

# **CAFE MANAGEMENT SYSTEM**

## **MINOR PROJECT REPORT**

By

**S.Yaffin (RA2211032010053)**  
**Anshul Rohilla (RA2211032010064)**  
**Ayush Prajapati (RA2211032010070)**

Under the guidance of

**Dr. Lakshmi Narayanan R**

*In partial fulfilment for the Course*

of

**21CSC203P – ADVANCED PROGRAMMING PRACTICE**

in Networking and Communications



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR**

**NOVEMBER 2023**

# **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

(Under Section 3 of UGC Act, 1956)

## **BONAFIDE CERTIFICATE**

Certified that this minor project report for the course **21CSC203P ADVANCED PROGRAMMING PRACTICE** entitled in "CAFE MANAGEMENT SYSTEM" is the bonafide work of **S.Yaffin (RA2211032010053)**, **Anshul Rohilla (RA2211032010064)** and **Ayush Prajapati (RA2211032010070)** who carried out the work under my supervision.

### **SIGNATURE**

DR. Lakshmi Narayanan R

**Assistant Professor**

**Networking and Communications**

SRM Institute of Science and Technology

Kattankulathur

## ABSTRACT

The "Cafe Management System" project is a comprehensive software application designed to revolutionize and simplify cafe management operations. In a world where efficiency and customer satisfaction are paramount, this system offers a user-friendly and feature-rich solution for cafe owners and employees alike.

The project is built on the foundation of Python, a versatile and widely-used programming language, and leverages the Tkinter graphical user interface (GUI) library for its intuitive and visually appealing interface. Its primary objective is to streamline the process of order management and bill calculation, ultimately enhancing the customer experience and facilitating the day-to-day operations of cafes.

### Features and Functionality:

1. **Order Management:** This application provides a simplified platform for ordering various cafe items, including tea, coffee, sandwiches, cakes, burgers, pizzas, fries, and Pepsi. Users can effortlessly input the quantity of items they desire, making the order process quick and error-free.
2. **Calculator:** In addition to order management, the system includes a built-in calculator. This functionality allows users to perform basic arithmetic operations, offering both convenience and versatility.
3. **Total Bill Calculation:** One of the core functions of the system is its ability to calculate the total bill. Based on the items ordered, service costs, and applicable tax rates, the project provides an accurate and transparent billing experience.

## ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal**, for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman**, for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G , Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems and Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **Dr. Lakshmi Narayanan R, Assistant Professor , Networking and Communications**, for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our HoD of the Department, **Dr K. Annaprani Panaiyappan, Professor, Networking and Communications** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

## **TABLE OF CONTENTS**

<b>CHAPTER NO</b>	<b>CONTENTS</b>	<b>PAGE NO</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Motivation	1
	1.2 Objective	2
	1.3 Problem Statement	3
	1.4 Challenges	4
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>5</b>
<b>3</b>	<b>REQUIREMENT</b>	<b>6</b>
	<b>ANALYSIS</b>	
<b>4</b>	<b>ARCHITECTURE &amp;</b>	<b>7</b>
	<b>DESIGN</b>	
<b>5</b>	<b>IMPLEMENTATION</b>	<b>8</b>
<b>6</b>	<b>EXPERIMENT RESULTS</b>	<b>12</b>
	<b>&amp; ANALYSIS</b>	
<b>7</b>	<b>CONCLUSION</b>	<b>15</b>
<b>8</b>	<b>REFERENCES</b>	<b>16</b>

# **1. INTRODUCTION**

## **1.1 MOTIVATION**

The "Cafe Management System" project is born out of a deep-seated motivation to transform the way cafes operate and to elevate the overall cafe experience for both customers and cafe owners. Cafes hold a unique place in our daily lives, serving as hubs for social interaction, relaxation, and culinary delight. However, managing a bustling cafe efficiently and effectively can be a formidable challenge. This project is motivated by a commitment to overcome these challenges and introduce a system that redefines the cafe management landscape.

Cafe management presents an array of intricate challenges, including the need for precise order processing, transparent billing, and operational efficiency. Inaccuracies in order management can lead to customer dissatisfaction and financial discrepancies. Additionally, manual bill calculations are time-consuming and susceptible to errors. Our motivation is rooted in the aspiration to address these issues comprehensively. We seek to simplify the complexities of cafe management, enabling cafe staff to excel in their roles while ensuring customers enjoy a seamless and gratifying experience.

## **1.2 OBJECTIVE**

The "Cafe Management System" project is designed with a clear set of objectives aimed at enhancing cafe management and customer experience. Its core objectives include streamlining order management to eliminate errors and inefficiencies, automating bill calculations for transparency, and fostering an overall positive customer experience. The system also seeks to boost operational efficiency by reducing time spent on administrative tasks, allowing cafe staff to focus on providing exceptional service. In addition, the project is adaptable, with plans to incorporate advanced features such as user authentication and order history tracking, ensuring its long-term relevance. With a user-friendly interface and a focus on error reduction, the project aspires to revolutionize the cafe management landscape, making cafes more efficient, customer-friendly, and profitable for owners.

### **1.3 PROBLEM STATEMENT**

The cafe management industry encounters various challenges that affect operational efficiency and customer satisfaction. Manual order processing often results in inaccuracies, leading to customer dissatisfaction and financial discrepancies. Billing complexities, including time-consuming calculations and transparency issues, pose administrative burdens. Furthermore, operational inefficiencies stemming from extensive administrative tasks hinder the ability of cafe staff to provide superior customer service. These challenges compromise the customer experience, impeding the cafes' capacity to thrive. The "Cafe Management System" project aims to tackle these issues comprehensively by automating order management, simplifying billing, and enhancing operational efficiency. Its mission is to create a seamless and enjoyable cafe experience for customers while alleviating the administrative load on cafe staff. This project looks ahead, with the potential to evolve and adapt to future industry demands, making it a valuable asset for cafe owners.



## 1.4 CHALLENGES

The development and implementation of the "Cafe Management" mini-project entail several key challenges. First and foremost, crafting a user-friendly interface that simplifies the cafe management process for both staff and customers is of paramount importance. This challenge involves ensuring that the interface is intuitive, easy to navigate, and minimizes the learning curve for users.

Another critical challenge revolves around data handling and security. Protecting sensitive customer orders, billing information, and transaction records from potential breaches or data loss demands robust data security measures. The system must be designed to safeguard this information while providing essential functionality.

Effective error handling is an essential challenge, as the system should gracefully manage errors in order processing, billing, and user interactions to maintain a seamless cafe experience. Achieving compatibility with the diverse hardware and software configurations commonly found in cafes adds complexity to the project, necessitating extensive testing and adaptability.

Providing real-time updates on order status and bill calculations poses technical challenges to ensure that customers and cafe staff receive accurate information promptly. Scalability is also a concern, as the system should cater to cafes of various sizes, from small local establishments to larger franchises.

Furthermore, the project must address the challenge of user adoption by overcoming resistance to change and offering sufficient training and support for cafe staff. Balancing development and maintenance costs with the benefits the system provides to cafe owners is another key challenge, particularly for smaller businesses.

## 2. LITERATURE SURVEY

S.no	Paper Title	Author	Year	Publisher	Keywords	Algorithms/Tools/Techniques Used with drawback
1.	USING TKINTER OF PYTHON TO CREATE GRAPHICAL USER INTERFACE (GUI) FOR SCRIPTS IN LNLS	D. B. Beniz† , A. M. Espindola	2016	www.researchgate.net	Tkinter,Python	Tk GUI toolkit
2.	COLLEGE MANAGEMENT SYSTEM USING PYTHON PROGRAMMING	TATHAGAT A MUKHERJEE ABHISHEK MITRA	2022	Jetir.org	College Management System , Python Programming	Python,tkinter

### **3. REQUIREMENTS ANALYSIS**

#### **3.1 Requirement Analysis**

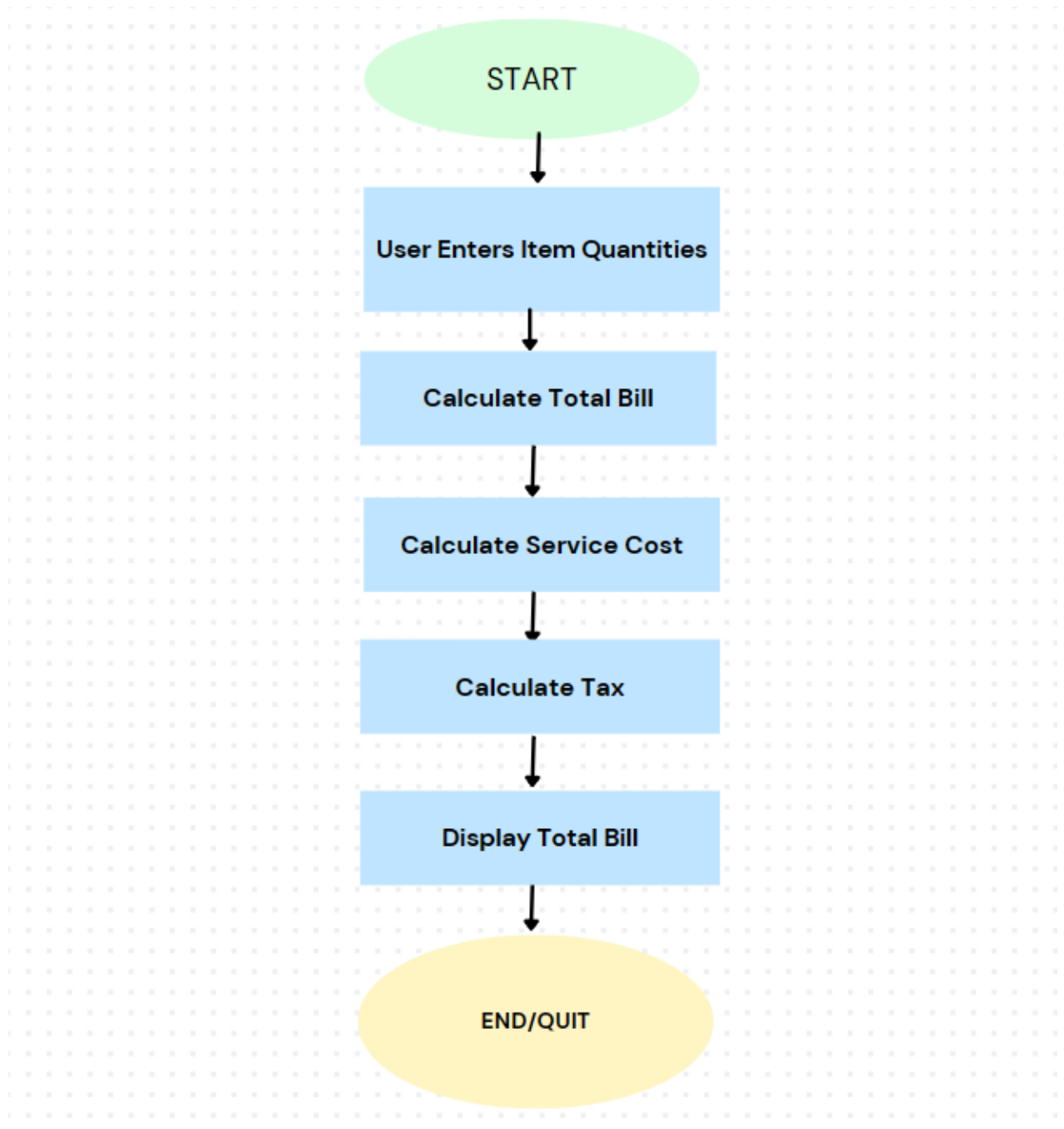
From the given scenario, we draw the following requirements:

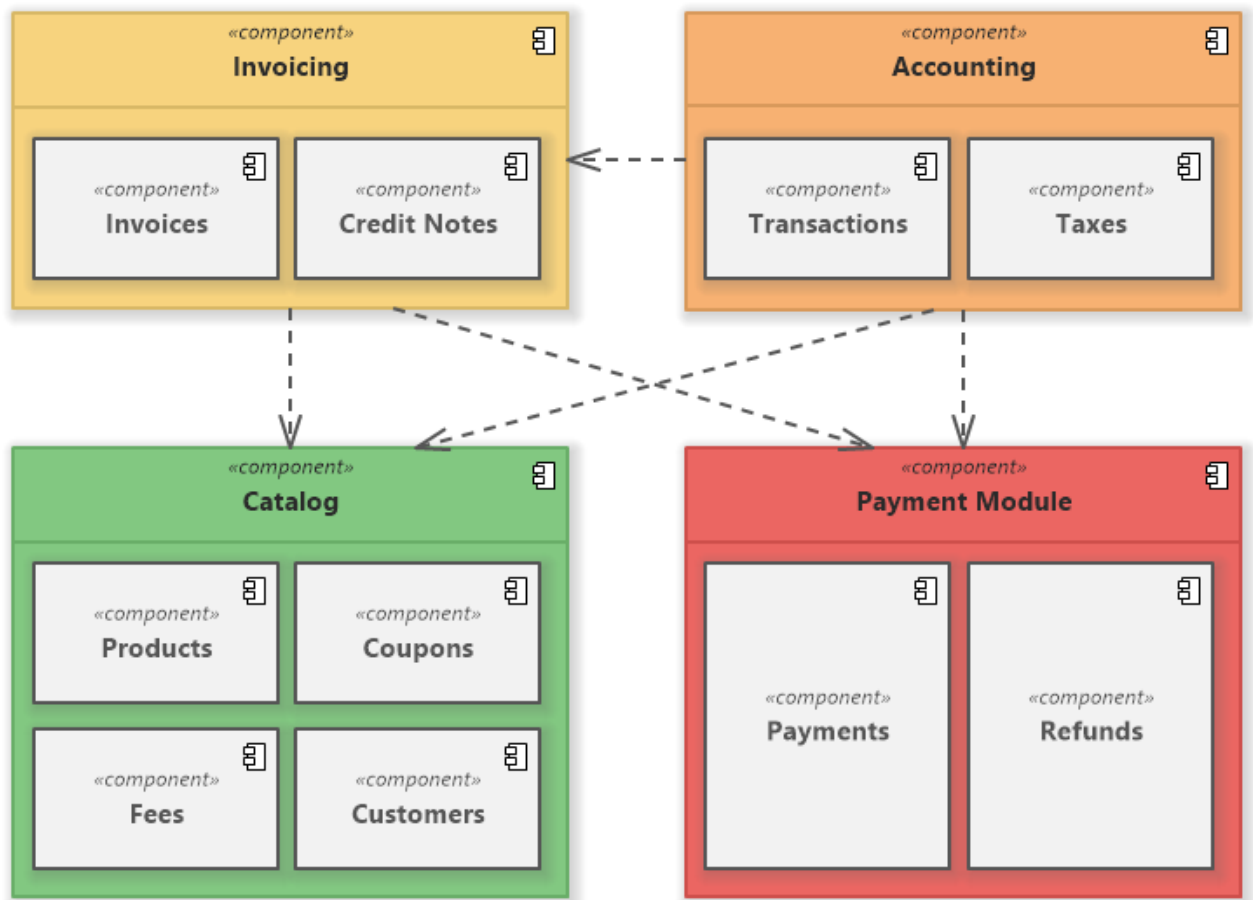
1. **Intuitive Interface:** Develop a user-friendly interface for customers and staff to streamline order placement and payment processes.
2. **Order Management:** Enable efficient order processing, tracking, and updates for cafe staff.
3. **Billing and Payment:** Calculate bills accurately, including item costs, service charges, and taxes, offering various payment options.
4. **Data Security:** Implement robust data security measures to protect customer information and transaction records.
5. **Error Handling:** Include effective error-handling mechanisms to ensure a seamless customer experience.
6. **Compatibility:** Ensure compatibility with different devices and operating systems used in cafes.
7. **Scalability:** Adapt to the needs of cafes of varying sizes without compromising performance.
8. **Reporting and Analytics:** Provide reporting and analytics features for cafe owners to make informed decisions. .

## 4. ARCHITECTURE AND DESIGN

### a. Architecture

The architecture is as follows:





## 5. IMPLEMENTATION

```
from tkinter import *
import tkinter as tk
from datetime import datetime
from PIL import ImageTk, Image
from tkinter import messagebox

class cafe_management():

    # ===== Total Bill Code =====

    def Total_Bill(self):
        self.tea_price = 10
        self.coffee_price = 20
        self.sandwitch_price = 50
        self.cake_price = 100
        self.burger_price = 50
        self.pizza_price = 150
        self.fries_price = 80
        self.pepsi_price = 80

        if self.tea_item.get() != "":
            self.tea_cost = self.tea_price * int(self.tea_item.get())
        else:
            self.tea_cost = 0
        if self.coffee_item.get() != "":
            self.coffee_cost = self.coffee_price * int(self.coffee_item.get())
        else:
            self.coffee_cost = 0
        if self.sandwitch_item.get() != "":
            self.sandwitch_cost = self.sandwitch_price * int(self.sandwitch_item.get())
        else:
            self.sandwitch_cost = 0
        if self.cake_item.get() != "":
            self.cake_cost = self.cake_price * int(self.cake_item.get())
        else:
            self.cake_cost = 0
        if self.burger_item.get() != "":
```

```

else:
    self.cake_cost = 0
if self.burger_item.get() != "":
    self.burger_cost = self.burger_price * int(self.burger_item.get())
else:
    self.burger_cost = 0
if self.pizza_item.get() != "":
    self.pizza_cost = self.pizza_price * int(self.pizza_item.get())
else:
    self.pizza_cost = 0
if self.fries_item.get() != "":
    self.fries_cost = self.fries_price * int(self.fries_item.get())
else:
    self.fries_cost = 0
if self.pepsi_item.get() != "":
    self.pepsi_cost = self.pepsi_price * int(self.pepsi_item.get())
else:
    self.pepsi_cost = 0

self.Total_Bill = self.pepsi_cost + self.fries_cost + self.pizza_cost + self.burger_cost + self.cake_cost + self.sandwich_cost

if self.items_cost != "":
    self.items_cost.delete(0,END)
    self.items_cost.insert(END,self.Total_Bill)
else:
    self.items_cost.insert(END,self.Total_Bill)
if self.service_cost != "":
    self.service_cost.delete(0,END)
    self.service_cost.insert(END,10.0)
else:
    self.service_cost.insert(END,10.0)
if self.sub_cost != "":
    self.sub_cost.delete(0,END)
    self.sub_cost.insert(END,int(self.items_cost.get()) + float(self.service_cost.get()))
else:
    self.sub_cost.insert(END,int(self.items_cost.get()) + float(self.service_cost.get()))
if self.paid_tax != "":
    self.paid_tax.delete(0,END)

```

```

if self.paid_tax != "":
    self.paid_tax.delete(0,END)
    self.paid_tax.insert(END,float(self.sub_cost.get())*8/100)
else:
    self.paid_tax.insert(END,float(self.sub_cost.get())*8/100)

if self.total_bill != "":
    self.total_bill.delete(0,END)
    self.total_bill.insert(END,float(self.sub_cost.get())+float(self.paid_tax.get()))
else:
    self.total_bill.insert(END,float(self.sub_cost.get())+float(self.paid_tax.get()))

```

# ===== Calculator code =====

```

def nine(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","9")
    else:
        self.result.insert("end","9")

```

```

def eight(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","8")
    else:
        self.result.insert("end","8")

```

```

def seven(self):
    if 'error' in self.result.get() or '=' in self.result.get():

```



```
def seven(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","7")
    else:
        self.result.insert("end","7")

def six(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","6")
    else:
        self.result.insert("end","6")

def five(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","5")
    else:
        self.result.insert("end","5")

def four(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","4")
    else:
        self.result.insert("end","4")

def three(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","3")
    else:
        self.result.insert("end","3")

def two(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","2")
```

```
def two(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","2")
    else:
        self.result.insert("end","2")

def one(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","1")
    else:
        self.result.insert("end","1")

def zero(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","0")
    else:
        self.result.insert("end","0")

def plus(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","+")
    else:
        self.result.insert("end","+")

def minus(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","-")
    else:
        self.result.insert("end","-")

def mul(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","*")
```

```

def mul(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","*")
    else:
        self.result.insert("end","*")

def divide(self):
    if 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")
        self.result.insert("end","/")
    else:
        self.result.insert("end","/")

def equal(self):

    if self.result.get() == "":
        self.result.insert("end","error")
    elif self.result.get()[0] == "0" or self.result.get()[0] == "+" or self.result.get()[0] == "*" or self.result.get()[0] == "/":
        self.result.delete(0,"end")
        self.result.insert("end","error")
    elif 'error' in self.result.get() or '=' in self.result.get():
        self.result.delete(0,"end")

    else:
        self.res = self.result.get()
        self.res = eval(self.res)
        self.result.insert("end"," = ")
        self.result.insert("end",self.res)

# ===== Clear Fields =====
def clear(self):
    self.result.delete(0,"end")
def Clear(self):
    self.tea_item.delete(0,"end")

```

```

def Clear(self):
    self.tea_item.delete(0,"end")
    self.coffee_item.delete(0,"end")
    self.sandwitch_item.delete(0,"end")
    self.burger_item.delete(0,"end")
    self.cake_item.delete(0,"end")
    self.fries_item.delete(0,"end")
    self.pizza_item.delete(0,"end")
    self.pepsi_item.delete(0,"end")
    self.items_cost.delete(0,"end")
    self.service_cost.delete(0,"end")
    self.sub_cost.delete(0,"end")
    self.paid_tax.delete(0,"end")
    self.total_bill.delete(0,"end")

# ==== Exit button code =====
def Quit(self):
    self.message = messagebox.askquestion('Exit',"Do you want to exit the application")
    if self.message == "yes":
        self.root.destroy()
    else:
        "return"

#===== end =====

def __init__(self):
    self.root = tk.Tk()
    self.root.geometry('500x300')
    self.root.title("Cafe Management System")
    self.root.maxsize(500,300)
    self.root.minsize(500,300)
    self.root['bg'] = "white"

    self.heading = Label(self.root,text="          Our Proton Cafe",font=('verdana',20,'bold'),fg="#248aa2",bg="white")
    self.heading.place(x=60,y=5)

    self.style1 = Label(self.root,bg="#248aa2",height=1,width=17)

```

```

self.burger = Label(self.frame1,text="Burger",font=('verdana',10,'bold'),bg="white")
self.burger.place(x=3,y=80)
self.burger_item = Entry(self.frame1,width=7,borderwidth=4,relief=SUNKEN,bg="#248aa2")
self.burger_item.place(y=80,x=85)

self.pizza = Label(self.frame1,text="Pizza",font=('verdana',10,'bold'),bg="white")
self.pizza.place(x=3,y=100)
self.pizza_item = Entry(self.frame1,width=7,borderwidth=4,relief=SUNKEN,bg="#248aa2")
self.pizza_item.place(y=100,x=85)

self.fries = Label(self.frame1,text="Fries",font=('verdana',10,'bold'),bg="white")
self.fries.place(x=3,y=120)
self.fries_item = Entry(self.frame1,width=7,borderwidth=4,relief=SUNKEN,bg="#248aa2")
self.fries_item.place(y=120,x=85)

self.pepsi = Label(self.frame1,text="Pepsi",font=('verdana',10,'bold'),bg="white")
self.pepsi.place(x=3,y=140)
self.pepsi_item = Entry(self.frame1,width=7,borderwidth=4,relief=SUNKEN,bg="#248aa2")
self.pepsi_item.place(y=140,x=85)

# ===== Items Bill =====

self.frame2 = LabelFrame(self.root,text="Cafe Items Bills",width=180,height=160,font=('verdana',10,
self.frame2.place(x=180,y=120)

self.item_cost_lb = Label(self.frame2,text="Items Cost",font=('verdana',10,'bold'),bg="white")
self.item_cost_lb.place(x=3,y=1)
self.items_cost = Entry(self.frame2,width=9,borderwidth=4,relief=SUNKEN,bg="#248aa2")
self.items_cost.place(y=1,x=100)

self.service_cost_lb = Label(self.frame2,text="Service Cost",font=('verdana',10,'bold'),bg="white")
self.service_cost_lb.place(x=3,y=20)
self.service_cost = Entry(self.frame2,width=9,borderwidth=4,relief=SUNKEN,bg="#248aa2")
self.service_cost.place(y=20,x=100)

self.sub_cost_lb = Label(self.frame2,text="Sub Cost",font=('verdana',10,'bold'),bg="white")
self.sub_cost_lb.place(x=3,y=40)
self.sub_cost = Entry(self.frame2,width=9,borderwidth=4,relief=SUNKEN,bg="#248aa2")
self.sub_cost.place(y=40,x=100)

```

```

self.sub_cost = Entry(self.frame2,width=9,borderwidth=4,relief=SUNKEN,bg="#248aa2")
self.sub_cost.place(y=40,x=100)

self.paid_tax_lb = Label(self.frame2,text="Paid Tax",font=('verdana',10,'bold'),bg="white")
self.paid_tax_lb.place(x=3,y=80)
self.paid_tax = Entry(self.frame2,width=9,borderwidth=4,relief=SUNKEN,bg="#248aa2")
self.paid_tax.place(y=80,x=100)

self.total_bill_lb = Label(self.frame2,text="Total Bill",font=('verdana',10,'bold'),bg="white")
self.total_bill_lb.place(x=3,y=100)
self.total_bill = Entry(self.frame2,width=9,borderwidth=4,relief=SUNKEN,bg="#248aa2")
self.total_bill.place(y=100,x=100)

# ===== Calculator =====
self.frame3 = LabelFrame(self.root,text="Calculator",font=('verdana',10,'bold'),fg="#248aa2",bg="white",highlightbackground="white",width=135,height=135)
self.frame3.place(x=360,y=90)

self.result = Entry(self.frame3,width=19,relief=SUNKEN,borderwidth=3)
self.result.place(x=2,y=0)

self.nine = Button(self.frame3,text="9",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.nine)
self.nine.place(x=0,y=24)
self.eight = Button(self.frame3,text="8",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.eight)
self.eight.place(x=32,y=24)
self.seven = Button(self.frame3,text="7",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.seven)
self.seven.place(x=64,y=24)
self.plus = Button(self.frame3,text="+",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='white',fg="black",command=self.plus)
self.plus.place(x=96,y=24)

self.six = Button(self.frame3,text="6",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.six)
self.six.place(x=0,y=50)
self.five = Button(self.frame3,text="5",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.five)
self.five.place(x=32,y=50)
self.four = Button(self.frame3,text="4",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.four)
self.four.place(x=64,y=50)
self.minus = Button(self.frame3,text="-",padx=8,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='white',fg="black",command=self.minus)
self.minus.place(x=96,y=50)

self.three = Button(self.frame3,text="3",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.three)
self.three.place(x=0,y=76)
self.two = Button(self.frame3,text="2",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.two)
self.two.place(x=32,y=76)
self.one = Button(self.frame3,text="1",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.one)
self.one.place(x=64,y=76)
self.multiply = Button(self.frame3,text="*",padx=7,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='white',fg="black",command=self.mul)
self.multiply.place(x=96,y=76)

self.zero = Button(self.frame3,text="0",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.zero)
self.zero.place(x=0,y=102)
self.clear = Button(self.frame3,text="C",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.clear)
self.clear.place(x=32,y=102)
self.equal = Button(self.frame3,text="=",padx=6,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.equal)
self.equal.place(x=64,y=102)
self.divide = Button(self.frame3,text="/",padx=7,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='white',fg="black",command=self.divide)
self.divide.place(x=96,y=102)

self.Total_Bills_btn = Button(self.root,text="Total",relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.Total_Bill)
self.Total_Bills_btn.place(x=360,y=245)

self.Clear_btn = Button(self.root,text="Clear",relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",command=self.Clear)
self.Clear_btn.place(x=410,y=245)

self.icon = ImageTk.PhotoImage(Image.open('false.png'))
self.Quit_btn = Button(self.root,image=self.icon,relief=RAISED,borderwidth=2,font=('verdana',10,'bold'),bg='#248aa2',fg="white",padx=5,command=self.Quit)
self.Quit_btn.place(x=463,y=245)

self.root.mainloop()

```


```

if __name__ == '__main__':
    cafe_management()

```

## 6. RESULTS AND DISCUSSION

### i. ORDER INTERFACE



**Cafe Management System**

2023-11-08 21:44:44.833853

**Cafe Items**

Tea	
Coffee	
Sandwich	
Cake	
Burger	
Pizza	
Fries	
Pepsi	

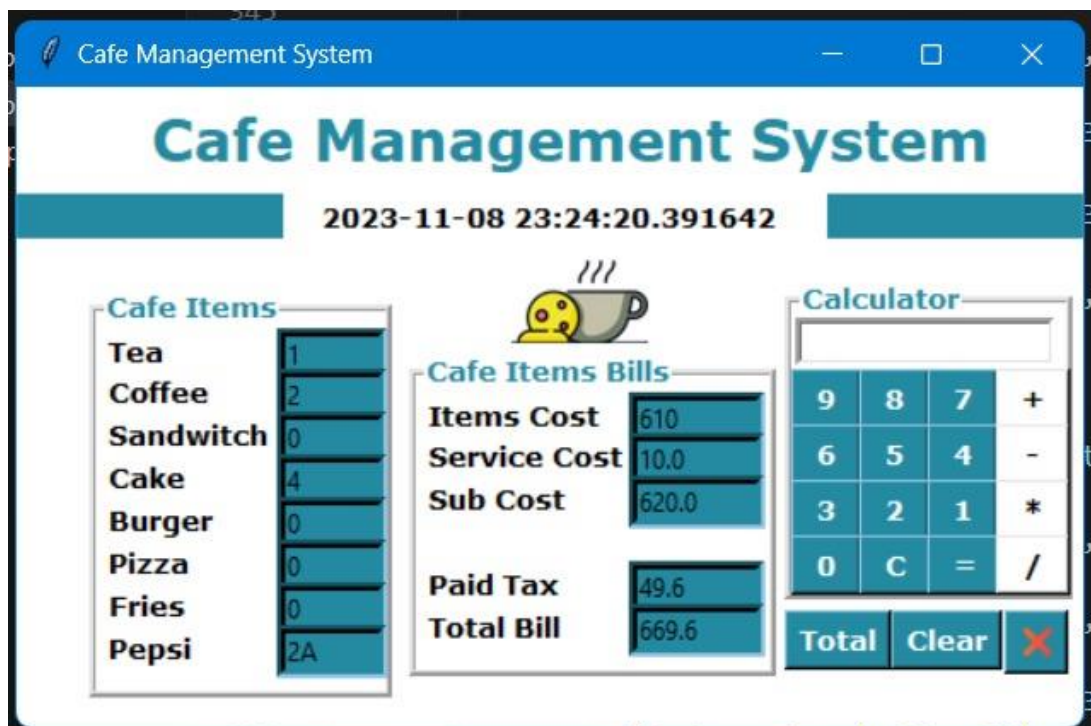
**Cafe Items Bills**

Items Cost	
Service Cost	
Sub Cost	
Paid Tax	
Total Bill	

**Calculator**

9	8	7	+
6	5	4	-
3	2	1	*
0	C	=	/
Total		Clear	X

### b. CALCULATES TOTAL BILL



**Cafe Management System**

2023-11-08 23:24:20.391642

**Cafe Items**

Tea	1
Coffee	2
Sandwich	0
Cake	4
Burger	0
Pizza	0
Fries	0
Pepsi	2A

**Cafe Items Bills**

Items Cost	610
Service Cost	10.0
Sub Cost	620.0
Paid Tax	49.6
Total Bill	669.6

**Calculator**

9	8	7	+
6	5	4	-
3	2	1	*
0	C	=	/
Total		Clear	X

## 7. CONCLUSION

The development of the "Cafe Management System" represents a significant stride in enhancing the efficiency and customer experience in cafe operations. This project has been motivated by the need to streamline order management, improve billing accuracy, and ensure data security while complying with regulatory requirements.

Through a meticulous requirement analysis, this system is poised to address the core needs of cafe owners, staff, and customers. The intuitive user interface, efficient order processing, and versatile payment options cater to the diverse demands of cafes. Robust data security measures and error-handling mechanisms guarantee a secure and error-free cafe experience.

The "Cafe Management System" acknowledges the importance of compatibility with various hardware configurations and scalability to meet the needs of cafes of varying sizes. Reporting and analytics features empower cafe owners with insights to make informed decisions, while user training and support promote user adoption.

With a focus on regulatory compliance and cost-effective implementation, this project aims to provide a solution that not only enhances cafe management but also aligns with industry standards and budgetary constraints. The design principles of future readiness ensure that the system can adapt to evolving industry trends and technological advancements.

In summary, the "Cafe Management System" project aspires to revolutionize the cafe industry by delivering a comprehensive solution that enhances operational efficiency, customer satisfaction, and data security. It is a testament to the power of technology in elevating traditional businesses to meet the demands of the modern world.



## 8. REFERENCES

- <https://wiki.python.org/moin/TkInter>
- <https://www.w3schools.com/python/>
- <https://www.researchgate.net/>
- <https://www.ijeat.org/>
- <https://www.jstor.org/>
- <https://www.researchgate.net/publication>
- <https://papers.ssrn.com/sol3/papers.cfm?/lite/guide>  
<https://docs.python.org/3/>