Online Hotel Room Booking

A Project Report

Submitted in Partial fulfilment

of the Degree of

Masters of Computer Applications

Supervisor's Name-:
Ms. Ruchika Sharma

Submitted By-: Himanshu Kaushik
Enrolment No: 030222016
Semester: 2nd

Jagan Nath University

Bahadurgarh (NCR)

(2022-24)

Project Certificate

This is to certify that the project report entitled Online Hotel Room Booking submitted to **Jaganath University, BAHADURGARH** in partial fulfilment of the requirement for the award of the degree of **MASTERS OF COMPUTER APPLICATIONS (MCA),** is an original work carried out by Mr. **Himanshu Kaushik** Enrolment No: **030222016** under the guidance of Ms **Ruchika Sharma**. The matter embodied in this project is a genuine work done by the student and has not been submitted whether to this University or to any other University / Institute for the fulfilment of the requirement of any course of study.

(Signature of the Student)	(Signature of the Guide)
Name of the student: Himanshu Kaushik	Name of the guide: Ms Ruchika Sharma
Enrolment No.:030222016	

Acknowledgement

I offer my sincere thanks and humble regards to **JAGANNATH UNIVERSITY**, **BAHADURGARH** for imparting us very valuable professional training in **MCA**.

I pay my gratitude and sincere regards to Ruchika Sharma, my project Guide for giving methe cream of her knowledge. I am thankful to her as he has been a constant source of advice, motivation, and inspiration. I am also thankful to her for giving his suggestions and encouragement throughout the project work. I take the opportunity to express my gratitude and thanks to our computer Lab staff andlibrary staff for allowing me to utilize their resources to complete the project.

I am also thankful to my family and friends for constantly motivating me to complete the projectand providing me with an environment, which enhanced my knowledge.

Name: Mr. Himanshu Kaushik

Course: MCA 2nd Semester

Enroll. No: 03022201

TABLE OF CONTENT

INTRODUCTION

The project "Hotel Booking System" is design to keep the records of daily booking of hotel rooms and meals. This project will ease the burden of cashier and make his way faster. This project is divided into various parts that perform different functions. This console-based application has the listing of rooms, their prices, an inbuilt calculator and separate window for reports. The software made for the room booking is made for the Full Proof Functioning of different organization by effectively reducing the tedious, time consuming & expensive paper work. The software helps in reducing the hassles of paper & unnecessary cost of records and maintenance.

Now a days IT is the backbone of the business sector. In today's scenario all the sectors are adopting the IT techniques for simplifying their work and solving the day-to-day problems. Technology is serious stuff. If we talk about manual or semi-automated system, we can easily figure out how time consuming and prone to error the manual system is. In today's fast-track era, where "time is money" one has to be fully automate the existing system to beat the competitive world.

This rule will also reduce the paper work, which will also result in reduction of complexity and chaos and will provide timely information. This system will process data speedily and accurately and provide information when and where it is required. The system will be used to store data, produce booking reports and handle management inquiries. If you are an owner of a busy hotel, motel or a resort or if you want to be on in the same industry, let's be sure, it is very important for you to have an automated room booking system, as for any hotels not having this is costing them a fortune. Either be it in a smartphone or any other electronic device, having a food ordering online system is really essential.

Objectives of the Hotel Booking System GUI Application for Admin

The objectives of the Hotel Booking System GUI Application are focused on providing hotel administrators with a user-friendly interface to efficiently manage and oversee the hotel's operations. The application aims to streamline administrative tasks and enhance productivity. The key objectives include:

1. Provides an intuitive and informative front page that presents a comprehensive overview of the hotel's operations. This may include adding customers, creating a new account for admins, update/delete the saved data, revenue.

2. Room Management: Enable administrators to manage room inventory efficiently. This includes adding new rooms, updating room details (such as room type, and pricing).

3. Reservation Management: Allow administrators to view and manage reservations made by customers. This includes the ability to modify or cancel reservations, assign rooms to guests, and handle special requests or requirements

4. Booking: Generate reports and analytics related to hotel bookings and revenue. This information can assist administrators in making data-driven decisions, identifying trends, and optimizing pricing and availability.

Some major benefits of room booking system: -

- Fully automated service
- User friendly interface
- Easy to keep records
- Runs effectively
- Simple to manage room window
- Lessens the burden on the user

OBJECTIVES

THE OBJECTIVE OF THE STUDY

The objective of a room booking GUI-based application for admin usage is to provide hotel administrators with a user-friendly interface to efficiently manage and oversee the hotel's room booking operations. The application aims to streamline administrative tasks related to room reservations and enhance productivity. The key purposes include

1. Simplifying Room Management: The GUI-based application allows administrators to easily manage the availability, pricing, and details of different rooms within the hotel. They can view, add, edit, or remove rooms as needed, ensuring accurate and up-to-date information.
2. Streamlining Reservation Process: The application enables administrators to handle room reservations efficiently. They can view, modify, or cancel existing bookings based on customer requests or changes in availability. This helps ensure a smooth reservation process and minimizes errors or double bookings.
3. Maximizing Room Occupancy: By having a clear overview of room availability and reservations, administrators can make informed decisions to maximize room occupancy. They can identify periods of high demand and adjust pricing or promotions accordingly to optimize revenue and occupancy rates.
4. Managing Room Types: The application allows administrators to define and manage different room types and their associated amenities. They can set the pricing, occupancy limits, and specific features for each room type, ensuring accurate representation to customers during the booking process.

Overall, the purpose of a room booking GUI-based application for admin usage is to provide hotel administrators with a comprehensive and efficient tool to manage room reservations, optimize occupancy, enhance customer service, and improve overall operational effectiveness.

PURPOSE

The purpose of a room booking system is to facilitate the process of booking rooms from hotels or other stay establishments. This can be done through an online platform, a mobile app, or a dedicated ordering system at the hotel. A room booking system can make it easier and more convenient for the users, as they can browse room menus, select the required rooms and assign to the customer.

In addition to these benefits, a room booking system can also help hotels increase their revenue and enhance the customer experience, as it can provide a more seamless and enjoyable way for users to book stays, hence making the customers to wait lesser. Finally, a room booking system can provide valuable data and insights that can be used to make informed business decisions.

Some benefits of a room booking system include:

- Convenience for users: Users can easily book a room, add a new customer, update the room availability etc.
- Improved efficiency and accuracy: A room booking system can streamline the booking process, reducing errors and ensuring that orders are placed and fulfilled quickly.
- Increased revenue: By making it easier for customers to book stays, a room booking system can help hotels increase their sales and revenue.
- Data and insights: A room booking system can also help hotels track and analyze customer orders and preferences, which can be used to make informed business decisions.

REQUIREMENTS

A high-level requirements specification is required. The purpose of the requirements analysis is to identify requirements for the proposed system. The emphasis is on the discovery of user requirements. Each requirement (or problem) must be defined and documented in the requirements catalogue.

HARDWARE & SOFTWARE

Hardware Interface:

Hardware requirements for running this project are as follows:

Processor: - Pentium III or above, and any other processor with more than 1.65GHz

RAM: - 128 MB or above.

HD: - 20 GB or above.

Software Interface: -

Operating System: Windows 2000/XP/Vista/Linux

Python

My SQL

ANALYSIS DOCUMENT

DATA FLOW DIAGRAM

(DFD)

It is a pictorial representation of Business processes (functions/services/activities), along with the data flow. It includes issuing ticket,cancelling ticket. Here what we focus is on what data flows and not how the data flows. When all the analysis is being made then we develop a diagram to depict the analysis, and following symbols are being used:-

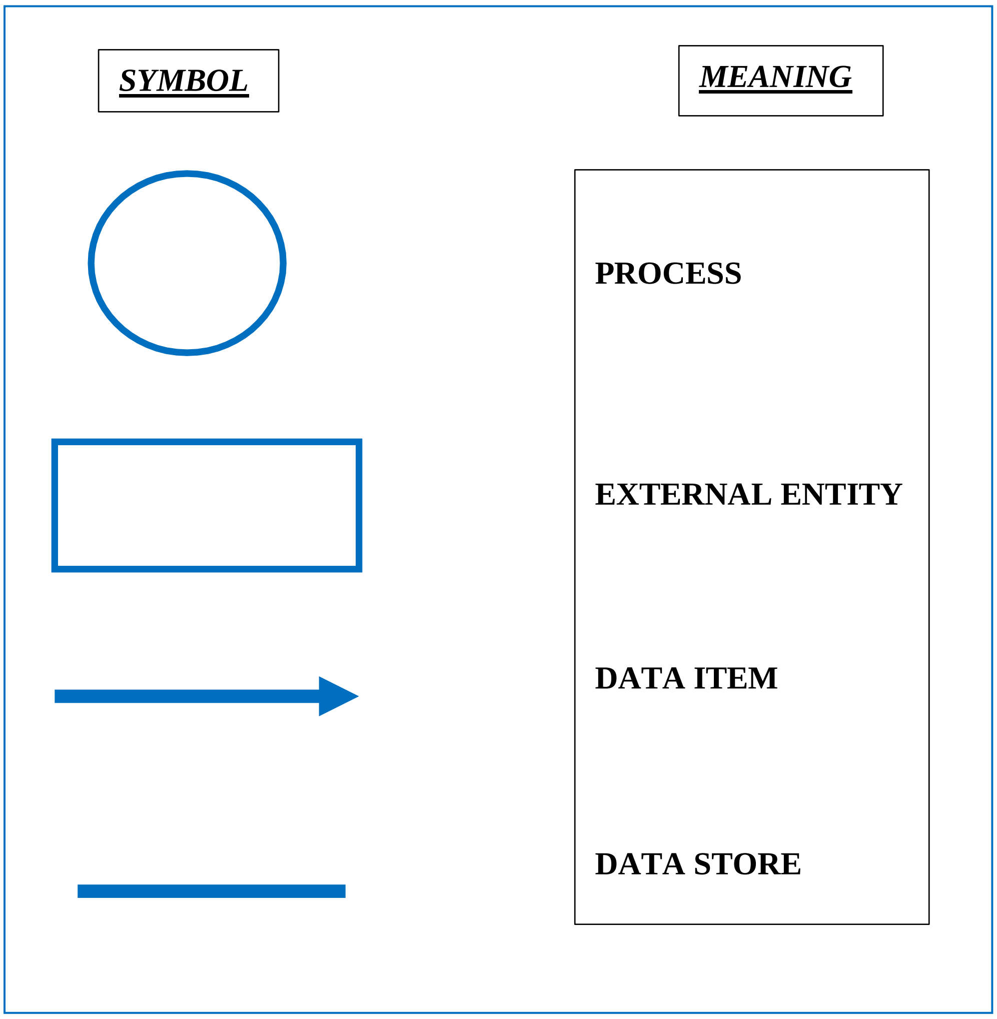


Fig 4.1

0 LEVEL DFD : -

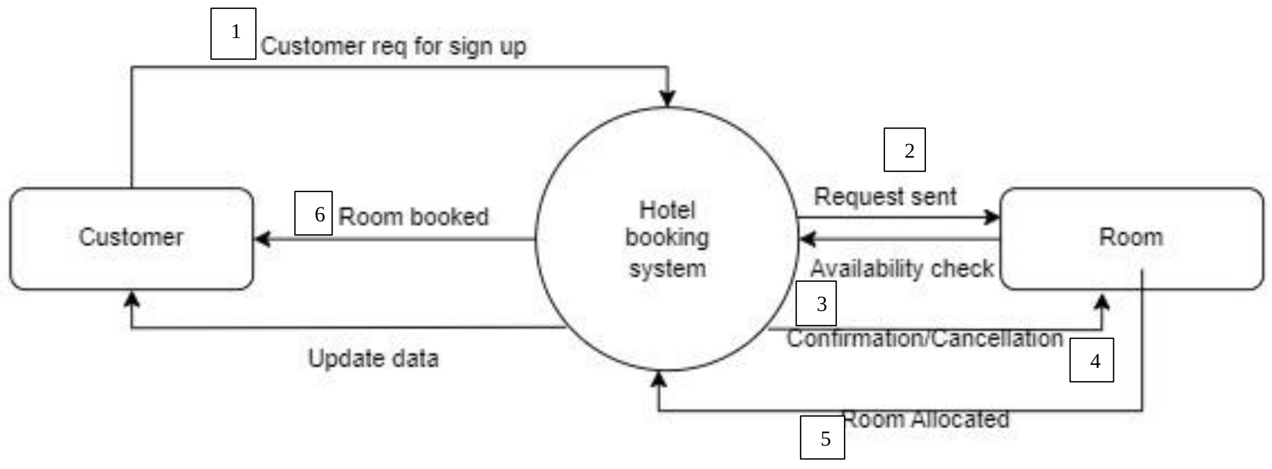


Fig 4.2

1 LEVEL DFD : -

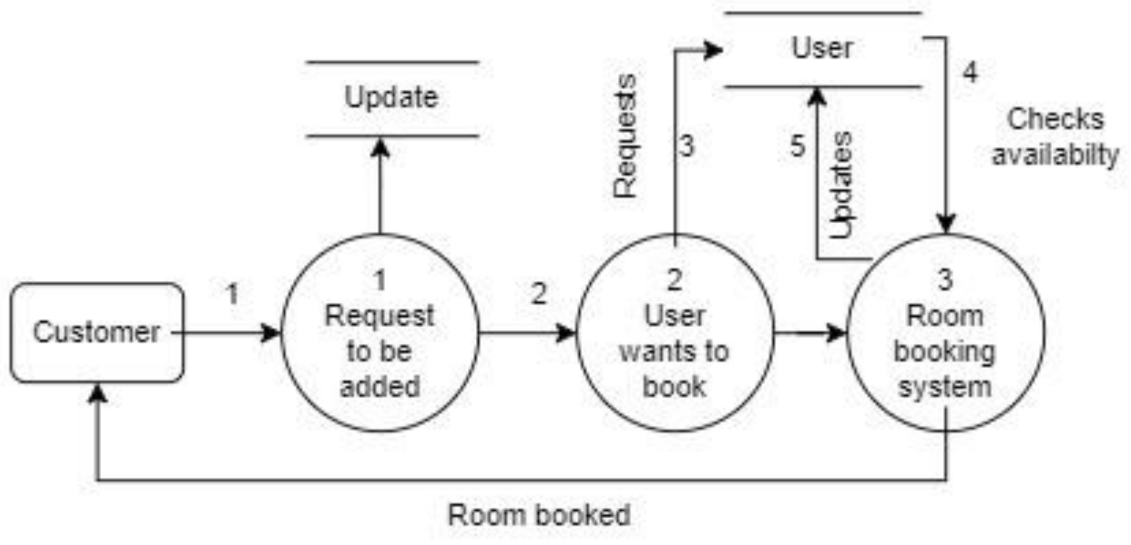


Fig 4.3

ENTITY RELATIONSHIP DIAGRAM (ER DIAGRAM):

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology (IT) system. An ERD uses data modelling techniques that can help define business processes and serve as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a reference point, should any debugging or business process re-engineering be needed later on.

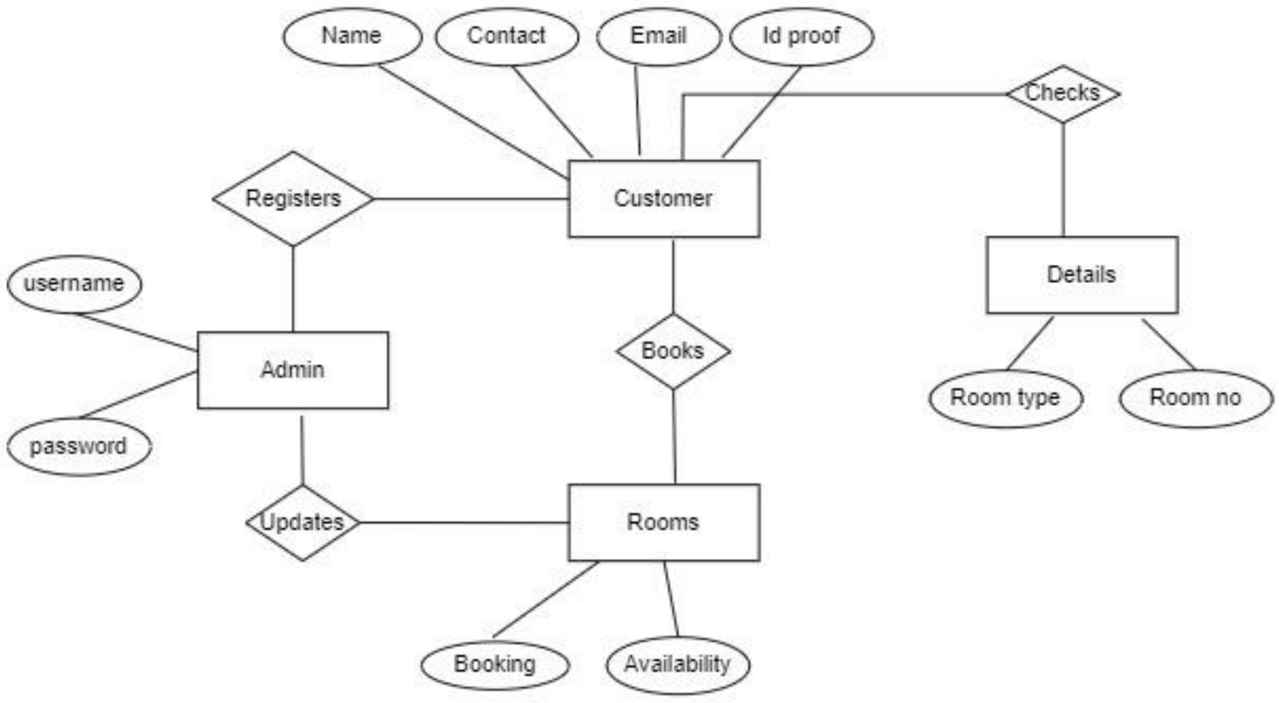


Fig 4.4

DESIGN DOCUMENT

1. **Modularisatio**
   When the system is broken down into small modules i.e. functions this process is referred to as modularization. In this project There are various modules which perform different functions. The modules are as follows :-

   Main Menu Form
   In this page, a list of options are available to the user each leading to a different form for eg :- the user can choose between 3 options for room booking, customer registrations, reports management and can choose the last option to exit the console.

   Item's Details
   Item's details is the page that will be available to the user after he/she login into the page. In this form, the details of all the available rooms and customers entries along with their bookings will be available to the admin e.g :- type , genre, price,

2. **Billing Interface**

   The user interface is made in such a way that user can use it easily. Just as I declared earlier this project is dependent software's that supports almost all IDE's.

3. **Integrity Constraints**

- Integrity Constraints are a set of rules. It is used to maintain the quality information.
- Integrity Constraints Ensure that the data insertion, deletion, and other processes have to be performed in such a way that data integrity is not affected
- Thus, integrity constraints are used to guard against accidental damage to the database.

PROGRAM CODE

module 1 SIGN UP

```
import tkinter as tk
from tkinter import *
from tkinter import messagebox
from PIL import Image, ImageTk
import re
import mysql.connector


def validate_email(email):
    # Regular expression pattern to validate email format
    pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'
    return re.match(pattern, email)


def validate_password(password):
    # Password validation criteria (example):
    # At least 8 characters long and contains at least one digit and one special character
    pattern = r'^(?=.*\d)(?=.*[@#$%^&+=]).{8,}$'
    return re.match(pattern, password)


def validate_fields():
    email_id = songInfo1.get()
    username = songInfo2.get()
    password = songInfo3.get()
    confirm_password = songInfo4.get()
    if email_id == "" or username == "" or password == "" or confirm_password == "":
        messagebox.showerror("Error", "Please fill in all the fields.")
        return False

    if not validate_email(email_id):
        messagebox.showerror("Error", "Invalid email format.")
        return False

    if len(username) < 6:
        messagebox.showerror("Error", "Username must be at least 6 characters long.")
        return False

    if not validate_password(password):
        messagebox.showerror("Error", "Invalid password format. Password must be at least 8
characters long and contain at least one digit and one special character.")
        return False

    if password != confirm_password:
        messagebox.showerror("Error", "Passwords do not match.")
        return False

    return True



def songadd():
    if not validate_fields():
        return

    email_id = songInfo1.get()
    username = songInfo2.get()
    password = songInfo3.get()
    query = "INSERT INTO login (email_id, username, password) VALUES (%s, %s, %s)"
    values = (email_id, username, password)

    try:
        cur.execute(query, values)
        con.commit()
        messagebox.showinfo("Success", "Account added successfully.")
        root.destroy()
        import login_page
    except mysql.connector.Error as e:
        messagebox.showerror("Error", f"Error adding Account: {e}")


def addsong():
    global img, songInfo1, songInfo2, songInfo3, songInfo4, songInfo5, songInfo6, con, cur,
root

    root = tk.Tk()
    root.title("SIGN UP PAGE")
    root.minsize(width=400, height=400)
    root.geometry("1550x800+0+0")

    con = mysql.connector.connect(host="localhost", user="root",
password="kaushik@2918", database="python_connector")
    cur = con.cursor()

    headingFrame1 = tk.Frame(root, bg="#2f2e2e", bd=15)
    headingFrame1.place(relx=0.25, rely=0.1, relwidth=0.5, relheight=0.13)

    headingLabel = tk.Label(headingFrame1, text="Please Sign Up First", font='Helvetica 14
bold', bg="#FAFAD2", fg='black')
    headingLabel.place(relx=0, rely=0, relwidth=1, relheight=1)
    labelFrame = tk.Frame(root, bg="#1abc9c")
    labelFrame.place(relx=0.1, rely=0.4, relwidth=0.8, relheight=0.4)

    lb1 = tk.Label(labelFrame, text="Email id:", font='Helvetica 13 bold', bg="#1abc9c",
fg='white')
    lb1.place(relx=0.05, rely=0.10, relheight=0.08)

    songInfo1 = tk.Entry(labelFrame)
    songInfo1.place(relx=0.3, rely=0.10, relwidth=0.62, relheight=0.08)

    lb2 = Label(labelFrame, text="Username:", font='Helvetica 13 bold', bg="#1abc9c",
fg='white')
    lb2.place(relx=0.05, rely=0.26, relheight=0.08)

    songInfo2 = Entry(labelFrame)
    songInfo2.place(relx=0.3, rely=0.26, relwidth=0.62, relheight=0.08)

    lb3 = Label(labelFrame, text="Password:", font='Helvetica 13 bold', bg="#1abc9c",
fg='white')
    lb3.place(relx=0.05, rely=0.42, relheight=0.08)

    songInfo3 = Entry(labelFrame, show="*")  # Use show="*" to hide password input
    songInfo3.place(relx=0.3, rely=0.42, relwidth=0.62, relheight=0.08)

    lb4 = Label(labelFrame, text="Confirm Password:", font='Helvetica 13 bold',
bg="#1abc9c", fg='white')
    lb4.place(relx=0.05, rely=0.58, relheight=0.08)

    songInfo4 = Entry(labelFrame, show="*")  # Use show="*" to hide password input
    songInfo4.place(relx=0.3, rely=0.58, relwidth=0.62, relheight=0.08)
    SubmitBtn = Button(root, text="Submit", font='Helvetica 12 bold', bg='#FF5733',
fg='white', command=songadd)
    SubmitBtn.place(relx=0.28, rely=0.9, relwidth=0.18, relheight=0.08)

    quitBtn = Button(root, text="Quit", font='Helvetica 12 bold', bg='#FF5733', fg='white',
command=root.destroy)
    quitBtn.place(relx=0.53, rely=0.9, relwidth=0.18, relheight=0.08)

    root.mainloop()


addsong()
```

module 2 LOGIN PAGE

```
import os
import tkinter as tk
from tkinter import *
from tkinter import messagebox
from PIL import Image, ImageTk
import re
import mysql.connector


def validate_login():
    username = songInfo1.get()
    password = songInfo2.get()

    if username == "" or password == "":
        messagebox.showerror("Error", "Please enter username and password.")
        return False

    # Check if the username and password exist in the database
    query = "SELECT * FROM login WHERE username = %s AND password = %s"
    values = (username, password)

    cur.execute(query, values)
    result = cur.fetchone()

    if result:
        messagebox.showinfo("Success", "Login successful.")
        #root.destroy()
        mainpage()

    else:
        messagebox.showerror("Error", "Invalid username or password.")



def mainpage():
    root.destroy()
    os.system("python hotel.py")



def forgot_password():
    global root_forgot

    root_forgot = tk.Tk()
    root_forgot.title("FORGOT PASSWORD")
    root_forgot.geometry("1550x800+0+0")

    forgot_frame = tk.Frame(root_forgot, bg="#2f2e2e", bd=25)
    forgot_frame.place(relx=0.25, rely=0.1, relwidth=0.5, relheight=0.2)
    forgot_label = tk.Label(forgot_frame, text="Forgot Password", font='Helvetica 14 bold',
bg="#FAFAD2", fg='black')
    forgot_label.place(relx=0, rely=0, relwidth=1, relheight=1)

    username_label = tk.Label(root_forgot, text="Username:", font='Helvetica 12 bold')
    username_label.place(relx=0.1, rely=0.4, relheight=0.1)

    username_entry = tk.Entry(root_forgot)
    username_entry.place(relx=0.35, rely=0.4, relwidth=0.5, relheight=0.1)

    password_label = tk.Label(root_forgot, text="Password:", font='Helvetica 12 bold')
    password_label.place(relx=0.1, rely=0.5, relheight=0.1)

    password_entry = tk.Entry(root_forgot)
    password_entry.place(relx=0.35, rely=0.5, relwidth=0.5, relheight=0.1)

    confirm_password_label = tk.Label(root_forgot, text="Confirm Password:",
font='Helvetica 12 bold')
    confirm_password_label.place(relx=0.1, rely=0.6, relheight=0.1)

    confirm_password_entry = tk.Entry(root_forgot)
    confirm_password_entry.place(relx=0.35, rely=0.6, relwidth=0.5, relheight=0.1)

    def reset_password():
        if username_entry.get() == '' or password_entry.get() == '' or
confirm_password_entry.get() == '':
            messagebox.showerror('Error', 'All Fields Are Required', parent=root_forgot)
        elif password_entry.get() != confirm_password_entry.get():
            messagebox.showerror('Error', 'Password and confirm password do not match',
parent=root_forgot)
        else:
            con = mysql.connector.connect(host="localhost", user="root",
password="kaushik@2918", database="python_connector")
            mycursor = con.cursor()
            query = 'SELECT * FROM login WHERE username=%s'
            mycursor.execute(query, (username_entry.get(),))
            row = mycursor.fetchone()
            if row is None:
                messagebox.showerror('Error', 'Incorrect Username', parent=root_forgot)
            else:
                query = 'UPDATE login SET password=%s WHERE username=%s'
                mycursor.execute(query, (password_entry.get(), username_entry.get()))
                con.commit()
                con.close()
                messagebox.showinfo('Success', 'Password has been reset. Please login with the
new password',
                                    parent=root_forgot)
                root_forgot.destroy()

    reset_button = tk.Button(root_forgot, text="Reset Password", font='Helvetica 14 bold',
bg='#FF5733', fg='white', command=reset_password)
    reset_button.place(relx=0.35, rely=0.7, relwidth=0.2, relheight=0.1)
    cancel_button = tk.Button(root_forgot, text="Cancel", font='Helvetica 14 bold',
bg='#FF5733', fg='white',command=root_forgot.destroy)
    cancel_button.place(relx=0.65, rely=0.7, relwidth=0.2, relheight=0.1)

    root_forgot.mainloop()



def login_pa():
    root.destroy()
    os.system("python signup.py")
```

```
def login_page():
    global img, songInfo1, songInfo2, con, cur, root

    root = tk.Tk()
    root.title("LOGIN PAGE")
    root.geometry("1550x800+0+0")

    con = mysql.connector.connect(host="localhost", user="root",
password="kaushik@2918", database="python_connector")
    cur = con.cursor()

    headingFrame1 = tk.Frame(root, bg="#2f2e2e", bd=15)
    headingFrame1.place(relx=0.25, rely=0.1, relwidth=0.5, relheight=0.13)

    headingLabel = tk.Label(headingFrame1, text="Login to continue", font='Helvetica 14
bold', bg="#FAFAD2", fg='black')
    headingLabel.place(relx=0, rely=0, relwidth=1, relheight=1)

    labelFrame = tk.Frame(root, bg="#1abc9c")
    labelFrame.place(relx=0.1, rely=0.4, relwidth=0.8, relheight=0.4)

    lb1 = tk.Label(labelFrame, text="Username:", font='Helvetica 13 bold', bg="#1abc9c",
fg='white')
    lb1.place(relx=0.05, rely=0.3, relheight=0.08)

    songInfo1 = tk.Entry(labelFrame)
    songInfo1.place(relx=0.3, rely=0.3, relwidth=0.62, relheight=0.08)

    lb2 = Label(labelFrame, text="Password:", font='Helvetica 13 bold', bg="#1abc9c",
fg='white')
    lb2.place(relx=0.05, rely=0.5, relheight=0.08)
    songInfo2 = Entry(labelFrame, show="*")
    songInfo2.place(relx=0.3, rely=0.5, relwidth=0.62, relheight=0.08)

    loginBtn = Button(root, text="Login", font='Helvetica 12 bold', bg='#FF5733', fg='white',
command=validate_login)
    loginBtn.place(relx=0.30, rely=0.7, relwidth=0.18, relheight=0.08)

    singupBtn = Button(root, text="Sign up", font='Helvetica 12 bold', bg='#FF5733',
fg='white', command=login_pa)
    singupBtn.place(relx=0.10, rely=0.7, relwidth=0.18, relheight=0.08)

    forgotBtn = Button(root, text="Forgot Password", font='Helvetica 10 bold', bg='#FF5733',
fg='white', command=forgot_password)
    forgotBtn.place(relx=0.50, rely=0.7, relwidth=0.18, relheight=0.08)

    quitBtn = Button(root, text="Quit", font='Helvetica 12 bold', bg='#FF5733', fg='white',
command=root.destroy)
    quitBtn.place(relx=0.70, rely=0.7, relwidth=0.18, relheight=0.08)

    root.mainloop()


login_page()
```

module 3 HOTEL

```
from tkinter import*
from PIL import Image,ImageTk
from customer import Customer_window
from room import Roombooking
from details import DetailsRoom
from reports import ReportsRoom

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



        #top photo

img1=Image.open(r"C:\Users\HIMANSHU\Desktop\hmspycharm\images\wallpaperflare.co
m_wallpaper.png")
        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)

        #logo of hotel
        img2=Image.open(r"C:\Users\HIMANSHU\Desktop\hmspycharm\images\logo.png")
        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)



        # title
        lbl_title=Label(self.root,text="WE'VE BEEN EXPECTING YOU!",font=("times new
roman",40,"bold"),bg="black",fg="gold",bd=4,relief=RIDGE)
        lbl_title.place(x=0,y=140,width=1550,height=50)
        #frame start
        main_frame=Frame(self.root,bd=4,relief=RIDGE)
        main_frame.place(x=0,y=190,width=1550,height=620)

        #main menu
        lbl_menu = Label(main_frame, text="MENU", font=("times new roman", 20, "bold"),
bg="black",
                   fg="gold", bd=4, relief=RIDGE)
        lbl_menu.place(x=0, y=0, width=230)

        # frame of all options
        button_frame = Frame(main_frame, bd=4, relief=RIDGE)
        button_frame.place(x=0, y=35, width=228, height=190)

        customer_button=Button(button_frame,text="CUSTOMER",
command=self.customer_details,width=22 ,font=("times new roman", 14, "bold"),
bg="black",fg="gold",bd=1,cursor="hand1")
        customer_button.grid(row=0,column=0)

        rooms_button = Button(button_frame, text="ROOMS BOOKING",
command=self.roombooking,width=22, font=("times new roman", 14, "bold"),
                          bg="black", fg="gold", bd=1, cursor="hand1")
        rooms_button.grid(row=1, column=0)

        details_button = Button(button_frame, text="ROOM DETAILS",
command=self.details_room,width=22, font=("times new roman", 14, "bold"),
                          bg="black", fg="gold", bd=1, cursor="hand1")
        details_button.grid(row=2, column=0)

        reports_button = Button(button_frame, text="REPORTS",
command=self.Report_room,width=22,font=("times new roman", 14, "bold"),
                              bg="black", fg="gold", bd=1, cursor="hand1")
        reports_button.grid(row=3, column=0)

        logout_button = Button(button_frame, text="EXIT", width=22,
command=root.destroy,font=("times new roman", 14, "bold"),
                              bg="black", fg="gold", bd=1, cursor="hand1")
        logout_button.grid(row=4, column=0)

        img3 =
Image.open(r"C:\Users\HIMANSHU\Desktop\hmspycharm\images\roombed.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)

        #menu down images
        img4 =
Image.open(r"C:\Users\HIMANSHU\Desktop\hmspycharm\images\reception.jpg")
        img4 = img4.resize((230, 210), Image.LANCZOS)
        self.photoimg4 = ImageTk.PhotoImage(img4)
        lblimg1 = Label(main_frame, image=self.photoimg4, bd=4, relief=RIDGE)
        lblimg1.place(x=0, y=225, width=230, height=210)

        img5 =
Image.open(r"C:\Users\HIMANSHU\Desktop\hmspycharm\images\maldives.jpg")
        img5 = img5.resize((230, 190), Image.LANCZOS)
        self.photoimg5 = ImageTk.PhotoImage(img5)
        lblimg1 = Label(main_frame, image=self.photoimg5, bd=4, relief=RIDGE)
        lblimg1.place(x=0, y=420, width=230, height=190)

    def customer_details(self):
        self.new_window=Toplevel(self.root)
        self.app=Customer_window(self.new_window)


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

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

    def Report_room(self):
        self.new_window = Toplevel(self.root)
        self.app = ReportsRoom(self.new_window)



if __name__ == '__main__':
    root=Tk()
    obj = HotelManagementSystem(root)
    root.mainloop()
```

module 4 CUSTOMER

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

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



        #variables of sql and random no generate
        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_pincode = 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()

        def validate_contact(event=None):
            contact = txtMobile.get()
            if contact and not re.match(r'^\d{10}$', contact):
                messagebox.showerror("Invalid Input", "Contact should be a 10-digit number")
                txtMobile.focus_set()

        def validate_email(event):
            email = txtEmail.get()
            if email and not re.match(r'^[\w\.-]+@[\w\.-]+\.\w+$', email):
                messagebox.showerror("Invalid Input", "Invalid email format")
                txtEmail.focus_set()

        def validate_pincode(event=None):
            pincode = txtPinCode.get()
            if pincode and not re.match(r'^\d{6}$', pincode):
                messagebox.showerror("Invalid Input", "Pincode should be a 6-digit number")
                txtPinCode.focus_set()

        def validate_name(event=None):
            name = txtcname.get()
            if name and len(name) < 3:
                messagebox.showerror("Invalid Input", "Name should have at least 3 characters")
                txtcname.focus_set()



        def validate_id(event=None):
            id = txtIdNumber.get()
            if id and not re.match(r'^\d{12}$', id):
                messagebox.showerror("Invalid Input", "Input should be a 12-digit number")
                txtIdNumber.focus_set()

        # title
        lbl_title = Label(self.root, text="ADD CUSTOMER DETAILS",font=("times new
roman", 18, "bold"), bg="black",fg="gold", bd=4, relief=RIDGE)
        lbl_title.place(x=0, y=0, width=1295, height=50)
        # logo of hotel
        img2 = Image.open(r"C:\Users\HIMANSHU\Desktop\hmspycharm\images\logo.png")
        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)

        # label frames
        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 entries
        #customer reference
        lbl_cust_ref=Label(labelframeleft,text="Customer Ref:",font=("arial", 12,
"bold"),padx=2, pady=6)
        lbl_cust_ref.grid(row=0,column=0,sticky=W)

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

        #customer 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,width=29,
font=("times new roman", 13, "bold"))
        txtcname.grid(row=1, column=1)
        txtcname.bind("<FocusOut>", validate_name)

        #mothers name
        lblmname = Label(labelframeleft,font=("arial", 12, "bold"),text="Mother's Name:",
padx=2, pady=6)
        lblmname.grid(row=2, column=0, sticky=W)
        txtmname = ttk.Entry(labelframeleft,  textvariable=self.var_mother,width=29,
font=("times new roman", 13, "bold"))
        txtmname.grid(row=2, column=1)
        #gender selection
        label_gender = Label(labelframeleft,  font=("arial", 12, "bold"),text="Gender:", padx=2,
pady=6)
        label_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"] = ("Select","Male", "Female", "Others")
        combo_gender.current(0)
        combo_gender.grid(row=3, column=1)

        #pincode
        lblPinCode = Label(labelframeleft,  font=("arial", 12, "bold"), text="Pincode:",padx=2,
pady=6)
        lblPinCode.grid(row=4, column=0, sticky=W)
        txtPinCode = ttk.Entry(labelframeleft,textvariable=self.var_pincode,width=29,
font=("times new roman", 13, "bold"))
        txtPinCode.grid(row=4, column=1)
        txtPinCode.bind("<FocusOut>", validate_pincode)

        # contact 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, width=29,
font=("arial", 13, "bold"))
        txtMobile.grid(row=5, column=1)
        txtMobile.bind("<FocusOut>", validate_contact)

        # mail address
        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, width=29,
font=("arial", 13, "bold"))
        txtEmail.grid(row=6, column=1)
        txtEmail.bind("<FocusOut>", validate_email)

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

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

        # idtype
        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"] = ("Adhaar Card","Others")
        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,width=29,
font=("arial", 13, "bold"))
        txtIdNumber.grid(row=9, column=1)
        txtIdNumber.bind("<FocusOut>", validate_id())

        # 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,width=29,
font=("arial", 13, "bold"))
        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", 11,
"bold"),bg="black",fg="gold",width=10)
        btnAdd.grid(row=0,column=0,padx=1)

        btnUpdate = Button(btn_frame, text="Update",command=self.update, font=("arial", 11,
"bold"), bg="black", fg="gold", width=10)
        btnUpdate.grid(row=0, column=1, padx=1)

        btnDelete = Button(btn_frame, text="Delete",command=self.mDelete
                    , font=("arial", 11, "bold"), bg="black", fg="gold", width=10)
        btnDelete.grid(row=0, column=2, padx=1)

        btnReset = Button(btn_frame, text="Reset", command=self.reset,font=("arial", 11,
"bold"), bg="black", fg="gold", width=10)
```

```
        btnReset.grid(row=0, column=3, padx=1)


        #label frame and search options
        Table_Frame = LabelFrame(self.root, bd=2, relief=RIDGE, text="View Details and
Search",
                                    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="red",fg="white", padx=2, pady=6)
        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", "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, width=24,
font=("arial", 13, "bold"))
        txtSearch.grid(row=0, column=2,padx=2)


        btnSearch = Button(Table_Frame, text="Search", command=self.search,font=("arial",
11, "bold"), bg="black", fg="gold", width=10)
        btnSearch.grid(row=0, column=3, padx=1)


        btnShowallTable_Frame = Button(Table_Frame, text="Show all",
command=self.fetch_data,font=("arial", 11, "bold"), bg="black", fg="gold", width=10)
        btnShowallTable_Frame.grid(row=0, column=4, padx=1)
        # data tables
        details_table = Frame(Table_Frame, bd=2, relief=RIDGE)
        details_table.place(x=0, y=50, width=860, height=350)

        scroll_x=ttk.Scrollbar(details_table,orient=HORIZONTAL)
        scroll_y=ttk.Scrollbar(details_table,orient=VERTICAL)

        self.Cust_Details_Table=ttk.Treeview(details_table, column=("ref", "name", "mother
name", "gender", "pincode" ,"mobile",
                                                        "email", "nationality", "idproof", "id number",
"address")
                                        , xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)

        scroll_x.pack(side=BOTTOM,fill=X)
        scroll_y.pack(side=RIGHT, fill=Y)

        scroll_x.config(command=self.Cust_Details_Table.xview)
        scroll_y.config(command=self.Cust_Details_Table.yview)

        self.Cust_Details_Table.heading("ref",text="Refer No")
        self.Cust_Details_Table.heading("name",text="Name")
        self.Cust_Details_Table.heading("mother name",text="Mother's Name")
        self.Cust_Details_Table.heading("gender",text="Gender")
        self.Cust_Details_Table.heading("pincode",text="Pincode")
        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="IDProof")
        self.Cust_Details_Table.heading("id number",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 name",width=100)
        self.Cust_Details_Table.column("gender",width=100)
        self.Cust_Details_Table.column("pincode",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("id number",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","Please fill all the details",parent=self.root)
        else :
            try:

conn=mysql.connector.connect(host="localhost",username="root",password="kaushik@291
8",database="python_connector")
                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_pincode.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 data added",parent=self.root)
            except Exception as es:
                messagebox.showwarning("Something went wrong:{str(es)}",parent=self.root)

    def fetch_data(self):
        conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                    database="python_connector")
        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_pincode.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="localhost", username="root",
password="kaushik@2918",
                                           database="python_connector")
            my_cursor = conn.cursor()
            my_cursor.execute("update customer set Name=%s,`Mother
Name`=%s,Gender=%s,Pincode=%s,Mobile=%s,Email=%s,Nationality=%s,Idproof=%s,Idn
umber=%s,Address=%s where Ref=%s",(
                self.var_cust_name.get(),
                self.var_mother.get(),
                self.var_gender.get(),
                self.var_pincode.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("Updated","Customer details has been updated
successfully",parent=self.root)

    def mDelete(self):
        mDelete=messagebox.askyesno("Hotel Management System", "Do you want to delete
the selected data?",parent=self.root)
        if mDelete>0:
            conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                           database="python_connector")
            my_cursor = conn.cursor()
            query="delete from customer where Ref=%s"
            value=(self.var_ref.get(),)
            my_cursor.execute(query,value)
        else:
            if not mDelete:
                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_pincode.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="localhost", username="root",
password="kaushik@2918",
                                       database="python_connector")
        my_cursor = conn.cursor()
        my_cursor.execute("select * from customer where "+str(self.search_var.get())+" LIKE
'%"+str(self.txt_search.get())+"%'")
        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()
```

```
if __name__ == '__main__':
    root=Tk()
    obj=Customer_window(root)
    root.mainloop()
```

module 5 ROOM BOOKING

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

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

        #variables to store data

        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_actualcost = StringVar()
        self.var_total = StringVar()

        # title
        lbl_title = Label(self.root, text="Room Booking Details", font=("times new roman", 18,
"bold"), bg="black",
                          fg="gold", bd=4, relief=RIDGE)
        lbl_title.place(x=0, y=0, width=1295, height=50)

        # logo of hotel
        img2 = Image.open(r"C:\Users\HIMANSHU\Desktop\hmspycharm\images\logo.png")
        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)

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

        # customer reference
        lbl_cust_contact= Label(labelframeleft, text="Customer CONTACT:", font=("arial",
12, "bold"), padx=2, pady=6)
        lbl_cust_contact.grid(row=0, column=0, sticky=W)
        entry_contact = ttk.Entry(labelframeleft,
textvariable=self.var_contact,width=20,font=("times new roman", 13, "bold"))
        entry_contact.grid(row=0, column=1,sticky=W)




        # button for fetching data
        btnFetchdata = Button(labelframeleft, text="Fetch Data",
command=self.Fetch_contact,font=("arial", 8, "bold"), bg="black", fg="gold",
                    width=8)

        btnFetchdata.place(x=347,y=4)

        #check in date
        today = datetime.date.today()
        check_in_date=Label(labelframeleft,font=("arial",12,"bold"),text="Check-in
Date:",padx=2,pady=6)
        check_in_date.grid(row=1,column=0,sticky=W)

txtcheck_in_date=DateEntry(labelframeleft,textvariable=self.var_checkin,mindate=today,fo
nt=("arial",13,"bold"),width=22)
        txtcheck_in_date.grid(row=1,column=1)

        # check out date
        check_out_date = Label(labelframeleft, font=("arial", 12, "bold"), text="Check-out
Date:", padx=2, pady=6)
        check_out_date.grid(row=2, column=0, sticky=W)
        txtcheck_out_date = DateEntry(labelframeleft,
textvariable=self.var_checkout,mindate=today,font=("arial", 13, "bold"), width=22)
        txtcheck_out_date.grid(row=2, column=1)

        # room type
        lblRoomType = Label(labelframeleft, font=("arial", 12, "bold"), text="Room Type:",
padx=2, pady=6)
        lblRoomType.grid(row=3, column=0, sticky=W)

        conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                       database="python_connector")
        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=("ari
al",12,"bold"),width=27,state="readonly")
        combo_RoomType["value"]=ide
        combo_RoomType.current(0)
        combo_RoomType.grid(row=3,column=1)

        # room availability
        lblRoomAvailable = Label(labelframeleft, font=("arial", 12, "bold"), text="Available
Rooms:", padx=2, pady=6)
        lblRoomAvailable.grid(row=4, column=0, sticky=W)
        #txtRoomAvailable = ttk.Entry(labelframeleft,
textvariable=self.var_roomavailable,font=("arial", 13, "bold"), width=29)
        #txtRoomAvailable.grid(row=4, column=1)

        conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                       database="python_connector")
        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)



        #meals
        lblMeal = Label(labelframeleft, font=("arial", 12, "bold"), text="Food:", padx=2,
pady=6)
        lblMeal.grid(row=5, column=0, sticky=W)
        combo_Meal = ttk.Combobox(labelframeleft, textvariable=self.var_meal, font=("arial",
12, "bold"),
                                    width=27, state="readonly")
        combo_Meal["value"] = ("Select", "Breakfast & Lunch (Rs 300)", "Only Lunch (Rs
150)", "Lunch & Dinner (Rs 350)", "Breakfast & Dinner (Rs 300)", "3 Meals (Rs 450)", "No
meals")
        combo_Meal.current(0)
        combo_Meal.grid(row=5, column=1)



        #number 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)
        txtNoOfDays = ttk.Entry(labelframeleft, textvariable=self.var_noOfdays,font=("arial",
13, "bold"), width=29)
        txtNoOfDays.grid(row=6, column=1)




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



        # sub total
        lblNoOfDays = Label(labelframeleft, font=("arial", 12, "bold"), text="Room charge:",
padx=2, pady=6)
        lblNoOfDays.grid(row=8, column=0, sticky=W)
        txtNoOfDays = ttk.Entry(labelframeleft,
textvariable=self.var_actualcost,font=("arial",13, "bold"), width=29)
        txtNoOfDays.grid(row=8, column=1)



        # total calculation
        lblIDNumber = Label(labelframeleft, font=("arial", 12, "bold"), text="Total Cost :",
padx=2, pady=6)
        lblIDNumber.grid(row=9, column=0, sticky=W)
        txtIDNumber = ttk.Entry(labelframeleft,textvariable=self.var_total, font=("arial", 13,
"bold"), width=29)
        txtIDNumber.grid(row=9, column=1)



        # bill button on room

        btnBill = Button(labelframeleft, text="BILL",command=self.total, font=("arial", 11,
"bold"), bg="black", fg="gold",
                    width=10)
        btnBill.grid(row=10, column=0, padx=1, sticky=W)
        # buttons from customer frame

        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", 11,
"bold"), bg="black", fg="gold",
                        width=10)
        btnAdd.grid(row=0, column=0, padx=1)

        btnUpdate = Button(btn_frame, text="Update",command=self.update,font=("arial", 11,
"bold"), bg="black",
                           fg="gold", width=10)
        btnUpdate.grid(row=0, column=1, padx=1)

        btnDelete = Button(btn_frame, text="Delete"
                           ,command=self.mDelete, font=("arial", 11, "bold"), bg="black", fg="gold",
width=10)
        btnDelete.grid(row=0, column=2, padx=1)

        btnReset = Button(btn_frame, text="Reset", command=self.reset,font=("arial", 11,
"bold"), bg="black",
                          fg="gold", width=10)
        btnReset.grid(row=0, column=3, padx=1)


        # right side image in room window

        img3 =
Image.open(r"C:\Users\HIMANSHU\Desktop\hmspycharm\images\roombed.jpg")
```

```
        img3 = img3.resize((520, 300), 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)



        #search system in room window

        Table_Frame = LabelFrame(self.root, bd=2, relief=RIDGE, text="View Details and
Search",
                                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="red", fg="white", padx=2,
                            pady=6)
        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"] = ("Contact", "Room")
        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, width=24,
font=("arial", 13, "bold"))
        txtSearch.grid(row=0, column=2, padx=2)
        btnSearch = Button(Table_Frame, text="Search", command=self.search,font=("arial",
11, "bold"), bg="black",
                        fg="gold", width=10)
        btnSearch.grid(row=0, column=3, padx=1)

        btnShowallTable_Frame = Button(Table_Frame, text="Show all",
command=self.fetch_data,
                                font=("arial", 11, "bold"), bg="black", fg="gold", width=10)
        btnShowallTable_Frame.grid(row=0, column=4, padx=1)



        # data tables from customer window

        details_table = Frame(Table_Frame, bd=2, relief=RIDGE)
        details_table.place(x=0, y=50, width=860, height=180)

        scroll_x = ttk.Scrollbar(details_table, orient=HORIZONTAL)
        scroll_y = ttk.Scrollbar(details_table, orient=VERTICAL)

        self.room_table = ttk.Treeview(details_table,
                                       column=("contact", "checkin", "checkout",
"roomtype","roomavailable", "meal", "noOfdays")
                                       , xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)

        scroll_x.pack(side=BOTTOM, fill=X)
        scroll_y.pack(side=RIGHT, fill=Y)

        scroll_x.config(command=self.room_table.xview)
        scroll_y.config(command=self.room_table.yview)

        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 available")
        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()



    # adding data in boxes
    def add_data(self):
        if self.var_contact.get()=="" or self.var_checkin.get()=="":
            messagebox.showerror("ERROR","Please fill all the details",parent=self.root)
        else :
            try:

conn=mysql.connector.connect(host="localhost",username="root",password="kaushik@291
8",database="python_connector")
```

```
            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("Something went wrong:{str(es)}",parent=self.root)

    #fetching data from db
    def fetch_data(self):
        conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                       database="python_connector")
        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 button
    def update(self):
        if self.var_contact.get()=="":
            messagebox.showerror("ERROR","Please enter mobile number",parent=self.root)
        else:
            conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                           database="python_connector")
            my_cursor = conn.cursor()
            my_cursor.execute("update room set
check_in=%s,check_out=%s,roomtype=%s,roomavailable=%s,meal=%s,noOfdays=%s
where contact=%s",(
            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()
            ))



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



    # delete button work
    def mDelete(self):
        mDelete=messagebox.askyesno("Hotel Management System", "Do you want to delete
the selected data?",parent=self.root)
        if mDelete>0:
            conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                          database="python_connector")
            my_cursor = conn.cursor()
            query="delete from room where contact=%s"
            value=(self.var_contact.get(),)
            my_cursor.execute(query,value)
        else:
            if not mDelete:
                return
        conn.commit()
        self.fetch_data()
        conn.close()
    #reset button work
    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_actualcost.set("")
        self.var_total.set("")



    #data fetch by conatct


    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="localhost", username="root",
password="kaushik@2918",
                                    database="python_connector")
            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","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)

        # data fetch by gender

        conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                       database="python_connector")
        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)

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

        # data fetch by email

        conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                              database="python_connector")
        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)

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



        # data fetching from nationality

        conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                              database="python_connector")
        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)

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

```
        # data fetching from address

        conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                       database="python_connector")
        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)

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

  #search system in room window
  def search(self):
    conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                   database="python_connector")
    my_cursor = conn.cursor()
    my_cursor.execute("select * from room where "+str(self.search_var.get())+" LIKE
'%"+str(self.txt_search.get())+"%'")
    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()

# working on bill button to calculate no of days and amounts
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 & Lunch" and self.var_roomtype.get()=="Single"):
        q1=float(300)
        q2=float(800)
        q3=float(self.var_noOfdays.get())
        q4=float(q1+q2)
        q5=float(q3 * q4)
        Tax="Rs."+str("%.2f"%((q5)*0.1))
        ST="Rs."+str("%.2f"%((q5)))
        TT="Rs."+str("%.2f"%(q5+((q5)*0.1)))
        self.var_paidtax.set(Tax)
        self.var_actualcost.set(ST)
        self.var_total.set(TT)

    if (self.var_meal.get() == "Only Lunch" and self.var_roomtype.get() == "Single"):
        q1 = float(100)
        q2 = float(800)
        q3 = float(self.var_noOfdays.get())
        q4 = float(q1 + q2)
        q5 = float(q3 * q4)
        Tax = "Rs." + str("%.2f" % ((q5) * 0.1))
        ST = "Rs." + str("%.2f" % ((q5)))
```

```
            TT = "Rs." + str("%.2f" % (q5 + ((q5) * 0.1)))
            self.var_paidtax.set(Tax)
            self.var_actualcost.set(ST)
            self.var_total.set(TT)

        if (self.var_meal.get() == "Lunch & Dinner" and self.var_roomtype.get() == "Single"):
            q1 = float(100)
            q2 = float(800)
            q3 = float(self.var_noOfdays.get())
            q4 = float(q1 + q2)
            q5 = float(q3 * q4)
            Tax = "Rs." + str("%.2f" % ((q5) * 0.1))
            ST = "Rs." + str("%.2f" % ((q5)))
            TT = "Rs." + str("%.2f" % (q5 + ((q5) * 0.1)))
            self.var_paidtax.set(Tax)
            self.var_actualcost.set(ST)
            self.var_total.set(TT)

        if (self.var_meal.get() == "Breakfast & Dinner" and self.var_roomtype.get() ==
"Single"):
            q1 = float(250)
            q2 = float(800)
            q3 = float(self.var_noOfdays.get())
            q4 = float(q1 + q2)
            q5 = float(q3 * q4)
            Tax = "Rs." + str("%.2f" % ((q5) * 0.1))
            ST = "Rs." + str("%.2f" % ((q5)))
            TT = "Rs." + str("%.2f" % (q5 + ((q5) * 0.1)))
            self.var_paidtax.set(Tax)
            self.var_actualcost.set(ST)
            self.var_total.set(TT)
```

```
    if (self.var_meal.get() == "3 Meals" and self.var_roomtype.get() == "Single"):
        q1 = float(400)
        q2 = float(800)
        q3 = float(self.var_noOfdays.get())
        q4 = float(q1 + q2)
        q5 = float(q3 * q4)
        Tax = "Rs." + str("%.2f" % ((q5) * 0.1))
        ST = "Rs." + str("%.2f" % ((q5)))
        TT = "Rs." + str("%.2f" % (q5 + ((q5) * 0.1)))
        self.var_paidtax.set(Tax)
        self.var_actualcost.set(ST)
        self.var_total.set(TT)

    if (self.var_meal.get() == "Breakfast & Lunch" and self.var_roomtype.get() ==
"Luxury"):
        q1 = float(300)
        q2 = float(1500)
        q3 = float(self.var_noOfdays.get())
        q4 = float(q1 + q2)
        q5 = float(q3 * q4)
        Tax = "Rs." + str("%.2f" % ((q5) * 0.1))
        ST = "Rs." + str("%.2f" % ((q5)))
        TT = "Rs." + str("%.2f" % (q5 + ((q5) * 0.1)))
        self.var_paidtax.set(Tax)
        self.var_actualcost.set(ST)
        self.var_total.set(TT)

    if (self.var_meal.get() == "Only Lunch" and self.var_roomtype.get() == "Luxury"):
        q1 = float(100)
        q2 = float(1500)
        q3 = float(self.var_noOfdays.get())
        q4 = float(q1 + q2)
```

```
            q5 = float(q3 * q4)
            Tax = "Rs." + str("%.2f" % ((q5) * 0.1))
            ST = "Rs." + str("%.2f" % ((q5)))
            TT = "Rs." + str("%.2f" % (q5 + ((q5) * 0.1)))
            self.var_paidtax.set(Tax)
            self.var_actualcost.set(ST)
            self.var_total.set(TT)

        if (self.var_meal.get() == "Lunch & Dinner" and self.var_roomtype.get() == "Luxury"):
            q1 = float(100)
            q2 = float(1500)
            q3 = float(self.var_noOfdays.get())
            q4 = float(q1 + q2)
            q5 = float(q3 * q4)
            Tax = "Rs." + str("%.2f" % ((q5) * 0.1))
            ST = "Rs." + str("%.2f" % ((q5)))
            TT = "Rs." + str("%.2f" % (q5 + ((q5) * 0.1)))
            self.var_paidtax.set(Tax)
            self.var_actualcost.set(ST)
            self.var_total.set(TT)

        if (self.var_meal.get() == "Breakfast & Dinner" and self.var_roomtype.get() ==
"Luxury"):
            q1 = float(250)
            q2 = float(1500)
            q3 = float(self.var_noOfdays.get())
            q4 = float(q1 + q2)
            q5 = float(q3 * q4)
            Tax = "Rs." + str("%.2f" % ((q5) * 0.1))
            ST = "Rs." + str("%.2f" % ((q5)))
            TT = "Rs." + str("%.2f" % (q5 + ((q5) * 0.1)))
            self.var_paidtax.set(Tax)
            self.var_actualcost.set(ST)
            self.var_total.set(TT)

        if (self.var_meal.get() == "3 Meals" and self.var_roomtype.get() == "Luxury"):
            q1 = float(400)
            q2 = float(1500)
            q3 = float(self.var_noOfdays.get())
            q4 = float(q1 + q2)
            q5 = float(q3 * q4)
            Tax = "Rs." + str("%.2f" % ((q5) * 0.1))
            ST = "Rs." + str("%.2f" % ((q5)))
            TT = "Rs." + str("%.2f" % (q5 + ((q5) * 0.1)))
            self.var_paidtax.set(Tax)
            self.var_actualcost.set(ST)
            self.var_total.set(TT)

if __name__ == '__main__':
    root=Tk()
    obj=Roombooking(root)
    root.mainloop()
```

module 6 ROOM DETAILS

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


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

        # title
        lbl_title = Label(self.root, text="Room Booking Details", font=("times new roman", 18,
"bold"), bg="black",
                          fg="gold", bd=4, relief=RIDGE)
        lbl_title.place(x=0, y=0, width=1295, height=50)

        # logo of hotel
        img2 = Image.open(r"C:\Users\HIMANSHU\Desktop\hmspycharm\images\logo.png")
        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)

        # labels and entries
        labelframeleft = LabelFrame(self.root, bd=2, relief=RIDGE, text="Add a new room",
                                    font=("times new roman", 12, "bold"), padx=2)
        labelframeleft.place(x=5, y=50, width=540, height=350)

        # floor number
        lbl_floor = Label(labelframeleft, text="FLOOR", font=("arial", 12, "bold"), padx=2,
pady=6)
        lbl_floor.grid(row=0, column=0, sticky=W)
        self.var_floor=StringVar()
        entry_floor = ttk.Entry(labelframeleft,textvariable=self.var_floor, width=20,
                                font=("times new roman", 13, "bold"))
        entry_floor.grid(row=0, column=1, sticky=W)

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

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



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

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

        # buttons from customer frame

        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", 11,
"bold"), bg="black", fg="gold",
                        width=10)
        btnAdd.grid(row=0, column=0, padx=1)

        btnUpdate = Button(btn_frame, text="Update",  command=self.update,font=("arial",
11, "bold"), bg="black",
                         fg="gold", width=10)
        btnUpdate.grid(row=0, column=1, padx=1)

        btnDelete = Button(btn_frame, text="Delete"
                         ,  command=self.mDelete,font=("arial", 11, "bold"), bg="black", fg="gold",
width=10)
        btnDelete.grid(row=0, column=2, padx=1)

        btnReset = Button(btn_frame, text="Reset",  command=self.reset_data,font=("arial",
11, "bold"), bg="black",
                        fg="gold", width=10)
        btnReset.grid(row=0, column=3, padx=1)


        # search system in room window

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

        scroll_x = ttk.Scrollbar(Table_Frame, orient=HORIZONTAL)
        scroll_y = ttk.Scrollbar(Table_Frame, orient=VERTICAL)

        self.room_table = ttk.Treeview(Table_Frame,
                                    column=("floor", "roomno", "roomType")
                        , xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)

        scroll_x.pack(side=BOTTOM, fill=X)
        scroll_y.pack(side=RIGHT, fill=Y)

        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()



    #add button
    def add_data(self):
        if self.var_floor.get()=="" or self.var_RoomType.get()=="":
            messagebox.showerror("ERROR","Please fill all the details",parent=self.root)
        else :
            try:

conn=mysql.connector.connect(host="localhost",username="root",password="kaushik@291
8",database="python_connector")
                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",parent=self.root)
            except Exception as es:
                messagebox.showwarning("Something went wrong:{str(es)}",parent=self.root)



    # fetching data from details table from db
    def fetch_data(self):
        conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                       database="python_connector")
        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()



    #get cursor
    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 button
    def update(self):
        if self.var_floor.get()=="":
            messagebox.showerror("ERROR","Please enter floor number",parent=self.root)
        else:
            conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                           database="python_connector")
            my_cursor = conn.cursor()
            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("Updated","Room details has been updated
successfully",parent=self.root)
```

```
    # delete button work
    def mDelete(self):
        mDelete = messagebox.askyesno("Hotel Management System", "Do you want to delete
the selected data?",
                                      parent=self.root)
        if mDelete > 0:
            conn = mysql.connector.connect(host="localhost", username="root",
password="kaushik@2918",
                                           database="python_connector")
            my_cursor = conn.cursor()
            query = "delete from details where RoomNo=%s"
            value = (self.var_roomNo.get(),)
            my_cursor.execute(query, value)
        else:
            if not mDelete:
                return
        conn.commit()
        self.fetch_data()
        conn.close()


    #reset button functioning
    def reset_data(self):
        self.var_floor.set(""),
        self.var_roomNo.set(""),
        self.var_RoomType.set("")


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

module 7 REPORTS

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

class ReportsRoom:
    def __init__(self, root):
        self.root = root
        self.root.title("Reports Room")
        self.root.geometry("1295x550+230+220")

        # title
        lbl_title = Label(self.root, text="ALL RECORDS", font=("times new roman", 18,
"bold"), bg="black",
                          fg="gold", bd=4, relief=RIDGE)
        lbl_title.place(x=0, y=0, width=1295, height=50)

        # logo of hotel
        img2 = Image.open(r"C:\Users\HIMANSHU\Desktop\hmspycharm\images\logo.png")
        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)
        # search system in room window
        Table_Frame = LabelFrame(self.root, bd=2, relief=RIDGE, text="View Details and
Search",
                                 font=("times new roman", 12, "bold"), padx=2)
        Table_Frame.place(x=0, y=0, width=1295, height=350)

        # Treeview widget to display records
        scroll_x = Scrollbar(Table_Frame, orient=HORIZONTAL)
        scroll_y = Scrollbar(Table_Frame, orient=VERTICAL)
        self.room_table = ttk.Treeview(Table_Frame, columns=("name", "mobile", "checkin",
"checkout", "roomtype",
                                                  "roomno", "noOfdays"), xscrollcommand=scroll_x.set,
                                   yscrollcommand=scroll_y.set)
        scroll_x.pack(side=BOTTOM, fill=X)
        scroll_y.pack(side=RIGHT, fill=Y)
        scroll_x.config(command=self.room_table.xview)
        scroll_y.config(command=self.room_table.yview)
        self.room_table.heading("name", text="contact")
        self.room_table.heading("mobile", text="Check_in")
        self.room_table.heading("checkin", text="Check-Out")
        self.room_table.heading("checkout", text="Room type")
        self.room_table.heading("roomtype", text="Room No")
        self.room_table.heading("roomno", text="Meal")
        self.room_table.heading("noOfdays", text="no of days")
        self.room_table["show"] = "headings"
        self.room_table.pack(fill=BOTH, expand=1)

        # Call function to fetch and display records
        self.display_records()

    def display_records(self):
        # Connect to MySQL database and fetch records
        conn = mysql.connector.connect(host="localhost", user="root",
password="kaushik@2918", database="python_connector")
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM room")
        records = cursor.fetchall()



        # Clear existing records
        self.room_table.delete(*self.room_table.get_children())

        # Insert fetched records into Treeview
        for record in records:
            self.room_table.insert("", END, values=record)

        # Close cursor and connection
        cursor.close()
        conn.close()

if __name__ == '__main__':
    root = Tk()
    obj = ReportsRoom(root)
    root.mainloop()
```

TESTING & VALIDATIONS

SYSTEM TESTING

The definition of the quality software is that it meets the clients functional and performance requirements, has been developed and documented in adherence to sound standards and practices, is maintainable and can absorb changes by being flexible. The main aim of testing is not show the absence of errors but their presence. Testing is often conducted in a planned manner. This project follows a phased, feature-based approach and hence testing cannot be precisely broken down into a structured pattern. Although the unit, integration and acceptance test plans are followed, yet it is more of a feature-oriented testing.

System testing tests the integration of each module in the system. It also tests to find discrepancies between the system and its original objective, current specification, and system documentation. The primary concern is the compatibility of individual modules. The entire system is working properly or not will be tested here, and specified path ODBC connection will be correct or not, and giving output or not are tested here these verifications andvalidations are done by giving input values to the system and by comparing with the expectedoutput. Top-down testing implementing here.

TESTING

Testing is a process of executing a program with the intent of finding an error. Testing is a crucial element of software quality assurance and presents an ultimate review of specification, design, and coding.

A good test case has a high probability of finding an as undiscovered error. A successful test uncovers an as undiscovered error.

TESTING OBJECTIVE

1. Testing is a process of executing a program with the intent of finding an error
2. A good test case has a probability of finding an as yet undiscovered error
3. A successful test uncovers an undiscovered error.

TESTING PRINCIPLES

- All tests should be traceable to end-user requirement
- Tests should be planned long before testing begins
- Testing should begin on a small scale and progress towards testing on larg
- Exhaustive testing is not possible
- To be most effective testing should be conducted by an independent third party
- The primary objective for test case design is to derive a set of tests that has the highest livelihood for uncovering defects in software. To accomplish this objective two different categories of test case design techniques are used.

TEST CASES

Test cases are derived to ensure that all statements in the program have been executedat least once during testing and that all logical conditions have been executed.
Using White-Box testing methods, the software engineer can drive test cases that

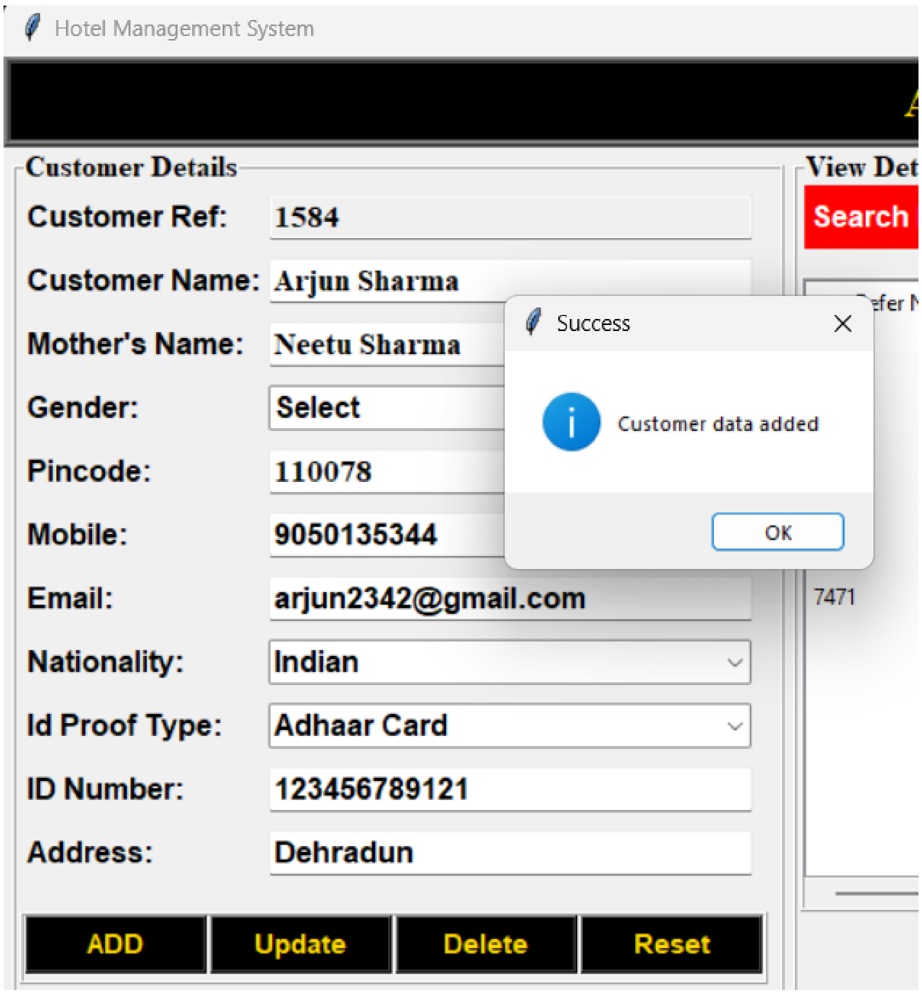
- Guarantee that logical decision on their true and false sides.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structure to assure their validity.

The test case specification for system testing has to be submitted for review before system testing commences.

Skipping any details on customer window Test Case

input	Expected result	Actual output	Result
All details filled	Customer data added	Customer data added	Pass
Details missing	Please fill all the details.	Please fill all the details.	pass

All details filled (Validated)



Hotel Management System

Customer Details

Customer Ref: 1584

Customer Name: Arjun Sharma

Mother's Name: Neetu Sharma

Gender: Select

Pincode: 110078

Mobile: 9050135344

Email: arjun2342@gmail.com

Nationality: Indian

Id Proof Type: Adhaar Card

ID Number: 123456789121

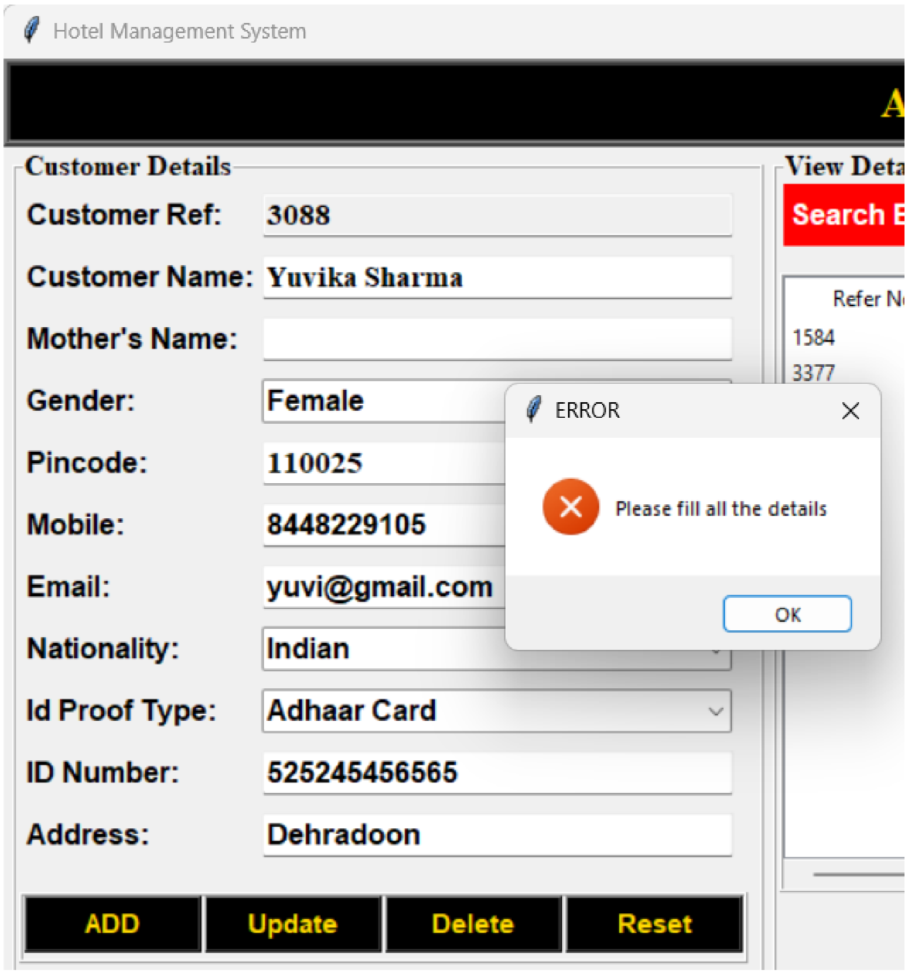
Address: Dehradun

Success

Customer data added

Fig 7.1

Missing any details (Validated)


Hotel Management System

Customer Details

Customer Ref: 3088

Customer Name: Yuvika Sharma

Mother's Name:

Gender: Female

Pincode: 110025

Mobile: 8448229105

Email: yuvi@gmail.com

Nationality: Indian

Id Proof Type: Adhaar Card

ID Number: 525245456565

Address: Dehradoon

ERROR

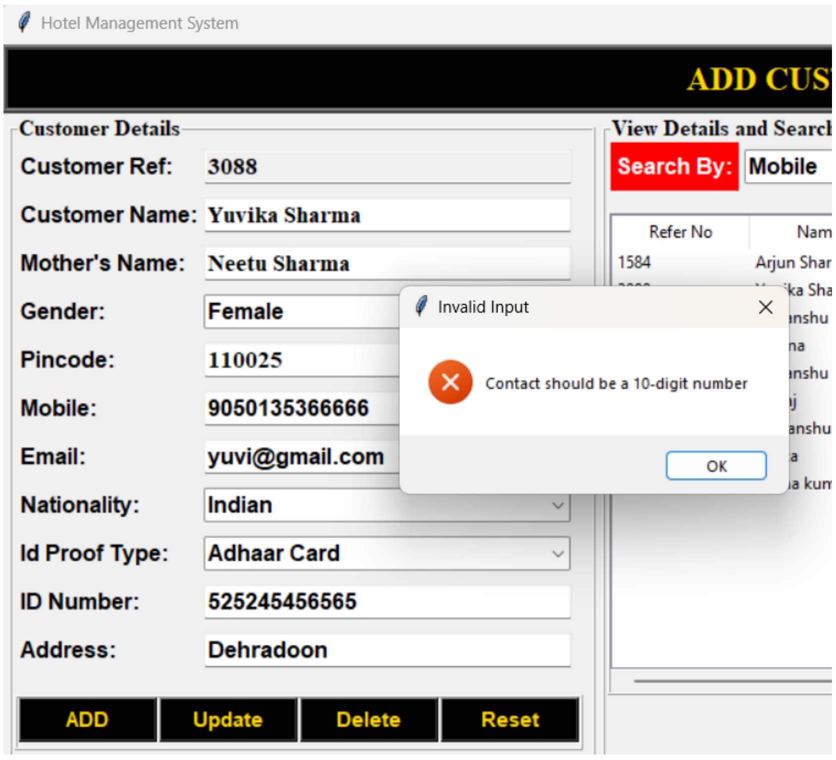
Please fill all the details

Fig 7.2

Invalid contact number

input	Expected output	Actual output	result
10 digit no	Customer data added	Customer data added	Pass
>10 & <10 digit number	Please enter a valid number	Please enter a valid number :	pass

Number entered greater than 10 digits (Validated)



Customer Ref: 3088
Customer Name: Yuvika Sharma
Mother's Name: Neetu Sharma
Gender: Female
Pincode: 110025
Mobile: 9050135366666
Email: yuvi@gmail.com
Nationality: Indian
Id Proof Type: Adhaar Card
ID Number: 525245456565
Address: Dehradoon

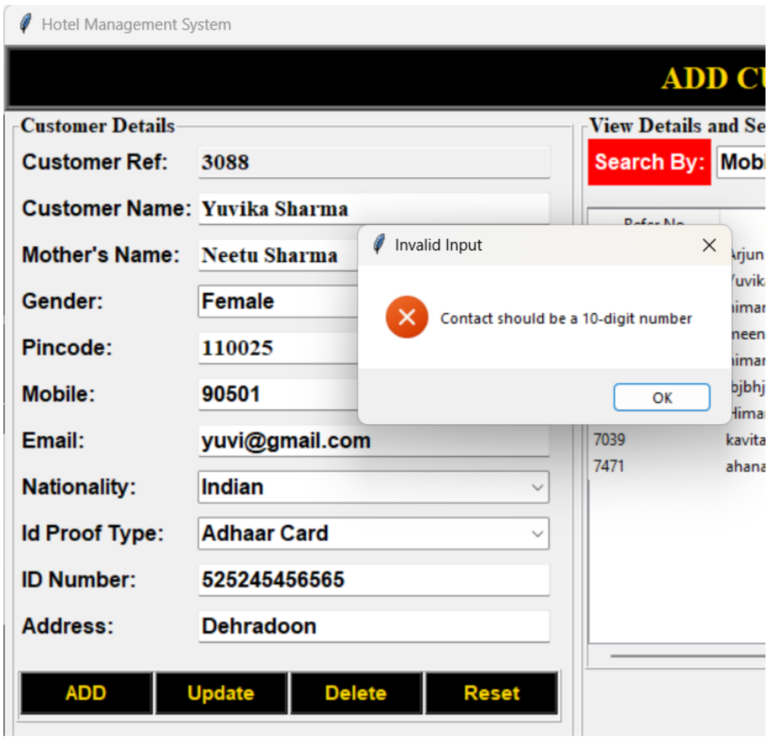
Invalid Input

Contact should be a 10-digit number

OK

Fig 7.3

Number entered lesser than 10 digits (Validated)



Customer Ref: 3088
Customer Name: Yuvika Sharma
Mother's Name: Neetu Sharma
Gender: Female
Pincode: 110025
Mobile: 90501
Email: yuvi@gmail.com
Nationality: Indian
Id Proof Type: Adhaar Card
ID Number: 525245456565
Address: Dehradoon

Invalid Input

Contact should be a 10-digit number

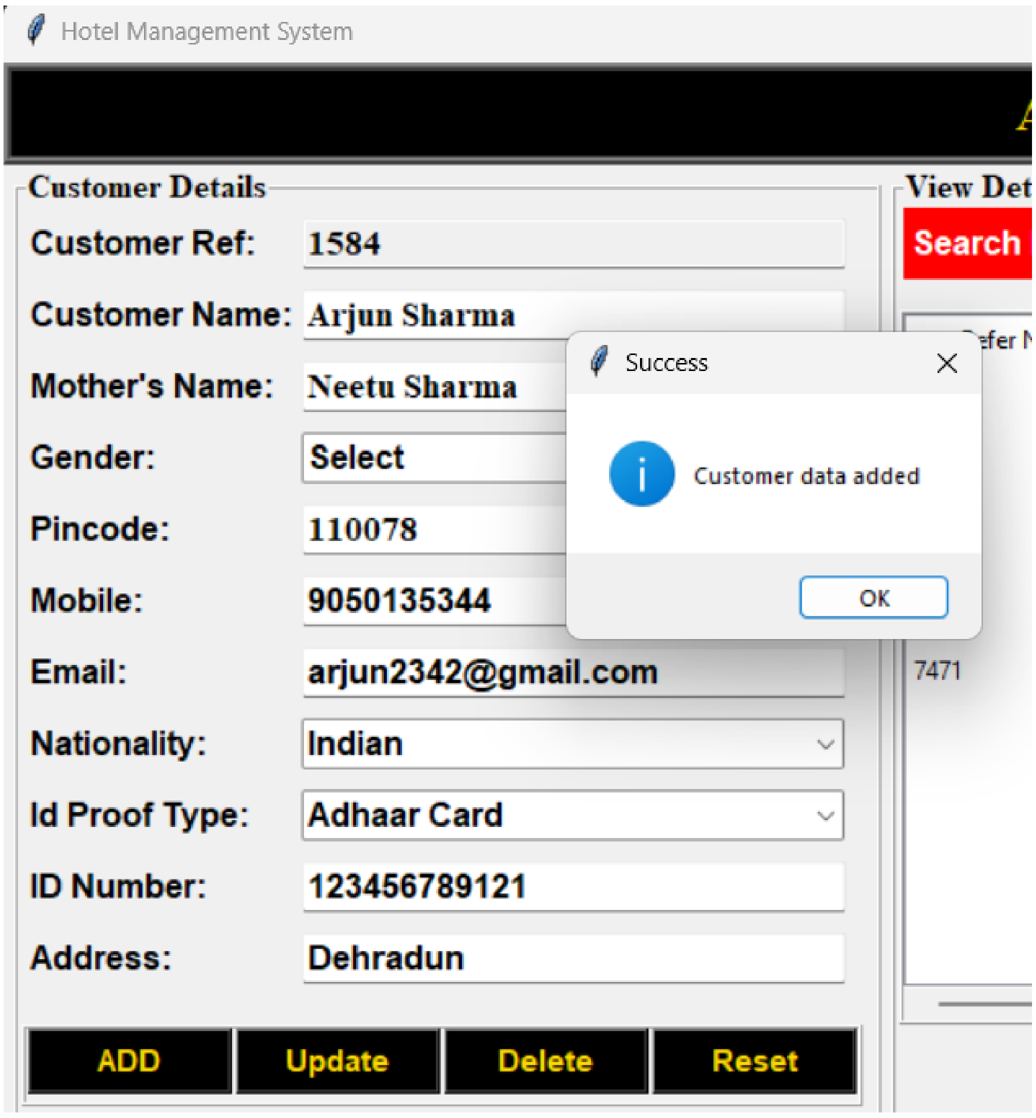
OK

Fig 7.4

Invalid pin code number

input	Expected output	Actual output	result
6 digit no	Customer data added	Customer data added	Pass
>6 & <6 digit number	Please enter a valid number	Please enter a valid number :	pass

Pin code when entered correct (Validated)



Customer Details

Customer Ref:	1584
Customer Name:	Arjun Sharma
Mother's Name:	Neetu Sharma
Gender:	Select
Pincode:	110078
Mobile:	9050135344
Email:	arjun2342@gmail.com
Nationality:	Indian
Id Proof Type:	Adhaar Card
ID Number:	123456789121
Address:	Dehradun

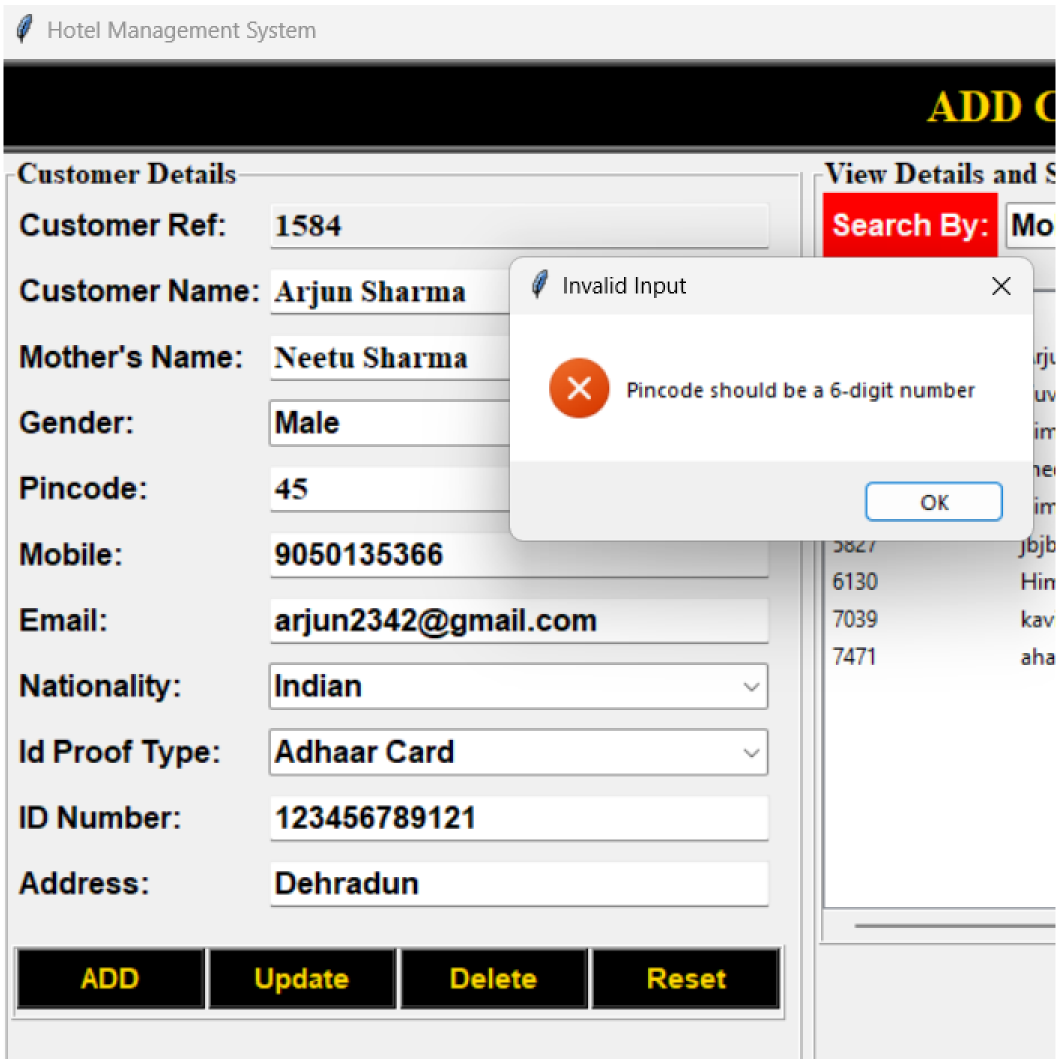
Success

Customer data added

OK

Fig 7.5

Invalid pin code entered (Validated)



Customer Details

Customer Ref: 1584

Customer Name: Arjun Sharma

Mother's Name: Neetu Sharma

Gender: Male

Pincode: 45

Mobile: 9050135366

Email: arjun2342@gmail.com

Nationality: Indian

Id Proof Type: Adhaar Card

ID Number: 123456789121

Address: Dehradun

Invalid Input

Pincode should be a 6-digit number

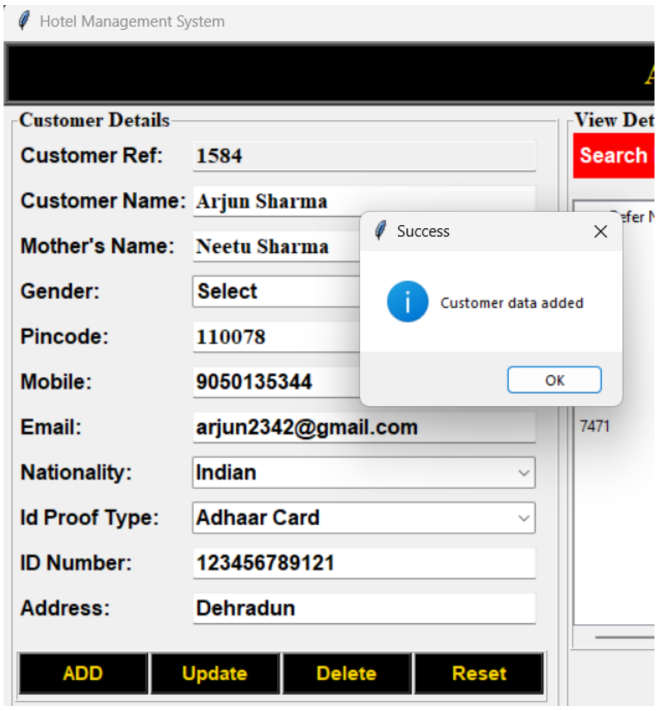
	rju
	uv
	im
	he
	im
5827	jbjb
6130	Hin
7039	kav
7471	aha

Fig 7.6

Invalid email address

input	Expected output	Actual output	result
Valid mail address	Customer data added	Customer data added	Pass
Invalid mail address	Please enter a valid address	Please enter a valid address :	pass

Email entered correctly (Validated)



Hotel Management System

Customer Details

Customer Ref: 1584

Customer Name: Arjun Sharma

Mother's Name: Neetu Sharma

Gender: Select

Pincode: 110078

Mobile: 9050135344

Email: arjun2342@gmail.com

Nationality: Indian

Id Proof Type: Adhaar Card

ID Number: 123456789121

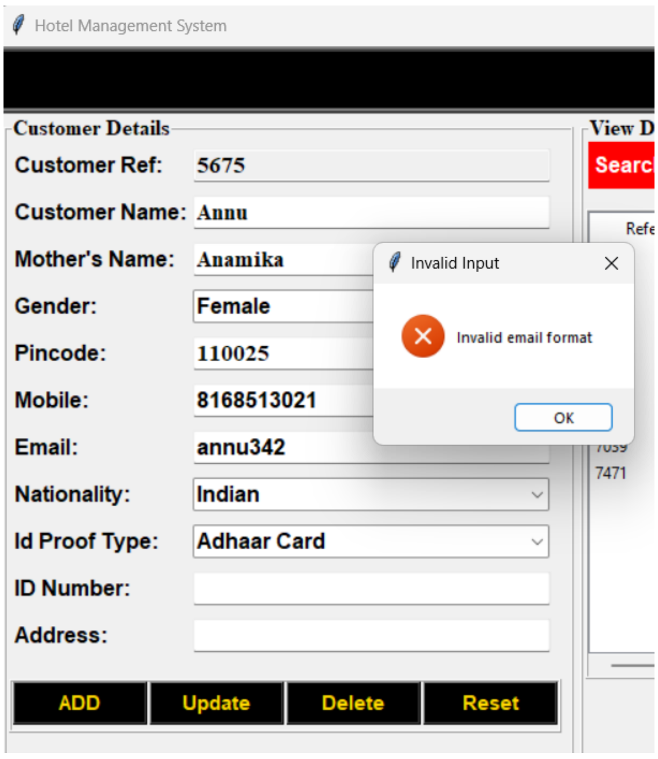
Address: Dehradun

Success

Customer data added

Fig 7.7

Email entered incorrectly (Validated)



Hotel Management System

Customer Details

Customer Ref: 5675

Customer Name: Annu

Mother's Name: Anamika

Gender: Female

Pincode: 110025

Mobile: 8168513021

Email: annu342

Nationality: Indian

Id Proof Type: Adhaar Card

ID Number:

Address:

Invalid Input

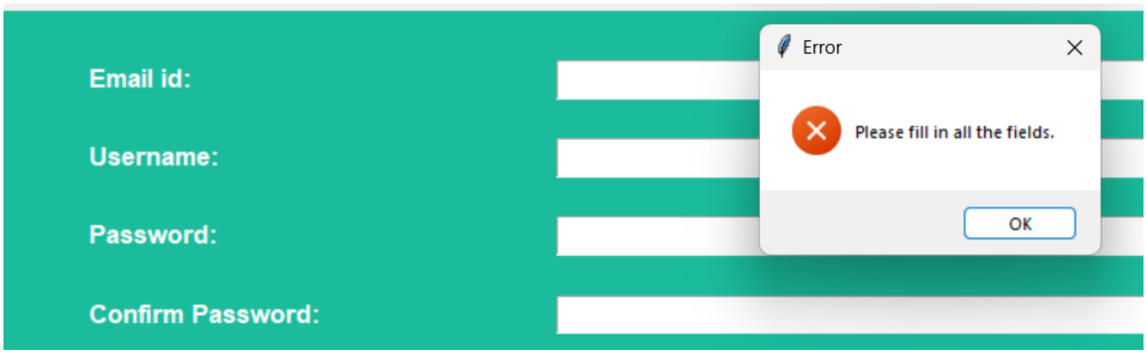
Invalid email format

Fig 7.8

Sign up validations

input	Expected result	Actual output	Result
All details filled	Sign up successful	Account added successfully.	Pass
Details missing	Please fill all the details.	Please fill all the details.	pass

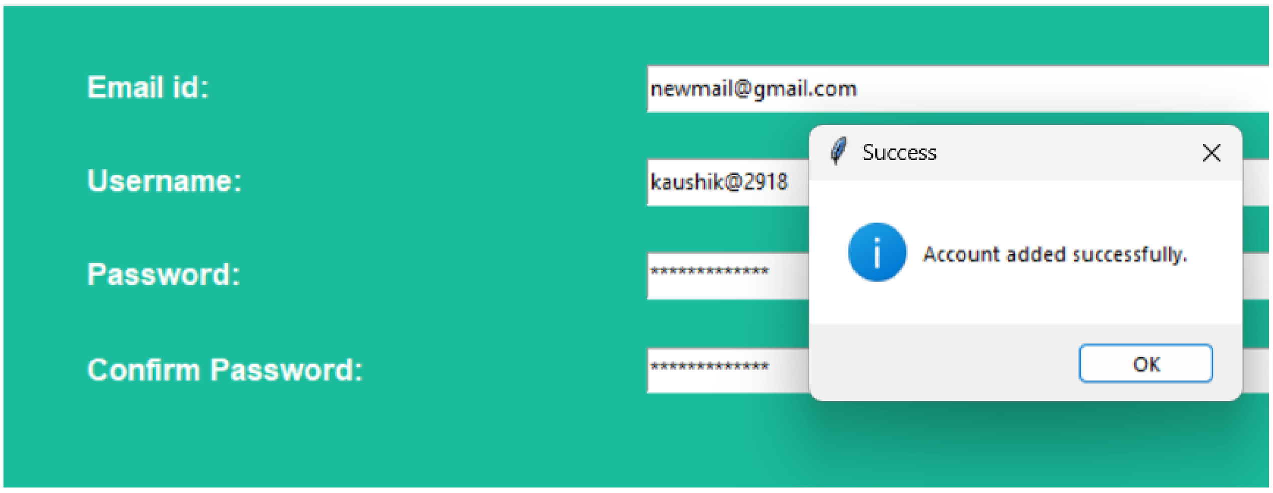
Details missing



Please fill in all the fields.

Fig 7.9

Details filled correctly



newmail@gmail.com
kaushik@2918
****
****
Account added successfully.

Fig 7.10

Incorrect details filled during sign up of a user

input	Expected result	Actual output	Result
All details filled	Sign up successful	Account added successfully	Pass
Email incorrect	Invalid email format.	Invalid email format.	pass

Email entered incorrectly

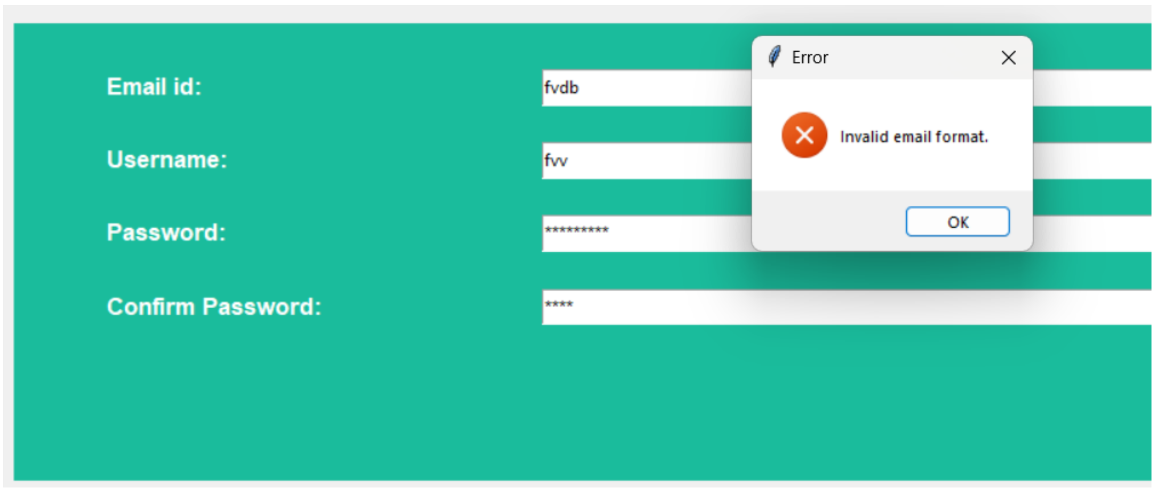

Email id: fvdb
Username: fvv
Password: *********
Confirm Password: ****
Error
Invalid email format.
OK

Fig 7.11

Email entered correctly

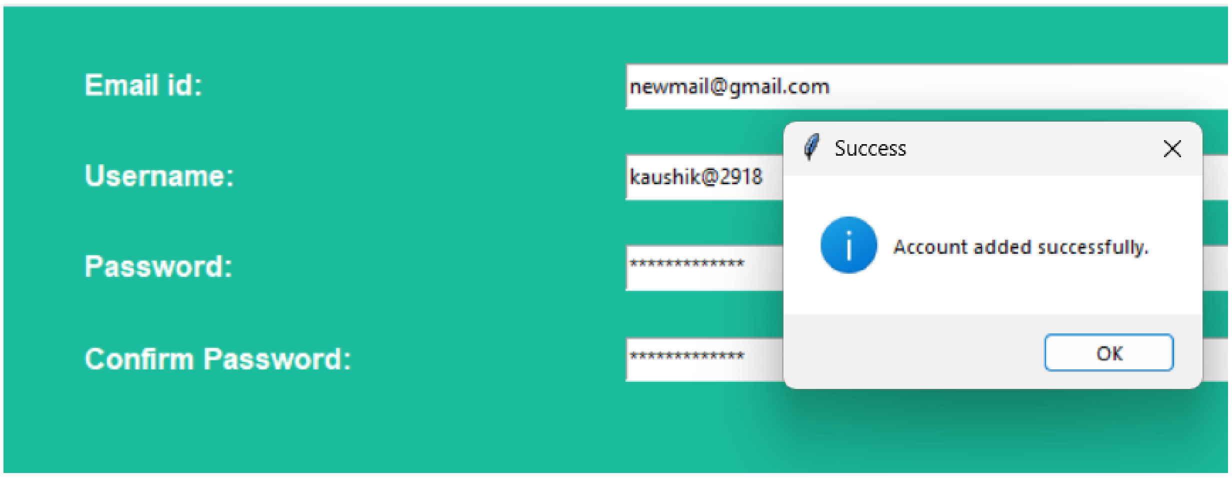
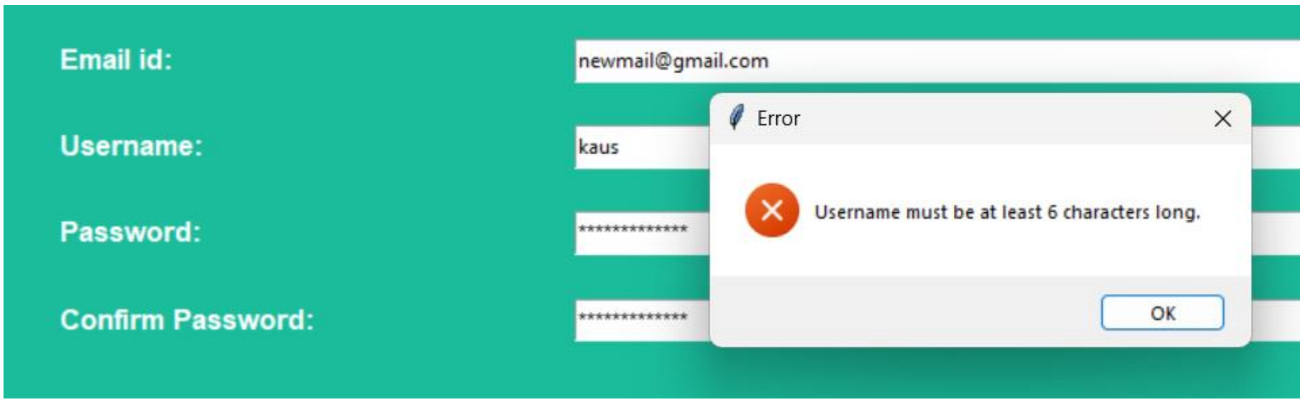

Email id: newmail@gmail.com
Username: kaushik@2918
Password: *************
Confirm Password: *************
Success
Account added successfully.
OK

Fig 7.12

input	Expected result	Actual output	Result
Username is 6 characters	Sign up successful	Account added successfully.	Pass
Username incorrect	Username must be 6 character long	Username must be 6 character long	pass

Username entered incorrectly



Email id: newmail@gmail.com

Username: kaus

Password: *************

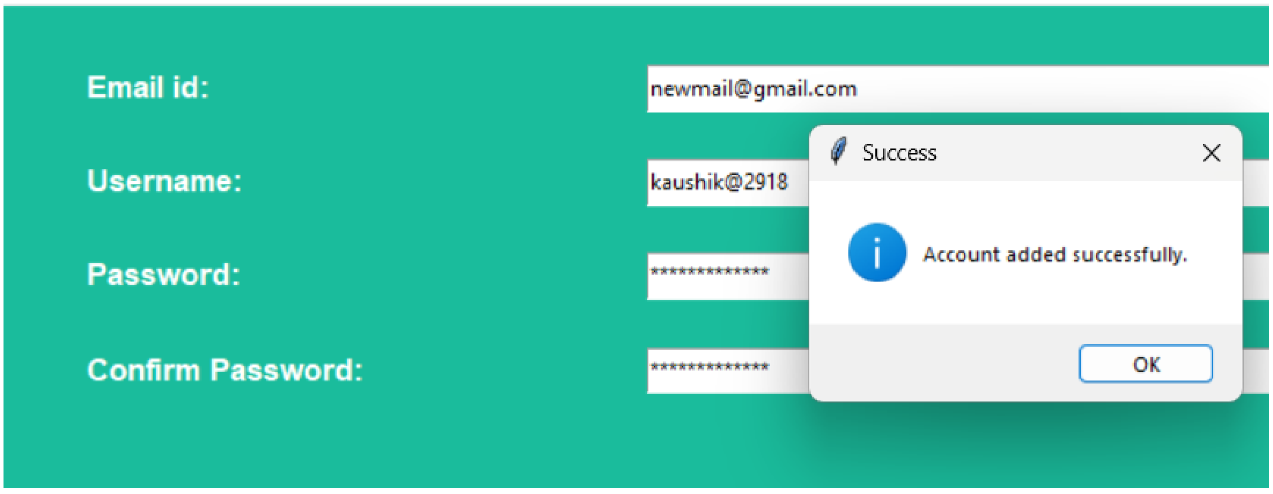
Confirm Password: *************

Error

Username must be at least 6 characters long.

Fig 7.13

Username entered correctly



Email id: newmail@gmail.com

Username: kaushik@2918

Password: *************

Confirm Password: *************

Success

Account added successfully.

Fig 7.14

INPUT & OUTPUT SCREENS

Main menu screen

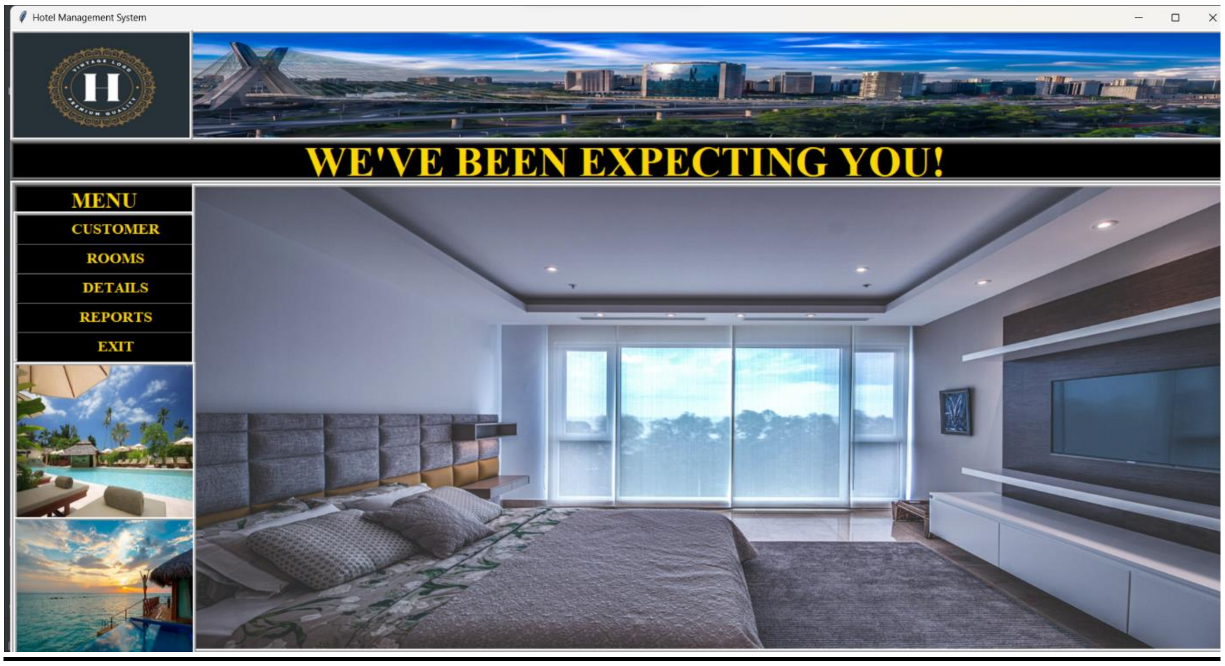
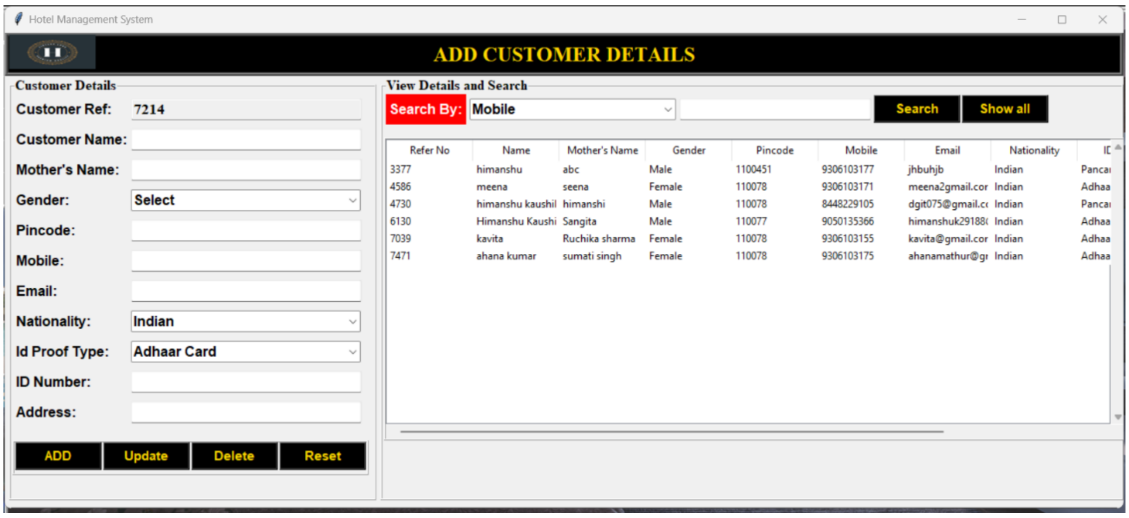


Fig 8.1

Customer menu screen



Customer Details

Customer Ref:	7214
Customer Name:	
Mother's Name:	
Gender:	Select
Pincode:	
Mobile:	
Email:	
Nationality:	Indian
Id Proof Type:	Adhaar Card
ID Number:	
Address:	

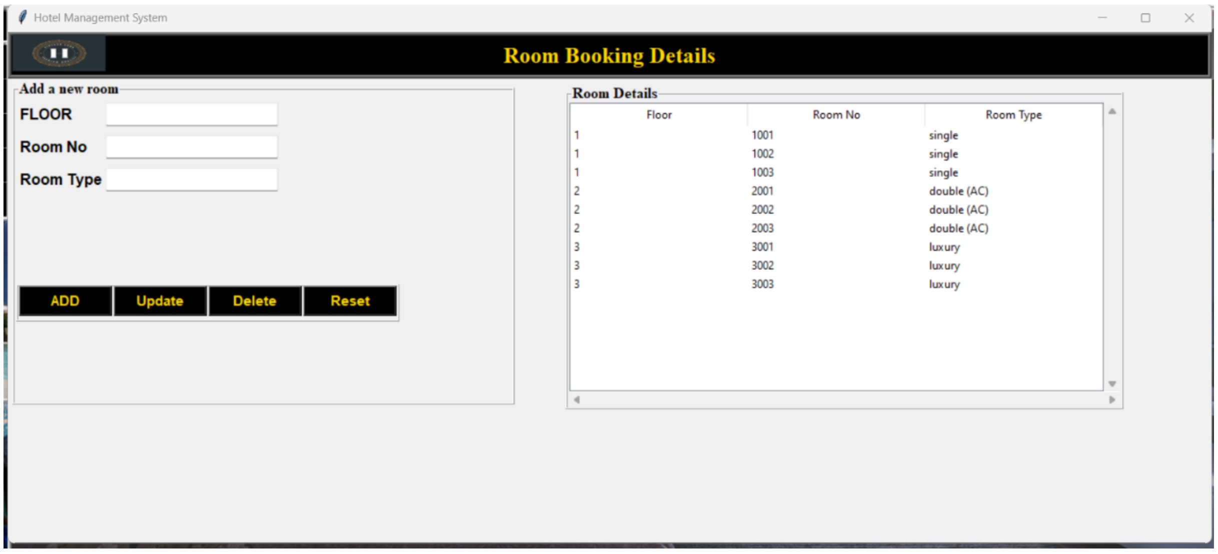
View Details and Search

Search By: Mobile

Refer No	Name	Mother's Name	Gender	Pincode	Mobile	Email	Nationality	IC
3377	himanshu	abc	Male	1100451	9306103177	jhbuhjb	Indian	Pancai
4586	meena	seena	Female	110078	9306103171	meena2gmail.cor	Indian	Adhaa
4730	himanshu kaushil	himanshi	Male	110078	8448229105	dgit075@gmail.cc	Indian	Pancai
6130	Himanshu Kaushi	Sangita	Male	110077	9050135366	himanshuk29188(	Indian	Adhaa
7039	kavita	Ruchika sharma	Female	110078	9306103155	kavita@gmail.cor	Indian	Adhaa
7471	ahana kumar	sumati singh	Female	110078	9306103175	ahanamathur@gı	Indian	Adhaa

Fig 8.2

Room menu screen



Floor	Room No	Room Type
1	1001	single
1	1002	single
1	1003	single
2	2001	double (AC)
2	2002	double (AC)
2	2003	double (AC)
3	3001	luxury
3	3002	luxury
3	3003	luxury

Fig 8.3

Sign up menu

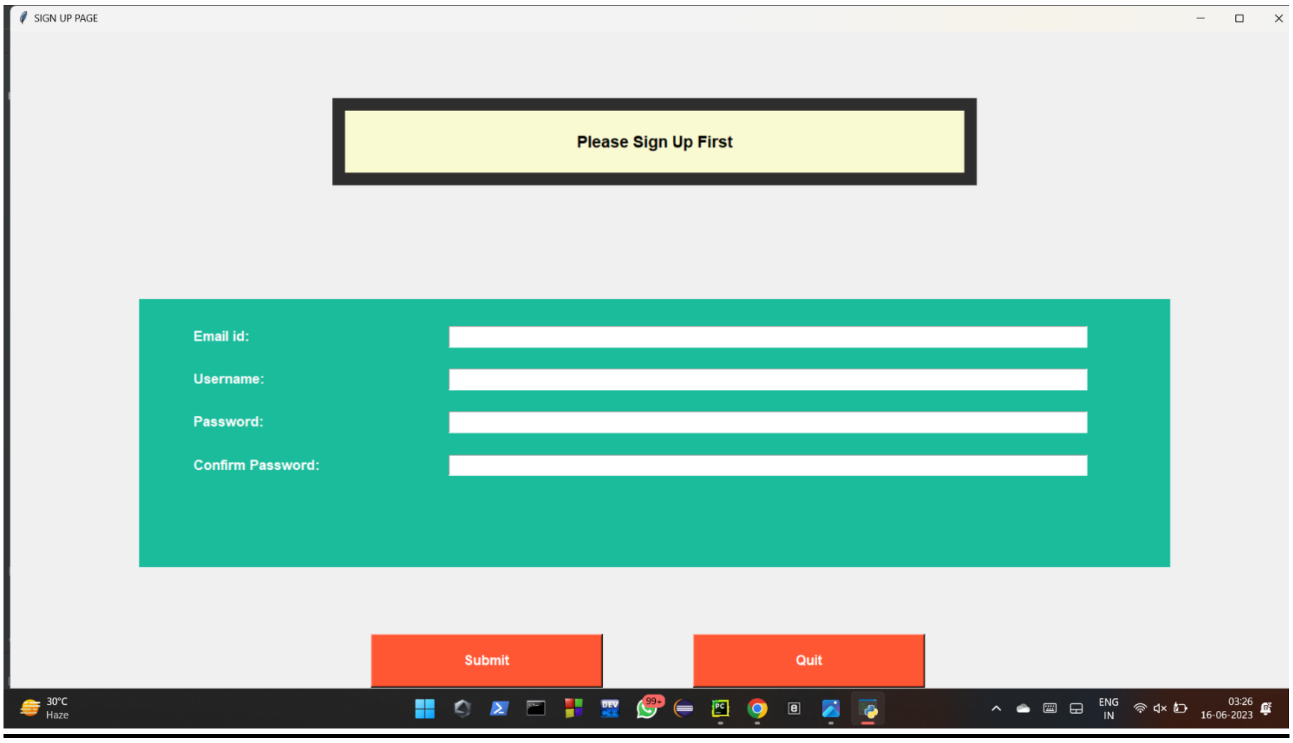


Fig 8.4

Login page view

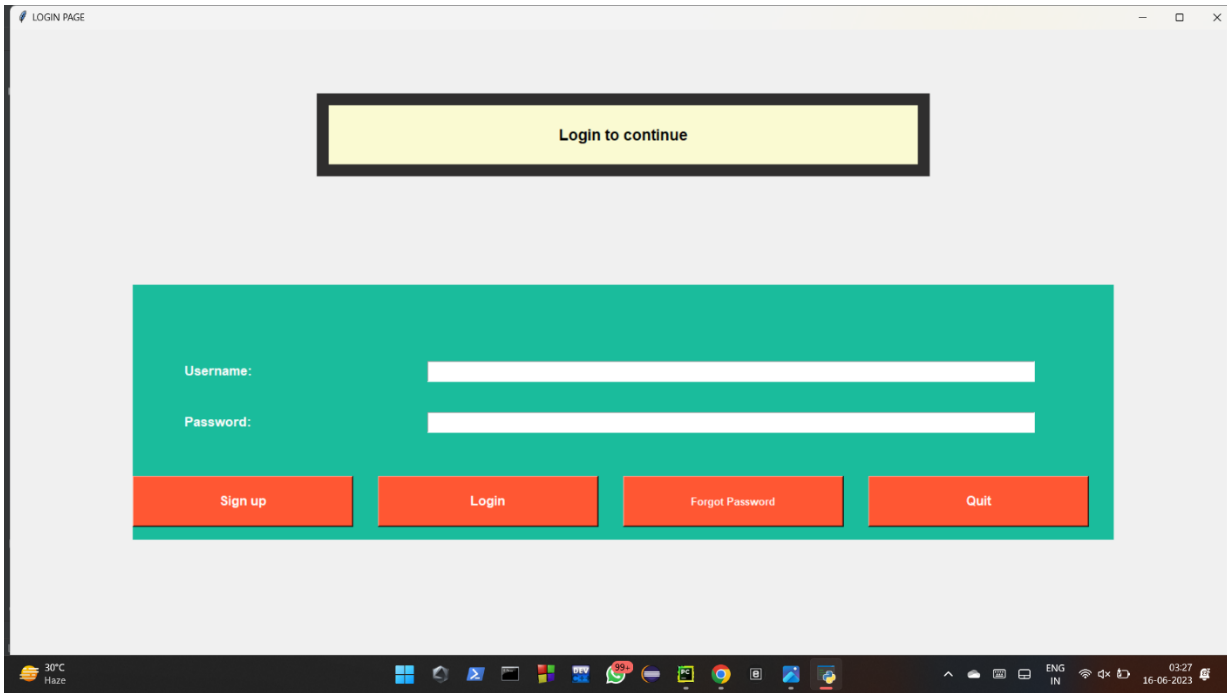


Fig 8.5

Forgot password menu

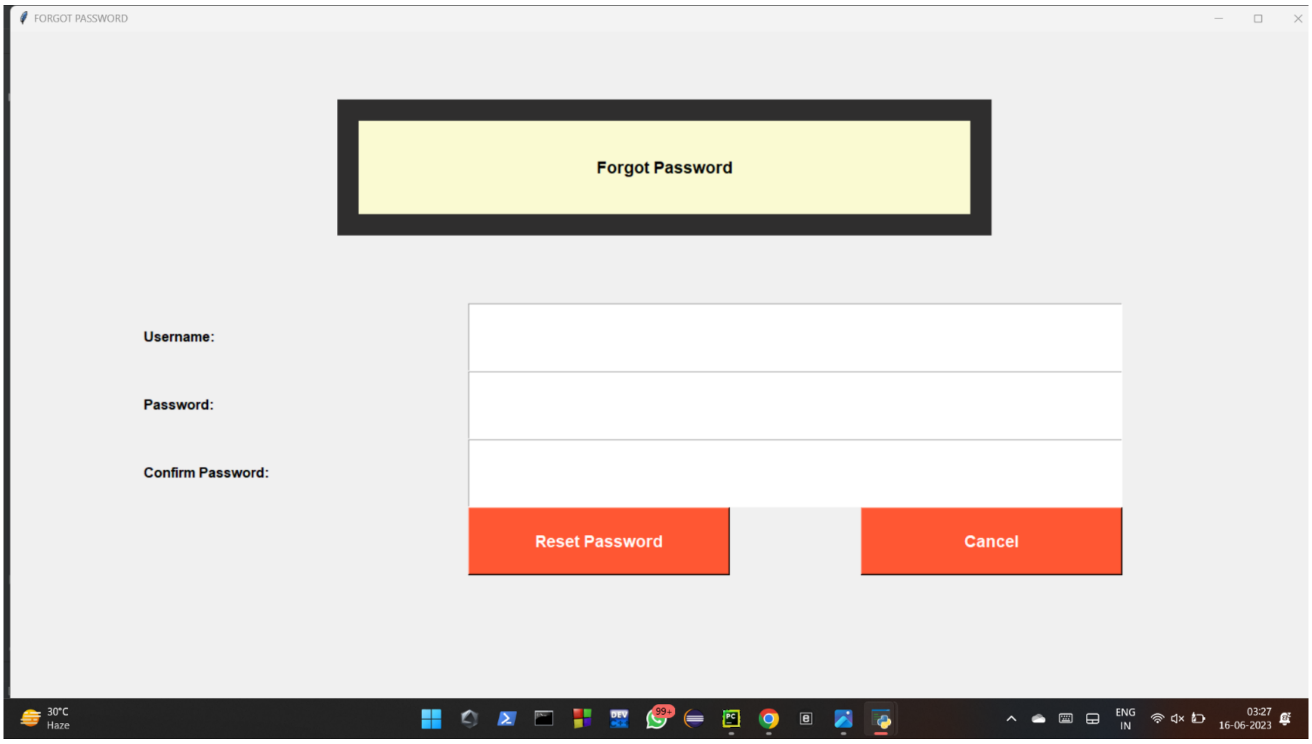


Fig 8

LIMITATIONS OF THE PROJECT

Limitations for Online Hotel Booking System :

Application based hotel booking systems have a number of limitations, including:

- Payment issues: There may be issues with payment processing, such as declined credit cards or problems with the payment gateway.
- Technical problems: The system may experience technical issues, such as outages or slow loading times, which can be frustrating for users.
- Limited customization options: Users may not be able to fully customize their orders, such as requesting special modifications or substitutions in rooms and meals.
- Limited availability: The room booking system may not be available in all areas or at all times.
- Security concerns: Some customers may be concerned about the security of their personal and financial information when using a paying online.
- Admin dependency: The GUI relies heavily on the admin's actions and inputs. This can lead to delays or bottlenecks if the admin is unavailable or unable to perform necessary tasks promptly.

FUTURE APPLICATIONS OF THE PROJECT

Here are some future scopes of a hotel booking system:

1. Limited Customization: Many hotel booking systems have limited options for customization. Administrators may not have the flexibility to tailor the system to their specific needs or integrate it with other internal systems.

2. Laundry addition: A new attached option of laundry can be added just like meals to ease the stay of the customers as well as the hotel working staff.

3. Doctor on call: This special feature will be handled as an emergency toggle button in the GUI application to tackle any medical situation.

4. . Data Security Concerns: Hotel booking systems store sensitive customer data, such as personal information and payment details. Administrators must ensure that the system has robust security measures in place to protect this data from unauthorized access or breaches.

5. Mobile application: Develop a mobile application version of the room booking system GUI, allowing admins to manage bookings and perform administrative tasks on-the-go. This provides flexibility and accessibility, enabling admins to handle tasks from anywhere at any time.

6. Multilingual support: Incorporate multilingual support within the GUI to cater to users from different regions or language preferences. This expands the system's usability and user base.

7. Room customization and amenities management: Provide options to configure room types, amenities, and additional services within the GUI. This allows admins to easily manage and update room details, including descriptions, images, pricing, and availability.

BIBLIOGRAPHY

Books

- Let Us C by Yashwant Kanetkar.
- Let us C++ by Yashwant Kanetkar.
- C in Depth by S.K Srivastava.

Websites

- www.google.com
- www.youtube.com
- www.w3schools.com
- www.lucid.com
- www.geeksforgeeks.com

Thank You