CATERING MANAGEMENT SYSTEM

Submitted by team members:

Nandhana Suffin(U2103148)
Nikhil Stephen (U2103155)
Niveditha B (U2103162)
Rachel Jacob (U2103168)

ACKNOWLEDGEMENT

We extend our sincere gratitude to Ms. Jomina John,Dr.Renu Mary Daniel,Dr.Uma Narayanan our esteemed subject teachers and Mr Steve our lab in charge , whose invaluable guidance and unwavering support were instrumental in the successful completion of this project. Their expertise and encouragement greatly enriched my learning experience.

I would like to express my heartfelt thanks to Mr. Sreejith, our respected principal, for providing me with the golden opportunity to undertake this project on "Catering Management System." His belief in my capabilities served as a motivating force, and I am truly grateful for the chance to delve into this fascinating topic in-depth.

We also wish to acknowledge and thank our fellow schoolmates for their collaboration and assistance throughout the project. Their collective efforts enhanced the overall quality of our work, and we appreciate the spirit of teamwork that prevailed.

In addition, I want to acknowledge the support and encouragement received from my family and friends. Their belief in my abilities and words of encouragement provided the motivation needed to overcome challenges and strive for excellence.

Lastly, I extend my gratitude to all those who, directly or indirectly, contributed to the realization of this project. Your collective efforts have been instrumental in its success.

ABSTRACT

The Catering Management System is an innovative software solution designed to streamline and enhance the efficiency of catering businesses by automating various aspects of their operations. This comprehensive system integrates advanced technologies to manage the entire catering process, from order creation to delivery, ensuring seamless coordination and optimal resource utilization.

The system features a user-friendly interface that allows clients to place catering orders online, customize menus, and specify event details. Caterers can efficiently manage these orders, track inventory levels, and optimize resource allocation using the centralized platform.

Key functionalities of the Catering Management System include order management, menu customization, inventory tracking and financial reporting. The system employs data analytics to provide valuable insights into customer preferences, helping caterers tailor their services to meet evolving demands.

Security measures, such as authentication protocols and data encryption, safeguard sensitive information, instilling trust among clients and ensuring compliance with data protection regulations. The system is scalable, accommodating the varying needs of catering businesses, whether small-scale operations or large-scale enterprises.

By leveraging the Catering Management System, businesses in the catering industry can optimize their processes, improve customer satisfaction, and achieve sustainable growth. The integration of technology not only enhances operational efficiency but also positions catering businesses to adapt to the dynamic landscape of the modern culinary industry.

TABLE OF CONTENTS

- ACKNOWLEDGEMENT……1
- ABSTRACT……2
- PROJECT OVERVIEW……..4
- DATABASE DESIGN……7
- E-R DIAGRAM……..8
- ARCHITECTURE DIAGRAM…….……9
- FUNCTIONS CREATED…….10
- SOURCE CODE……11
- OUTPUT……....27
- FRONT END/BACK END ……..…..31
- BIBLIOGRAPHY……..33
- CONCLUSION……..……34

PROJECT OVERVIEW

Introduction:

The Catering Management System is a comprehensive software solution designed to revolutionize and streamline the operations of catering businesses. In response to the evolving demands of the catering industry, this project aims to leverage advanced technologies to enhance efficiency, improve customer satisfaction, and provide valuable insights for business growth.

Objectives:

Efficient Order Management: Implement a user-friendly platform for clients to place catering orders, customize menus, and specify event details.

Resource Optimization: Develop features for caterers to manage orders, track inventory, and optimize resource allocation, ensuring timely and accurate delivery of catering services.

Security Measures: Ensure the security of sensitive information through robust authentication protocols and data encryption, maintaining client trust and compliance with data protection regulations.

Scalability: Design the system to be scalable, catering to the needs of both small-scale catering businesses and large-scale enterprises.

Key Features:

User Authentication: The system requires user authentication with a username and password. If the user is not registered, there is an option to sign up.

Main Menu: After successful login or signup, the user is presented with a main menu for various actions.

Order Placement: Users can place catering orders through the "PLACE ORDER" option. The order details include personal information (name, email, phone number) and event details (type of dining, package, number of people, time of event, date of event, and venue).

Order Editing: Users can edit their existing orders using the "EDIT REQUEST" option. Editing involves modifying details such as the type of dining, package, number of people, time of event, date of event, and venue.

Order Cancellation: Users can cancel their placed orders through the "CANCEL ORDER" option. Cancellation involves a confirmation step, and upon confirmation, the order is deleted from the database.

Payment Processing: Payments are handled through the "MAKE PAYMENT" option. The payment form includes fields for selecting the payment method (credit card or debit card) and entering card details (number, name, expiry date, CVV). A unique transaction ID is generated for each payment, and payment details are stored in the database.

Invoice Generation: Users can generate invoices for their orders using the "PRINT INVOICE" option. The invoice includes order details and payment information. Order Details and Printing: Users can view and print details of their placed orders using the "PRINT ORDER DETAILS" option. Users need to provide the order ID to retrieve and display the order details.

User Interface: The user interface is built using the Tkinter library for GUI development in Python. The interface includes buttons, entry fields, labels, and dropdown menus for user interaction.

Database Integration: The system is connected to a MySQL database for storing and retrieving user information, order details, and payment information. SQL queries are used to interact with the database and perform operations like insertion, updating, and deletion.

Error Handling: The system includes error handling mechanisms using message boxes to notify users of invalid actions or errors.

Order ID Generation: unique order IDs are generated for each placed order.

Scanned Attachments: The email seems to contain a scanned attachment, but its content is not displayed in the provided code snippet.

Benefits:

Operational Efficiency: Streamline catering processes, reducing manual effort and improving overall efficiency.

Customer Satisfaction: Enhance the customer experience with online order management and customizable menus.

Business Growth: Gain insights into customer preferences and market trends, informing strategic decisions for business expansion.

Adaptability: Accommodate the varying needs of catering businesses, supporting growth and scalability.

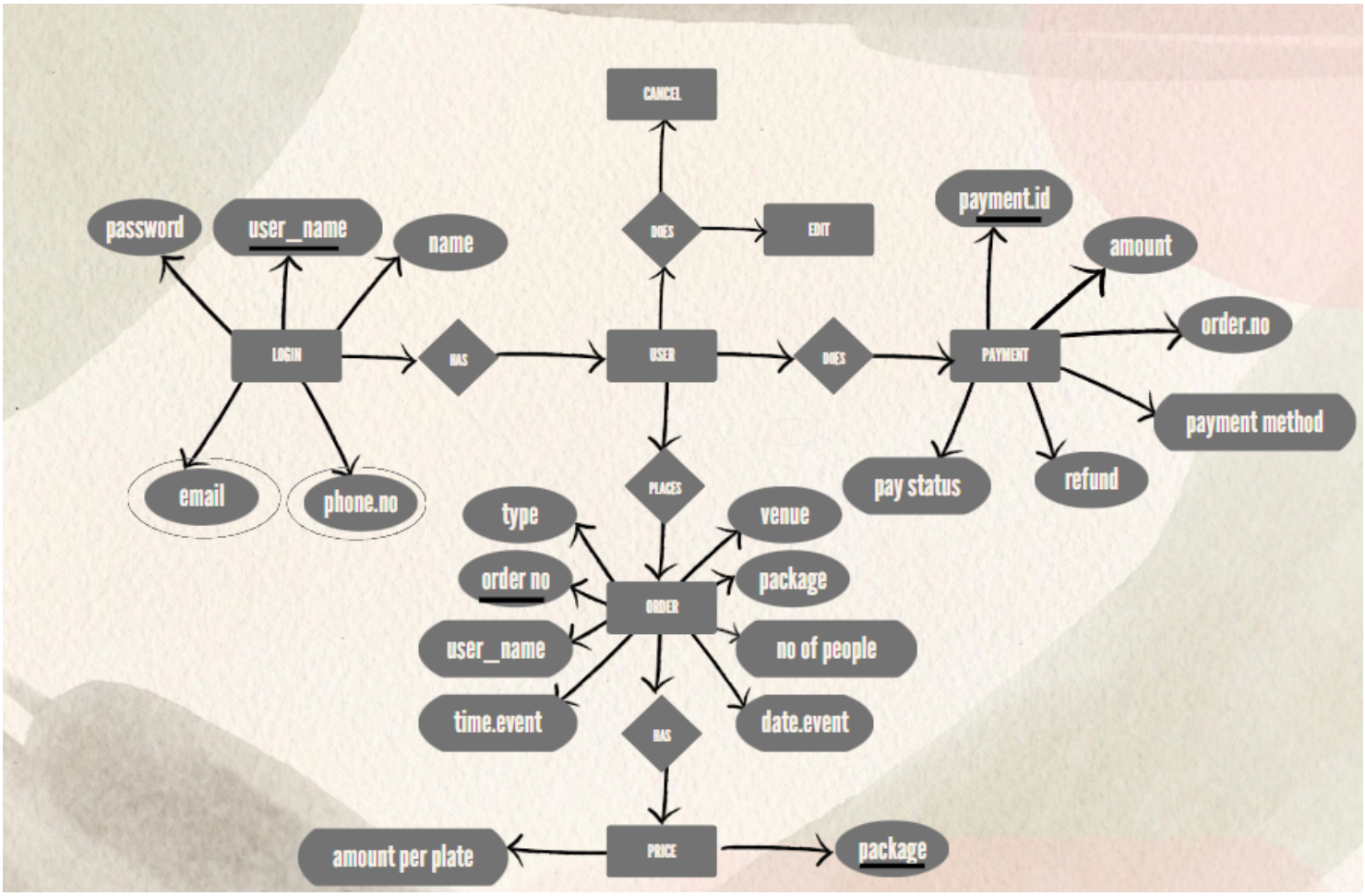
DATABASE DESIGN

TABLES

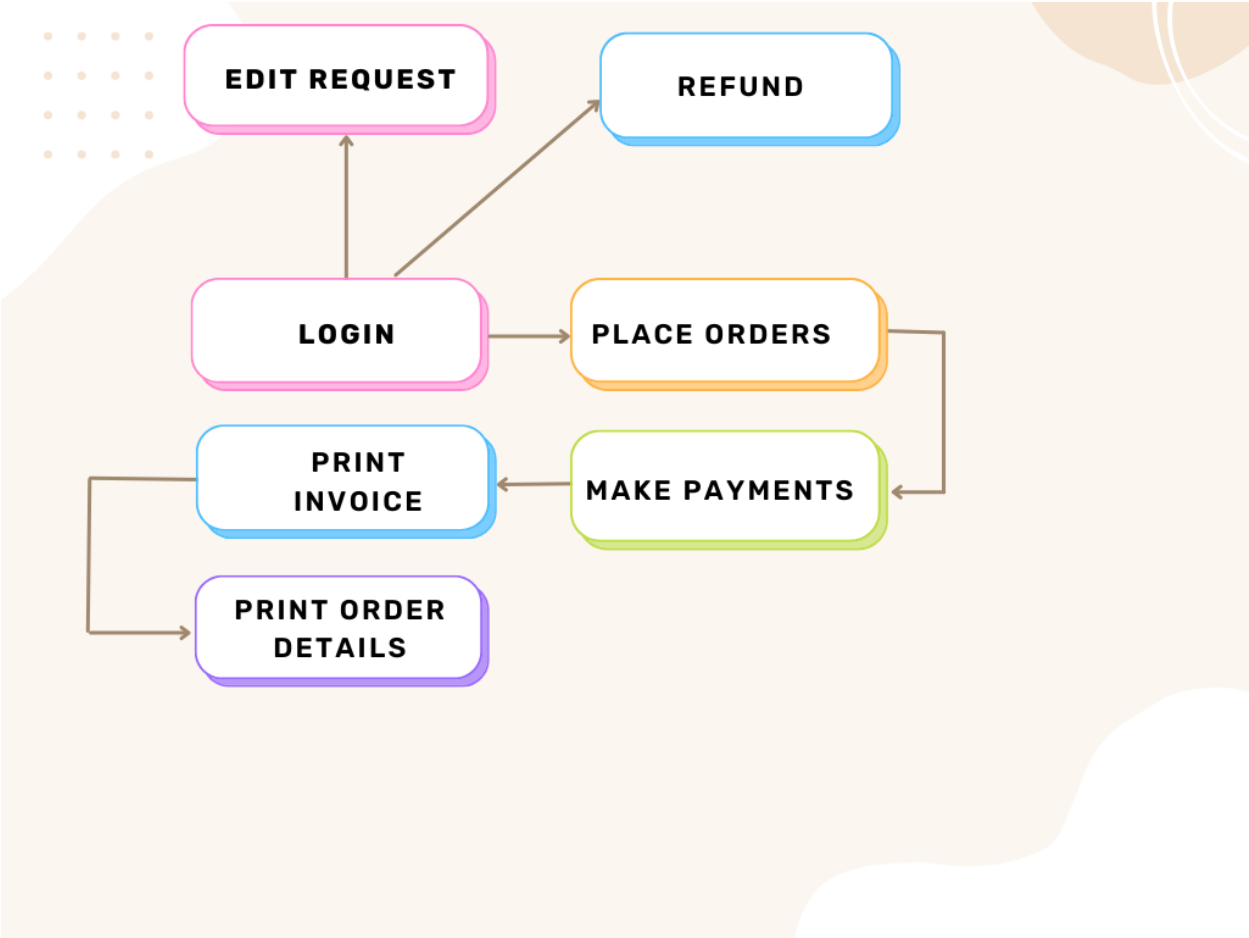
1.login table 2.price table 3.orders table 4.payment table

```
mysql> desc price;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| package          | varchar(50)  | NO   | PRI |         |       |
| amount_per_plate | decimal(10,2)| YES  |     | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
2 rows in set (0.02 sec)

mysql> desc login;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| User_name | varchar(20) | NO   | PRI | NULL    |       |
| password  | varchar(10) | YES  |     | NULL    |       |
| name      | varchar(20) | YES  |     | NULL    |       |
| email     | varchar(20) | YES  |     | NULL    |       |
| phone_no  | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)

mysql> desc orders;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| user_name  | varchar(20)  | YES  |     | NULL    |       |
| order_no   | int(10)      | NO   | PRI | NULL    |       |
| type       | varchar(20)  | YES  |     | NULL    |       |
| package    | varchar(20)  | YES  |     | NULL    |       |
| no_of_ppl  | int(10)      | YES  |     | NULL    |       |
| date_event | varchar(20)  | YES  |     | NULL    |       |
| time_event | varchar(20)  | YES  |     | NULL    |       |
| venue      | varchar(50)  | YES  |     | NULL    |       |
| amount     | decimal(10,2)| YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
9 rows in set (0.01 sec)

mysql> desc payment;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| payment_id     | varchar(20)  | NO   | PRI | NULL    |       |
| order_no       | int(20)      | YES  |     | NULL    |       |
| payment_method | varchar(20)  | YES  |     | NULL    |       |
| amount         | decimal(10,2)| YES  |     | NULL    |       |
| refund         | varchar(20)  | YES  |     | NULL    |       |
| payment_status | varchar(20)  | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

E-R DIAGRAM


CANCEL
password
user_name
name
DOES
EDIT
payment.id
amount
order.no
LOGIN
HAS
USER
DOES
PAYMENT
payment method
email
phone.no
PLACES
pay status
refund
type
venue
order no
package
ORDER
user_name
no of people
time.event
HAS
date.event
amount per plate
PRICE
package

ARCHITECTURE DIAGRAM



FUNCTIONS CREATED

- Cancel_order
- Make_payment
- Place_order
- Edit_order
- Order_details
- Invoice
- Page2
- Login_form
- Create_login

PROGRAM SOURCE CODE

```
import random
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
from tkinter import scrolledtext
from PIL import Image, ImageTk
import mysql.connector
mydb=mysql.connector.connect(host='localhost',database='catering',user='root',password='1234')
cursor=mydb.cursor()


fpage = Tk()
fpage.title("Order details")
fpage.geometry("1500x1500")
fpage.configure(bg="#FFB3BA")



def cancel_order():
    cancel = Tk()
    cancel.title("Cancel order")
    cancel.geometry("2000x1900")
    cancel.configure(bg="#3EB489")  # Choose your desired color
    def cancelo():
        order_no = e1.get()
        sql='select * from orders where order_no=%s and user_name=%s'
        cursor.execute(sql, (order_no,u,))
        r = cursor.fetchone()
        if r is  None:
            messagebox.showerror("Error","Invalid Order Id")
        else:

            delete_sql = 'DELETE FROM orders WHERE order_no = %s and user_name=%s'
            delete_values = (order_no,u)
            r="Refunded"
            try:
                cursor.execute(delete_sql, delete_values)
                mydb.commit()
```

```
            messagebox.showinfo("Confirmation", "Order Cancelled for Order No:
{}".format(order_no))
            cancel.destroy()
        except mysql.connector.Error as err:
            messagebox.showerror("Error", f"Error deleting order: {err}")
        update_sql = 'UPDATE payment SET refund=%s WHERE order_no=%s '
        update_values = (r,order_no,u,)
        cursor.execute(update_sql, update_values)
        mydb.commit()

    lab1 = Label(cancel, text="CANCEL YOUR ORDER", width=20, font=("bold",
30),bg="#3EB489")
    lab1.grid(row=0, column=0, sticky=W, pady=20,padx=400)

    lab2 = Label(cancel, text="Enter the order number:", width=18, font=("bold", 25),
anchor=W,bg="#3EB489")
    lab2.grid(row=7, column=0, sticky=W, pady=150,padx=270)

    e1 = Entry(cancel, width=25, font=('Arial 15'))
    e1.grid(row=7, column=0, sticky=W, padx=650)  # Adjust the padx value as needed

    button = Button(cancel, text="CONFIRM CANCEL", command=cancelo, font=("bold", 20),
bg="green", fg="white")
    button.grid(row=8, column=0,sticky=W, pady=20, padx=500)

    cancel.mainloop()


def make_payment(oid,amt):
    def cl():
        payment.destroy()
    def pay(oid,amt):
        pid=random.randint(10000, 9999999)
        meth=e1.get()
        refund="NO"
        status="TXN_SUC"
        insert_sql = 'INSERT INTO payment (payment_id, order_no, payment_method, amount,
refund, payment_status) VALUES (%s, %s, %s, %s, %s, %s)'
        insert_values = (pid, oid, meth, amt, refund, status)
        cursor.execute(insert_sql, insert_values)
```

```
        mydb.commit()
        T1 = Text(payment, height=2, width=25)
        T1.grid()
        T1.insert(END,"Order Id    :"+str(oid)+"\nTransaction Id:"+str(pid))
        T1.configure(state=DISABLED,font=18,fg='black')
        button_close=Button(payment,text="CLOSE",font=('arial
bold',20),fg='black',command=cl,bg='orange',height=1,width=10).grid(row=29,column=0,pady=
10,sticky=W,padx=200)

    payment = Tk()
    payment.title(" MAKE PAYMENT")
    payment.geometry("1300x1200")
    lab1 = Label(payment, text="MAKE PAYMENT", width=60, font=("bold", 30)).grid(row = 0,
column = 0, sticky = W, pady = 2)

    lab2 = Label(payment, text="Payment Method : ", width=30, font=("bold", 20)).grid(row =
10, column = 0, sticky = W, pady = 10)
    payment_options = ["CREDIT CARD","DEBIT CARD"]
    e1 = ttk.Combobox(payment,width=23, values=payment_options, font=('Arial
15'),state="readonly")
    e1.grid(row=10,sticky=W,padx=400)

    lab3 = Label(payment, text="Card Number     : ", width=30, font=("bold", 20)).grid(row = 13,
column = 0, sticky = W, pady = 10)
    e2 = Entry(payment, width=23, font=('Arial 15'))
    e2.grid(row=13,sticky=W,padx=400)

    lab4 = Label(payment, text="Card Name       : ", width=30, font=("bold", 20)).grid(row = 16,
column = 0, sticky = W, pady = 10)
    e3 = Entry(payment, width=23, font=('Arial 15'))
    e3.grid(row=16,sticky=W,padx=400)

    lab5 = Label(payment, text="Expiry Date      : ", width=30, font=("bold", 20)).grid(row = 19,
column = 0, sticky = W, pady = 10)
    e4 = Entry(payment, width=23, font=('Arial 15'))
    e4.grid(row=19,sticky=W,padx=400)

    lab6 = Label(payment, text="CVV              : ", width=30, font=("bold", 20)).grid(row = 22,
column = 0, sticky = W, pady = 10)
    e5 = Entry(payment, width=23, font=('Arial 15'))
```

```
    e5.grid(row=22,sticky=W,padx=400)
    labels= Label(payment, text="Total Amount:"+str(amt), width=60, font=("bold",
30)).grid(row = 24, column = 0, sticky = W, pady = 2)
    button_pay=Button(payment,text="PAY",font=('arial
bold',20),fg='white',command=lambda:pay(oid,amt),bg='orange',height=1,width=10).grid(row=2
6,column=0,pady=10,sticky=W,padx=200)


def place_order():
    form = Tk()
    form.title("Place Order")
    form.geometry("1500x800")

    form.configure(bg="#3EB489")  # Choose your desired color


    def dostuff():
        # Personal details
        n = e1.get()
        em = e2.get()
        pno = e3.get()

        # Order details
        dining_type = e4.get()
        package = e5.get()
        num_people = e6.get()
        time_of_event = e7.get()
        date_of_event = e8.get()
        v=e9.get()
        def generate_random_two_digit():
            return random.randint(10000, 9999999)
        o = generate_random_two_digit()
        # Update login table
        update_sql = 'UPDATE login SET name=%s, email=%s, phone_no=%s WHERE
user_name=%s'
        update_values = (n, em, pno, u)
        cursor.execute(update_sql, update_values)
        mydb.commit()

        # Insert into orders table
```

```
        insert_sql = 'INSERT INTO orders (user_name, type,order_no, package, no_of_ppl,
time_event, date_event,venue) VALUES (%s, %s, %s, %s, %s, %s, %s,%s)'
        insert_values = (u, dining_type, o,package, num_people, time_of_event, date_of_event,v)
        cursor.execute(insert_sql, insert_values)
        mydb.commit()

        stm = 'select amount_per_plate from price where package=%s'
        cursor.execute(stm, (package,))
        r = cursor.fetchone()
        if r is not None:
            amt = r[0]
        mydb.commit()
        amt = float(amt)  # Convert decimal.Decimal to float
        tot = amt * int(num_people)  # Convert num_people to int and perform multiplication

        update_sql1 = 'UPDATE orders SET amount=%s WHERE order_no=%s'
        update_values = (tot, o)
        cursor.execute(update_sql1, update_values)
        mydb.commit()
        form.destroy()
        messagebox.showinfo("Your order has been placed successfully!", "Proceed to pay")
        make_payment(o,tot)


    lab1 = Label(form, text="PLACE YOUR ORDER", width=60, font=("bold",
30),bg="#3EB489").grid(row = 0, column = 0, sticky = W, pady = 2)
    lab2 = Label(form, text="Personal Details:", width=38, font=("bold",
25),bg="#3EB489",anchor=W).grid(row=3,sticky=W,pady=4)

    lab3 = Label(form, text="Name:", width=45, font=("bold",
20),bg="#3EB489",anchor=W).grid(row=4,column=0,sticky=W,columnspan=20)
    e1=Entry(form,width=25,font=('Arial 15'))
    e1.grid(row=4,column=0,sticky=W,padx=200)
    lab4 = Label(form, text="Email Id:", width=45, font=("bold",
20),bg="#3EB489",anchor="w").grid(row=5,column=0,columnspan=20,sticky=W)
    e2=Entry(form,width=25,font=('Arial 15'))
    e2.grid(row=5,sticky=W,padx=200)
    lab6 = Label(form, text="Phone No:", width=45, font=("bold",
20),bg="#3EB489",anchor="w").grid(row=6,column=0,columnspan=20,sticky=W)
    e3=Entry(form,width=25,font=('Arial 15'))
```

```
    e3.grid(row=6,sticky=W,padx=200)
    lab7 = Label(form, text="Order Details:     ", width=38, font=("bold",
25),bg="#3EB489",anchor=W).grid(row=8, sticky=W, pady=4)
    lab8 = Label(form, text="type of dining:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=9, column=0, columnspan=20,sticky=W)



    # Create a list of cuisines for the drop-down
    cuisine_options = ["buffet","table sevice"]

    # Create a Combobox and set the values
    e4 = ttk.Combobox(form,width=23, values=cuisine_options, font=('Arial
15'),state="readonly")
    e4.grid(row=9,sticky=W,padx=200)

    lab9 = Label(form, text="package:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=10, column=0, columnspan=20,sticky=W)
    # Create a list of cuisines for the drop-down
    package_options = ["indian p1", "indian p2", "chinese p1","chinese p2", "indochinese
p1","indochinese p2"]

    e5 = ttk.Combobox(form,width=23, values=package_options, font=('Arial
15'),state="readonly")
    e5.grid(row=10,sticky=W,padx=200)

    lab10 = Label(form, text="No.of people:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=11, column=0, columnspan=20,sticky=W)
    e6 = Entry(form, width=25, font=('Arial 15'))
    e6.grid(row=11,sticky=W,padx=200)

    lab11 = Label(form, text="time of event:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=12, column=0, columnspan=20,sticky=W)
    e7 = Entry(form, width=25, font=('Arial 15'))
    e7.grid(row=12,sticky=W,padx=200)

    lab12 = Label(form, text="date of event:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=13, column=0, columnspan=20,sticky=W)
    e8 = Entry(form, width=25, font=('Arial 15'))
    e8.grid(row=13,sticky=W,padx=200)
```

```
    lab13 = Label(form, text="Venue:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=14, column=0, columnspan=20,sticky=W)
    e9 = Entry(form, width=25, font=('Arial 15'))
    e9.grid(row=14,sticky=W,padx=200)
    button=Button(form,text="Proceed To Pay",font=('arial
bold',20),fg='white',command=dostuff,bg='orange',height=1,width=15).grid(row=16,column=0,p
ady=10,sticky=W,padx=200)
    form.mainloop()


def edit_order():
    form = Tk()
    form.title("Place Order")
    form.geometry("1500x800")

    form.configure(bg="#3EB489")  # Choose your desired color

    def update_order():


        order_id = e3.get()
        dining_type = e4.get()
        package = e5.get()
        num_people = e6.get()
        time_of_event = e7.get()
        date_of_event = e8.get()
        v=e9.get()
        sql='select * from orders where order_no=%s and user_name=%s'
        cursor.execute(sql,(order_id,u,))
        r=cursor.fetchone()
        if r is None:
            messagebox.showerror("Invalid Order_Id", "Retry")
        # Update login table
        else:
            update_sql = 'UPDATE orders SET package=%s, no_of_ppl=%s, date_event=%s
,time_event=%s ,venue=%s,type=%s WHERE order_no=%s'
            update_values =
(package,num_people,date_of_event,time_of_event,v,dining_type,order_id)
            cursor.execute(update_sql, update_values)
```

```
        mydb.commit()
        st="Refunded"
        stm = 'select amount_per_plate from price where package=%s'
        cursor.execute(stm, (package,))
        r = cursor.fetchone()
        if r is not None:
            amt = r[0]
        mydb.commit()
        amt = float(amt)  # Convert decimal.Decimal to float
        tot = amt * int(num_people)  # Convert num_people to int and perform multiplication

        update_sql1 = 'UPDATE orders SET amount=%s WHERE order_no=%s'
        update_values = (tot, order_id)
        cursor.execute(update_sql1, update_values)
        mydb.commit()
        update='update payment set refund=%s where order_no=%s'
        update_values = (st,order_id)
        cursor.execute(update, update_values)
        mydb.commit()
        form.destroy()
        messagebox.showinfo("ORDER UPDATED", "Money Refunded\nMake New
Payment!")
        make_payment(order_id,tot)

    lab7 = Label(form, text="Order Details:    ", width=38, font=("bold",
25),bg="#3EB489",anchor=W).grid(row=8, sticky=W, pady=4)
    lab1 = Label(form, text="Order ID:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=9, column=0, columnspan=20,sticky=W)
    e3 = Entry(form, width=25, font=('Arial 15'))
    e3.grid(row=9,sticky=W,padx=200)
    lab8 = Label(form, text="type of dining:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=10, column=0, columnspan=20,sticky=W)



    # Create a list of cuisines for the drop-down
    cuisine_options = ["buffet","table sevice"]

    # Create a Combobox and set the values
```

```
    e4 = ttk.Combobox(form,width=23, values=cuisine_options, font=('Arial
15'),state="readonly")
    e4.grid(row=10,sticky=W,padx=200)

    lab9 = Label(form, text="package:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=11, column=0, columnspan=20,sticky=W)
    # Create a list of cuisines for the drop-down
    package_options = ["indian p1", "indian p2", "chinese p1","chinese p2", "indochinese
p1","indochinese p2"]

    e5 = ttk.Combobox(form,width=23, values=package_options, font=('Arial
15'),state="readonly")
    e5.grid(row=11,sticky=W,padx=200)

    lab10 = Label(form, text="No.of people:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=12, column=0, columnspan=20,sticky=W)
    e6 = Entry(form, width=25, font=('Arial 15'))
    e6.grid(row=12,sticky=W,padx=200)

    lab11 = Label(form, text="time of event:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=13, column=0, columnspan=20,sticky=W)
    e7 = Entry(form, width=25, font=('Arial 15'))
    e7.grid(row=13,sticky=W,padx=200)

    lab12 = Label(form, text="date of event:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=14, column=0, columnspan=20,sticky=W)
    e8 = Entry(form, width=25, font=('Arial 15'))
    e8.grid(row=14,sticky=W,padx=200)

    lab13 = Label(form, text="Venue:", width=45, font=("bold", 20),bg="#3EB489",
anchor="w").grid(row=15, column=0, columnspan=20,sticky=W)
    e9 = Entry(form, width=25, font=('Arial 15'))
    e9.grid(row=15,sticky=W,padx=200)

    #button=Button(form,text="Place Order",font=('arial
bold',20),fg='white',command=dostuff,bg='orange',height=1,width=10).grid(row=16,column=0,p
ady=10,sticky=W,padx=200)
    button2=Button(form,text="Update Order",font=('arial
bold',20),fg='white',command=update_order,bg='orange',height=1,width=10).grid(row=17,colum
n=0,pady=10,sticky=W,padx=200)
```

```
    form.mainloop()


def order_details():
    custom_font = ("times new roman", 30, "bold italic")
    custom_font1= ("times new roman", 15, "bold")
    def showd():
        order_no=e3.get()
        sql='select * from orders where order_no=%s and user_name=%s'
        cursor.execute(sql, (order_no,u,))
        r = cursor.fetchone()
        if r is  None:
            messagebox.showerror("Error","Invalid Order Id")
        else:
            stm = 'select  order_no,package ,no_of_ppl,date_event, time_event, venue, amount,type
from orders where order_no=%s and user_name=%s'
            cursor.execute(stm, (order_no,u))
            r = cursor.fetchone()
            if r is not None:
                lab2 = Label(details, text="Order No       :"+str(r[0]), width=30,
font=custom_font1,anchor="w").grid(row=4,column=0,pady=10)
                lab3 = Label(details, text="Package        :"+str(r[1]), width=30,
font=custom_font1,anchor="w").grid(row=5,column=0)
                lab4 = Label(details, text="No. of people :"+str(r[2]),
width=30,font=custom_font1,anchor="w").grid(row=6,column=0,pady=10)
                lab5 = Label(details, text="Date of Event:"+str(r[3]), width=30,
font=custom_font1,anchor="w").grid(row=7,column=0,pady=10)
                lab6 = Label(details, text="Time of Event:"+str(r[4]), width=30,
font=custom_font1,anchor="w").grid(row=8,column=0,pady=10)
                lab7 = Label(details, text="Venue          :"+str(r[5]), width=30,
font=custom_font1,anchor="w").grid(row=9,column=0,pady=10)
                lab8 = Label(details, text="Amount         :"+str(r[6]), width=30,
font=custom_font1,anchor="w").grid(row=10,column=0,pady=10)
                lab9 = Label(details, text="Type           :"+str(r[7]), width=30,
font=custom_font1,anchor="w").grid(row=11,column=0,pady=10)
            else:
                messagebox.showerror("No Record Found", "Invalid Order Id")
    details= Tk()
    details.title("Order details")
    details.geometry("1600x900")
```

```
    details.configure(bg="#F0D9FF")
    lab1 = Label(details, text="Order ID:", width=45, font=custom_font1,bg="#F0D9FF",
anchor="w").grid(row=2, column=0, columnspan=20,sticky=W)
    e3 = Entry(details, width=25, font=('Arial 15'))
    e3.grid(row=2,sticky=W,padx=200)
    button2=Button(details,text="Show
Details",font=custom_font1,fg='white',command=showd,bg='orange',height=1,width=10).grid(ro
w=3,column=0,pady=10,sticky=W,padx=200)
    lab1 = Label(details, text="ORDER DETAILS", width=60,
font=custom_font,bg="#F0D9FF").grid(row = 0, column = 0, sticky = W, pady = 2)



    details.mainloop()

def invoice():
    custom_font = ("times new roman", 30, "bold italic")
    custom_font1= ("times new roman", 15, "bold")
    invoice= Tk()
    invoice.title("Invoice")
    invoice.geometry("1600x900")
    invoice.configure(bg="#CDF0EA")
    lab1 = Label(  invoice, text="INVOICE", width=60,
font=custom_font,bg="#CDF0EA").grid(row = 0, column = 0, sticky = W, pady =12)
    def showi():
        o=e3.get();
        sql='select user_name from orders where order_no=%s'
        cursor.execute(sql,(o,))
        r=cursor.fetchone()
        if r[0]==u:

            stm = 'select  payment_id,payment_method,amount,refund,payment_status from
PAYMENT where ORDER_NO=%s and payment_id=%s '
            cursor.execute(stm, (o,e4.get()))

            r = cursor.fetchone()
            if r is not None:
                lab2 = Label(  invoice, text="Order No.          :"   +str(o), width=30,
font=custom_font1,anchor="w").grid(row=4,column=0,pady=15)
```

```
            lab3 = Label(  invoice, text="Payment ID        :"  +str(r[0]), width=30,
font=custom_font1,anchor="w").grid(row=5,column=0,pady=15)
            lab4 = Label(  invoice, text="Payment Method:"    +str(r[1]),
width=30,font=custom_font1,anchor="w").grid(row=6,column=0,pady=15)
            lab5 = Label(  invoice, text="Amount              :"          +str(r[2]), width=30,
font=custom_font1,anchor="w").grid(row=7,column=0,pady=15)
            lab6 = Label(  invoice, text="Refund              :"          +str(r[3]), width=30,
font=custom_font1,anchor="w").grid(row=8,column=0,pady=15)
            lab7 = Label(  invoice, text="Payment Status   :"    +str(r[4]), width=30,
font=custom_font1,anchor="w").grid(row=9,column=0,pady=15)
         else:
            messagebox.showerror("No Record Found", "Invalid Order Id")
      else:
         messagebox.showerror("No Record Found", "Invalid Order Id")
   lab1 = Label(invoice, text="Order ID:", width=45, font=custom_font1,
anchor="w",bg="#CDF0EA").grid(row=2, column=0, columnspan=20,sticky=W)
   e3 = Entry(invoice, width=25, font=('Arial 15'))
   e3.grid(row=2,sticky=W,padx=200)
   lab2= Label(invoice, text="Transaction ID:", width=45, font=custom_font1,
anchor="w",bg="#CDF0EA").grid(row=3, column=0, columnspan=20,sticky=W)
   e4 = Entry(invoice, width=25, font=('Arial 15'))
   e4.grid(row=3,sticky=W,padx=200)
   button2=Button(invoice,text="Show
Invoice",font=custom_font1,fg='white',command=showi,bg='orange',height=1,width=10).grid(ro
w=5,column=0,pady=10,sticky=W,padx=200)

   invoice.mainloop()


def page2():

   root = Tk()
   root.geometry("800x700")
   root.title("Bon Appétit")
   custom_font = ("times new roman", 30, "bold italic")
   custom_font1= ("times new roman", 15, "bold")
   root.configure(bg="#FFC0D9")

   style = ttk.Style()
   style.configure("TLabel", font=("TkDefaultFont", 50, "bold"))
```

```
    title_label = ttk.Label(root, text="Bon Appétit",
style="TLabel",font=custom_font,background="#FFC0D9")
    title_label.pack(pady=70)
    title_label.config(foreground="black")

    button = Button(root, text="PLACE ORDER", command=place_order,font=custom_font1)
    button.pack(padx=10, pady=10, side="top")

    button = Button(root, text="PRINT INVOICE", command=invoice,font=custom_font1)
    button.pack(padx=10, pady=20, side="top")

    button = Button(root, text="PRINT ORDER DETAILS",
command=order_details,font=custom_font1)
    button.pack(padx=10, pady=10, side="top")

    button = Button(root, text="EDIT REQUEST", command=edit_order,font=custom_font1)
    button.pack(padx=10, pady=30, side="top")

    button = Button(root, text="CANCEL ORDER", command=cancel_order,font=custom_font1)
    button.pack(padx=10, pady=10, side="top")



    root.mainloop()

def login_form():
    global resized_image_tk
    def validate_login():

    # Get the entered username and password
        global u
        u= e1.get()
        p = e2.get()
        v=(u,p)
        sql='select * from login where user_name=%s and password= %s';(u,p)
        k=cursor.execute(sql,v)
        r=cursor.fetchall()
        mydb.commit()
        if r!=[]:
            login.destroy()
```

```
            messagebox.showinfo("Login Successful", "Welcome, " + u + "!")
            page2()
        else:
            messagebox.showerror("Login Failed", "Invalid username or password")
    def create_login():
        global u
        u= e1.get()
        p= e2.get()
        v=(u,p)
        sql='select * from login where user_name=%s and password= %s'
        k=cursor.execute(sql,v)
        r=cursor.fetchall()
        mydb.commit()
        if r!=[]:
            messagebox.showerror("Sign Up Failed", "Username already exists\nCreate a new one")
        else:
            sql='insert into login(user_name,password) values(%s,%s)'
            cursor.execute(sql,v)
            mydb.commit()
            login.destroy()
            messagebox.showinfo("Login Successful", "Welcome, " + u + "!")
            page2()
    # Create the main window
    login = Tk()
    login.title("Login Page")
    login.geometry("2000x1900")
    login.configure(bg="#0097b2")

    custom_font = ("times new roman", 30, "bold italic")
    custom_font2=("times new roman", 20, "bold")

    # Create and place widgets (labels, entries, buttons) in the window
    lab1 = Label(login, text="WELCOME", width=20,
font=custom_font,bg="#0097b2",fg="white")
    lab1.grid(row=0, column=0, sticky=W, pady=20,padx=400)

    lab2 = Label(login, text="USER NAME:", width=13, font=custom_font2,bg="#0097b2",
anchor=W,fg="white")
    lab2.grid(row=7, column=0, sticky=W, pady=50,padx=400)
```

```
    e1 = Entry(login, width=20, font=('Arial 15'))
    e1.grid(row=7, column=0, sticky=W, padx=750)
    lab3 = Label(login, text="PASSWORD:", width=13, font=custom_font2,bg="#0097b2",
anchor=W,fg="white")
    lab3.grid(row=8, column=0, sticky=W, pady=50,padx=400)

    e2 = Entry(login, width=20, font=('Arial 15'),show="*")
    e2.grid(row=8, column=0, sticky=W, padx=750)


    button1 = Button(login,
text="LOGIN",width=10,height=1,font=custom_font2,command=validate_login)
    button1.grid(row=10,column=0,sticky=W,pady=20, padx=550)
    button2 = Button(login, text="SIGN
UP",width=10,height=1,font=custom_font2,command=create_login)
    button2.grid(row=10,column=0,sticky=W,pady=20, padx=750)




    login.mainloop()


# Set the font style and size
custom_font = ("times new roman", 30, "bold italic")
custom_font2 = ("times new roman", 15, "roman")

lab1 = Label(fpage, text="Bon Appétit", width=40, font=custom_font,bg="#FFB3BA")
lab1.grid(row=0, column=0, sticky="ns", pady=7)

lab2 = Label(fpage, text="'Indulge your senses with our exquisite catering services.\n From
elegant weddings to corporate gatherings, we create unforgettable culinary \nexperiences.
Discover a symphony of flavors, meticulous presentation, and \nunparalleled service. Your event,
our passion – where every bite tells a story.'", width=120, font=custom_font2,bg="#FFB3BA")
lab2.grid(row=2, column=0, sticky=W, pady=10)

lab3 = Label(fpage, text="INDIAN: regular package: 1.Chicken Biryani 2.Paneer Tikka 3.Butter
Chicken 4.Vegetable Korma 5.Tandoori Naan 6.Palak Paneer\n Rs.300 per plate ", width=120,
font=custom_font2,bg="#FFDFBA")
```

```
lab3.grid(row=3, column=0, sticky=W,padx=100)
lab4 = Label(fpage, text="INDIAN: large package: 1.Chicken Biryani: 2.Paneer Tikka 3.Butter Chicken: 4.Vegetable Korma 5.Tandoori Naan 6.Palak Paneer \n7.Samosas: 8.Chana Masala 9.Dal Makhani 10.Aloo Gobi 11.Rogan Josh 12.Gulab Jamun\n Rs.600 per plate ", width=120, font=custom_font2,bg="#FFDFBA")


lab4.grid(row=4, column=0, sticky=W,padx=100)
lab5 = Label(fpage, text="CHINESE: regular package: 1.Sweet and Sour Chicken 2.Beef and Broccoli 3.Chow Mein 4.Kung Pao Shrimp 5.egg Rolls 6.General Tso's Tofu\n Rs.400 per plate ", width=120, font=custom_font2,bg="#FFDFBA")
lab5.grid(row=5, column=0, sticky=W,padx=100)
lab6 = Label(fpage, text="CHINESE: large package: 1.Sweet,padx=100 and Sour Chicken 2.Beef and Broccoli 3.Chow Mein 4.Kung Pao Shrimp 5.egg Rolls 6.General Tso's Tofu \n7.Perking Duck 8.Shrimp Fried Rice 9.Mongolian Beef 10.Dim Sum Platter 11.Mapo Tofu 12.Hot And Sour Soup\n Rs.700 per plate ", width=120, font=custom_font2,bg="#FFDFBA")

lab6.grid(row=6, column=0, sticky=W,padx=100)
lab7 = Label(fpage, text="INDOCHINESE: regular package: 1.Gobi Manchurian 2.Chicken Manchurian 3.Hakka Noodles 4.Chilli Chicken 5.Paneer Chilli 6.Schezwan Fried Rice  \n Rs.350 per plate ", width=120, font=custom_font2,bg="#FFDFBA")
lab7.grid(row=7, column=0, sticky=W,padx=100)
lab8 = Label(fpage, text="INDOCHINESE: large package: 1.Gobi Manchurian 2.Chicken Manchurian 3.Hakka Noodles 4.Chilli Chicken 5.Paneer Chilli 6.Schezwan Fried Rice \n7.Dragon Chicken 8.Vegetable Manchow Soup 9.Spring Roll Chaat 10.Chowmein Bhel 11.Schezwan Paneer Pizza 12.Singapore Fried Prawns\n Rs.650 per plate ", width=120, font=custom_font2,bg="#FFDFBA")
lab8.grid(row=8, column=0, sticky=W,padx=100)
button = Button(fpage, text="LOGIN/SIGN UP",
command=login_form,height=3,width=15,bg="#FFDFBA")
button.grid(row=10, column=0, pady=30)


# Create a Canvas widge
#canvas = Canvas(fpage, width=300, height=200)
#canvas.grid()
fpage.mainloop()
```

OUTPUT

```
mysql> select * from price;
+-----------------+-------------------+
| package         | amount_per_plate  |
+-----------------+-------------------+
| chinese p1      |            400.00 |
| chinese p2      |            700.00 |
| indian p1       |            300.00 |
| indian p2       |            600.00 |
| indochinese p1  |            350.00 |
| indochinese p2  |            650.00 |
+-----------------+-------------------+
6 rows in set (0.00 sec)
```

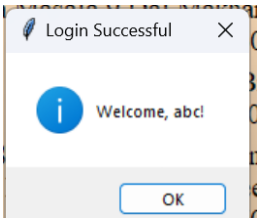
```
mysql> select * from payment;
+------------+----------+----------------+-----------+----------+----------------+
| payment_id | order_no | payment_method | amount    | refund   | payment_status |
+------------+----------+----------------+-----------+----------+----------------+
| 1209863    |  8742980 | CREDIT CARD    |   1400.00 | NO       | TXN_SUC        |
| 2042459    |  8742980 | CREDIT CARD    |   1400.00 | NO       | TXN_SUC        |
| 2206642    |  5496013 | DEBIT CARD     | 150000.00 | NO       | TXN_SUC        |
| 3057449    |  8742980 | CREDIT CARD    |   1400.00 | NO       | TXN_SUC        |
| 3759150    |  1209894 | DEBIT CARD     |   1200.00 | NO       | TXN_SUC        |
| 6935734    |  5487694 | CREDIT CARD    | 650000.00 | NO       | TXN_SUC        |
| 7104459    |  5496013 | DEBIT CARD     | 700000.00 | Refunded | TXN_SUC        |
| 7471574    |  3919944 | CREDIT CARD    |  41400.00 | NO       | TXN_SUC        |
| 8929138    |  5487694 | CREDIT CARD    |  27600.00 | Refunded | TXN_SUC        |
+------------+----------+----------------+-----------+----------+----------------+
9 rows in set (0.00 sec)
```

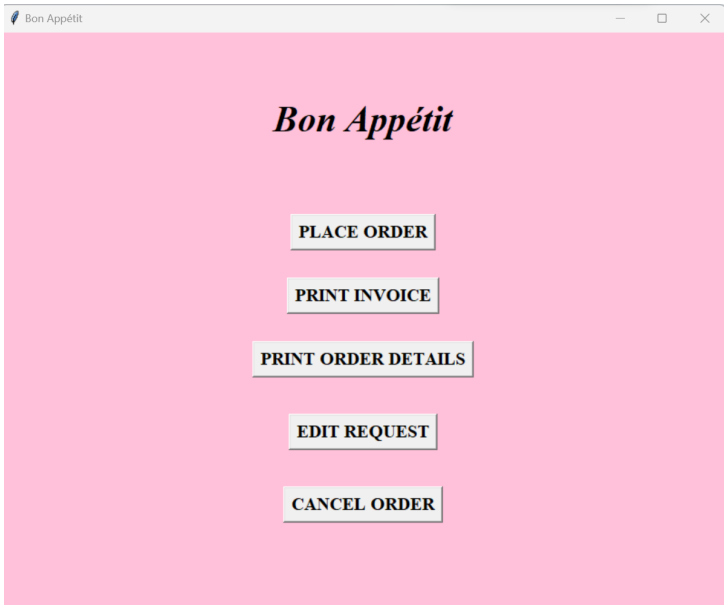
```
mysql> select * from orders;
+-----------+----------+-------------+----------------+-----------+------------+------------+----------+------------+
| user_name | order_no | type        | package        | no_of_ppl | date_event | time_event | venue    | amount     |
+-----------+----------+-------------+----------------+-----------+------------+------------+----------+------------+
| Nikita    |  1209894 | buffet      | indian p2      |         2 | 2023-12-20 | 2:30       | Bali     |    1200.00 |
| yoo       |  3919944 | buffet      | indian p2      |        69 | 31         | 12         | rset     |   41400.00 |
| yoo       |  5070860 | table sevice | indochinese p1 |      3000 | 12-4-24    | 12         | bali     | 1050000.00 |
| yoo       |  5448018 | buffet      | indian p2      |        69 | 30         | 12         | rset     |   41400.00 |
| yoo       |  6475640 | table sevice | chinese p2     |      1000 | 3-9-2023   | 12         | Kerala   |  700000.00 |
| yoo       |  8152152 | table sevice | chinese p1     |      1000 | 12-9-2023  | 12         | Chennai  |  400000.00 |
| anu       |  8742980 | buffet      | indochinese p1 |         4 | 1-1-24     | 9.00pm     | Lal Kila |    1400.00 |
| yoo       |  9837434 | table sevice | indochinese p1 |     10000 | 12-4-24    | 21         | bali     | 3500000.00 |
+-----------+----------+-------------+----------------+-----------+------------+------------+----------+------------+
8 rows in set (0.00 sec)
```

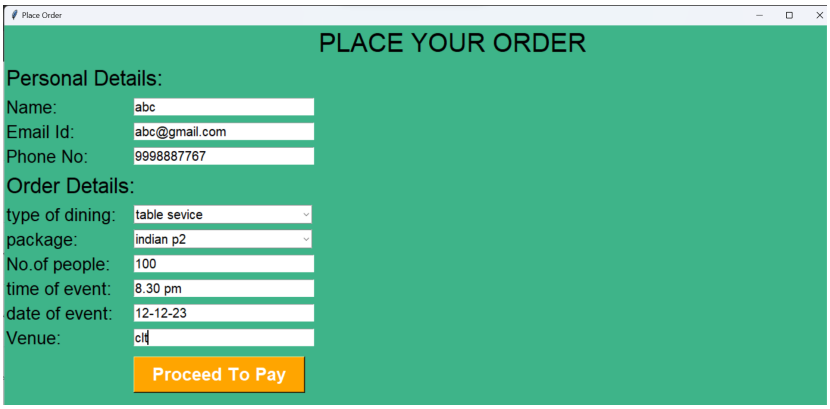
```
mysql> select *from login;
+-----------+------------+---------------+----------------------+------------+
| User_name | password   | name          | email                | phone_no   |
+-----------+------------+---------------+----------------------+------------+
| anu       | goodgirl   | ananya suffin | anusuffin4@gmail.com | 1312335345 |
| Nikita    | 123@nikita | Nikita        | ygsj                 | 12345678   |
| yoo       | 123        | nandhana      | nan@gmail.com        | 123445432  |
+-----------+------------+---------------+----------------------+------------+
3 rows in set (0.00 sec)
```

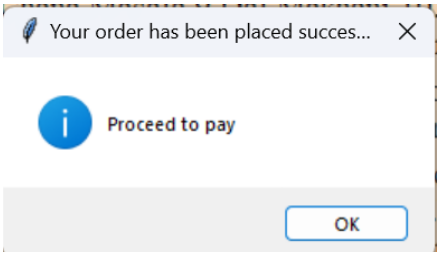

Order details
Bon Appétit
'Indulge your senses with our exquisite catering services.
From elegant weddings to corporate gatherings, we create unforgettable culinary
experiences. Discover a symphony of flavors, meticulous presentation, and
unparalleled service. Your event, our passion – where every bite tells a story.'
INDIAN: regular package: 1.Chicken Biryani 2.Paneer Tikka 3.Butter Chicken 4.Vegetable Korma 5.Tandoori Naan 6.Palak Pane
Rs.300 per plate
INDIAN: large package: 1.Chicken Biryani: 2.Paneer Tikka 3.Butter Chicken: 4.Vegetable Korma 5.Tandoori Naan 6.Palak Panee
7.Samosas: 8.Chana Masala 9.Dal Makhani 10.Aloo Gobi 11.Rogan Josh 12.Gulab Jamun
Rs.600 per plate
CHINESE: regular package: 1.Sweet and Sour Chicken 2.Beef and Broccoli 3.Chow Mein 4.Kung Pao Shrimp 5.egg Rolls 6.General Ts
Rs.400 per plate
CHINESE: large package: 1.Sweet,padx=100 and Sour Chicken 2.Beef and Broccoli 3.Chow Mein 4.Kung Pao Shrimp 5.egg Rolls 6.Genera
7.Perking Duck 8.Shrimp Fried Rice 9.Mongolian Beef 10.Dim Sum Platter 11.Mapo Tofu 12.Hot And Sour Soup
Rs.700 per plate
INDOCHINESE: regular package: 1.Gobi Manchurian 2.Chicken Manchurian 3.Hakka Noodles 4.Chilli Chicken 5.Paneer Chilli 6.Schezwan
Rs.350 per plate
INDOCHINESE: large package: 1.Gobi Manchurian 2.Chicken Manchurian 3.Hakka Noodles 4.Chilli Chicken 5.Paneer Chilli 6.Schezwan
7.Dragon Chicken 8.Vegetable Manchow Soup 9.Spring Roll Chaat 10.Chowmein Bhel 11.Schezwan Paneer Pizza 12.Singapore Fried
Rs.650 per plate
LOGIN/SIGN UP

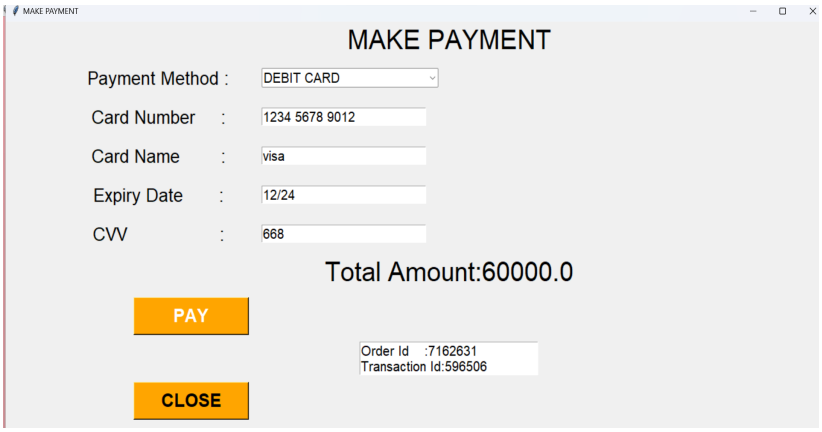

Login Page
WELCOME
USER NAME:
abc
PASSWORD:
LOGIN
SIGN UP

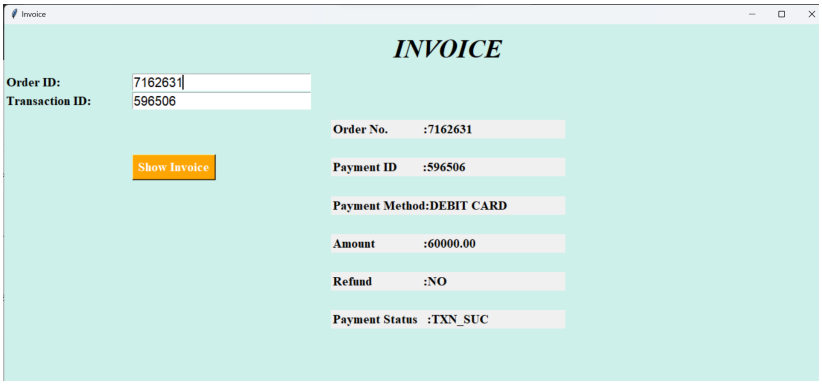

Login Successful
Welcome, abc!
OK


Bon Appétit
Bon Appétit
PLACE ORDER
PRINT INVOICE
PRINT ORDER DETAILS
EDIT REQUEST
CANCEL ORDER

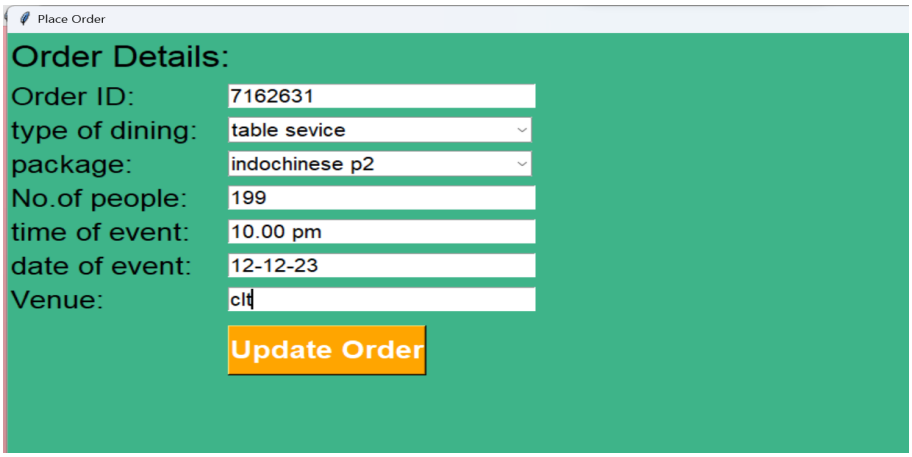

Place Order
PLACE YOUR ORDER
Personal Details:
Name:
abc
Email Id:
abc@gmail.com
Phone No:
9998887767
Order Details:
type of dining:
table sevice
package:
indian p2
No.of people:
100
time of event:
8.30 pm
date of event:
12-12-23
Venue:
clt
Proceed To Pay

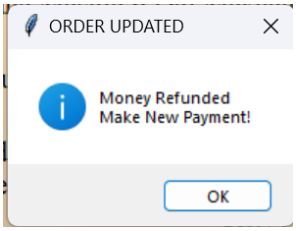

Your order has been placed succes...
Proceed to pay
OK

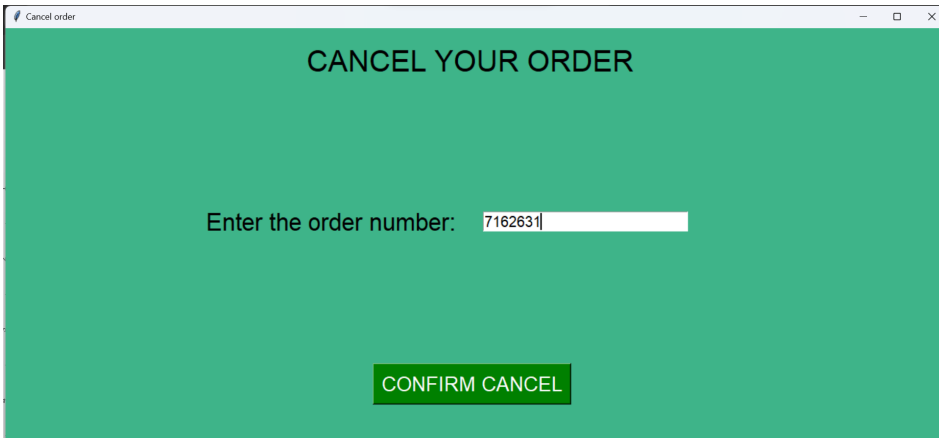

MAKE PAYMENT
MAKE PAYMENT
Payment Method :
DEBIT CARD
Card Number :
1234 5678 9012
Card Name :
visa
Expiry Date :
12/24
CVV :
668
Total Amount:60000.0
PAY
Order Id :7162631
Transaction Id:596506
CLOSE

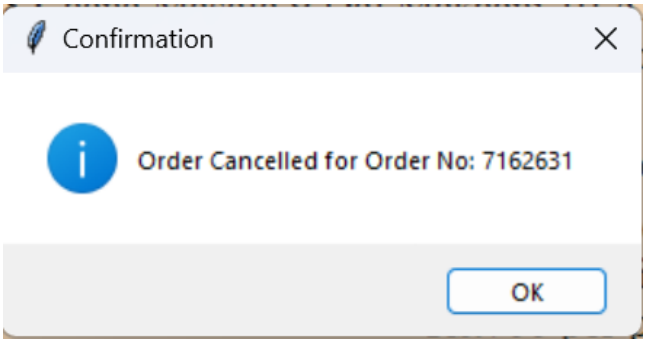

Invoice
INVOICE
Order ID:
7162631
Transaction ID:
596506
Show Invoice
Order No. :7162631
Payment ID :596506
Payment Method:DEBIT CARD
Amount :60000.00
Refund :NO
Payment Status :TXN_SUC


Order details
ORDER DETAILS
Order ID:
7162631
Show Details
Order No :7162631
Package :indian p2
No. of people :100
Date of Event:12-12-23
Time of Event:8.30 pm
Venue :clt
Amount :60000.00
Type :table sevice


Place Order
Order Details:
Order ID:
7162631
type of dining:
table sevice
package:
indochinese p2
No.of people:
199
time of event:
10.00 pm
date of event:
12-12-23
Venue:
clt
Update Order


ORDER UPDATED
Money Refunded
Make New Payment!
OK


Cancel order
CANCEL YOUR ORDER
Enter the order number:
7162631
CONFIRM CANCEL


Confirmation
Order Cancelled for Order No: 7162631
OK

FRONT END/BACK END

Front end: Python(Tkinter)

Tkinter is the standard GUI (Graphical User Interface) toolkit that comes bundled with Python. It provides a set of tools and libraries for creating desktop applications with graphical interfaces. Tkinter is based on the Tk GUI toolkit and is the de facto standard for building GUI applications using Python.

Key Features:

Simplicity and Accessibility:

Tkinter is easy to learn and use, making it a popular choice for beginners who want to create simple GUI applications.

Cross-Platform:

Tkinter is platform-independent and works on various operating systems, including Windows, macOS, and Linux.

Integration with Python:

As Tkinter is included in the Python standard library, there is no need for additional installations or dependencies. It seamlessly integrates with Python.
Widgets:

Tkinter provides a variety of GUI elements called widgets, including buttons, labels, entry fields, text boxes, and more. These widgets form the building blocks of a GUI.

Back end: MySQL

MySQL is an open-source relational database management system (RDBMS) that uses SQL (Structured Query Language). It is one of the most popular databases in the world, known for its reliability, ease of use, and strong community support. MySQL is commonly used for web applications and is a key component of the LAMP (Linux, Apache, MySQL, PHP/Python/Perl) stack.

Key Features:

Relational Database:

MySQL follows the relational database model, allowing the organization of data into tables with defined relationships.
SQL Support:

MySQL supports SQL, which is a standardized language for managing and querying relational databases. Users can perform operations like SELECT, INSERT, UPDATE, and DELETE.
Open Source:

MySQL is an open-source database system, providing free access to its source code. This has contributed to its widespread adoption and continuous improvement by the community.
Cross-Platform Compatibility:

MySQL is designed to run on various platforms, including Linux, Windows, and macOS, making it versatile for different deployment environments.
Scalability:

MySQL is scalable and can handle small to large-scale databases. It is used by both small businesses and large enterprises.

BIBLIOGRAPHY

- https://www.geeksforgeeks.org/
- https://www.tutorialspoint.com/index.htm
- https://www.w3schools.com/
- CHATGPT

CONCLUSION

In conclusion, the development and implementation of the Catering Management System represent a significant stride forward in redefining how catering businesses operate and deliver services. This comprehensive system addresses the complexities of the catering industry by integrating advanced technologies and streamlining various aspects of the catering process.

The system's success lies in its ability to enhance operational efficiency, optimize resource utilization, and provide valuable insights for informed decision-making. By introducing user-friendly interfaces for clients to place orders and customize menus, the Catering Management System not only meets but exceeds customer expectations, contributing to increased satisfaction and loyalty.

Security measures embedded in the system instill trust by safeguarding sensitive information, aligning with data protection regulations. The system's scalability ensures its applicability across diverse catering enterprises, from small-scale operations to large-scale ventures, fostering adaptability and sustainable growth.

In essence, the Catering Management System emerges as a transformative solution, positioning catering businesses to thrive in a competitive and dynamic industry. By embracing technology, optimizing processes, and prioritizing customer satisfaction, this system heralds a new era in catering management, where efficiency, innovation, and adaptability converge for unparalleled success.