CAPSTONE PROJECT REPORT

(Project Term January-May 2023)

(RESTAURANT MANAGEMENT SYSTEM)

Submitted By

Name of the Student : TOLUPUNOORI RAJSHEKAR

Registration Number : 12100335

Section: K21PU | Roll no: B42

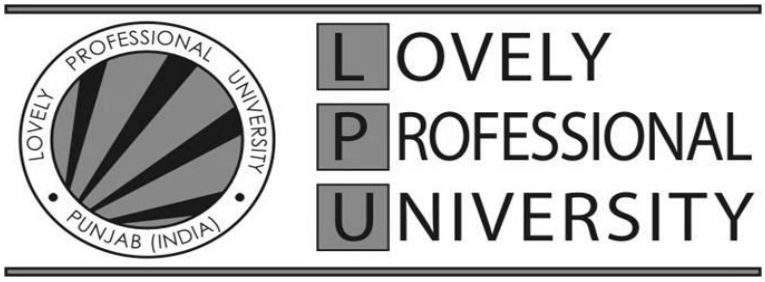
Course Code: INT216

Under the Guidance of

Ved Prakash Chaubey(63892)

(Name of faculty mentor with designation)

School of Computer Science and Engineering



PAC Form

PROVISIONAL ACCEPTANCE CERTIFICATE

This is to certify that Rajshekar, a student of Computer Science And Engineering at Lovely Professional University, has successfully submitted a project proposal entitled Restaurant management System using tkinter for 4th Semester.
The proposal has been reviewed and approved by the Upgrad Team on 26th April 2023. The project is now provisionally accepted and Rajshekar is authorized to proceed with the project as per the guidelines provided by the department/committee.
This certificate is valid until the completion of the project and submission of the final project report.

Team Upgrad,
Lovely Professional University,
26th April 2023.

Congratulations on receiving the Provisional Acceptance Certificate for your college project!

DECLARATION

I hereby declare that the project work entitled “Restaurant Management System using Tkinter” is an authentic record of our own work carried out as requirements of Capstone Project for the award of B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Ved Prakash Chaubey(63892), during January to May 2023. All the information furnished in this capstone project report is based on my own intensive work and is genuine.

Name of Student : Tolupunoori Rajshekar
Registration Number: 12100335

(Signature of Student)
Rajshekar

Date: 26th April 2023

CERTIFICATE

This is to certify that the declaration statement made by this student is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of B.Tech degree in Computer Science And Engineering from Lovely Professional University, Phagwara.

Signature and Name of the Mentor

Designation : Lectuer

School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab.

Date : 26th April 2023

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have supported me in completing my college project titled "Restaurant Management System".

First and foremost, I would like to thank my project supervisor Ved Prakash Chaubey for their guidance, support, and valuable suggestions throughout the project. Their constant feedback helped me to improve the project and achieve my goals.

I would also like to thank the faculty members of School of Computer Science at Lovely Professional University for their support and encouragement. Their expertise in the field and their willingness to share their knowledge has been invaluable.

I would like to extend my appreciation to my friends and family who provided me with the motivation, encouragement, and emotional support needed to complete this project. Their support has been essential in helping me to overcome the challenges faced during the course of this project.

Finally, I would like to acknowledge the contribution of various online resources, tutorials, and forums, which helped me to learn and implement the various aspects of the project.

Once again, thank you to everyone who has contributed to the successful completion of this project.

TABLE OF CONTENTS

Inner first page…….(i)

PAC form……(ii)

Declaration…..……..(iii)

Certificate……..…….……(iv)

Acknowledgement……...…….....(v)

Table of Contents……...……(vi)

1. Introduction……..7

2. Problem Statement……..……8

3. Existing System.……9

3.1 Introduction

3.2 Exiting Software

4. Problem Analysis……10

4.1 Product Definition

4.2 Feasibility Analysis

5. Software Requirement Analysis……11

5.1 Introduction

5.2 General Description

5.3Specific Requirements

6.Design…….12

6.1 System Design

7.Testing……13

7.1Functional Testing

7.2structural testing

8.User Manual: Restaurant Management System.……14

9.Sourcecode……..15

1 Introduction

Below are two subheadings that will be discussed in this tutorial:

Tkinter for the graphical user interface. The purpose of this system is to help restaurant staff manage orders and calculate the cost of meals, including taxes and service charges. The code features a calculator that can perform basic arithmetic operations and a user interface that allows users to input the quantities of various food items.
The code begins by importing the necessary libraries, such as Tkinter, random, time, twilio.rest.Client (for sending notifications), and PIL (Python Imaging Library) for displaying images.
The program then sets up the main window (root) using Tkinter, configuring the window's dimensions, title, and creating frames for organizing the layout. The time and date are also displayed at the top of the window.
The calculator is implemented using functions that handle button clicks, clearing the display, and evaluating expressions. The calculator buttons are created with the Tkinter Button widget and arranged using the grid layout manager.
The main part of the Restaurant Management System consists of several input fields for entering the quantities of different food items, such as fries, burgers, and drinks. The cost of each item is calculated and displayed, along with the subtotal, tax, service charge, and total cost.
The application provides buttons for various actions such as calculating the bill, resetting the input fields, and exiting the application.
Overall, this Restaurant Management System is a handy tool for managing orders and calculating costs in a restaurant setting, demonstrating the flexibility and ease of use of Tkinter for creating graphical applications in Python.

1.1 **Creating the GUI with Tkinter**:

In this section, we will cover how to create a simple GUI for the Restaurant Management System game using Tkinter.We will create a input functions to take input for the all the items ordered by the customer so that then we can create a bill.

1.2 **Implementing the logic**:

First, we import the necessary modules. In this case, we use tkinter for the graphical user interface (GUI) and tkinter.messagebox for popup messages.

We define a class Application which inherits from the tk.Tk class. This class represents our application and will contain all the logic for our restaurant management system.

Inside the __init__ method of the Application class, we initialize our application. We set the title of the window to "Restaurant Management System".

We then define the menu items for the restaurant and their corresponding prices in a dictionary called self.menu. The keys of the dictionary are the names of the items and the values are the prices. We also create a dictionary called self.order to keep track of the order. Initially, the quantities of all items in the order are set to 0.

We then create widgets for each item in the menu using a for loop. For each item, we create a label displaying the name of the item and an

entry widget where the user can enter the quantity of the item they want to order. The entry widget is configured to only accept integer inputs.

After creating widgets for all items, we create a button labeled 'Total'. When this button is clicked, it calls the self.calculate_total method.

The self.calculate_total method is not defined in the given code, but it would typically be implemented to calculate the total cost of the order by multiplying the quantity of each item by its price and summing up the results.

2. Problem Statement

Running a restaurant involves managing a plethora of activities like taking orders, managing inventory, handling reservations, managing employee schedules, and ensuring customer satisfaction. In the current digital age, managing all these activities manually has become a burdensome task, prone to human errors, and inefficient, leading to potential financial losses and reduced customer satisfaction.

Rationale:

Efficiency and Accuracy: Utilizing a digital RMS can significantly improve the accuracy of orders and inventory management. It can eliminate human errors such as incorrect order entries or miscalculations in inventory, which can lead to financial losses. By automating these tasks, the system can also speed up the service, leading to improved customer satisfaction.

Scope of the Study:

This study aims to develop a comprehensive and user-friendly Restaurant Management System (RMS) using Python and suitable database management technologies. The system is intended to streamline the processes involved in running a restaurant by providing a unified platform for tasks such as table reservations, online ordering, inventory management, and employee scheduling.

Product Definition

The product is a simple restaurant management system with a graphical user interface (GUI) for placing orders. The system allows users to select quantities of different menu items and calculate the total cost of their order. The product is designed to provide a user-friendly and efficient way for customers to order food in a restaurant setting.

Feasibility Analysis

Technical Feasibility: The implementation of the code is straightforward using the Tkinter library in Python. Tkinter provides a simple and easy-to-use framework for building GUI applications, and the code provided uses its basic functionality. Given that the logic and structure of the code are already in place, the technical feasibility of the product is high.

Operational Feasibility: The code demonstrates a basic restaurant ordering system that would be easy for users to understand and interact with. However, for real-world deployment, additional features and improvements may be necessary. For instance, the system should be able to handle menu updates, special promotions, and various payment methods. Additionally, integrating the system with the restaurant's existing point-of-sale (POS) and inventory management systems would be necessary to ensure a seamless operational experience.

Economic Feasibility: The development costs for this product are relatively low, as it is built using open-source technologies (Python and Tkinter). However, as the product lacks advanced features and customizations, it may not provide significant cost savings or increased efficiency for the restaurant. To enhance its economic feasibility, the product could be expanded to include additional functionalities like order tracking, table reservations, or customer feedback.

Legal Feasibility: There are no apparent legal concerns with the implementation of the code. However, when deploying the system, the restaurant must ensure that they comply with all relevant regulations and guidelines, such as data protection and privacy laws.

Schedule Feasibility: The current implementation is a basic prototype, and it would require more development and testing before being deployed in a real-world setting. The timeline for completion would depend on the complexity of additional features and integrations required by the restaurant. However, given the simplicity of the current code and the availability of development resources, the schedule feasibility is likely achievable.

System Design

The current system is relatively simple, but here's a basic overview of its design:

User Interface (UI): The UI is designed using Python's Tkinter library. It consists of a graphical user interface (GUI) that presents a list of food items along with their prices. Users can select the quantity of each item they wish to order. The UI also provides a button to calculate and display the total cost of the selected items.

Backend Logic: The backend of the system is also implemented using Python. It processes the user's selections, calculates the total cost based on quantities and item prices, and displays the result on the UI.

Data Storage: Currently, the system does not include any form of data storage. It processes orders in real-time, and the data does not persist once the application is closed.

For a more robust system design, we could consider the following enhancements:

Database Integration: Integrate a database system to store order details, customer information, menu items, and prices. This could be a simple file-based database or a more complex SQL or NoSQL database, depending on the scale of the restaurant and the number of transactions.

Error Handling and Validation: Implement error handling to account for invalid user inputs and other potential issues. Validation checks could be included to ensure that input fields are not left blank and that quantities entered are valid numbers.

User Accounts and Authentication: If the system were to be used by multiple staff members or across multiple devices, user accounts and authentication might be necessary. This would allow for tracking of orders by staff member and provide additional security.

Integration with Other Systems: If the restaurant has other software systems in place, such as for inventory management or accounting, the order management system could be designed to integrate with these. This would allow for real-time updates of inventory based on orders and automatic recording of sales data.

Expanded UI Features: The UI could be expanded to include additional features, such as a view of the current order, options to modify or cancel items in the order, special requests, or promotional codes.

Scalability and Performance: As the restaurant grows, the system should be designed in a way that it can handle increasing numbers of orders and menu items without a decrease in performance. This could involve optimizing the backend code, using a more efficient database system, or even transitioning to a server-based system if necessary.

Functional Testing

Functional testing focuses on validating the system's functionality against the specified requirements. The goal is to ensure that the system behaves as expected. For the Restaurant Order Management System, we can perform the following functional tests:

Test if the GUI loads properly and displays all the menu items with their respective prices.
Test the quantity input field by entering valid and invalid inputs, such as negative numbers, non-integer values, or empty inputs.
Test the "Calculate Total" button functionality by selecting various combinations of menu items and quantities, and checking if the total cost is calculated correctly.
Test the error handling and validation features, such as input validation for the quantity fields and proper error messages.

Structural Testing

Structural testing, also known as white-box testing, focuses on the internal structure of the system, including code and logic. The goal is to ensure that the system's internal components are functioning correctly.

For the Restaurant Order Management System, we can perform the following structural tests:

Code Coverage: Ensure that all lines of code and branches in the logic are executed during testing. This can be achieved by designing test cases that cover all possible paths through the code.
Boundary Testing: Test the system's behavior at the boundaries of input values, such as the minimum and maximum quantity of items that can be ordered.
Unit Testing: Test individual components of the system, such as the function responsible for calculating the total cost. This can be done by writing test cases with various inputs and checking if the output is as expected.
Integration Testing: Test the interaction between different components of the system, such as the UI and the backend logic, to ensure that they work together seamlessly.
These testing approaches will help in identifying and resolving any issues in the Restaurant Order Management System and ensuring that it functions as expected. Performing both functional and structural testing will provide a comprehensive assessment of the system's quality and reliability.

User Manual for the Restaurant Order Management System

1. Introduction

The Restaurant Order Management System is a simple and efficient tool that assists in managing orders in a restaurant setting. It displays a list of menu items along with their prices and allows users to input the quantity of each item they wish to order. The system calculates the total cost of the order in real-time.

2. Getting Started

2.1 System Requirements

The system requires a standard web browser (Google Chrome, Mozilla Firefox, Safari, etc.). Ensure that JavaScript is enabled for the best experience.

2.2 Accessing the System

The Restaurant Order Management System is web-based. Simply open your web browser and navigate to the provided URL.

3. Using the System

3.1 Viewing the Menu

Upon accessing the system, you will see a list of menu items along with their respective prices. Scroll through to view all available items.

3.2 Placing an Order

Next to each menu item, there's an input field where you can enter the quantity of that item you wish to order.

Input the quantity in the text box next to the desired menu item(s).
Make sure to input only numerical values (e.g., 1, 2, 3, etc.). Non-numerical inputs will not be accepted.
There is no limit on the quantity you can order, but it needs to be a positive number.

3.3 Calculating the Total Cost

After inputting the quantities for all items you wish to order, click the "Calculate Total" button. The total cost of your order will be displayed immediately.

4. Troubleshooting

4.1 Invalid Input

If you input a non-numerical value or a negative number as a quantity, an error message will appear prompting you to input a valid quantity. Please input a positive numerical value.

4.2 System Errors

If the system is not calculating the total cost correctly or is not responding, try refreshing the page. If the problem persists, contact the system administrator.

5. Conclusion

The Restaurant Order Management System is an easy-to-use tool that streamlines the ordering process, making it quick, efficient, and accurate. It is intuitive to use, and the above guide should cover all the basic operations you need to know. If you encounter any issues or have any further questions, please don't hesitate to contact the system support team. Enjoy your dining experience!

CODE:

```
    def Ref():
        x=random.randint(12980, 50876)
        randomRef = str(x)
        # rand.set(randomRef)

        cof =float(Fries.get())
        colfries= float(Largefries.get())
        cob= float(Burger.get())
        cofi= float(Filet.get())
        cochee= float(Cheese_burger.get())
        codr= float(Drinks.get())

        costoffries = cof*25
        costoflargefries = colfries*40
        costofburger = cob*35
        costoffilet = cofi*50
        costofcheeseburger = cochee*50
        costofdrinks = codr*35

        costofmeal = "Rs.",str('%.2f'% (costoffries +  costoflargefries + costofburger + costoffilet + costofcheeseburger + costofdrinks))
        PayTax=((costoffries +  costoflargefries + costofburger + costoffilet +  costofcheeseburger + costofdrinks)*0.33)
        Totalcost=(costoffries +  costoflargefries + costofburger + costoffilet  + costofcheeseburger + costofdrinks)
        Ser_Charge=((costoffries +  costoflargefries + costofburger + costoffilet + costofcheeseburger + costofdrinks)/99)
        Service="Rs.",str('%.2f'% Ser_Charge)
        OverAllCost="Rs.",str( PayTax + Totalcost + Ser_Charge)
        PaidTax="Rs.",str('%.2f'% PayTax)

        Service_Charge.set(Service)
        cost.set(costofmeal)
        Tax.set(PaidTax)
        Subtotal.set(costofmeal)
        Total.set(OverAllCost)


    def qexit():
        root.destroy()

    def reset():
        rand.set("")
        Fries.set("")
        Largefries.set("")
        Burger.set("")
        Filet.set("")
        Subtotal.set("")
        Total.set("")
        Service_Charge.set("")
        Drinks.set("")
        Tax.set("")
        cost.set("")
        Cheese_burger.set("")


    btn7=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="7",bg="black", command=lambda: btnclick(7) )
    btn7.grid(row=2,column=0)

    btn8=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="8",bg="black", command=lambda: btnclick(8) )



    btn7=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="7",bg="black", command=lambda: btnclick(7) )
    btn7.grid(row=2,column=0)

    btn8=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="8",bg="black", command=lambda: btnclick(8) )
    btn8.grid(row=2,column=1)

    btn9=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="9",bg="black", command=lambda: btnclick(9) )
    btn9.grid(row=2,column=2)

    Addition=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="+",bg="black", command=lambda: btnclick("+") )
    Addition.grid(row=2,column=3)
    #---------------------------------------------------------------------------------------------
    btn4=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="4",bg="black", command=lambda: btnclick(4) )
    btn4.grid(row=3,column=0)

    btn5=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="5",bg="black", command=lambda: btnclick(5) )
    btn5.grid(row=3,column=1)

    btn6=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="6",bg="black", command=lambda: btnclick(6) )
    btn6.grid(row=3,column=2)

    Substraction=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="-",bg="black", command=lambda: btnclick("-") )
    Substraction.grid(row=3,column=3)
    #---------------------------------------------------------------------------------------------
    btn1=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="1",bg="black", command=lambda: btnclick(1) )
    btn1.grid(row=4,column=0)

    btn2=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="2",bg="black", command=lambda: btnclick(2) )
    btn2.grid(row=4,column=1)

    btn3=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="3",bg="black", command=lambda: btnclick(3) )
    btn3.grid(row=4,column=2)

    multiply=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="*",bg="black", command=lambda: btnclick("*") )
    multiply.grid(row=4,column=3)
    #---------------------------------------------------------------------------------------------
    btn0=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="0",bg="black", command=lambda: btnclick(0) )
    btn0.grid(row=5,column=0)

    btnc=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="c",bg="black", command=clrdisplay)
    btnc.grid(row=5,column=1)

    btnequal=Button(f2,padx=16,pady=16,bd=4,width = 16, fg="black", font=('ariel', 20 ,'bold'),text="=",bg="black",command=eqals)
    btnequal.grid(columnspan=4)

    Decimal=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text=".",bg="black", command=lambda: btnclick(".") )
    Decimal.grid(row=5,column=2)

    Division=Button(f2,padx=16,pady=16,bd=4, fg="black", font=('ariel', 20 ,'bold'),text="/",bg="black", command=lambda: btnclick("/") )
    Division.grid(row=5,column=3)
    status = Label(f2,font=('aria', 15, 'bold'),width = 16, text="~By Rajashekar",bd=2,relief=SUNKEN)
    status.grid(row=7,columnspan=3)

    #---------------------------------------------------------------------------------------------
    rand = StringVar()
```

```
    #-------------------------------------------------------------------------------------------
    rand = StringVar()
    Fries = StringVar()
    Largefries = StringVar()
    Burger = StringVar()
    Filet = StringVar()
    Subtotal = StringVar()
    Total = StringVar()
    Service_Charge = StringVar()
    Drinks = StringVar()
    Tax = StringVar()
    cost = StringVar()
    Cheese_burger = StringVar()


    lblreference = Label(f1, font=( 'aria' ,16, 'bold' ),text="Order No.",fg="steel blue",bd=10,anchor='w')
    lblreference.grid(row=0,column=0)
    txtreference = Entry(f1,font=('ariel' ,16,'bold'), textvariable=rand , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtreference.grid(row=0,column=1)

    lblfries = Label(f1, font=( 'aria' ,16, 'bold' ),text="Chicken Biryani",fg="steel blue",bd=10,anchor='w')
    lblfries.grid(row=1,column=0)
    txtfries = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Fries , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtfries.grid(row=1,column=1)

    lblLargefries = Label(f1, font=( 'aria' ,16, 'bold' ),text="Mutton Biryani",fg="steel blue",bd=10,anchor='w')
    lblLargefries.grid(row=2,column=0)
    txtLargefries = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Largefries , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtLargefries.grid(row=2,column=1)


    lblburger = Label(f1, font=( 'aria' ,16, 'bold' ),text="Mutton Curry",fg="steel blue",bd=10,anchor='w')
    lblburger.grid(row=3,column=0)
    txtburger = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Burger , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtburger.grid(row=3,column=1)

    lblFilet = Label(f1, font=( 'aria' ,16, 'bold' ),text="Veg Palao",fg="steel blue",bd=10,anchor='w')
    lblFilet.grid(row=4,column=0)
    txtFilet = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Filet , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtFilet.grid(row=4,column=1)

    lblCheese_burger = Label(f1, font=( 'aria' ,16, 'bold' ),text="Pulihora",fg="steel blue",bd=10,anchor='w')
    lblCheese_burger.grid(row=5,column=0)
    txtCheese_burger = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Cheese_burger , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtCheese_burger.grid(row=5,column=1)

    #-------------------------------------------------------------------------------------------
    lblDrinks = Label(f1, font=( 'aria' ,16, 'bold' ),text="Drinks",fg="steel blue",bd=10,anchor='w')
    lblDrinks.grid(row=0,column=2)
    txtDrinks = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Drinks , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtDrinks.grid(row=0,column=3)

    lblcost = Label(f1, font=( 'aria' ,16, 'bold' ),text="Cost",fg="steel blue",bd=10,anchor='w')
    lblcost.grid(row=1,column=2)
    txtcost = Entry(f1,font=('ariel' ,16,'bold'), textvariable=cost , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtcost.grid(row=1,column=3)


    lblService_Charge = Label(f1, font=( 'aria' ,16, 'bold' ),text="Service Charge",fg="steel blue",bd=10,anchor='w')
    lblService_Charge.grid(row=2,column=2)
    txtService_Charge = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Service_Charge , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtService_Charge.grid(row=2,column=3)

    lblTax = Label(f1, font=( 'aria' ,16, 'bold' ),text="Tax",fg="steel blue",bd=10,anchor='w')
    lblTax.grid(row=3,column=2)
    txtTax = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Tax , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtTax.grid(row=3,column=3)

    lblSubtotal = Label(f1, font=( 'aria' ,16, 'bold' ),text="Subtotal",fg="steel blue",bd=10,anchor='w')
    lblSubtotal.grid(row=4,column=2)
    txtSubtotal = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Subtotal , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtSubtotal.grid(row=4,column=3)

    lblTotal = Label(f1, font=( 'aria' ,16, 'bold' ),text="Total",fg="steel blue",bd=10,anchor='w')
    lblTotal.grid(row=5,column=2)
    txtTotal = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Total , bd=6,insertwidth=4,bg="black" ,justify='left')
    txtTotal.grid(row=5,column=3)

    #-------------------------------------buttons-----------------------------------------
    lblTotal = Label(f1,text="----------------------",fg="white")
    lblTotal.grid(row=6,columnspan=3)

    btnTotal=Button(f1,padx=16,pady=8, bd=10 ,fg="black",font=('ariel' ,16,'bold'),width=10, text="TOTAL", bg="black",command=Ref)
    btnTotal.grid(row=7, column=1)

    btnreset=Button(f1,padx=16,pady=8, bd=10 ,fg="black",font=('ariel' ,16,'bold'),width=10, text="RESET", bg="black",command=reset)
    btnreset.grid(row=7, column=2)

    btnexit=Button(f1,padx=16,pady=8, bd=10 ,fg="black",font=('ariel' ,16,'bold'),width=10, text="EXIT", bg="black",command=qexit)
    btnexit.grid(row=7, column=3)









    def price():
        roo = Tk()
        roo.geometry("600x220+0+0")
        roo.title("Price List")
        lblinfo = Label(roo, font=('aria', 15, 'bold'), text="ITEM", fg="black", bd=5)
        lblinfo.grid(row=0, column=0)
        lblinfo = Label(roo, font=('aria', 15,'bold'), text="_____________", fg="white", anchor=W)
        lblinfo.grid(row=0, column=2)
        lblinfo = Label(roo, font=('aria', 15, 'bold'), text="PRICE", fg="black", anchor=W)
        lblinfo.grid(row=0, column=3)
        lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Fries Meal", fg="steel blue", anchor=W)
        lblinfo.grid(row=1, column=0)
        lblinfo = Label(roo, font=('aria', 15, 'bold'), text="25", fg="steel blue", anchor=W)
        lblinfo.grid(row=1, column=3)
        lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Lunch Meal", fg="steel blue", anchor=W)
        lblinfo.grid(row=2, column=0)
        lblinfo = Label(roo, font=('aria', 15, 'bold'), text="40", fg="steel blue", anchor=W)
```

```
btnTotal=Button(f1,padx=16,pady=8, bd=10 ,fg="black",font=('ariel' ,16,'bold'),width=10, text="TOTAL", bg="black",command=Ref)
btnTotal.grid(row=7, column=1)

btnreset=Button(f1,padx=16,pady=8, bd=10 ,fg="black",font=('ariel' ,16,'bold'),width=10, text="RESET", bg="black",command=reset)
btnreset.grid(row=7, column=2)

btnexit=Button(f1,padx=16,pady=8, bd=10 ,fg="black",font=('ariel' ,16,'bold'),width=10, text="EXIT", bg="black",command=qexit)
btnexit.grid(row=7, column=3)









def price():
    roo = Tk()
    roo.geometry("600x220+0+0")
    roo.title("Price List")
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="ITEM", fg="black", bd=5)
    lblinfo.grid(row=0, column=0)
    lblinfo = Label(roo, font=('aria', 15,'bold'), text="_____________", fg="white", anchor=W)
    lblinfo.grid(row=0, column=2)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="PRICE", fg="black", anchor=W)
    lblinfo.grid(row=0, column=3)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Fries Meal", fg="steel blue", anchor=W)
    lblinfo.grid(row=1, column=0)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="25", fg="steel blue", anchor=W)
    lblinfo.grid(row=1, column=3)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Lunch Meal", fg="steel blue", anchor=W)
    lblinfo.grid(row=2, column=0)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="40", fg="steel blue", anchor=W)
    lblinfo.grid(row=2, column=3)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Burger Meal", fg="steel blue", anchor=W)
    lblinfo.grid(row=3, column=0)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="35", fg="steel blue", anchor=W)
    lblinfo.grid(row=3, column=3)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Pizza Meal", fg="steel blue", anchor=W)
    lblinfo.grid(row=4, column=0)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="50", fg="steel blue", anchor=W)
    lblinfo.grid(row=4, column=3)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Cheese Burger", fg="steel blue", anchor=W)
    lblinfo.grid(row=5, column=0)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="30", fg="steel blue", anchor=W)
    lblinfo.grid(row=5, column=3)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="Drinks", fg="steel blue", anchor=W)
    lblinfo.grid(row=6, column=0)
    lblinfo = Label(roo, font=('aria', 15, 'bold'), text="35", fg="steel blue", anchor=W)
    lblinfo.grid(row=6, column=3)

    roo.mainloop()

btnprice=Button(f1,padx=16,pady=8, bd=10 ,fg="black",font=('ariel' ,16,'bold'),width=10, text="PRICE", bg="black",command=price)
btnprice.grid(row=7, column=0)

root.mainloop()
```

OUTPUT: