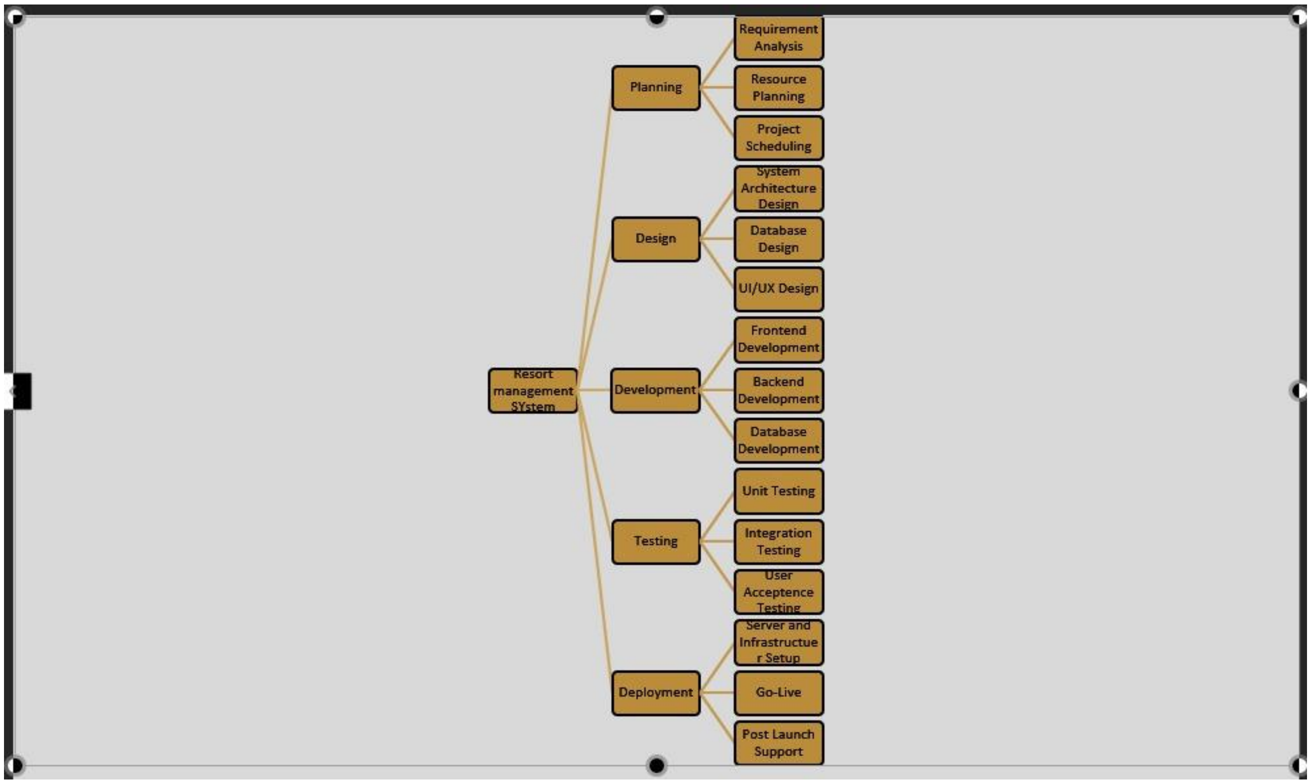
- Define the booking flow:
  - Validate customer details.
  - Confirm room availability.
  - Register the booking, update the room's availability status, and calculate the total cost.
- Handle **cancellation** requests by marking bookings as canceled and updating the room's availability status.

4. GUI Design with Tkinter

Design Principles:

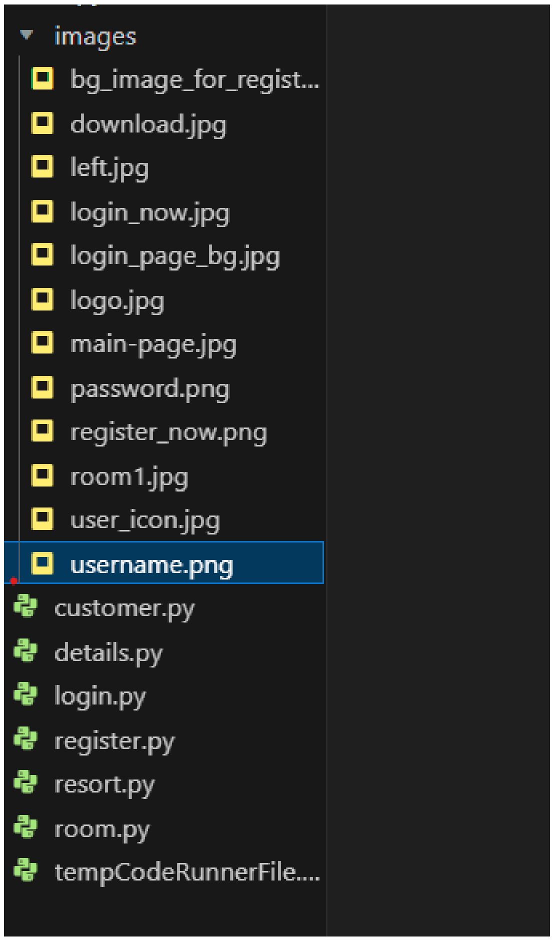
- Use Tkinter to create a user-friendly interface for resort management system.
- Split the interface into multiple screens or frames:
  - **Login Screen**: For secure access.
  - **Customer Registration**: Allows adding new customer information by the admin.
  - **Booking Interface**: Provides options to check room availability, input booking details.
  - **Cancellation Interface**: Allows retrieval of booking details and cancels them if required, But through contacting the admin.

Work breakdown structure



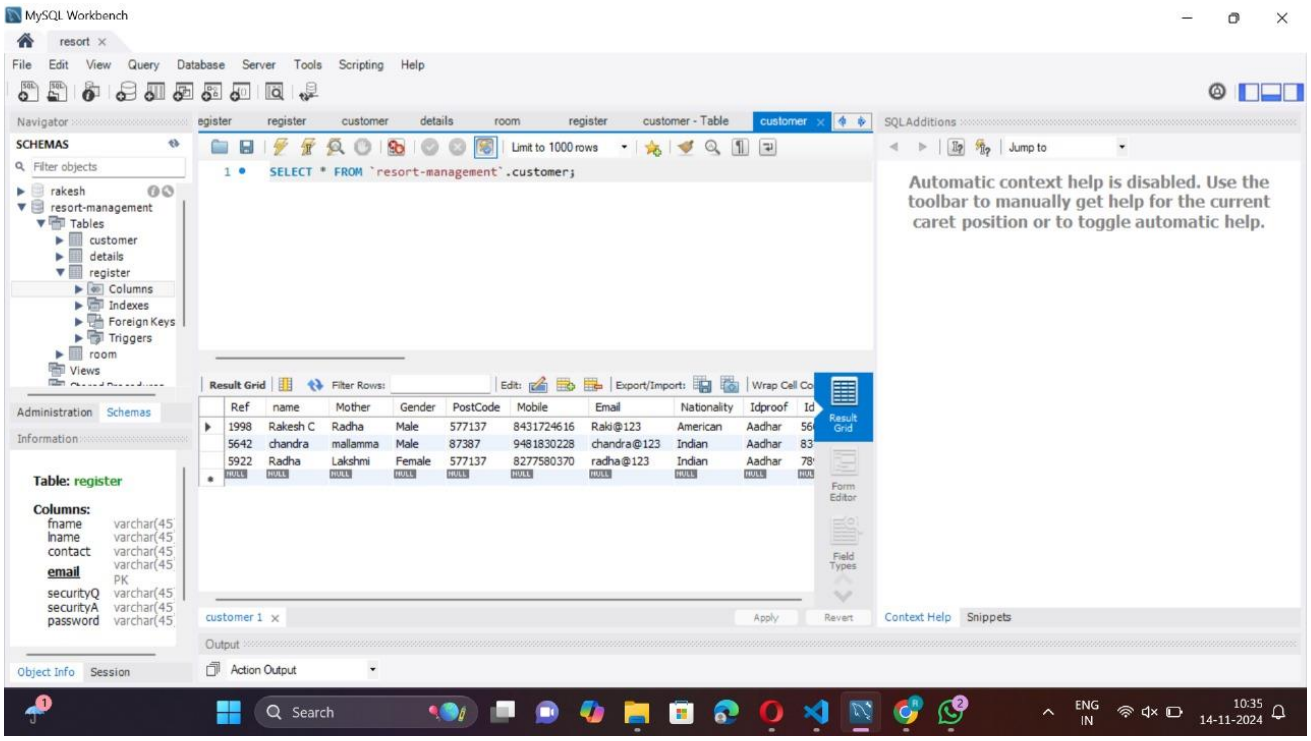
Frontend :

For better view of web the images downloaded and stored in the same folder


```
images
  bg_image_for_regist...
  download.jpg
  left.jpg
  login_now.jpg
  login_page_bg.jpg
  logo.jpg
  main-page.jpg
  password.png
  register_now.png
  room1.jpg
  user_icon.jpg
  username.png
customer.py
details.py
login.py
register.py
resort.py
room.py
tempCodeRunnerFile....
```

The workbench database looks like this

Registrations



```
SELECT * FROM `resort-management`.customer;
```

	Ref	name	Mother	Gender	PostCode	Mobile	Email	Nationality	Idproof	Id
▶	1998	Rakesh C	Radha	Male	577137	8431724616	Raki@123	American	Aadhar	56
	5642	chandra	mallamma	Male	87387	9481830228	chandra@123	Indian	Aadhar	83
	5922	Radha	Lakshmi	Female	577137	8277580370	radha@123	Indian	Aadhar	78
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

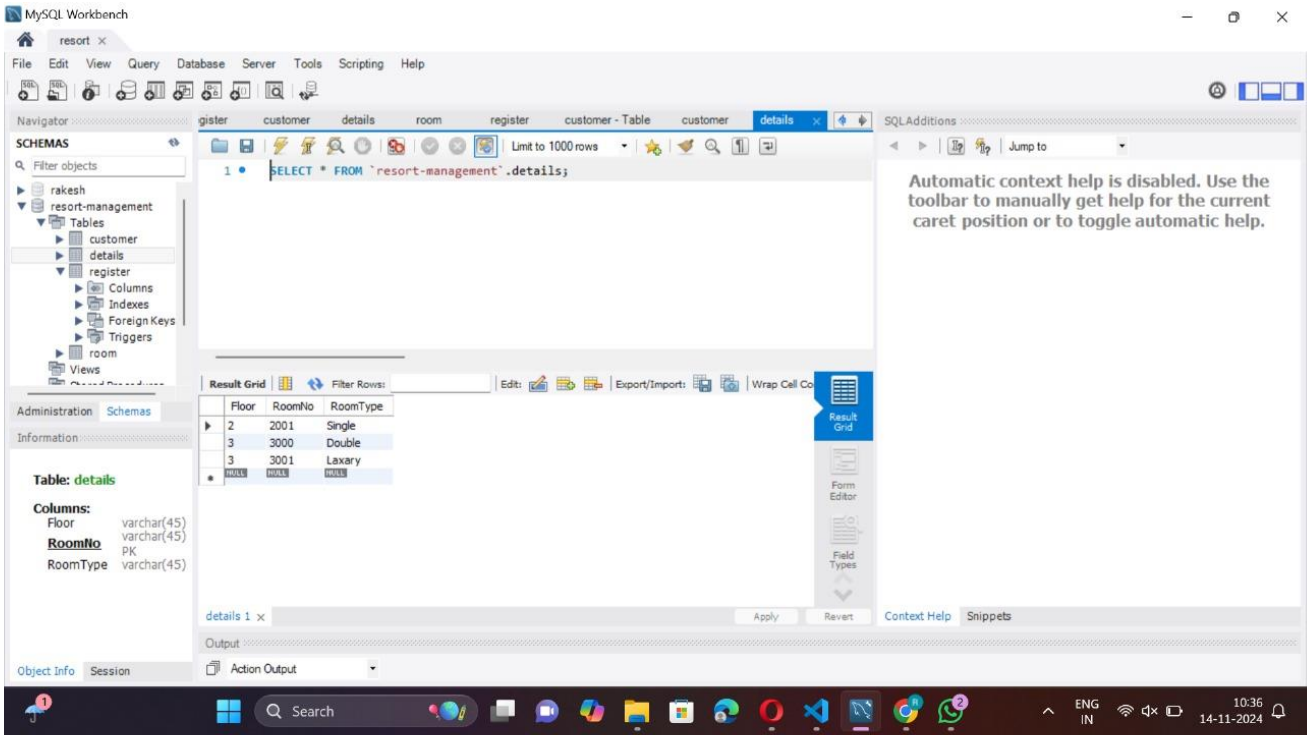
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: register

Columns:

fname	varchar(45
lname	varchar(45
contact	varchar(45
email	varchar(45 PK
securityQ	varchar(45
securityA	varchar(45
password	varchar(45

Rooms availability in Resort



```
SELECT * FROM `resort-management`.details;
```

	Floor	RoomNo	RoomType
▶	2	2001	Single
	3	3000	Double
	3	3001	Laxary
*	NULL	NULL	NULL

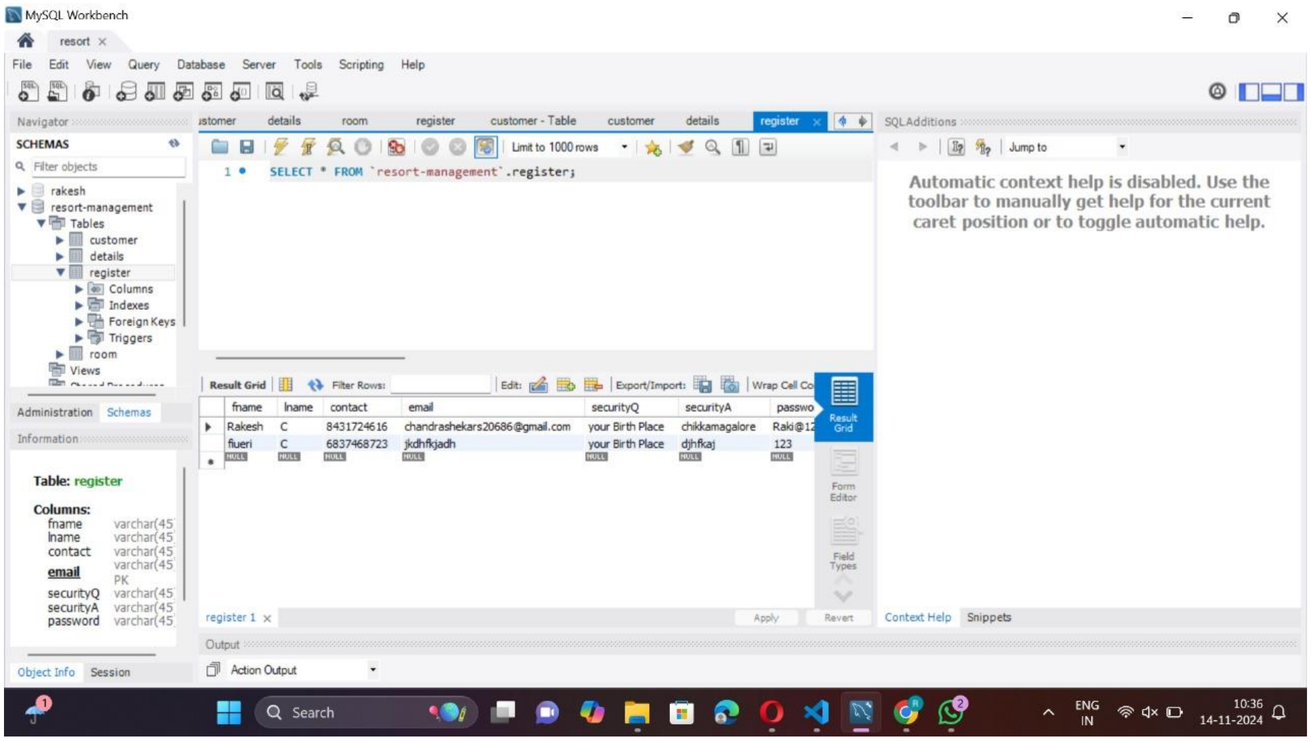
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: details

Columns:

Floor	varchar(45)
RoomNo	varchar(45) PK
RoomType	varchar(45)

Regestration details stored in this format



```
SELECT * FROM `resort-management`.register;
```

fname	lname	contact	email	securityQ	securityA	passwo
Rakesh	C	8431724616	chandrashekars20686@gmail.com	your Birth Place	chikkamagalore	Raki@12
fiueri	C	6837468723	jkdhfkjadh	your Birth Place	djhfkaj	123
NULL	NULL	NULL	NULL	NULL	NULL	NULL

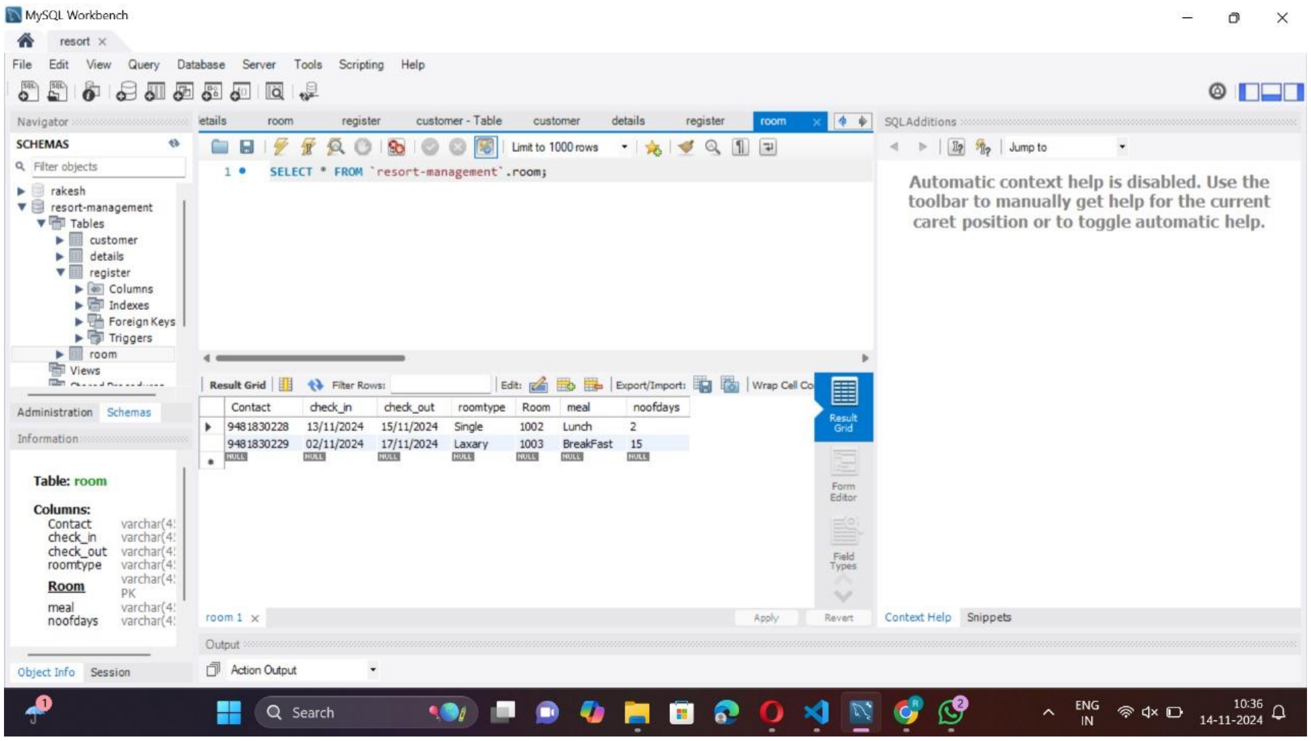
Table: register

Columns:

fname	varchar(45
lname	varchar(45
contact	varchar(45
email	varchar(45 PK
securityQ	varchar(45
securityA	varchar(45
password	varchar(45

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Booking details stored in this format



```
SELECT * FROM `resort-management`.room;
```

Contact	check_in	check_out	roomtype	Room	meal	noofdays
9481830228	13/11/2024	15/11/2024	Single	1002	Lunch	2
9481830229	02/11/2024	17/11/2024	Laxary	1003	BreakFast	15
NULL	NULL	NULL	NULL	NULL	NULL	NULL

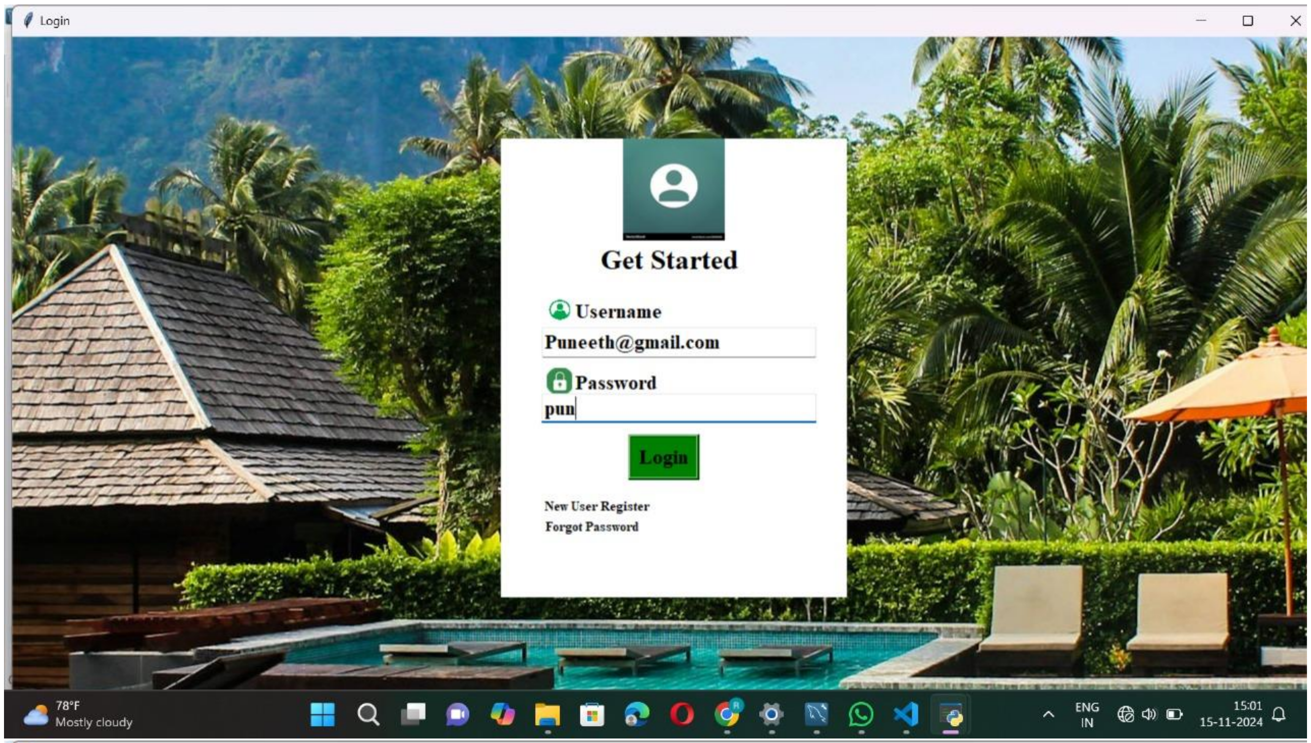
Table: room

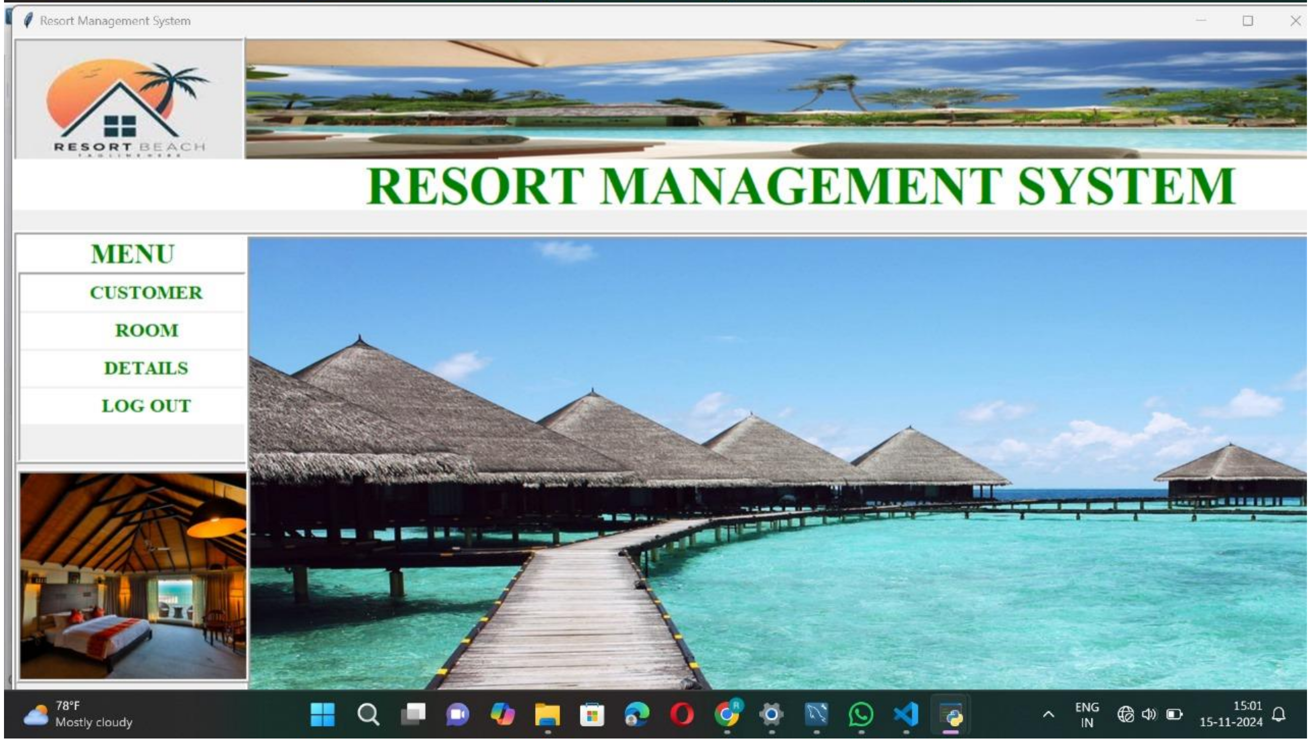
Columns:

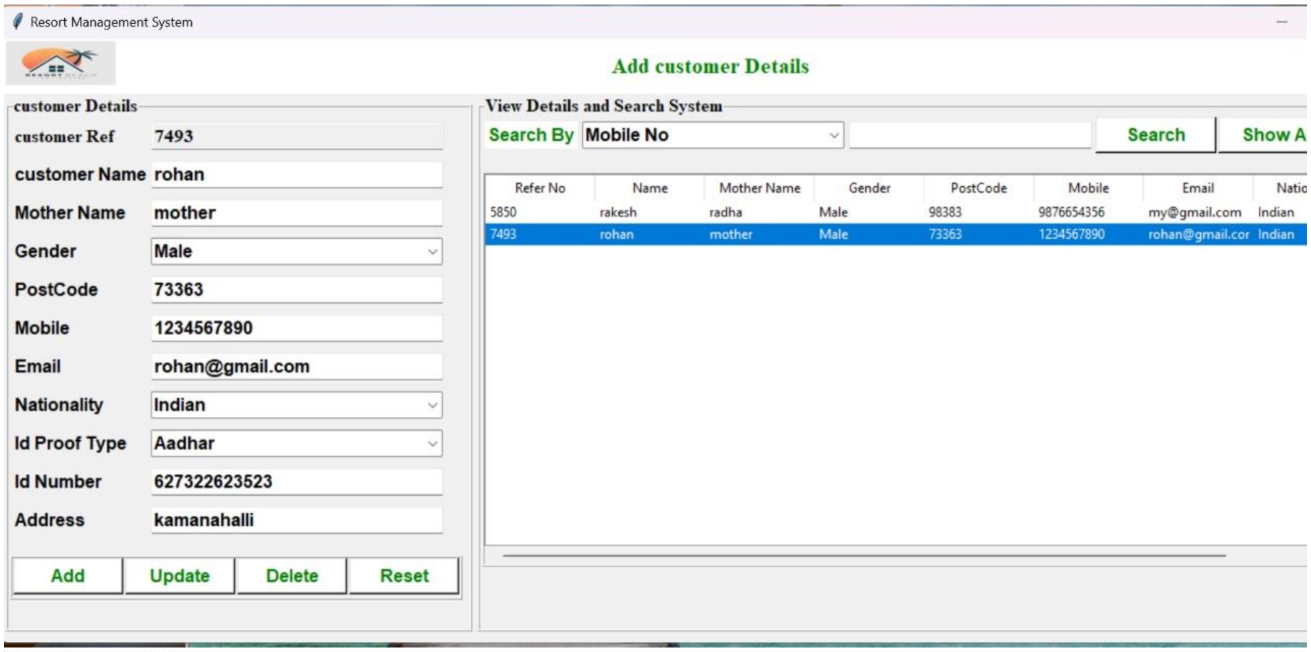
Contact	varchar(4:
check_in	varchar(4:
check_out	varchar(4:
roomtype	varchar(4:
Room	varchar(4: PK
meal	varchar(4:
noofdays	varchar(4:

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Frontend


Login
Get Started
Username
Puneeth@gmail.com
Password
pun
Login
New User Register
Forgot Password
78°F
Mostly cloudy
ENG
IN
15:01
15-11-2024


Resort Management System
RESORT BEACH
RESORT MANAGEMENT SYSTEM
MENU
CUSTOMER
ROOM
DETAILS
LOG OUT
78°F
Mostly cloudy
ENG
IN
15:01
15-11-2024


Resort Management System

Add customer Details

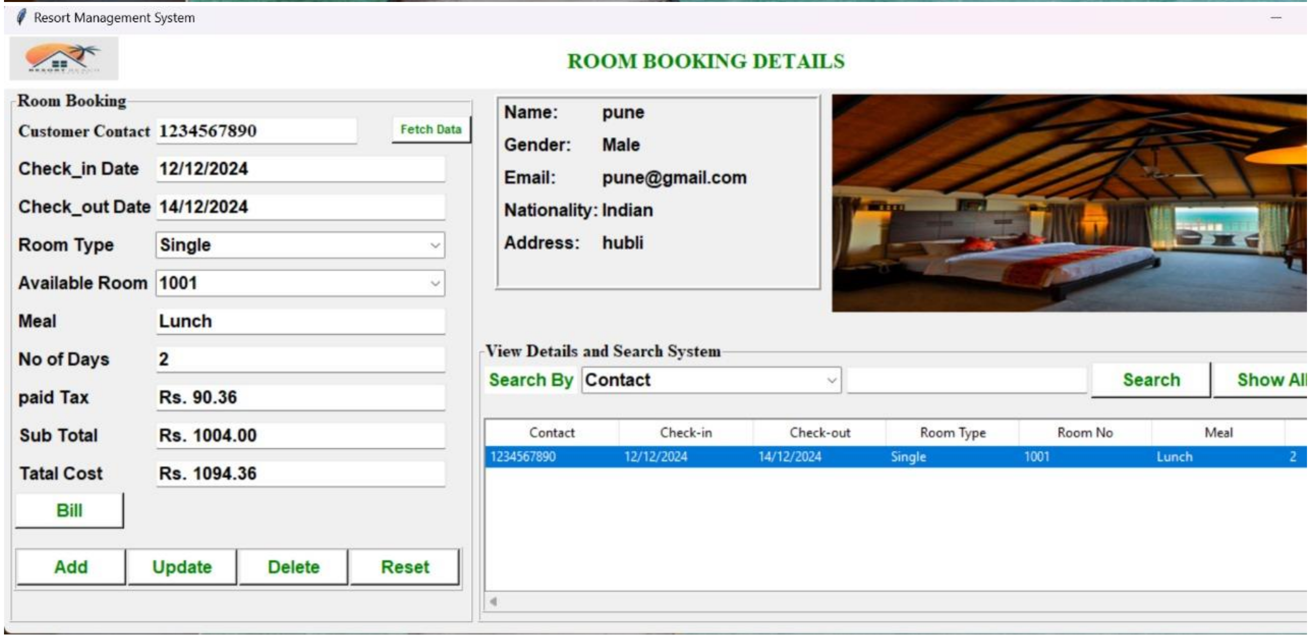
customer Details

customer Ref	7493
customer Name	rohan
Mother Name	mother
Gender	Male
PostCode	73363
Mobile	1234567890
Email	rohan@gmail.com
Nationality	Indian
Id Proof Type	Aadhar
Id Number	627322623523
Address	kamanahalli

View Details and Search System

Search By Mobile No

Refer No	Name	Mother Name	Gender	PostCode	Mobile	Email	Natio
5850	rakesh	radha	Male	98383	9876654356	my@gmail.com	Indian
7493	rohan	mother	Male	73363	1234567890	rohan@gmail.cor	Indian


Resort Management System

ROOM BOOKING DETAILS

Room Booking

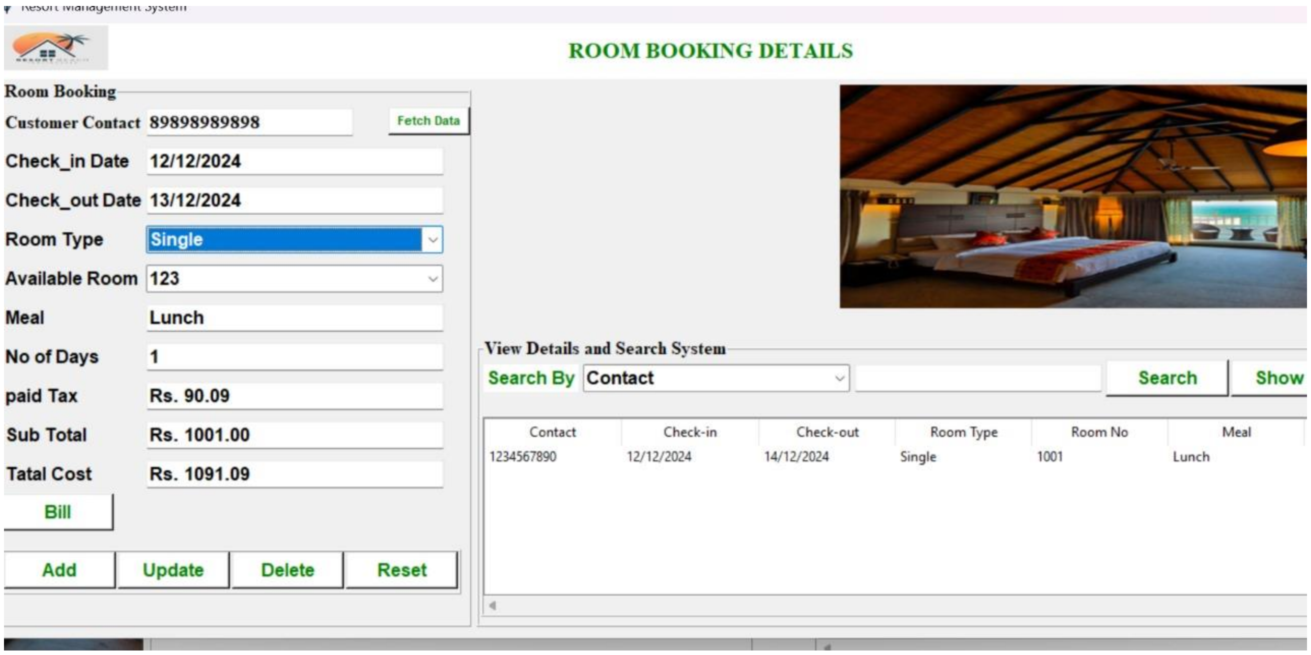
Customer Contact	1234567890
Check_in Date	12/12/2024
Check_out Date	14/12/2024
Room Type	Single
Available Room	1001
Meal	Lunch
No of Days	2
paid Tax	Rs. 90.36
Sub Total	Rs. 1004.00
Tatal Cost	Rs. 1094.36

Name:	pune
Gender:	Male
Email:	pune@gmail.com
Nationality:	Indian
Address:	hubli

View Details and Search System

Search By Contact

Contact	Check-in	Check-out	Room Type	Room No	Meal	
1234567890	12/12/2024	14/12/2024	Single	1001	Lunch	2


Resort Management System

ROOM BOOKING DETAILS

Room Booking

Customer Contact	89898989898
Check_in Date	12/12/2024
Check_out Date	13/12/2024
Room Type	Single
Available Room	123
Meal	Lunch
No of Days	1
paid Tax	Rs. 90.09
Sub Total	Rs. 1001.00
Tatal Cost	Rs. 1091.09

View Details and Search System

Search By: Contact

Contact	Check-in	Check-out	Room Type	Room No	Meal
1234567890	12/12/2024	14/12/2024	Single	1001	Lunch

Code :

Resort.py

```
from tkinter import*
from PIL import Image,ImageTk
from customer import Cust_Win
from  room import Roombooking
from details import DetailsRoom


class ResortManagementSystem:
    def __init__(self,root):
        self.root=root
        self.root.title("Resort Management System")
        self.root.geometry("1550x800+0+0")


        img1=Image.open(r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-management-
system\images\download.jpg")
        img1 = img1.resize((1550, 140), Image.LANCZOS)


        self.photoimg1=ImageTk.PhotoImage(img1)


        lblimg = Label(self.root,image=self.photoimg1, bd=4, relief=RIDGE)
        lblimg.place(x=0,y=0,width=1550,height=140)
        img2=Image.open(r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-management-
system\images\logo.jpg")
        img2 = img2.resize((230, 140), Image.LANCZOS)

        self.photoimg2=ImageTk.PhotoImage(img2)

        lblimg = Label(self.root,image=self.photoimg2, bd=4, relief=RIDGE)
        lblimg.place(x=0,y=0,width=230,height=140)



        lbl_title=Label(self.root,text="RESORT MANAGEMENT SYSTEM",font=("times new
roman",40,"bold"),bg="white",fg="green")
        lbl_title.place(x=0,y=120,width=1550,height=50)

        main_frame=Frame(self.root,bd=4,relief=RIDGE)
        main_frame.place(x=0,y=190,width=1550,height=620)

        lbl_menu=Label(main_frame,text="MENU",font=("times new
roman",20,"bold"),bg="white",fg="green")
        lbl_menu.place(x=0,y=0,width=230)

        btn_frame=Frame(main_frame,bd=4,relief=RIDGE)
        btn_frame.place(x=0,y=35,width=228,height=190)

        cust_btn=Button(btn_frame,text="CUSTOMER",command=self.Cust_details,width=22,font=("ti
mes new roman",14,"bold"),bg="white",fg="green",bd=0,cursor="hand1")
        cust_btn.grid(row=0,column=0,pady=1)

        room_btn=Button(btn_frame,text="ROOM",command=self.Roombooking,width=22,font=("times
new roman",14,"bold"),bg="white",fg="green",bd=0,cursor="hand1")
        room_btn.grid(row=1,column=0,pady=1)

        details_btn=Button(btn_frame,text="DETAILS",command=self.DetailsRoom,width=22,font=("ti
mes new roman",14,"bold"),bg="white",fg="green",bd=0,cursor="hand1")
        details_btn.grid(row=2,column=0,pady=1)

        report_btn=Button(btn_frame,text="REPORT",width=22,font=("times new
roman",14,"bold"),bg="white",fg="green",bd=0,cursor="hand1")
        report_btn.grid(row=3,column=0,pady=1)
        logout_btn=Button(btn_frame,text="LOG OUT",width=22,font=("times new
roman",14,"bold"),bg="white",fg="green",bd=0,cursor="hand1")
        logout_btn.grid(row=4   ,column=0,pady=1)


        img3=Image.open(r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-management-
system\images\main-page.jpg")
        img3 = img3.resize((1310, 590), Image.LANCZOS)


        self.photoimg3=ImageTk.PhotoImage(img3)


        lblimg1 = Label(main_frame,image=self.photoimg3, bd=4, relief=RIDGE)
        lblimg1.place(x=225,y=0,width=1310,height=590)


        img4=Image.open(r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-management-
system\images\room1.jpg")
        img4 = img4.resize((230, 210), Image.LANCZOS)


        self.photoimg4=ImageTk.PhotoImage(img4)


        lblimg2 = Label(main_frame,image=self.photoimg4, bd=4, relief=RIDGE)
        lblimg2.place(x=0,y=230,width=230,height=210)


    def Cust_details(self):
        self.new_window=Toplevel(self.root)
        self.app=Cust_Win(self.new_window)


    def Roombooking(self):
        self.new_window=Toplevel(self.root)
        self.app=Roombooking(self.new_window)


    def DetailsRoom(self):
        self.new_window=Toplevel(self.root)
        self.app=DetailsRoom(self.new_window)


if __name__ == "__main__":
    root=Tk()
    obj=ResortManagementSystem(root)
    root.mainloop()
```

Register.py

```
from tkinter import*
from tkinter import ttk
from PIL import Image,ImageTk
from tkinter import messagebox
import mysql.connector

class Register:
    def __init__(self,root):
        self.root=root
        self.root.title("Register")
        self.root.geometry("1600x900+0+0")

        #==============variables==========
        self.var_fname = StringVar()
        self.var_lname = StringVar()
        self.var_contact = StringVar()
        self.var_email = StringVar()
        self.var_securityQ = StringVar()
        self.var_SecurityA = StringVar()
        self.var_pass = StringVar()
        self.var_confpass = StringVar()

#==============background image============
        self.bg=ImageTk.PhotoImage(file=r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-
management-system\images\bg_image_for_register.jpeg")
        bg_lbl=Label(self.root,image=self.bg)
        bg_lbl.place(x=0,y=0,relwidth=1,relheight=1)

        #==============left image==============
        self.bg1=ImageTk.PhotoImage(file=r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-
management-system\images\left.jpg")
        left_lbl=Label(self.root,image=self.bg1)
        left_lbl.place(x=50,y=100,width=470,height=550)

    #===========mail frame
        frame=Frame(self.root,bg="white")
        frame.place(x=520,y=100,width=800,height=550)
        #
        register_lbl=Label(frame,text="REGISTER HERE", font=("times new
roman",20,"bold"),fg="green")
        register_lbl.place(x=20,y=20)

        #============label and entry=============

        #==================row1
        fname=Label(frame,text="First Name",font=("times new
roman",15,"bold"),fg="black",bg="white")
        fname.place(x=50,y=100)

        fname_entry=ttk.Entry(frame,textvariable=self.var_fname,font=("times new roman",20,"bold"))
        fname_entry.place(x=50,y=130,width=250)

        l_name=Label(frame,text="Last Name",font=("times new
roman",15,"bold"),fg="black",bg="white")
        l_name.place(x=370,y=100)

        self.txt_lname=ttk.Entry(frame,textvariable=self.var_lname,font=("times new roman",15))
        self.txt_lname.place(x=370,y=130,width=250)

        #============row2=============
        contact=Label(frame,text="Contact No",font=("times new
roman",15,"bold"),bg="white",fg="black")
        contact.place(x=50,y=170)

        self.txt_contact=ttk.Entry(frame,textvariable=self.var_contact,font=("times new
roman",15,"bold"))
        self.txt_contact.place(x=50,y=200,width=250)



        email=Label(frame,text="Email",font=("times new roman",15,"bold"),fg="black",bg="white")
        email.place(x=370,y=170)



        self.txt_email=ttk.Entry(frame,textvariable=self.var_email,font=("times new roman",15,"bold"))
        self.txt_email.place(x=370,y=200,width=250)
        #==============row3=============
        security_Q=Label(frame,text="Select security question",font=("times new
roman",15,"bold"),fg="black",bg="white")
        security_Q.place(x=50,y=240)

        self.combo_Security_Q=ttk.Combobox(frame,textvariable=self.var_securityQ,font=("times new
roman",15,"bold"),state="readonly")
        self.combo_Security_Q["values"]=("select","your Birth Place","Your Girlsfriends name","your
pet name")
        self.combo_Security_Q.place(x=50,y=270,width=250)
        self.combo_Security_Q.current(0)

        security_A=Label(frame,text="Security Answer",font=("times new
roman",15,"bold"),bg="white",fg="black")
        security_A.place(x=370,y=240)

        self.txt_security=ttk.Entry(frame,textvariable=self.var_SecurityA,font=("times new
roman",15,"bold"))
        self.txt_security.place(x=370,y=270,width=250)


        #===========row4===================
        pswd=Label(frame,text="Password",font=("times new
roman",15,"bold"),fg="black",bg="white")
        pswd.place(x=50,y=310)

        self.txt_pswd=ttk.Entry(frame,textvariable=self.var_pass,font=("times new roman",15,"bold"))
        self.txt_pswd.place(x=50,y=340,width=250)

        confirm_pswd=Label(frame,text=" Confirm Password",font=("times new
roman",15,"bold"),fg="black",bg="white")
        confirm_pswd.place(x=370,y=310)

        self.txt_confirm_pswd=ttk.Entry(frame,textvariable=self.var_confpass,font=("times new
roman",15,"bold"))
        self.txt_confirm_pswd.place(x=370,y=340,width=250)

        #============check button=============
        self.var_check=IntVar()
        checkbtn=Checkbutton(frame,variable=self.var_check,text="I agree the terms and
condition",font=("times new roman",12,"bold"),onvalue=1,offvalue=0)
        checkbtn.place(x=50,y=380)


        #=================buttons===========
        img=Image.open(r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-management-
system\images\register_now.png")
        img=img.resize((200,50),Image.LANCZOS)
        self.photoimage=ImageTk.PhotoImage(img)
        b1=Button(frame,image=self.photoimage,command=self.register_data,borderwidth=0,cursor="ha
nd2")
        b1.place(x=10,y=420,width=200,height=50)


        img1=Image.open(r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-management-
system\images\login_now.jpg")
        img1=img1.resize((200,50),Image.LANCZOS)
        self.photoimage1=ImageTk.PhotoImage(img1)
        b1=Button(frame,image=self.photoimage1,borderwidth=0,cursor="hand2")
        b1.place(x=330,y=420,width=200,height=50)




    def register_data(self):
        if self.var_fname.get()=="" or self.var_email.get()=="" or self.var_securityQ.get()=="Select":
            messagebox.showerror("Error","All fields are required")
        elif self.var_pass.get()!=self.var_confpass.get():
            messagebox.showerror("Error","Password and confirm password must be same")
        elif self.var_check.get()==0:
            messagebox.showerror("Error","Please Agree to our terms and conditions")
        else:
            conn=mysql.connector.connect(host="127.0.0.1",username="root",password="rakesh@123",da
tabase="resort-management")
            my_cursor=conn.cursor()
            query=("select * from register where email=%s")
            value=(self.var_email.get(),)
            my_cursor.execute(query,value)
            row=my_cursor.fetchone()
            if row!=None:
                messagebox.showerror("Error","User already exist please try another email")
            else:
```

```
        my_cursor.execute("insert into register values(%s,%s,%s,%s,%s,%s,%s)",(
                                                                   self.var_fname.get(),
                                                                   self.var_lname.get(),
                                                                   self.var_contact.get(),
                                                                   self.var_email.get(),
                                                                   self.var_securityQ.get(),
                                                                   self.var_SecurityA.get(),
                                                                   self.var_pass.get()
                                                                 ))
       conn.commit()
       conn.close()
       messagebox.showinfo("Success", "Registered Successfullt")


if __name__ == "__main__":
    root=Tk()
    app=Register(root)
    root.mainloop()
```

Room.py

```
from tkinter import*
from PIL import Image,ImageTk
from tkinter import ttk
import random
from time import strptime
from datetime import datetime
import mysql.connector
from tkinter import messagebox

class Roombooking:
      def __init__(self,root):
       self.root=root
       self.root.title("Resort Management System")
       self.root.geometry("1295x550+230+220")

       #===========variables==========
       self.var_contact=StringVar()
       self.var_CheckIn=StringVar()
```

```
        self.var_CheckOut=StringVar()
        self.var_RoomType=StringVar()
        self.var_RoomAvailable=StringVar()
        self.var_Meal=StringVar()
        self.var_NoOfDays=StringVar()
        self.var_PaidTax=StringVar()
        self.var_ActualTotal=StringVar()
        self.var_Total=StringVar()
        #==================title==============


        lbl_title=Label(self.root,text="ROOM BOOKING DETAILS ",font=("times new
roman",15,"bold"),bg="white",fg="green")
        lbl_title.place(x=0,y=0,width=1295,height=50)
        #==================logo=======================


        img2=Image.open(r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-management-
system\images\logo.jpg")
        img2 = img2.resize((100, 40), Image.LANCZOS)

        self.photoimg2=ImageTk.PhotoImage(img2)

        lblimg = Label(self.root,image=self.photoimg2, bd=0, relief=RIDGE)
        lblimg.place(x=5,y=2,width=100,height=40)
        #======================lableframe====================


        labelframeleft=LabelFrame(self.root,bd=2,relief=RIDGE,text="Room Booking",font=("times new
roman",12,"bold"),padx=2)
        labelframeleft.place(x=5,y=50,width=425,height=490)



        #==========labels and entrys==============#


        #constomer contact
        lbl_cust_contact=Label(labelframeleft,text="Customer Contact",font=("times new
roman",12,"bold"),padx=2,pady=6)
        lbl_cust_contact.grid(row=0,column=0,sticky=W)
        enty_contact=ttk.Entry(labelframeleft,textvariable=self.var_contact,width=20,font=("times new
roman",13,"bold"))
        enty_contact.grid(row=0,column=1,sticky=W)
        #fetch data button

        btnFetchData=Button(labelframeleft,command=self.Fetch_contact,text="Fetch
Data",font=("arial",8,"bold"),bg="white",fg="green",width=9)
        btnFetchData.place(x=347,y=4)

        #Check-in data
        check_in_data=Label(labelframeleft,font=("arial",12,"bold"),text="Check_in
Date",padx=2,pady=6)
        check_in_data.grid(row=1,column=0,sticky=W)
        textcheck_in_date=ttk.Entry(labelframeleft,textvariable=self.var_CheckIn,font=("arial",13,"bold
"),width=29)
        textcheck_in_date.grid(row=1,column=1)

        #Check_out data
        check_out_data=Label(labelframeleft,font=("arial",12,"bold"),text="Check_out
Date",padx=2,pady=6)
        check_out_data.grid(row=2,column=0,sticky=W)
        textcheck_out_date=ttk.Entry(labelframeleft,textvariable=self.var_CheckOut,font=("arial",13,"b
old"),width=29)
        textcheck_out_date.grid(row=2,column=1)

        #Room type
        label_RoomType=Label(labelframeleft,font=("arial",12,"bold"),text="Room
Type",padx=2,pady=6)
        label_RoomType.grid(row=3,column=0,sticky=W)
        conn=mysql.connector.connect(host="127.0.0.1",username="root",password="rakesh@123",data
base="resort-management")
        my_cursor=conn.cursor()
        my_cursor.execute("select RoomType from details")
        ide=my_cursor.fetchall()

        combo_RoomType=ttk.Combobox(labelframeleft,textvariable=self.var_RoomType,font=("arial",1
2,"bold"),width=27,state="readonly")
        combo_RoomType["value"]=ide
        combo_RoomType.current(0)
        combo_RoomType.grid(row=3,column=1)
        #Available rooms
        lblRoomAvailable=Label(labelframeleft,font=("arial",12,"bold"),text="Available
Room",padx=2,pady=6)
        lblRoomAvailable.grid(row=4,column=0,sticky=W)
        #textRoomAvailable=ttk.Entry(labelframeleft,textvariable=self.var_RoomAvailable,font=("arial",
13,"bold"),width=29)
        #textRoomAvailable.grid(row=4,column=1)
        conn=mysql.connector.connect(host="127.0.0.1",username="root",password="rakesh@123",data
base="resort-management")
        my_cursor=conn.cursor()
        my_cursor.execute("select RoomNo from details")
        rows=my_cursor.fetchall()

        combo_RoomNo=ttk.Combobox(labelframeleft,textvariable=self.var_RoomAvailable,font=("arial
",12,"bold"),width=27,state="readonly")
        combo_RoomNo["value"]=rows
        combo_RoomNo.current(0)
        combo_RoomNo.grid(row=4,column=1)

        #Meal
        lblMeal=Label(labelframeleft,font=("arial",12,"bold"),text="Meal",padx=2,pady=6)
        lblMeal.grid(row=5,column=0,sticky=W)
        textMeal=ttk.Entry(labelframeleft,textvariable=self.var_Meal,font=("arial",13,"bold"),width=29)
        textMeal.grid(row=5,column=1)

        #No of Days
        lblNoOfDays=Label(labelframeleft,font=("arial",12,"bold"),text="No of Days",padx=2,pady=6)
        lblNoOfDays.grid(row=6,column=0,sticky=W)
        textNoOfDays=ttk.Entry(labelframeleft,textvariable=self.var_NoOfDays,font=("arial",13,"bold"),
width=29)
        textNoOfDays.grid(row=6,column=1)

        #paid tax
        lblNoOfDays=Label(labelframeleft,font=("arial",12,"bold"),text="paid Tax",padx=2,pady=6)
        lblNoOfDays.grid(row=7,column=0,sticky=W)
        textNoOfDays=ttk.Entry(labelframeleft,textvariable=self.var_PaidTax,font=("arial",13,"bold"),wi
dth=29)
        textNoOfDays.grid(row=7,column=1)

        #Sub total
        lblNoOfDays=Label(labelframeleft,font=("arial",12,"bold"),text="Sub Total",padx=2,pady=6)
        lblNoOfDays.grid(row=8,column=0,sticky=W)
        textNoOfDays=ttk.Entry(labelframeleft,textvariable=self.var_ActualTotal,font=("arial",13,"bold"
),width=29)
        textNoOfDays.grid(row=8,column=1)

        #Total Cost
        lblNoOfDays=Label(labelframeleft,font=("arial",12,"bold"),text="Tatal Cost",padx=2,pady=6)
        lblNoOfDays.grid(row=9,column=0,sticky=W)
        textNoOfDays=ttk.Entry(labelframeleft,textvariable=self.var_Total,font=("arial",13,"bold"),widt
h=29)
        textNoOfDays.grid(row=9,column=1)

        #================Bill button
        btnBill=Button(labelframeleft,text="Bill",command=self.total,font=("arial",12,"bold"),bg="white
",fg="green",width=9)
        btnBill.grid(row=10,column=0,padx=1,sticky=W)



         #=========buttons==========
        btn_frame=Frame(labelframeleft,bd=2,relief=RIDGE)
        btn_frame.place(x=0,y=400,width=412,height=40)

        btnAdd=Button(btn_frame,text="Add",command=self.add_data,font=("arial",12,"bold"),bg="w
hite",fg="green",width=9)
        btnAdd.grid(row=0,column=0,padx=1)



        btnupdate=Button(btn_frame,text="Update",command=self.update,font=("arial",12,"bold"),bg=
"white",fg="green",width=9)
        btnupdate.grid(row=0,column=1,padx=1)



        btnDelete=Button(btn_frame,text="Delete",command=self.Delete,font=("arial",12,"bold"),bg="w
hite",fg="green",width=9)
        btnDelete.grid(row=0,column=2,padx=1)

        btnreset=Button(btn_frame,text="Reset",command=self.reset,font=("arial",12,"bold"),bg="white
",fg="green",width=9)
        btnreset.grid(row=0,column=3,padx=1)
        #=============Right side image===========
        img3=Image.open(r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-management-
system\images\room1.jpg")
        img3 = img3.resize((520, 200), Image.LANCZOS)

        self.photoimg3=ImageTk.PhotoImage(img3)

        lblimg = Label(self.root,image=self.photoimg3, bd=0, relief=RIDGE)
        lblimg.place(x=760,y=55,width=520,height=200)

        #=============table frame for search system===============

        Table_Frame=LabelFrame(self.root,bd=2,relief=RIDGE,text="View Details and Search
System",font=("times new roman",12,"bold"),padx=2)
        Table_Frame.place(x=435,y=280,width=860,height=260)

        lblSearchBy=Label(Table_Frame,font=("arial",12,"bold"),text="Search
By",bg="white",fg="green")
        lblSearchBy.grid(row=0,column=0,sticky=W,padx=2)

        self.search_var=StringVar()

        combo_Search=ttk.Combobox(Table_Frame,font=("arial",12,"bold"),width=24,state="readonly")
        combo_Search["value"]=("Contact","Room")
        combo_Search.current(0)
        combo_Search.grid(row=0,column=1,padx=2)

        self.txt_search=StringVar()
        txtSearch=ttk.Entry(Table_Frame,font=("arial",13,"bold"),width=24)
        txtSearch.grid(row=0,column=2,padx=2)

        btnSearch=Button(Table_Frame,text="Search",command=self.search,font=("arial",12,"bold"),bg
="white",fg="green",width=10)
        btnSearch.grid(row=0,column=3,padx=1)
        btnShowAll=Button(Table_Frame,text="Show
All",command=self.fetch_data,font=("arial",12,"bold"),bg="white",fg="green",width=10)
        btnShowAll.grid(row=0,column=4,padx=1)


         #=============show data table==============
        details_table = Frame(Table_Frame, bd=2, relief=RIDGE)
        details_table.place(x=0, y=50, width=880, height=180)


# Creating horizontal and vertical scrollbars
        Scroll_x = ttk.Scrollbar(details_table, orient=HORIZONTAL)
        Scroll_y = ttk.Scrollbar(details_table, orient=VERTICAL)


# Configuring the Treeview and attaching scrollbars
        self.room_table = ttk.Treeview(
        details_table,
        columns=("contact", "CheckIn", "CheckOut", "RoomType", "RoomAvailable", "Meal",
"NoOfDays",),
        xscrollcommand=Scroll_x.set,
        yscrollcommand=Scroll_y.set)


# Packing the scrollbars
        Scroll_x.pack(side=BOTTOM, fill=X)
        Scroll_y.pack(side=RIGHT, fill=Y)


# Configuring the scroll commands to link with Treeview
        Scroll_x.config(command=self.room_table.xview)
        Scroll_y.config(command=self.room_table.yview)


# Packing the Treeview widget to occupy the frame
        self.room_table.pack(fill=BOTH, expand=1)


        self.room_table.heading("contact",text="contact")
        self.room_table.heading("CheckIn",text="Check-in")
        self.room_table.heading("CheckOut",text="Check-out")
        self.room_table.heading("RoomType",text="Room Type")
        self.room_table.heading("RoomAvailable",text="Room No")
        self.room_table.heading("Meal",text="Meal")
        self.room_table.heading("NoOfDays",text="NoOfDays")
```

```
 self.room_table["show"]="headings"

 self.room_table.column("contact",width=100)
 self.room_table.column("CheckIn",width=100)
 self.room_table.column("CheckOut",width=100)
 self.room_table.column("RoomType",width=100)
 self.room_table.column("RoomAvailable",width=100)
 self.room_table.column("Meal",width=100)
 self.room_table.column("NoOfDays",width=100)
 self.room_table.pack(fill=BOTH,expand=1)

 self.room_table.bind("<ButtonRelease-1>",self.get_cursor)
 self.fetch_data()

def add_data(self):
 if self.var_contact.get()=="" or self.var_CheckIn.get()=="":
   messagebox.showerror("Error","All fields are required",parent=self.root)
 else:
   try:
       conn = mysql.connector.connect(
       host="127.0.0.1",   # MySQL server IP (local in this case)
       username="root",         # MySQL username
       password="rakesh@123",  # Replace with your password
       database="resort-management"  # Replace with your database name
       )
       my_cursor=conn.cursor()
       my_cursor.execute("insert into room values(%s,%s,%s,%s,%s,%s,%s)",(
                                                          self.var_contact.get(),
                                                          self.var_CheckIn.get(),
                                                          self.var_CheckOut.get(),
                                                          self.var_RoomType.get(),
                                                          self.var_RoomAvailable.get(),
                                                          self.var_Meal.get(),
                                                          self.var_NoOfDays.get()
                                                      ))
```

```
            conn.commit()
            self.fetch_data()
            conn.close()
            messagebox.showinfo("Success","Room Booked",parent=self.root)
        except Exception as es:
            messagebox.showwarning("Worning",f"something went wrong: {str(es)}",parent=self.root)

            #fetch data
    def fetch_data(self):
        conn=mysql.connector.connect(host="127.0.0.1",username="root",password="rakesh@123",da
tabase="resort-management")
        my_cursor=conn.cursor()
        my_cursor.execute("select * from room")
        rows=my_cursor.fetchall()
        if len(rows)!=0:
            self.room_table.delete(*self.room_table.get_children())
            for i in rows:
                self.room_table.insert("",END,values=i)
        conn.commit()
        conn.close()

    def get_cursor(self,event=""):
        cursor_row=self.room_table.focus()
        content=self.room_table.item(cursor_row)
        row=content["values"]

        self.var_contact.set(row[0]),
        self.var_CheckIn.set(row[1]),
        self.var_CheckOut.set(row[2]),
        self.var_RoomType.set(row[3]),
        self.var_RoomAvailable.set(row[4]),
        self.var_Meal.set(row[5]),
        self.var_NoOfDays.set(row[6])

        #update function
    def update(self):
            if self.var_contact.get() == "":
                messagebox.showerror("Error", "Please enter mobile number", parent=self.root)
            else:
                conn = mysql.connector.connect(host="127.0.0.1", username="root",
password="rakesh@123", database="resort-management")
                my_cursor = conn.cursor()


        # Corrected UPDATE query with consistent number of placeholders and variables
            my_cursor.execute(""" UPDATE room SET contact=%s, check_in=%s, check_out=%s,
roomtype=%s, roomavailable=%s, meal=%s, noofdays=%s WHERE contact=%s """, (

self.var_contact.get(),    # contact value to be updated

self.var_CheckIn.get(),

self.var_CheckOut.get(),

self.var_RoomType.get(),

self.var_RoomAvailable.get(),

self.var_Meal.get(),

self.var_NoOfDays.get(),

self.var_contact.get()     # contact value in WHERE condition
                                                                                        ))


            conn.commit()
            self.fetch_data()
            conn.close()
            messagebox.showinfo("Update", "Room details have been updated successfully",
parent=self.root)


        #delete function
        def Delete(self):
          Delete=messagebox.askyesno("Resort management System","Do you want to delete this
customer",parent=self.root)
          if Delete>0:
              conn=mysql.connector.connect(host="127.0.0.1",username="root",password="rakesh@123
",database="resort-management")
              my_cursor=conn.cursor()
              query="delete from room where contact=%s"
              value=(self.var_contact.get(),)
```

```
                my_cursor.execute(query,value)
            else:
                if not Delete:
                    return
            conn.commit()
            self.fetch_data()
            conn.close()

        #reset function
        def reset(self):
            self.var_contact.set(""),
            self.var_CheckIn.set(""),
            self.var_CheckOut.set(""),
            self.var_RoomType.set(""),
            self.var_RoomAvailable.set(""),
            self.var_Meal.set(""),
            self.var_NoOfDays.set("")
            self.var_PaidTax.set("")
            self.var_ActualTotal.set("")
            self.var_Total.set("")



        #=============all data fetch==============


        def Fetch_contact(self):
            if self.var_contact.get()=="":
                messagebox.showerror("Error","Please Enter contact number",parent=self.root)
            else:
                conn = mysql.connector.connect(host="127.0.0.1", username="root",
password="rakesh@123", database="resort-management")
                my_cursor = conn.cursor()
                query=("select Name from customer where Mobile=%s")
                value=(self.var_contact.get(),)
                my_cursor.execute(query,value)
                row=my_cursor.fetchone()

                if row==None:
                    messagebox.showerror("Error","This number Not found",parent=self.root)
                else:
                    conn.commit()
                    conn.close()

                    showDataframe=Frame(self.root,bd=4,relief=RIDGE,padx=2)
                    showDataframe.place(x=450,y=55,width=300,height=180)

                    lblName=Label(showDataframe,text="Name:",font=("arial",12,"bold"))
                    lblName.place(x=0,y=0)

                    lbl=Label(showDataframe,text=row,font=("arial",12,"bold"))
                    lbl.place(x=90,y=0)
                    conn = mysql.connector.connect(host="127.0.0.1", username="root",
password="rakesh@123", database="resort-management")
                    my_cursor = conn.cursor()
                    query=("select Gender from customer where Mobile=%s")
                    value=(self.var_contact.get(),)
                    my_cursor.execute(query,value)
                    row=my_cursor.fetchone()

                    lblGender=Label(showDataframe,text="Gender:",font=("arial",12,"bold"))
                    lblGender.place(x=0,y=30)

                    lbl2=Label(showDataframe,text=row,font=("arial",12,"bold"))
                    lbl2.place(x=90,y=30)

                    #=============email=============
                    conn = mysql.connector.connect(host="127.0.0.1", username="root",
password="rakesh@123", database="resort-management")
                    my_cursor = conn.cursor()
                    query=("select Email from customer where Mobile=%s")
                    value=(self.var_contact.get(),)
                    my_cursor.execute(query,value)
                    row=my_cursor.fetchone()

                    lblEmail=Label(showDataframe,text="Email:",font=("arial",12,"bold"))
                    lblEmail.place(x=0,y=60)
```

```
                lbl3=Label(showDataframe,text=row,font=("arial",12,"bold"))
                lbl3.place(x=90,y=60)

                #==============nationality========
                conn = mysql.connector.connect(host="127.0.0.1", username="root",
password="rakesh@123", database="resort-management")
                my_cursor = conn.cursor()
                query=("select Nationality from customer where Mobile=%s")
                value=(self.var_contact.get(),)
                my_cursor.execute(query,value)
                row=my_cursor.fetchone()

                lblNationality=Label(showDataframe,text="Nationality:",font=("arial",12,"bold"))
                lblNationality.place(x=0,y=90)

                lbl4=Label(showDataframe,text=row,font=("arial",12,"bold"))
                lbl4.place(x=90,y=90)

                #============address============
                conn = mysql.connector.connect(host="127.0.0.1", username="root",
password="rakesh@123", database="resort-management")
                my_cursor = conn.cursor()
                query=("select Address from customer where Mobile=%s")
                value=(self.var_contact.get(),)
                my_cursor.execute(query,value)
                row=my_cursor.fetchone()

                lblAddress=Label(showDataframe,text="Address:",font=("arial",12,"bold"))
                lblAddress.place(x=0,y=120)

                lbl5=Label(showDataframe,text=row,font=("arial",12,"bold"))
                lbl5.place(x=90,y=120)

      def search(self):
       conn = mysql.connector.connect(
       host="127.0.0.1",
       username="root",
       password="rakesh@123",
       database="resort-management"
        )
       my_cursor = conn.cursor()

   # Retrieve the column name and search text
       column_name = str(self.search_var.get()).strip()
       search_text = str(self.txt_search.get()).strip()

   # Check if we are searching the 'Ref' column (assuming 'Ref' is a numeric field)
       if column_name == "Contact":
               query = f"SELECT * FROM room WHERE `Contact` = %s"
               my_cursor.execute(query, (search_text,))
       else:
       # Use LIKE for other text-based searches
               query = f"SELECT * FROM room WHERE `Room` LIKE %s"
               my_cursor.execute(query, ("%" + search_text + "%",))

       rows = my_cursor.fetchall()

       if len(rows) != 0:
               self.room_table.delete(*self.room_table.get_children())
               for row in rows:
                           self.room_table.insert("", END, values=row)

               conn.commit()
               conn.close()




      def total(self):
          inDate=self.var_CheckIn.get()
          outDate=self.var_CheckOut.get()
          inDate=datetime.strptime(inDate,"%d/%m/%Y")
          outDate=datetime.strptime(outDate,"%d/%m/%Y")
          self.var_NoOfDays.set(abs(outDate-inDate).days)

          if(self.var_Meal.get()=="BreakFast" and self.var_RoomType.get()=="Laxary"):
               q1=float(300)
               q2=float(700)
               q3=float(self.var_NoOfDays.get())
               q4=float(q1+q2)
               q5=float(q3+q4)
               Tax="Rs. "+str("%.2f"%((q5)*0.09))
               ST="Rs. "+str("%.2f"%((q5)))
               TT="Rs. "+str("%.2f"%(q5+((q5)*0.09)))
               self.var_PaidTax.set(Tax)
               self.var_ActualTotal.set(ST)
               self.var_Total.set(TT)

           elif (self.var_Meal.get()=="Lunch" and self.var_RoomType.get()=="Single"):
               q1=float(300)
               q2=float(700)
               q3=float(self.var_NoOfDays.get())
               q4=float(q1+q2)
               q5=float(q3+q4)
               Tax="Rs. "+str("%.2f"%((q5)*0.09))
               ST="Rs. "+str("%.2f"%((q5)))
               TT="Rs. "+str("%.2f"%(q5+((q5)*0.09)))
               self.var_PaidTax.set(Tax)
               self.var_ActualTotal.set(ST)
               self.var_Total.set(TT)

           elif (self.var_Meal.get()=="BreakFast" and self.var_RoomType.get()=="Duplex"):
               q1=float(500)
               q2=float(1000)
               q3=float(self.var_NoOfDays.get())
               q4=float(q1+q2)
               q5=float(q3+q4)
               Tax="Rs. "+str("%.2f"%((q5)*0.09))
               ST="Rs. "+str("%.2f"%((q5)))
               TT="Rs. "+str("%.2f"%(q5+((q5)*0.09)))
               self.var_PaidTax.set(Tax)
               self.var_ActualTotal.set(ST)
               self.var_Total.set(TT)
   if __name__ == "__main__":
      root=Tk()
      obj=Roombooking(root)
      root.mainloop()
```

Customer.py

```
from tkinter import*
from PIL import Image,ImageTk
from tkinter import ttk
import random
import mysql.connector
from tkinter import messagebox

class Cust_Win:
      def __init__(self,root):
       self.root=root
       self.root.title("Resort Management System")
       self.root.geometry("1295x550+230+220")



       #============= variables=============
       self.var_ref=StringVar()
       x=random.randint(1000,9999)
       self.var_ref.set(str(x))

       self.var_cust_name=StringVar()
       self.var_mother=StringVar()
       self.var_gender=StringVar()
       self.var_post=StringVar()
       self.var_mobile=StringVar()
       self.var_email=StringVar()
       self.var_nationality=StringVar()
       self.var_address=StringVar()
       self.var_id_proof=StringVar()
       self.var_id_number=StringVar()
```

```
        #===============title================

        lbl_title=Label(self.root,text="Add Customer Details",font=("times new 
roman",15,"bold"),bg="white",fg="green")
        lbl_title.place(x=0,y=0,width=1295,height=50)




        img2=Image.open(r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-management-
system\images\logo.jpg")
        img2 = img2.resize((100, 40), Image.LANCZOS)

        self.photoimg2=ImageTk.PhotoImage(img2)

        lblimg = Label(self.root,image=self.photoimg2, bd=0, relief=RIDGE)
        lblimg.place(x=5,y=2,width=100,height=40)

       ## =======================lableframe==============#
        labelframeleft=LabelFrame(self.root,bd=2,relief=RIDGE,text="Customer Details",font=("times 
new roman",12,"bold"),padx=2)
        labelframeleft.place(x=5,y=50,width=425,height=490)

        #==========labels and entrys==============#

        lbl_cust_ref=Label(labelframeleft,text="Customer Ref",font=("times new 
roman",12,"bold"),padx=2,pady=6)
        lbl_cust_ref.grid(row=0,column=0,sticky=W)

        enty_ref=ttk.Entry(labelframeleft,textvariable=self.var_ref,width=29,font=("times new 
roman",13,"bold"),state="readonly")
        enty_ref.grid(row=0,column=1)

        #cust_name
        cname=Label(labelframeleft,font=("arial",12,"bold"),text="Customer Name",padx=2,pady=6)
        cname.grid(row=1,column=0,sticky=W)
        txtcname=ttk.Entry(labelframeleft,textvariable=self.var_cust_name,font=("arial",13,"bold"),widt
h=29)
        txtcname.grid(row=1,column=1)

        #mother name
        lblmname=Label(labelframeleft,font=("arial",12,"bold"),text="Mother Name",padx=2,pady=6)
        lblmname.grid(row=2,column=0,sticky=W)
        txtmname=ttk.Entry(labelframeleft,textvariable=self.var_mother,font=("arial",13,"bold"),width=
29)
        txtmname.grid(row=2,column=1)

        #gender
        lable_gender=Label(labelframeleft,font=("arial",12,"bold"),text="Gender",padx=2,pady=6)
        lable_gender.grid(row=3,column=0,sticky=W)

        combo_gender=ttk.Combobox(labelframeleft,textvariable=self.var_gender,font=("arial",12,"bold
"),width=27,state="readonly")
        combo_gender["value"]=("Male","Female","Other")
        combo_gender.current(0)
        combo_gender.grid(row=3,column=1)



        #postcode
        lblPostCode=Label(labelframeleft,font=("arial",12,"bold"),text="PostCode",padx=2,pady=6)
        lblPostCode.grid(row=4,column=0,sticky=W)
        txtPostCode=ttk.Entry(labelframeleft,textvariable=self.var_post,font=("arial",13,"bold"),width=2
9)
        txtPostCode.grid(row=4,column=1)

        #mobile number
        lblMobile=Label(labelframeleft,font=("arial",12,"bold"),text="Mobile",padx=2,pady=6)
        lblMobile.grid(row=5,column=0,sticky=W)
        txtMobile=ttk.Entry(labelframeleft,textvariable=self.var_mobile,font=("arial",13,"bold"),width=2
9)
        txtMobile.grid(row=5,column=1)

        #email
        lblEmail=Label(labelframeleft,font=("arial",12,"bold"),text="Email",padx=2,pady=6)
        lblEmail.grid(row=6,column=0,sticky=W)
        txtEmail=ttk.Entry(labelframeleft,textvariable=self.var_email,font=("arial",13,"bold"),width=29)
        txtEmail.grid(row=6,column=1)

        #nationality
        lblNationality=Label(labelframeleft,font=("arial",12,"bold"),text="Nationality",padx=2,pady=6)
        lblNationality.grid(row=7,column=0,sticky=W)


        combo_Natinality=ttk.Combobox(labelframeleft,textvariable=self.var_nationality,font=("arial",12
,"bold"),width=27,state="readonly")
        combo_Natinality["value"]=("Indian","American","British")
        combo_Natinality.current(0)
        combo_Natinality.grid(row=7,column=1)


        #idproof type combobox
        lblIdProof=Label(labelframeleft,font=("arial",12,"bold"),text="Id Proof Type",padx=2,pady=6)
        lblIdProof.grid(row=8,column=0,sticky=W)


        combo_id=ttk.Combobox(labelframeleft,textvariable=self.var_id_proof,font=("arial",12,"bold"),
width=27,state="readonly")
        combo_id["value"]=("Aadhar","Voter Id","Driving Licence","Passport")
        combo_id.current(0)
        combo_id.grid(row=8,column=1)


        #id number
        lblIdNumber=Label(labelframeleft,font=("arial",12,"bold"),text="Id Number",padx=2,pady=6)
        lblIdNumber.grid(row=9,column=0,sticky=W)
        txtIdNumber=ttk.Entry(labelframeleft,textvariable=self.var_id_number,font=("arial",13,"bold"),
width=29)
        txtIdNumber.grid(row=9,column=1)


        #address
        lblAddress=Label(labelframeleft,font=("arial",12,"bold"),text="Address",padx=2,pady=6)
        lblAddress.grid(row=10,column=0,sticky=W)
        txtAddress=ttk.Entry(labelframeleft,textvariable=self.var_address,font=("arial",13,"bold"),width
=29)
        txtAddress.grid(row=10,column=1)



        #=========buttons==========
        btn_frame=Frame(labelframeleft,bd=2,relief=RIDGE)
        btn_frame.place(x=0,y=400,width=412,height=40)


        btnAdd=Button(btn_frame,text="Add",command=self.add_data,font=("arial",12,"bold"),bg="w
hite",fg="green",width=9)
        btnAdd.grid(row=0,column=0,padx=1)
        btnupdate=Button(btn_frame,text="Update",command=self.update,font=("arial",12,"bold"),bg=
"white",fg="green",width=9)
        btnupdate.grid(row=0,column=1,padx=1)



        btnDelete=Button(btn_frame,text="Delete",command=self.Delete,font=("arial",12,"bold"),bg="w
hite",fg="green",width=9)
        btnDelete.grid(row=0,column=2,padx=1)

        btnreset=Button(btn_frame,text="Reset",command=self.reset,font=("arial",12,"bold"),bg="white
",fg="green",width=9)
        btnreset.grid(row=0,column=3,padx=1)

        #=============table frame===============

        Table_Frame=LabelFrame(self.root,bd=2,relief=RIDGE,text="View Details and Search
System",font=("times new roman",12,"bold"),padx=2)
        Table_Frame.place(x=435,y=50,width=860,height=490)

        lblSearchBy=Label(Table_Frame,font=("arial",12,"bold"),text="Search
By",bg="white",fg="green")
        lblSearchBy.grid(row=0,column=0,sticky=W,padx=2)

        self.search_var=StringVar()

        combo_Search=ttk.Combobox(Table_Frame,textvariable=self.search_var,font=("arial",12,"bold")
,width=24,state="readonly")
        combo_Search["value"]=("Mobile No","Ref")
        combo_Search.current(0)
        combo_Search.grid(row=0,column=1,padx=2)

        self.txt_search=StringVar()
        txtSearch=ttk.Entry(Table_Frame,textvariable=self.txt_search,font=("arial",13,"bold"),width=24)
        txtSearch.grid(row=0,column=2,padx=2)

        btnSearch=Button(Table_Frame,text="Search",command=self.search,font=("arial",12,"bold"),bg
="white",fg="green",width=10)
        btnSearch.grid(row=0,column=3,padx=1)
        btnShowAll=Button(Table_Frame,text="Show
All",command=self.fetch_data,font=("arial",12,"bold"),bg="white",fg="green",width=10)
        btnShowAll.grid(row=0,column=4,padx=1)

        #=============show data table==============
        details_table = Frame(Table_Frame, bd=2, relief=RIDGE)
        details_table.place(x=0, y=50, width=880, height=360)

# Creating horizontal and vertical scrollbars
        Scroll_x = ttk.Scrollbar(details_table, orient=HORIZONTAL)
        Scroll_y = ttk.Scrollbar(details_table, orient=VERTICAL)

# Configuring the Treeview and attaching scrollbars
        self.Cust_Details_Table = ttk.Treeview(
        details_table,
        columns=("ref", "name", "mother", "gender", "post", "mobile", "email", "nationality",
"idproof", "idnumber", "address"),
        xscrollcommand=Scroll_x.set,
        yscrollcommand=Scroll_y.set)

# Packing the scrollbars
        Scroll_x.pack(side=BOTTOM, fill=X)
        Scroll_y.pack(side=RIGHT, fill=Y)

# Configuring the scroll commands to link with Treeview
        Scroll_x.config(command=self.Cust_Details_Table.xview)
        Scroll_y.config(command=self.Cust_Details_Table.yview)

# Packing the Treeview widget to occupy the frame
        self.Cust_Details_Table.pack(fill=BOTH, expand=1)

        self.Cust_Details_Table.heading("ref",text="Refer No")
        self.Cust_Details_Table.heading("name",text="Name")
        self.Cust_Details_Table.heading("mother",text="Mother Name")
        self.Cust_Details_Table.heading("gender",text="Gender")
        self.Cust_Details_Table.heading("post",text="PostCode")
```

```
 self.Cust_Details_Table.heading("mobile",text="Mobile")
 self.Cust_Details_Table.heading("email",text="Email")
 self.Cust_Details_Table.heading("nationality",text="Nationality")
 self.Cust_Details_Table.heading("idproof",text="Id Proof")
 self.Cust_Details_Table.heading("idnumber",text="Id Number")
 self.Cust_Details_Table.heading("address",text="Address")


 self.Cust_Details_Table["show"]="headings"

 self.Cust_Details_Table.column("ref",width=100)
 self.Cust_Details_Table.column("name",width=100)
 self.Cust_Details_Table.column("mother",width=100)
 self.Cust_Details_Table.column("gender",width=100)
 self.Cust_Details_Table.column("post",width=100)
 self.Cust_Details_Table.column("mobile",width=100)
 self.Cust_Details_Table.column("email",width=100)
 self.Cust_Details_Table.column("nationality",width=100)
 self.Cust_Details_Table.column("idproof",width=100)
 self.Cust_Details_Table.column("idnumber",width=100)
 self.Cust_Details_Table.column("address",width=100)


 self.Cust_Details_Table.pack(fill=BOTH,expand=1)
 self.Cust_Details_Table.bind("<ButtonRelease-1>",self.get_cursor)
 self.fetch_data()

def add_data(self):
 if self.var_mobile.get()=="" or self.var_mother.get()=="":
   messagebox.showerror("Error","All fields are required",parent=self.root)
 else:
   try:
        conn = mysql.connector.connect(
        host="127.0.0.1",   # MySQL server IP (local in this case)
        username="root",         # MySQL username
        password="rakesh@123",  # Replace with your password
        database="resort-management"  # Replace with your database name
        )
```

```
            my_cursor=conn.cursor()
            my_cursor.execute("insert into customer values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",(
                                                            self.var_ref.get(),
                                                            self.var_cust_name.get(),
                                                            self.var_mother.get(),
                                                            self.var_gender.get(),
                                                            self.var_post.get(),
                                                            self.var_mobile.get(),
                                                            self.var_email.get(),
                                                            self.var_nationality.get(),
                                                            self.var_id_proof.get(),
                                                            self.var_id_number.get(),
                                                            self.var_address.get()
                                                        ))
            conn.commit()
            self.fetch_data()
            conn.close()
            messagebox.showinfo("Success","Customer has been added",parent=self.root)
        except Exception as es:
            messagebox.showwarning("Worning",f"something went wrong: {str(es)}",parent=self.root)

    def fetch_data(self):
        conn=mysql.connector.connect(host="127.0.0.1",username="root",password="rakesh@123",da
tabase="resort-management")
        my_cursor=conn.cursor()
        my_cursor.execute("select * from Customer")
        rows=my_cursor.fetchall()
        if len(rows)!=0:
            self.Cust_Details_Table.delete(*self.Cust_Details_Table.get_children())
            for i in rows:
                self.Cust_Details_Table.insert("",END,values=i)
        conn.commit()
        conn.close()

    def get_cursor(self,event=""):
        cursor_row=self.Cust_Details_Table.focus()
        content=self.Cust_Details_Table.item(cursor_row)
        row=content["values"]
```

```
            self.var_ref.set(row[0]),
            self.var_cust_name.set(row[1]),
            self.var_mother.set(row[2]),
            self.var_gender.set(row[3]),
            self.var_post.set(row[4]),
            self.var_mobile.set(row[5]),
            self.var_email.set(row[6]),
            self.var_nationality.set(row[7]),
            self.var_id_proof.set(row[8]),
            self.var_id_number.set(row[9]),
            self.var_address.set(row[10])

        def update(self):
            if self.var_mobile.get() == "":
                messagebox.showerror("Error", "Please enter mobile number", parent=self.root)
            else:
                conn = mysql.connector.connect(host="127.0.0.1", username="root",
password="rakesh@123", database="resort-management")
                my_cursor = conn.cursor()

        # Use backticks around `Id Number` to avoid syntax errors
                my_cursor.execute(""" UPDATE customer
                SET Name=%s, Mother=%s, Gender=%s, PostCode=%s, Mobile=%s, Email=%s,
Nationality=%s,
                Idproof=%s, `Id Number`=%s, Address=%s  WHERE Ref=%s """, (
                    self.var_cust_name.get(),
                    self.var_mother.get(),
                    self.var_gender.get(),
                    self.var_post.get(),
                    self.var_mobile.get(),
                    self.var_email.get(),
                    self.var_nationality.get(),
                    self.var_id_proof.get(),
                    self.var_id_number.get(),
                    self.var_address.get(),
                    self.var_ref.get()
                ))
```

```
                    conn.commit()
                    self.fetch_data()
                    conn.close()
                    messagebox.showinfo("Update", "Customer details have been updated successfully",
parent=self.root)

    def Delete(self):
        Delete=messagebox.askyesno("Resort management System","Do you want to delete this
customer",parent=self.root)
        if Delete>0:
            conn=mysql.connector.connect(host="127.0.0.1",username="root",password="rakesh@123
",database="resort-management")
            my_cursor=conn.cursor()
            query="delete from Customer where Ref=%s"
            value=(self.var_ref.get(),)
            my_cursor.execute(query,value)
        else:
            if not Delete:
                return
        conn.commit()
        self.fetch_data()
        conn.close()

    def reset(self):
        self.var_ref.set(""),
        self.var_cust_name.set(""),
        self.var_mother.set(""),
        #self.var_gender.set(""),
        self.var_post.set(""),
        self.var_mobile.set(""),
        self.var_email.set(""),
        #self.var_nationality.set(""),
        #self.var_id_proof.set(""),
        self.var_id_number.set(""),
        self.var_address.set("")

        x=random.randint(1000,9999)
        self.var_ref.set(str(x))
```

```
   def search(self):
    conn = mysql.connector.connect(
    host="127.0.0.1",
    username="root",
    password="rakesh@123",
    database="resort-management"
     )
    my_cursor = conn.cursor()

 # Retrieve the column name and search text
    column_name = str(self.search_var.get()).strip()
    search_text = str(self.txt_search.get()).strip()

 # Check if we are searching the 'Ref' column (assuming 'Ref' is a numeric field)
    if column_name == "Ref":
        query = f"SELECT * FROM Customer WHERE `Ref` = %s"
        my_cursor.execute(query, (search_text,))
    else:
    # Use LIKE for other text-based searches
        query = f"SELECT * FROM Customer WHERE `Mobile` LIKE %s"
        my_cursor.execute(query, ("%" + search_text + "%",))

    rows = my_cursor.fetchall()

    if len(rows) != 0:
        self.Cust_Details_Table.delete(*self.Cust_Details_Table.get_children())
        for row in rows:
                self.Cust_Details_Table.insert("", END, values=row)

        conn.commit()
        conn.close()

if __name__ == "__main__":
 root=Tk()
 obj=Cust_Win(root)
 root.mainloop()
```

detailes.py

```
from tkinter import*
from PIL import Image,ImageTk
from tkinter import ttk
import random
from time import strptime
from datetime import datetime
import mysql.connector
from tkinter import messagebox

class DetailsRoom:
    def __init__(self,root):
        self.root=root
        self.root.title("Resort Management System")
        self.root.geometry("1295x550+230+220")

        #===================title==============



        lbl_title=Label(self.root,text="ROOM BOOKING DETAILS ",font=("times new
roman",15,"bold"),bg="white",fg="green")
        lbl_title.place(x=0,y=0,width=1295,height=50)
        #===================logo=======================

        img2=Image.open(r"C:\Users\CHANDRASHEKAR\OneDrive\Desktop\resort-management-
system\images\logo.jpg")
        img2 = img2.resize((100, 40), Image.LANCZOS)

        self.photoimg2=ImageTk.PhotoImage(img2)

        lblimg = Label(self.root,image=self.photoimg2, bd=0, relief=RIDGE)
        lblimg.place(x=5,y=2,width=100,height=40)
        #======================lableframe===================
        labelframeleft=LabelFrame(self.root,bd=2,relief=RIDGE,text=" New Add Room",font=("times new roman",12,"bold"),padx=2)
        labelframeleft.place(x=5,y=50,width=540,height=350)


        #Floor
        lbl_floor=Label(labelframeleft,text="Floor",font=("times new roman",12,"bold"),padx=2,pady=6)
        lbl_floor.grid(row=0,column=0,sticky=W)


        self.var_Floor=StringVar()
        enty_floor=ttk.Entry(labelframeleft,textvariable=self.var_Floor,width=20,font=("times new roman",13,"bold"))
        enty_floor.grid(row=0,column=1,sticky=W)


        #Room No
        lbl_RoomNo=Label(labelframeleft,text="Room No",font=("times new roman",12,"bold"),padx=2,pady=6)
        lbl_RoomNo.grid(row=1,column=0,sticky=W)


        self.var_RoomNo=StringVar()
        enty_RoomNo=ttk.Entry(labelframeleft,textvariable=self.var_RoomNo,width=20,font=("times new roman",13,"bold"))
        enty_RoomNo.grid(row=1,column=1,sticky=W)


        #Room type
        lbl_RoomType=Label(labelframeleft,text="Room Type",font=("times new roman",12,"bold"),padx=2,pady=6)
        lbl_RoomType.grid(row=2,column=0,sticky=W)


        self.var_RoomType=StringVar()
        enty_RoomType=ttk.Entry(labelframeleft,textvariable=self.var_RoomType,width=20,font=("times new roman",13,"bold"))
        enty_RoomType.grid(row=2,column=1,sticky=W)


         #=========buttons==========
        btn_frame=Frame(labelframeleft,bd=2,relief=RIDGE)
        btn_frame.place(x=0,y=200,width=412,height=40)
        btnAdd=Button(btn_frame,text="Add",command=self.add_data,font=("arial",12,"bold"),bg="w
hite",fg="green",width=9)
        btnAdd.grid(row=0,column=0,padx=1)



        btnupdate=Button(btn_frame,text="Update",command=self.update,font=("arial",12,"bold"),bg
="white",fg="green",width=9)
        btnupdate.grid(row=0,column=1,padx=1)



        btnDelete=Button(btn_frame,text="Delete",command=self.Delete,font=("arial",12,"bold"),bg="
white",fg="green",width=9)
        btnDelete.grid(row=0,column=2,padx=1)


        btnreset=Button(btn_frame,text="Reset",command=self.reset,font=("arial",12,"bold"),bg="whi
te",fg="green",width=9)
        btnreset.grid(row=0,column=3,padx=1)


          #=============table frame for search============
        Table_Frame=LabelFrame(self.root,bd=2,relief=RIDGE,text="Show Room Details",font=("times
new roman",12,"bold"),padx=2)
        Table_Frame.place(x=600,y=55,width=600,height=350)



# Creating horizontal and vertical scrollbars
        Scroll_x = ttk.Scrollbar(Table_Frame, orient=HORIZONTAL)
        Scroll_y = ttk.Scrollbar(Table_Frame, orient=VERTICAL)


# Configuring the Treeview and attaching scrollbars
        self.room_table = ttk.Treeview(
        Table_Frame,
        columns=("Floor", "RoomNo", "RoomType",),
        xscrollcommand=Scroll_x.set,
        yscrollcommand=Scroll_y.set)


# Packing the scrollbars
        Scroll_x.pack(side=BOTTOM, fill=X)
        Scroll_y.pack(side=RIGHT, fill=Y)
```

```
# Configuring the scroll commands to link with Treeview
    Scroll_x.config(command=self.room_table.xview)
    Scroll_y.config(command=self.room_table.yview)

    self.room_table.heading("Floor",text="Floor")
    self.room_table.heading("RoomNo",text="Room No")
    self.room_table.heading("RoomType",text="Room Type")




    self.room_table["show"]="headings"

    self.room_table.column("Floor",width=100)
    self.room_table.column("RoomNo",width=100)
    self.room_table.column("RoomType",width=100)
    self.room_table.pack(fill=BOTH,expand=1)
    self.room_table.bind("<ButtonRelease-1>",self.get_cursor)
    self.fetch_data()

   def add_data(self):
    if self.var_Floor.get()=="" or self.var_RoomType.get()=="":
      messagebox.showerror("Error","All fields are required",parent=self.root)
    else:
      try:
        conn = mysql.connector.connect(
        host="127.0.0.1",   # MySQL server IP (local in this case)
        username="root",       # MySQL username
        password="rakesh@123",  # Replace with your password
        database="resort-management"  # Replace with your database name
        )
        my_cursor=conn.cursor()
        my_cursor.execute("insert into Details values(%s,%s,%s)",(
                                                self.var_Floor.get(),
                                                self.var_RoomNo.get(),
                                       self.var_RoomType.get()

                                  ))
        conn.commit()
        self.fetch_data()
        conn.close()
        messagebox.showinfo("Success","New Room Added Successfully",parent=self.root)
      except Exception as es:
        messagebox.showwarning("Worning",f"something went wrong: {str(es)}",parent=self.root)


        #fetch data
   def fetch_data(self):
      conn=mysql.connector.connect(host="127.0.0.1",username="root",password="rakesh@123"
,database="resort-management")
      my_cursor=conn.cursor()
      my_cursor.execute("select * from details")
      rows=my_cursor.fetchall()
      if len(rows)!=0:
        self.room_table.delete(*self.room_table.get_children())
        for i in rows:
          self.room_table.insert("",END,values=i)
      conn.commit()
      conn.close()


   def get_cursor(self,event=""):
      cursor_row=self.room_table.focus()
      content=self.room_table.item(cursor_row)
      row=content["values"]


      self.var_Floor.set(row[0]),
      self.var_RoomNo.set(row[1]),
      self.var_RoomType.set(row[2])


      #update function
   def update(self):
        if self.var_Floor.get() == "":
            messagebox.showerror("Error", "Please enter Floor number", parent=self.root)
        else:
              conn = mysql.connector.connect(host="127.0.0.1", username="root",
password="rakesh@123", database="resort-management")
              my_cursor = conn.cursor()

  # Corrected UPDATE query with consistent number of placeholders and variables
        my_cursor.execute(""" UPDATE details SET Floor=%s, RoomType=%s  WHERE RoomNo=%s
""", (
                                                                  self.var_Floor.get(),
                                                                  self.var_RoomType.get(),
                                                                  self.var_RoomNo.get()

                                                                ))

        conn.commit()
        self.fetch_data()
        conn.close()
        messagebox.showinfo("Update", " New Room details have been updated successfully",
parent=self.root)

     #delete function
     def Delete(self):
         Delete=messagebox.askyesno("Resort management System","Do you want to delete this
Room",parent=self.root)
         if Delete>0:
             conn=mysql.connector.connect(host="127.0.0.1",username="root",password="rakesh@1
23",database="resort-management")
             my_cursor=conn.cursor()
             query="delete from details where RoomNo=%s"
             value=(self.var_RoomNo.get(),)
             my_cursor.execute(query,value)
         else:
        if not Delete:
           return
         conn.commit()
         self.fetch_data()
         conn.close()
```

```
    #reset function
    def reset(self):
        self.var_Floor.set(""),
        self.var_RoomNo.set(""),
        self.var_RoomType.set("")



if __name__ == "__main__":
    root=Tk()
    obj=DetailsRoom(root)
    root.mainloop()
```

TempCodeRunnerFile.py

```
if __name__ == "__main__":
    root=Tk()
    app=Login_Window(root)
    root.mainloop()
```