**TEA SPILLERS | Billing System**

A System Development Documentation

Submitted to the Faculty

of the Department of Information Technology

Cavite State University - Imus Campus

City of Imus, Cavite

In partial fulfillment

of the requirements in the subject,

DCIT 23 Computer Programming 2

Fabian, Meg Angeline V.

Koa, Kristine L.

Murillo, Pamela T.

Palima, Ginna V.

Velasco, Hazel Mae C.

**July 10, 2023**

**System Description**

This GUI application called Tea Spillers is designed to provide a user-friendly interface for customers visiting a coffee shop. With this application, customers can easily enter their details, browse the available products, calculate the total price of their order, and generate a receipt.

**System Features**

Here are the features of Tea Spillers Billing System:

1. Login and Sign-Up Functionality:

- Users can log in using predefined usernames and passwords.

- Users can sign up to create a new account.

2. Menu Section:

- Display names and prices of different coffee items.

- Prices are read-only and cannot be directly edited.

3. Order Section:

- You can click the images of the products to add quantity of different drinks.

- It has minus buttons, trash buttons, and billing area.

4. Receipt Section:

- Display the final bill or order details.

- Supports scrolling for longer receipts.

5. Action Buttons:

- Calculate Total: Calculate the total bill amount based on the entered quantities.

- Receipt: Generate and display the bill or order details in the receipt area.

- Print: Print the receipt.

- Reset: Clear all entered quantities and the receipt area.

- Exit: Close the application or go back to the login screen.

These features collectively provide functionality for logging in, choosing quantities of desserts and drinks, calculating the total bill, generating a receipt, and performing actions such as printing and resetting the order.

**System Information**

The system sets up a GUI application for a coffee shop where customers can enter their details, select products, calculate the total price, and print a receipt. The receipt can be displayed on the screen for the customer to review and, if desired, printed for their convenience.

**System Future Suggestion**

If you have any recommendations or face any problems while using the system, consider offering feedback to the system's creators or support team. Your suggestions can help us enhance the system for all users in the future.

**System Codes**

from tkinter import \*

from tkinter import messagebox

from PIL import Image, ImageTk

from fpdf import FPDF

root = Tk()

root.title('Tea Spillers | Sign in')

root.geometry("951x500+300+200")

root.configure(bg="#fff")

root.resizable(False, False)

img = PhotoImage(file='logongteaspillers.png')

Label(root, image=img, bg='white').place(x=10, y=10)

coffee\_shop = 'Tea Spillers'

shop\_address = 'Dream Villa Imus'

number = '930 870 0299'

message = 'May kape kana, may tsismis kapa. Arat na!'

import datetime

now = datetime.datetime.now()

date\_time = now.strftime("%Y-%m-%d %H:%M:%S")

receipt\_complete = False

def login():

    global selected\_option

    log = Toplevel()

    log.title("Tea Spillers")

    log.geometry("1355x650+100+140")

    log.resizable(width=False, height=False)

    frame = Frame(log, pady=250, padx=250)

    frame.pack()

    ######################################################Header#######################################################

    headerLabel = Label(log, text="Tea Spillers",

                        font=("times new roman", 30, "bold"), bg="RosyBrown", bd=12, relief=GROOVE)

    headerLabel.pack(fill=X)

    #####################################################Customer Details#############################################

    customer\_Details\_Frame = LabelFrame(log, text="Customer Details",

                                        font=("times new roman", 15, "bold"), bg="RosyBrown", bd=12, relief=GROOVE)

    customer\_Details\_Frame.pack(fill=X)

    cstmr\_name = Label(customer\_Details\_Frame, text="Nickname", font=("times new roman", 15, "bold"), bg="RosyBrown")

    cstmr\_name.grid(row=0, column=0, padx=20, pady=5)

    cstmr = "Enter a nickname"

    cstmr\_entry = Entry(customer\_Details\_Frame, font=('arial', 15), bd=7, width=18)

    cstmr\_entry.insert(0, cstmr)

    cstmr\_entry.grid(row=0, column=1, padx=8)

    cstmr\_entry.bind('<Button-1>', lambda event: delete\_zero(event, cstmr\_entry))

    phone\_num = Label(customer\_Details\_Frame, text="Phone No.", font=('times new roman', 15, 'bold'),

                      bg='RosyBrown')

    phone\_num.grid(row=0, column=2, padx=20, pady=5)

    phone = "Enter a number"

    phone\_num\_entry = Entry(customer\_Details\_Frame, font=('arial', 15), bd=7, width=18)

    phone\_num\_entry.insert(0, phone)

    phone\_num\_entry.grid(row=0, column=3, padx=8)

    phone\_num\_entry.bind('<Button-1>', lambda event: delete\_zero(event, phone\_num\_entry))

    bill\_num = Label(customer\_Details\_Frame, text="Status Type", font=('times new roman', 15, 'bold'),

                     bg='RosyBrown')

    bill\_num.grid(row=0, column=5, padx=20, pady=5)

    selected\_option = StringVar(customer\_Details\_Frame)

    # Create a list of options

    options = ["Student", "Regular", "N/A"]

    def check\_option(\*args):

        selected = selected\_option.get()

        if selected not in options:

            messagebox.showerror("Invalid User Type", "Please select a valid User Type.")

            return

    # Set the default selected option

    selected\_option.trace('w', check\_option)

    # Create the OptionMenu widget

    option\_menu = OptionMenu(customer\_Details\_Frame, selected\_option, \*options)

    option\_menu.config(width=25)

    option\_menu.grid(row=0, column=6)

    # Product Details

    prod\_container = Frame(log)

    prod\_container.pack()

    product\_frame = LabelFrame(prod\_container, text="Menu",

                               font=("times new roman", 18, "bold"), bd=8, relief=GROOVE, bg='RosyBrown', padx=25)

    product\_frame.grid(row=0,column=0, rowspan=1)

    def delete\_zero(event, entry):

        entry.delete(0, 'end')

    origIcedCoffeePrice = 120.00

    origMatchaPrice = 145.00

    origAmericanoPrice = 160.00

    origCaffucinoPrice = 150.00

    cinnamonPrice = StringVar()

    churrosPrice = StringVar()

    muffinsPrice = StringVar()

    chocolatepiePrice = StringVar()

    cinnamonPrice.set(origIcedCoffeePrice)

    churrosPrice.set(origMatchaPrice)

    muffinsPrice.set(origAmericanoPrice)

    chocolatepiePrice.set(origCaffucinoPrice)

    #######################################################################RESET###########################################

    def reset():

        # Clear other fields and variables

        cstmr\_entry.delete(0, END)

        matchaPrice.delete(0, END)

        textArea.config(state='normal')

        textArea.delete(1.0, END)

        textArea.config(state='disabled')

        americanoPrice.delete(0, END)

        caffucinoPrice.delete(0, END)

        cstmr\_entry.insert(0, cstmr)

        ice.set(0)

        matcha.set(0)

        americano0.set(0)

        caffucino0.set(0)

        global receipt\_complete

        receipt\_complete = False

        receiptButton.configure(state=DISABLED)

        # Reset the add\_to\_cart() functionality

        global total\_price, billing\_complete

        total\_price = 0

        billing\_complete = False

        receipt\_text\_widget.config(state='normal')

        receipt\_text\_widget.delete('1.0', END)

        receipt\_text\_widget.config(state='disabled')

        CashAmt.place\_forget()

        paymentBtn.place\_forget()

        CashAmt.config(text="Cash:")

        textArea.config(state='normal')

        textArea.delete(1.0, END)

        textArea.config(state='disabled')

    ##############################################################Print Page#########################################

    def print():

        global receipt\_complete

        global cash\_amount

        global payment\_entry, selected\_option

        # Check if the receipt is done

        if not receipt\_complete:

            messagebox.showerror("Error", "Please complete the receipt first.")

            return

        p = Toplevel()

        p.title("Tea Spillers | Your Receipt")

        p.geometry("1001x550+300+170")

        p.resizable(width=False, height=False)

        p.configure(bg="#fff")

        img = PhotoImage(file='logongteaspillers.png')

        Label(p, image=img, border=0, bg='white').place(x=5, y=15)

        p\_frame = Frame(p, width=350, height=390, bg='white')

        p\_frame.place(x=550, y=10)

        receiptHeader = Label(p\_frame, text='Please check your folder for the updated receipt copy.', bg='white', fg="salmon3", font=('Helvetica', 12, 'bold'), wraplength=340)

        receiptHeader.place(x=130, y=10)

        prod\_container = Frame(p, bg="RosyBrown")

        prod\_container.place(x=540, y=70)

        receiptFrame = Frame(prod\_container, bd=8, relief=GROOVE, bg='RosyBrown')

        receiptFrame.grid(row=0, column=5, padx=10)

        receiptArea = Label(receiptFrame, text="Bill Area", font=('times new roman', 15, 'bold'), bd=8, relief=GROOVE,

                            bg='RosyBrown')

        receiptArea.pack(fill=X)

        scrollbar = Scrollbar(receiptFrame, orient=VERTICAL)

        scrollbar.pack(side=RIGHT, fill=Y)

        textArea = Text(receiptFrame, width=50, height=25, yscrollcommand=scrollbar.set)

        textArea.pack()

        scrollbar.config(command=textArea.yview)

        totalFrame = LabelFrame(root, font=("times new roman", 15, "bold"), bg="RosyBrown", bd=15, relief=GROOVE)

        totalFrame.pack(fill=X)

        buyer\_name = cstmr\_entry.get()

        phone\_numm = phone\_num\_entry.get()

        textArea.config(state='normal')

        textArea.delete(1.0, END)

        receipt\_text = '\t\t{}\n\t\t{}\n\t\t{}\n'.format(coffee\_shop.title(), shop\_address.title(), number.title())

        textArea.insert(END, receipt\_text)

        receipt\_text1 = "-" \* 50 + "\n"

        textArea.insert(END, receipt\_text1)

        receipt\_text2 = f"{date\_time[0:10]}\t\t\t\t{date\_time[10:]}\n"

        textArea.insert(END, receipt\_text2)

        if phone\_numm.strip():

            receipt\_phone = f"Customer no. {phone\_numm}\n"

            textArea.insert(END, receipt\_phone)

        else:

            receipt\_phone1 = "Customer no.\n"

            textArea.insert(END, receipt\_phone1)

        receipt\_text3 = "-" \* 50 + "\n"

        textArea.insert(END, receipt\_text3)

        receipt\_text4 = '{}\n'.format(message.title())

        textArea.insert(END, receipt\_text4)

        receipt\_text5 = "-" \* 50 + "\n"

        textArea.insert(END, receipt\_text5)

        receipt\_text6 = "CAFE AND RESTAURANT\n\n"

        textArea.insert(END, receipt\_text6)

        receipt\_text7 = 'Quantity\t\tProduct Name\t\tProduct Price\n'

        textArea.insert(END, receipt\_text7)

        # Retrieve the quantities from Entry widgets

        iced\_coffee\_quantity = int(ice.get())

        matcha\_quantity = int(matcha.get())

        americano\_quantity = int(americano0.get())

        caffucino\_quantity = int(caffucino0.get())

        # Calculate the total price for each product

        iced\_coffee\_price = iced\_coffee\_quantity \* origIcedCoffeePrice

        matcha\_price = matcha\_quantity \* origMatchaPrice

        americano\_price = americano\_quantity \* origAmericanoPrice

        caffucino\_price = caffucino\_quantity \* origCaffucinoPrice

        # Include the quantities in the receipt if greater than zero

        if iced\_coffee\_quantity > 0:

            receipt\_text8 = f'{iced\_coffee\_quantity}\t\tIced Coffee\t\t{iced\_coffee\_price}\n'

            textArea.insert(END, receipt\_text8)

        if matcha\_quantity > 0:

            receipt\_text9 = f'{matcha\_quantity}\t\tMatcha\t\t{matcha\_price}\n'

            textArea.insert(END, receipt\_text9)

        if americano\_quantity > 0:

            receipt\_text10 = f'{americano\_quantity}\t\tAmericano\t\t{americano\_price}\n'

            textArea.insert(END, receipt\_text10)

        if caffucino\_quantity > 0:

            receipt\_text11 = f'{caffucino\_quantity}\t\tCaffucino\t\t{caffucino\_price}\n'

            textArea.insert(END, receipt\_text11)

        receipt\_text21 = "-" \* 50 + "\n"

        textArea.insert(END, receipt\_text21)

        # Define the original prices as a dictionary

        original\_prices = {

            'Iced Coffee': 120.00,

            'Matcha': 145.00,

            'Americano': 160.00,

            'Caffucino': 150.00

        }

        # Convert the quantity values to integers

        try:

            iced\_coffee\_quantity = int(iced\_coffee\_quantity)

            matcha\_quantity = int(matcha\_quantity)

            americano\_quantity = int(americano\_quantity)

            caffucino\_quantity = int(caffucino\_quantity)

        except ValueError:

            messagebox.showerror("Valid Quantity Error", "Please enter a valid quantity for each drink.")

            return

        # Calculate the subtotals

        iced\_coffee\_subtotal = iced\_coffee\_quantity \* original\_prices['Iced Coffee']

        matcha\_subtotal = matcha\_quantity \* original\_prices['Matcha']

        americano\_subtotal = americano\_quantity \* original\_prices['Americano']

        caffucino\_subtotal = caffucino\_quantity \* original\_prices['Caffucino']

        # Calculate the total price

        total\_price = iced\_coffee\_subtotal + matcha\_subtotal + americano\_subtotal + caffucino\_subtotal

        # Calculate the discount based on the user type

        user\_type = selected\_option.get()

        discount = 0

        if user\_type == "Student":

            discount = 0.05  # 5% discount for students

        elif user\_type == "Regular":

            discount = 0.1  # 10% discount for regular users

    # Check if an option is selected

        if not selected\_option.get():

            textArea.delete(1.0, END)  # Clear the textArea

            messagebox.showerror("Option Not Selected", "Please select an option.")

            return

        # Apply the discount to the total price

        discounted\_price = total\_price - (total\_price \* discount)

        receipt\_text12 = f'Subtotal:\t\t\t\t\t₱{total\_price:.2f}\n'

        textArea.insert(END, receipt\_text12)

        receipt\_text13 = f'Discount: \t\t\t\t\t{discount \* 100}%\n'

        textArea.insert(END, receipt\_text13)

        receipt\_text14 = f'Discounted Price: \t\t\t\t\t₱{discounted\_price:.2f}\n'

        textArea.insert(END, receipt\_text14)

        receipt\_text15 = ""

        payment\_method = selected\_option1.get()

        cash\_amount = 0.0  # Default value

        receiptButton.config(state=DISABLED)

        if payment\_method == "Cash" and payment\_entry:

            cash\_amount = payment\_entry.get()

            try:

                cash\_amount = float(cash\_amount)

            except ValueError:

                messagebox.showerror("Valid Cash Amount Error", "Please enter a valid cash amount.")

                return

            receipt\_text15 = f'Cash:\t\t\t\t\t₱{cash\_amount:.2f}\n'

        receipt\_text15 += f"Payment Method: \t\t\t\t\t {payment\_method}\n"

        textArea.insert(END, receipt\_text15)

        receipt\_text16 = ""

        if payment\_method == "Cash" and cash\_amount >= discounted\_price:

            change = cash\_amount - discounted\_price

            receipt\_text16 = f'Change:\t\t\t\t\t₱{change:.2f}\n'

            textArea.insert(END, receipt\_text16)

        receipt\_text17 = f'Total:\t\t\t\t\t₱{discounted\_price:.2f}\n'

        textArea.insert(END, receipt\_text17)

        receipt\_text18 = "-" \* 50 + "\n"

        textArea.insert(END, receipt\_text18)

        if buyer\_name != cstmr:

            ending\_message = f"Thank you for buying {buyer\_name}!"

        else:

            ending\_message = f"Thank you for your purchase!"

        ending\_message\_1 = " Find out about Tea Spillers at teaspillers.com!"

        ending\_message\_text = '{}\t\t\t{}'.format(ending\_message\_1, ending\_message)

        receipt\_text19 = f"{ending\_message\_text}\n"

        textArea.insert(END, receipt\_text19)

        receipt\_text20 = "=" \* 50 + "\n"

        textArea.insert(END, receipt\_text20)

        textArea.config(state='disabled')

        receipt\_complete = True

        #Generating PDF

        pdf = FPDF()

        pdf.add\_page()

        pdf.set\_font("Arial", size=12)

        txt = (

            receipt\_text +

            receipt\_text1 +

            receipt\_text2 +

            receipt\_text3 +

            receipt\_phone +

            receipt\_text4 +

            receipt\_text5 +

            receipt\_text6 +

            receipt\_text7 +

            receipt\_text8 +

            receipt\_text9 +

            receipt\_text10 +

            receipt\_text11 +

            receipt\_text12 +

            receipt\_text13 +

            receipt\_text14 +

            receipt\_text15 +

            receipt\_text16 +

            receipt\_text18 +

            receipt\_text19 +

            receipt\_text20 +

            receipt\_text21

            )

        txt = textArea.get("1.0", "end")

        txt = txt.encode("latin-1", "replace").decode("latin-1")

        pdf.multi\_cell(0,10, txt=txt)

        pdf.output("receipt.pdf")

        p.mainloop()

    ###################################################################Receipt Page

    def receipt():

        check\_option()

        global receipt\_complete, billing\_complete, selected\_option1

        buyer\_name = cstmr\_entry.get()

        phone\_numm = phone\_num\_entry.get()

        textArea.config(state='normal')

        textArea.delete(1.0, END)

        receipt\_text = '\t\t{}\n\t\t{}\n\t\t{}\n'.format(coffee\_shop.title(), shop\_address.title(), number.title())

        textArea.insert(END, receipt\_text)

        receipt\_text1 = "-" \* 50 + "\n"

        textArea.insert(END, receipt\_text1)

        receipt\_text2 = f"{date\_time[0:10]}\t\t\t\t{date\_time[10:]}\n"

        textArea.insert(END, receipt\_text2)

        if phone\_numm.strip():

            receipt\_phone = f"Customer no. {phone\_numm}\n"

            textArea.insert(END, receipt\_phone)

        else:

            receipt\_phone1 = "Customer no.\n"

            textArea.insert(END, receipt\_phone1)

        receipt\_text3 = "-" \* 50 + "\n"

        textArea.insert(END, receipt\_text3)

        receipt\_text4 = '{}\n'.format(message.title())

        textArea.insert(END, receipt\_text4)

        receipt\_text5 = "-" \* 50 + "\n"

        textArea.insert(END, receipt\_text5)

        receipt\_text6 = "CAFE AND RESTAURANT\n\n"

        textArea.insert(END, receipt\_text6)

        receipt\_text7 = 'Quantity\t\tProduct Name\t\tProduct Price\n'

        textArea.insert(END, receipt\_text7)

        # Retrieve the quantities from Entry widgets

        iced\_coffee\_quantity = int(ice.get())

        matcha\_quantity = int(matcha.get())

        americano\_quantity = int(americano0.get())

        caffucino\_quantity = int(caffucino0.get())

        # Calculate the total price for each product

        iced\_coffee\_price = iced\_coffee\_quantity \* origIcedCoffeePrice

        matcha\_price = matcha\_quantity \* origMatchaPrice

        americano\_price = americano\_quantity \* origAmericanoPrice

        caffucino\_price = caffucino\_quantity \* origCaffucinoPrice

        # Include the quantities in the receipt if greater than zero

        if iced\_coffee\_quantity > 0:

            receipt\_text8 = f'{iced\_coffee\_quantity}\t\tIced Coffee\t\t{iced\_coffee\_price}\n'

            textArea.insert(END, receipt\_text8)

        if matcha\_quantity > 0:

            receipt\_text9 = f'{matcha\_quantity}\t\tMatcha\t\t{matcha\_price}\n'

            textArea.insert(END, receipt\_text9)

        if americano\_quantity > 0:

            receipt\_text10 = f'{americano\_quantity}\t\tAmericano\t\t{americano\_price}\n'

            textArea.insert(END, receipt\_text10)

        if caffucino\_quantity > 0:

            receipt\_text11 = f'{caffucino\_quantity}\t\tCaffucino\t\t{caffucino\_price}\n'

            textArea.insert(END, receipt\_text11)

        receipt\_text18 = "-" \* 50 + "\n"

        textArea.insert(END, receipt\_text18)

        # Define the original prices as a dictionary

        original\_prices = {

            'Iced Coffee': 120.00,

            'Matcha': 145.00,

            'Americano': 160.00,

            'Caffucino': 150.00

        }

        # Convert the quantity values to integers

        try:

            iced\_coffee\_quantity = int(iced\_coffee\_quantity)

            matcha\_quantity = int(matcha\_quantity)

            americano\_quantity = int(americano\_quantity)

            caffucino\_quantity = int(caffucino\_quantity)

        except ValueError:

            messagebox.showerror("Valid Quantity Error", "Please enter a valid quantity for each drink.")

            return

        # Calculate the subtotals

        iced\_coffee\_subtotal = iced\_coffee\_quantity \* original\_prices['Iced Coffee']

        matcha\_subtotal = matcha\_quantity \* original\_prices['Matcha']

        americano\_subtotal = americano\_quantity \* original\_prices['Americano']

        caffucino\_subtotal = caffucino\_quantity \* original\_prices['Caffucino']

        # Calculate the total price

        total\_price = iced\_coffee\_subtotal + matcha\_subtotal + americano\_subtotal + caffucino\_subtotal

        # Calculate the discount based on the user type

        user\_type = selected\_option.get()

        discount = 0

        if user\_type == "Student":

            discount = 0.05  # 5% discount for students

        elif user\_type == "Regular":

            discount = 0.1  # 10% discount for regular users

        if not selected\_option.get():

            textArea.delete(1.0, END)  # Clear the textArea

            messagebox.showerror("Option Not Selected", "Please select an option.")

            return

        # Apply the discount to the total price

        discounted\_price = total\_price - (total\_price \* discount)

        receipt\_text12 = f'Subtotal:\t\t\t\t\t₱{total\_price:.2f}\n'

        textArea.insert(END, receipt\_text12)

        receipt\_text13 = f'Discount: \t\t\t\t\t{discount \* 100}%\n'

        textArea.insert(END, receipt\_text13)

        receipt\_text14 = f'Discounted Price: \t\t\t\t\t₱{discounted\_price:.2f}\n'

        textArea.insert(END, receipt\_text14)

        receipt\_text15 = ""

        payment\_method = selected\_option1.get()

        cash\_amount = 0.0  # Default value

        if payment\_method == "Cash" and payment\_entry:

            cash\_amount = payment\_entry.get()

            try:

                cash\_amount = float(cash\_amount)

            except ValueError:

                messagebox.showerror("Valid Cash Amount Error", "Please enter a valid cash amount.")

                receiptButton.configure(state=NORMAL)

                return

            receipt\_text15 = f'Cash:\t\t\t\t\t₱{cash\_amount:.2f}\n'

        receipt\_text15 += f"Payment Method: \t\t\t\t\t{payment\_method}\n"

        textArea.insert(END, receipt\_text15)

        if payment\_method == "Cash" and cash\_amount >= discounted\_price:

            change = cash\_amount - discounted\_price

            receipt\_text16 = f'Change:\t\t\t\t\t₱{change:.2f}\n'

            textArea.insert(END, receipt\_text16)

        receipt\_text17 = f'Total:\t\t\t\t\t₱{discounted\_price:.2f}\n'

        textArea.insert(END, receipt\_text17)

        receipt\_text18 = "-" \* 50 + "\n"

        textArea.insert(END, receipt\_text18)

        if buyer\_name != cstmr:

            ending\_message = f"Thank you for buying {buyer\_name}!"

        else:

            ending\_message = f"Thank you for your purchase!"

        ending\_message\_1 = " Find out about Tea Spillers at teaspillers.com!"

        ending\_message\_text = '{}\t\t\t{}'.format(ending\_message\_1, ending\_message)

        receipt\_text19 = f"{ending\_message\_text}\n"

        textArea.insert(END, receipt\_text19)

        receipt\_text20 = "=" \* 50 + "\n"

        textArea.insert(END, receipt\_text20)

        textArea.config(state='disabled')

        receipt\_complete = True

    # Desserts

    quantity1 = 0

    quantity2 = 0

    quantity3 = 0

    quantity4 = 0

    ice = StringVar()

    ice.set(str(quantity1))

    matcha = StringVar()

    matcha.set(str(quantity2))

    americano0 = StringVar()

    americano0.set(str(quantity3))

    caffucino0 = StringVar()

    caffucino0.set(str(quantity4))

      # Set an initial value for the StringVar

    def increment\_quantity(quantity\_var):

        current\_quantity = int(quantity\_var.get())

        new\_quantity = current\_quantity + 1

        quantity\_var.set(str(new\_quantity))

    def decrement\_quantity(quantity\_var1):

        current\_quantity = int(quantity\_var1.get())

        if current\_quantity > 0:

            new\_quantity = current\_quantity - 1

            quantity\_var1.set(str(new\_quantity))

    def reset\_quantities():

        global quantity1, quantity2, quantity3, quantity4

        global option\_label, payment\_entry

        quantity1 = 0

        ice.set(str(quantity1))

        quantity2 = 0

        matcha.set(str(quantity2))

        quantity3 = 0

        americano0.set(str(quantity3))

        quantity4 = 0

        caffucino0.set(str(quantity4))

    # Clear the receipt text

        receipt\_text\_widget.config(state="normal")  # Set the state to normal to enable modifications

        receipt\_text\_widget.delete("1.0", END)  # Clear the text area

        receipt\_text\_widget.config(state="disabled")  # Set the state to disabled to make it read-only

        paymentBtn.place\_forget()

        CashAmt.place\_forget()

        CashAmt.config(text="Cash:")

    def create\_options():

        global payment\_entry, cash, billing\_complete

        cash = Toplevel()

        cash.title("Tea Spillers | Payment Method")

        cash.geometry("900x500+300+200")

        cash.configure(bg="#fff")

        cash.resizable(width=False, height=False)

        img\_path = 'logongteaspillers.png'

        image = Image.open(img\_path)

        imguli = ImageTk.PhotoImage(image)

        label = Label(cash, image=imguli, bg='white')

        label.image = imguli  # Save a reference to the image to prevent it from being garbage collected

        label.place(x=10, y=10)

        receiptButton.config(state=DISABLED)

        cash\_frame = Frame(cash, width=350, height=420, bg='white')

        cash\_frame.place(x=550, y=50)

        payHeader = Label(cash\_frame, text='Payment Method', fg="salmon3", bg='white', font=('Helvetica', 23, 'bold'))

        payHeader.place(x=70, y=20)

        totalis = Label(cash\_frame, text=f"Your Total Price is : {total\_price}", bg='white').place(x=40,y=120)

        global option\_label, payment\_entry, selected\_option1

        selected\_option1 = StringVar(cash\_frame)

    # Create a list of options

        option\_label = Label(cash\_frame, text="Type of Payment:",bg="white").place(x=40,y=155)

        options = ["Cash", "E-Wallet(GCash)"]

    # Set the default selected option

        selected\_option1.set(options[0])

        def option\_selected(\*args):

            global payment\_entry, gcash\_entry

            selected = selected\_option1.get()

            if selected == "Cash":

                textt = "Enter payment Amount..."

                payment\_entry = Entry(cash\_frame, font=("Arial", 12), width=25)

                payment\_entry.insert(0, textt)

                payment\_entry.place(x=50, y=220)

                doneBtn = Button(cash\_frame, text="Done", bg="salmon3", command=done\_clicked).place(x=70, y=250)

                payment\_entry.bind('<Button-1>', lambda event: delete\_zero(event, payment\_entry))

            elif selected == "E-Wallet(GCash)":

                textt = "Enter 11-digit number..."

                gcash\_entry = Entry(cash\_frame, font=("Arial", 12), width=25)  # Use gcash\_entry instead of payment\_entry

                gcash\_entry.insert(0, textt)

                gcash\_entry.place(x=50, y=220)

                doneBtn = Button(cash\_frame, text="Done", bg="salmon3", command=done\_clicked).place(x=70, y=250)

                def delete\_text(event):

                    gcash\_entry.delete(0, END)

                gcash\_entry.bind('<Button-1>', delete\_text)

                gcash\_entry.bind('<FocusOut>', validate\_gcash\_number)

        def validate\_gcash\_number(event):

            gcash\_number = gcash\_entry.get()  # Use gcash\_entry instead of payment\_entry

            if len(gcash\_number) != 11 or not gcash\_number.isdigit():

                messagebox.showerror("Invalid GCash Number", "Please enter a valid 11-digit GCash number.")

                gcash\_entry.delete(0, END)

                gcash\_entry.insert(0, "Enter 11-digit number...")

                cash\_frame.focus\_force()

                CashAmt.config(text="")  # Clear the CashAmt label

            else:

                CashAmt.config(text="Paid by GCash")  # Update the CashAmt label

        def done\_clicked():

            receiptButton.config(state=DISABLED)

            global cash\_amt, total\_price, payment\_entry

            if selected\_option1.get() == "Cash":

                cash\_amt = payment\_entry.get()

            elif selected\_option1.get() == "E-Wallet(GCash)":

                cash\_amt = gcash\_entry.get()

            CashAmt.config(text=f"Cash: {cash\_amt}")

            if CashAmt.cget("text") == "Cash: ":  # Check if CashAmt label text is empty

                receiptButton.config(state=DISABLED)  # Disable the receipt button

            else:

                try:

                    cash\_amt = float(cash\_amt)

                    if cash\_amt >= total\_price:

                        # Payment is sufficient, proceed with further actions

                        cash.withdraw()  # Hide the cash window

                    else:

                        # Insufficient payment, display error message

                        messagebox.showerror("Insufficient Payment", "The payment amount is insufficient.")

                        if selected\_option1.get() == "Cash":

                            payment\_entry.delete(0, 'end')  # Reset the payment entry field

                            payment\_entry.insert(0, "Enter payment amount...")

                        elif selected\_option1.get() == "E-Wallet(GCash)":

                            gcash\_entry.delete(0, 'end')  # Reset the GCash entry field

                            gcash\_entry.insert(0, "Enter 11-digit number...")

                        CashAmt.config(text="")  # Clear the CashAmt label

                except ValueError:

                    # Invalid payment amount, display error message

                    messagebox.showerror("Invalid Payment", "Please enter a valid payment amount.")

                    if selected\_option1.get() == "Cash":

                        payment\_entry.delete(0, 'end')  # Reset the payment entry field

                        payment\_entry.insert(0, "Enter payment amount...")

                    elif selected\_option1.get() == "E-Wallet(GCash)":

                        gcash\_entry.delete(0, 'end')  # Reset the GCash entry field

                        gcash\_entry.insert(0, "Enter 11-digit number...")

                    CashAmt.config(text="")  # Clear the CashAmt label

            receiptButton.config(state=NORMAL)  # Enable the receipt button

    # Bind the option\_selected function to the OptionMenu widget

        selected\_option1.trace("w", option\_selected)

    # Create the OptionMenu widget

        option\_menu = OptionMenu(cash\_frame, selected\_option1, \*options)

        option\_menu.config(width=20)

    # Place the OptionMenu widget below the receipt text widget

        option\_menu.place(x=150,y=150)

        billing\_complete = False

        #TotalFrame

    coffee\_frame = Frame(prod\_container)

    coffee\_frame.grid(row=0, column=1, sticky="ns")

    drinks\_frame = LabelFrame(coffee\_frame, text="Billing Area", font=("times new roman", 18, "bold"), bd=8,

                              relief=GROOVE, bg='RosyBrown', padx=0, width=280, height=200)

    drinks\_frame.grid()

    receipt\_text7 = 'Quantity\tProduct Name\t\tProduct Price\n'

    receipt\_text\_widget = Text(drinks\_frame, font=("Times New Roman", 12), bg="RosyBrown", width=38, height=18)

    receipt\_text\_widget.insert("1.0", receipt\_text7)

    paymentBtn = Button(drinks\_frame, text="Click here to select payment method", bg="salmon3",command=create\_options)

    CashAmt = Label(drinks\_frame, text="Cash:",font=("Times New Roman", 12), bg="RosyBrown")

    receipt\_text\_widget.grid(padx=5, pady=10)

    original\_prices = {

        'Iced Coffee': 120.00,

        'Matcha': 145.00,

        'Americano': 160.00,

        'Caffucino': 150.00

    }

    def add\_to\_cart():

        global total\_price, billing\_complete

        quantities = [int(ice.get()), int(matcha.get()), int(americano0.get()), int(caffucino0.get())]

        products = ["Iced Coffee", "Matcha", "Americano", "Caffucino"]

        receipt\_text = "Quantity\tProduct Name\t\tProduct Price\n"

        global total\_price

        total\_price = 0

        for i in range(len(quantities)):

            if quantities[i] > 0:

                product\_name = products[i]

                quantity = quantities[i]

                product\_price = original\_prices[product\_name]

                line\_total = product\_price \* quantity

                receipt\_text += f"{quantity}\t{product\_name}\t\t{line\_total:.2f}\n"

                total\_price += line\_total

        receipt\_text += f"\nTotal Price: {total\_price:.2f}"

        CashAmt.place(x=8,y=150)

        paymentBtn.place(x=70,y=180)

        billing\_complete = TRUE

        receipt\_text\_widget.config(state="normal")

        receipt\_text\_widget.delete("1.0", END)

        receipt\_text\_widget.insert(END, receipt\_text)

        receipt\_text\_widget.config(state="disabled")

#Menu

    cinnamonLabel = Button(product\_frame,

                          bg="beige", borderwidth=10,command=lambda: increment\_quantity(ice))

    icedImg = PhotoImage(file="match.png")

    res = icedImg.subsample(2)

    cinnamonLabel.config(image=res, compound=LEFT)

    cinnamonLabel.config(width=70,height=50)

    cinnamonLabel.grid(row=1, column=0, sticky="w", padx=20)

    icedLabel = Label(product\_frame, text="Iced Coffee", font=("times new roman", 15, "bold"),

                         bg="RosyBrown")

    icedLabel.grid(row=1, column=1, sticky="w",padx=20)

    prodLabel = Label(product\_frame, text="Products", font=('arial', 15, 'bold'), bg='RosyBrown').grid(row=0, column=1)

    priceLabel = Label(product\_frame, text="Price", font=('arial', 15, 'bold'), bg='RosyBrown').grid(row=0,column=2)

    quantLabel = Label(product\_frame, text="Quantity", font=('arial', 15, 'bold'), bg='RosyBrown').grid(row=0, column=3)

    cinnamonPriceEntry = Entry(product\_frame, font=("arial", 15,), bd=10, width=5, justify='center',

                               textvariable=cinnamonPrice, state='readonly')

    cinnamonPriceEntry.grid(row=1, column=2, padx=10)

    icedCoffeePrice = Entry(product\_frame, font=("arial", 15), bd=10, width=3, justify='center', textvariable=ice)

    icedCoffeePrice.grid(row=1, column=3, padx=8, pady=8)

    icedCoffeePrice.bind('<Button-1>', lambda event: delete\_zero(event, icedCoffeePrice))

    minusBtn = Button(product\_frame,text="—", bd=10, width=3, justify='center', bg="tomato3",command=lambda: decrement\_quantity(ice)).grid(row=1, column=4)

    matchaLabel = Button(product\_frame, font=("times new roman", 15, "bold"),

                         bg="beige", borderwidth=10,command=lambda: increment\_quantity(matcha))

    matchaLabel.grid(row=2, column=0, sticky="w", padx=20)

    matchImg = PhotoImage(file="glass.png")

    res1 = matchImg.subsample(2)

    matchaLabel.config(image=res1, compound=LEFT)

    matchaLabel.config(width=70, height=50)

    matchaaLabel = Label(product\_frame, text="Matcha", font=("times new roman", 15, "bold"),

                         bg="RosyBrown")

    matchaaLabel.grid(row=2, column=1, sticky="w", padx=20)

    matchaPriceEntry = Entry(product\_frame, font=("arial", 15), bd=10, width=5, justify='center',

                              textvariable=churrosPrice, state='readonly')

    matchaPriceEntry.grid(row=2, column=2, padx=10, pady=10)

    matchaPrice = Entry(product\_frame, font=("arial", 15), bd=10, width=3, justify='center', textvariable=matcha)

    matchaPrice.grid(row=2, column=3, padx=8, pady=8)

    matchaPrice.bind('<Button-1>', lambda event: delete\_zero(event, matchaPrice))

    minusBtn = Button(product\_frame, text="—", bd=10, width=3, justify='center', bg="tomato3",command=lambda: decrement\_quantity(matcha)).grid(row=2, column=4)

    AmericanoLabel = Button(product\_frame, font=("times new roman", 15, "bold"),

                         bg="beige", borderwidth=10,command=lambda: increment\_quantity(americano0))

    AmericanoLabel.grid(row=3, column=0, sticky="w", padx=20)

    ameImg = PhotoImage(file="americano.png")

    res2 = ameImg.subsample(2)

    AmericanoLabel.config(image=res2, compound=LEFT)

    AmericanoLabel.config(width=70, height=50)

    AmeLabel = Label(product\_frame, text="Americano", font=("times new roman", 15, "bold"),

                         bg="RosyBrown")

    AmeLabel.grid(row=3, column=1, sticky="w", padx=20)

    muffinsPriceEntry = Entry(product\_frame, font=("arial", 15), bd=10, width=5, justify='center',

                              textvariable=muffinsPrice, state='readonly')

    muffinsPriceEntry.grid(row=3, column=2, padx=10, pady=10)

    americanoPrice = Entry(product\_frame, font=("arial", 15), bd=10, width=3, justify='center', textvariable=americano0)

    americanoPrice.grid(row=3, column=3, padx=8, pady=8)

    americanoPrice.bind('<Button-1>', lambda event: delete\_zero(event, americanoPrice))

    minusBtn = Button(product\_frame, text="—", bd=10, width=3, justify='center', bg="tomato3",command=lambda: decrement\_quantity(americano0)).grid(row=3, column=4)

    chocolatepieLabel = Button(product\_frame, font=("times new roman", 15, "bold"),

                              bg="beige", borderwidth=10,command=lambda: increment\_quantity(caffucino0))

    chocolatepieLabel.grid(row=4, column=0, sticky="w", padx=20)

    cafImg = PhotoImage(file="coffee.png")

    res3 = cafImg.subsample(2)

    chocolatepieLabel.config(image=res3, compound=LEFT)

    chocolatepieLabel.config(width=70, height=50)

    CafLabel = Label(product\_frame, text="Caffucino", font=("times new roman", 15, "bold"),

                         bg="RosyBrown")

    CafLabel.grid(row=4, column=1, sticky="w", padx=20)

    chocolatepiePriceEntry = Entry(product\_frame, font=("arial", 15), bd=10, width=5, justify='center',

                                   textvariable=chocolatepiePrice, state='readonly')

    chocolatepiePriceEntry.grid(row=4, column=2, padx=10, pady=10)

    caffucinoPrice = Entry(product\_frame, font=("arial", 15), bd=10, width=3, justify='center', textvariable=caffucino0)

    caffucinoPrice.grid(row=4, column=3, padx=8, pady=8)

    caffucinoPrice.bind('<Button-1>', lambda event: delete\_zero(event, caffucinoPrice))

    minusBtn = Button(product\_frame, text="—", bd=10, width=3, justify='center', bg="tomato3",command=lambda: decrement\_quantity(caffucino0)).grid(row=4, column=4)

    RemoveLabel = Button(product\_frame, font=("times new roman", 15, "bold"),width=5,padx=5, pady=5, bg="RosyBrown",command=reset\_quantities, borderwidth=0)

    binImg = PhotoImage(file="bin.png")

    res5 = binImg.subsample(22)

    RemoveLabel.config(image=res5, compound=LEFT)

    RemoveLabel.config(width=50,height=50)

    RemoveLabel.grid(row=5, column=4, sticky="w")

    addBtn = Button(product\_frame, text="Add", font=("times new roman", 15, "bold"),width=5,padx=5, pady=5, bg="beige", command=add\_to\_cart).grid(row=5, column=3, sticky="w")

    ##########################################################Receipt

    receiptFrame = Frame(prod\_container, bd=8, relief=GROOVE)

    receiptFrame.grid(row=0, column=2, padx=10, sticky='n')

    receiptArea = Label(receiptFrame, text="Click Receipt to view your Bill", font=('times new roman', 15, 'bold'),

                        bd=8, relief=GROOVE)

    receiptArea.pack(fill=X)

    scrollbar = Scrollbar(receiptFrame, orient=VERTICAL)

    scrollbar.pack(side=RIGHT, fill=Y)

    textArea = Text(receiptFrame, width=50, height=20, yscrollcommand=scrollbar.set)

    textArea.pack()

    scrollbar.config(command=textArea.yview)

    totalFrame = LabelFrame(log, font=("times new roman", 15, "bold"), bg="RosyBrown", bd=15, relief=GROOVE, height=50)

    totalFrame.pack(fill=X)

    #############################################################EXIT

    def Exit():

        result = messagebox.askyesnocancel("Billing System", "Do you want to order again?")

        if result is None:

            return  # User clicked on the 'Cancel' button, do nothing

        elif result:

            log.withdraw()  # Destroy the current window

            log.deiconify()  # Show the login window again

            reset()

            entry1.config(show="")

        else:

            log.withdraw()

            root.deiconify()

            reset()

            entry1.config(show="")

    totalButton = Button(totalFrame, text="Total", font=("times new roman", 15, "bold"), bg="beige", fg="black",

                         width=10, padx=5, pady=5, command=add\_to\_cart)

    totalButton.grid(row=0, column=0, padx=80, pady=10)

    billing\_complete = False

    receiptButton = Button(totalFrame, text="Receipt", font=("times new roman", 15, "bold"), bg="beige", fg="black",

                           width=10, padx=5, pady=5, command=receipt, state=DISABLED)

    receiptButton.grid(row=0, column=1, padx=50, pady=10)

    printButton = Button(totalFrame, text="Print", font=("times new roman", 15, "bold"), bg="beige", fg="black",

                         width=10, padx=5, pady=5, command=print)

    printButton.grid(row=0, column=2, padx=50, pady=10)

    resetButton = Button(totalFrame, text="Reset", command=reset, font=("times new roman", 15, "bold"), bg="beige",

                         fg="black", width=10, padx=5, pady=5)

    resetButton.grid(row=0, column=3, padx=50, pady=10)

    exitButton = Button(totalFrame, text="Exit", command=Exit, font=("times new roman", 15, "bold"), bg="beige",

                        fg="black", width=10, padx=5, pady=5)

    exitButton.grid(row=0, column=4, padx=50, pady=10)

    log.mainloop()

############################################################## Log in

# Creating Label, Entries, and button

user\_accounts = {

    "user": "Admin12345",

    "1": "2"

}

def reset():

    entry.delete(0, END)

    entry.insert(0, "Username")  # Set the default username value

    entry1.delete(0, END)

    entry1.insert(0, "Password")  # Set the default password value

def signin():

    entered\_username = entry.get()

    entered\_password = entry1.get()

    if entered\_username in user\_accounts and user\_accounts[entered\_username] == entered\_password:

        messagebox.showinfo("Login", "You have successfully log in!")

        root.withdraw()

        reset()

        login()

    else:

        messagebox.showerror("Login", "Invalid username or password.")

        reset()

# LOGIN

def clear\_username(event):

    entry.delete(0, END)

def clear\_password(event):

    entry1.delete(0, END)

    entry1.config(show="\*")

frame = Frame(root, width=350, height=350, bg='white')

frame.place(x=550, y=70)

signinHeader = Label(frame, text='Welcome Back', fg="salmon3", bg='white', font=('Helvetica', 23, 'bold'))

signinHeader.place(x=70, y=5)

entry = Entry(frame, width=25, fg='black', border=0, bg='white', font=('Helvetica', 11))

entry.place(x=30, y=100)

entry.insert(0, 'Username')

entry.bind("<Button-1>", clear\_username)

Frame(frame, width=295, height=2, bg='black').place(x=25, y=127)

entry1 = Entry(frame, width=25, fg='black', border=0, bg='white', font=('Helvetica', 11))

entry1.place(x=30, y=177)

entry1.insert(0, 'Password')

entry1.bind("<Button-1>", clear\_password)

Frame(frame, width=295, height=2, bg='black').place(x=25, y=207)

SigninBtn = Button(frame, width=15, pady=7, text='Sign in', bg='salmon3', fg='white', border=0, command=signin).place(

    x=55, y=274)

##################################################################SignUP

def signupp():

    root.withdraw()

    signup = Toplevel()

    signup.title("Sign up | Tea Spillers")

    signup.geometry("951x550+300+200")

    signup.resizable(width=False, height=False)

    signup.configure(bg="#fff")

    def create\_account():

        entered\_username = user.get()

        entered\_password = password.get()

        entered\_email = email.get()

        entered\_confirm\_password = password1.get()

        if entered\_username == "" or entered\_password == "" or entered\_email == "" or entered\_confirm\_password == "":

            messagebox.showerror("Registration", "Please fill in all the fields.")

        elif entered\_password != entered\_confirm\_password:

            messagebox.showerror("Registration", "Passwords do not match.")

            password.config(show="")

            password1.config(show="")

            reset1()

            signup.deiconify()

        elif not is\_password\_strong(entered\_password):

            messagebox.showerror("Registration", "Password is not strong enough.")

            password.config(show="")

            password1.config(show="")

            reset1()

        elif entered\_username in user\_accounts:

            messagebox.showerror("Registration", "Account already exists with this username.")

            password1.config(show="")

            reset1()

        else:

            user\_accounts[entered\_username] = entered\_password

            messagebox.showinfo("Registration", "Account created successfully! You can now log in.")

            signup.withdraw()

            root.deiconify()

    def is\_password\_strong(password):

        # Check if password meets the required strength criteria

        # At least 8 characters long, contains at least one uppercase letter, one lowercase letter, and one digit

        if len(password) < 8:

            return False

        has\_lowercase = False

        has\_uppercase = False

        has\_digit = False

        for char in password:

            if char.islower():

                has\_lowercase = True

            elif char.isupper():

                has\_uppercase = True

            elif char.isdigit():

                has\_digit = True

        return has\_lowercase and has\_uppercase and has\_digit

    img\_signup = PhotoImage(file='logongteaspillers.png')

    Label(signup, image=img\_signup, border=0, bg='white').place(x=20, y=20)

    signup\_frame = Frame(signup, width=350, height=420, bg='white')

    signup\_frame.place(x=550, y=50)

    signinHeader = Label(signup\_frame, text='Create Account', fg="salmon3", bg='white', font=('Helvetica', 23, 'bold'))

    signinHeader.place(x=70, y=20)

    def clear\_username1(event):

        user.delete(0, END)

    def clear\_email1(event):

        email.delete(0, END)

    def clear\_password1(event):

        password.delete(0, END)

        password.config(show="\*")

    def clear\_confirmpassword1(event):

        password1.delete(0, END)

        password1.config(show="\*")

    def reset1():

        user.delete(0, END)

        user.insert(0, "Username")

        email.delete(0, END)

        email.insert(0, "Email")

        password.delete(0, END)

        password.insert(0, "Password")

        password1.delete(0, END)

        password1.insert(0, "Confirm Password")

    user = Entry(signup\_frame, width=25, fg='black', border=0, bg='white', font=('Helvetica', 11))

    user.place(x=30, y=80)

    user.insert(0, 'Username')

    user.bind("<Button-1>", clear\_username1)

    Frame(signup\_frame, width=295, height=2, bg='black').place(x=25, y=117)

    email = Entry(signup\_frame, width=25, fg='black', border=0, bg='white', font=('Helvetica', 11))

    email.place(x=30, y=150)

    email.insert(0, 'Email')

    email.bind("<Button-1>", clear\_email1)

    Frame(signup\_frame, width=295, height=2, bg='black').place(x=25, y=187)

    password = Entry(signup\_frame, width=25, fg='black', border=0, bg='white', font=('Helvetica', 11))

    password.place(x=30, y=220)

    password.insert(0, 'Password')

    password.bind("<Button-1>", clear\_password1)

    Frame(signup\_frame, width=295, height=2, bg='black').place(x=25, y=257)

    password1 = Entry(signup\_frame, width=25, fg='black', border=0, bg='white', font=('Helvetica', 11))

    password1.place(x=30, y=290)

    password1.insert(0, 'Confirm Password')

    password1.bind("<Button-1>", clear\_confirmpassword1)

    Frame(signup\_frame, width=295, height=2, bg='black').place(x=25, y=327)

    Label(signup\_frame, text="The password should be at least 8 characters, 1 uppercase, 1 lowercase, and 1 digit.", font=('Arial', 8), bg="white", wraplength=305).place(x=25, y=330)

    SubmitBtn = Button(signup\_frame, width=15, pady=7, text='Submit', bg='salmon3', fg='white', border=0,

                       command=create\_account)

    SubmitBtn.place(x=45, y=389)

    def cancel():

        signup.withdraw()

        reset()

        root.deiconify()

        reset()

        entry1.config(show="")

    CancelBtn = Button(signup\_frame, width=15, pady=7, text='Cancel', bg='salmon3', fg='white', border=0,

                       command=cancel)

    CancelBtn.place(x=175, y=389)

    signup.mainloop()

SignupBtn = Button(frame, width=15, pady=7, text='Sign up', bg='salmon3', fg='white', border=0, command=signupp).place(

    x=185, y=274)

root.mainloop()

**System Output**







