|  |
| --- |
| **CAFÉ BILLING MANAGEMENT SYSTEM**  **21CSS101J – PROGRAMMING FOR PROBLEM-SOLVING**  **Mini Project Report**  *Submitted by*  **DAKSHINA PRASATH. M [Reg. No.: RA2311026010053]**  **B.Tech. CSE - AIML**  **SRMIST-01.jpg**  **SCHOOL OF COMPUTING**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  CHENGALPATTU DISTRICT  **November 2023**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  **SRMIST-01.jpg**  **BONAFIDE CERTIFICATE**  Certified that Mini project report titled CAFÉ BILLING MANAGEMENT SYSTEM is the bonafide work of Reg.No RA2311026010053 Name Dakshina Prasath. M who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.  **SIGNATURE**  **SIGNATURE**  **Dr. R. ANNIE UTHRA**  **PROFESSOR AND HEAD**  **Department Of Computational Intelligence School of Computing**  **SRM Institute of Science and Technology,**  **Kattankulathur**  **Dr. R. USHARANI**  **GUIDE**  **Assistant Professor**  **Department Of Computational Intelligence**  **School of Computing**  **SRM Institute of Science and Technology,**  **Kattankulathur** |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Title** | **Page No.** |
| 1 | Problem Statement | 1 |
| 2 | Methodology / Procedure/ Algorithm | 2 |
| 3 | Flowchart | 3 |
| 4 | Coding (Python) | 5 |
| 5 | Modules of the proposed work | 15 |
| 6 | Results/Screenshots | 16 |
| 7 | Conclusion | 20 |
| 8 | References | 21 |

1. **Problem Statement**

The project aims to develop a comprehensive Cafe Billing System with a user-friendly graphical interface using Python and Tkinter. The system should facilitate the efficient management of cafe orders, enabling staff to input customer preferences, calculate the total cost, generate bills, and maintain a record of order history. The solution should include functionalities such as order entry for various food items, dynamic total calculation based on quantities, bill generation with proper formatting, QR code generation for bills, and the ability to save and view order history. Additionally, the system should feature a secure user authentication mechanism to control access. The project report will encompass a detailed description of system requirements, architecture, user interface design, implementation details, testing and validation procedures, user authentication mechanisms, and an evaluation of the system's performance. The report will conclude with reflections on the project's success and propose potential future enhancements for the Cafe Billing System.

1. **Methodology / Procedure/ Algorithm**

The methodology for developing the Cafe Billing System involves a systematic approach encompassing various stages from planning to implementation. The following steps outline the methodology:

Requirements Analysis:

* Conduct a thorough analysis of the requirements by understanding the needs of the cafe and its staff.
* Identify key features, functionalities, and constraints of the system.

System Design:

* Develop a high-level system architecture outlining the major components and their interactions.
* Design the graphical user interface (GUI) with a focus on user-friendliness and efficiency.
* Plan the data structures and algorithms needed for order processing, calculations, and data storage.

Technology Selection:

* Choose appropriate technologies for implementation, considering the cafe's requirements and the development team's expertise.
* Select Python for application logic and Tkinter for the graphical user interface.

User Authentication Implementation:

* Develop a secure user authentication mechanism to control access to the Cafe Billing System.
* Ensure that login credentials are validated and unauthorized access is prevented.

Functionality Implementation:

* Implement each functionality of the system, such as order entry, total calculation, bill generation, QR code generation, and order history.
* Follow best practices for coding standards and ensure modular and maintainable code.

Testing:

* Conduct unit testing for individual components to ensure their correctness.
* Perform integration testing to verify the interaction between different modules.
* Test the system with various input scenarios to identify and fix bugs.

User Interface Testing:

* Validate the user interface by conducting usability tests with potential users.
* Gather feedback on the interface design and make necessary adjustments for enhanced user experience.

Documentation:

* Document the entire development process, including system requirements, design decisions, and implementation details.
* Provide clear and concise code documentation to aid future maintenance.

**3. Flow chart**

+-------------------+

| Start |

+-------------------+

↓

+-------------------+

| User Inputs Order |

+-------------------+

↓

+-------------------+

| Validate Inputs |

+-------------------+

↓

+-------------------+

| Process Order |

| Update Totals |

| Display Bill |

+---------------------+

↓

+-----------------------+

| User Authentication|

+----------------------+

↓

+---------------------+

| Successful Login|

| Show Main Screen|

+---------------------+

↓

+-----------------------+

| Unsuccessful Login |

|Show Error Message|

+----------------------+

↓

+-----------------------+

| User Logs Out |

|Return to Login Screen|

+----------------------+

↓

+-----------------+

| Generate Bill |

+-----------------+

↓

+-----------------------+

| Calculate Totals|

| Generate OR Code |

| Save Order to List|

+----------------------+

↓

+-----------------------+

| Print Bill|

| Order History|

| I Save to File|

+----------------------+

↓

+-----+

| End |

+-------+

**4. Coding (C/Python)**

from tkinter import\*

from tkinter import ttk

import random

from tkinter import messagebox

import time

import qrcode

from PIL import Image, ImageTk

import os

x=random.randint(12980, 50876)

randomRef = str(x)

root = Tk()

root.geometry("890x580+0+0")

root.title("CAFE BILLING SYSTEM")

Tops = Frame(root,bg="white",width = 1600,height=50,relief=SUNKEN)

Tops.pack(side=TOP)

f1 = Frame(root,width = 900,height=700,relief=SUNKEN)

f1.pack(side=LEFT)

f2 = Frame(root ,width = 400,height=700,relief=SUNKEN)

f2.pack(side=RIGHT)

#------------------TIME--------------

localtime=time.asctime(time.localtime(time.time()))

#-----------------INFO TOP------------

lblinfo = Label(Tops, font=( 'aria' ,30, 'bold' ),text="CAFE BILLING MANAGEMENT SYSTEM",fg="Black",bd=10,anchor='w')

lblinfo.grid(row=0,column=0)

lblinfo = Label(Tops, font=( 'aria' ,20, ),text=localtime,fg="steel blue",anchor=W)

lblinfo.grid(row=1,column=0)

def GenerateBill0():

# Create the bill content for the QR code

bill\_content = f"Order No: {rand.get()}\n"

bill\_content += f"French Fries: {Fries.get()}\n"

bill\_content += f"Lunch: {Largefries.get()}\n"

bill\_content += f"Burger: {Burger.get()}\n"

bill\_content += f"Pizza: {Filet.get()}\n"

bill\_content += f"Cheese Burger: {Cheese\_burger.get()}\n"

bill\_content += f"Drinks: {Drinks.get()}\n"

bill\_content += f"Subtotal: {Subtotal.get()}\n"

bill\_content += f"Tax: {Tax.get()}\n"

bill\_content += f"Service Charge: {Service\_Charge.get()}\n"

bill\_content += f"Total: {Total.get()}"

# Create a QR code for the bill content

qr = qrcode.QRCode(

version=1,

error\_correction=qrcode.constants.ERROR\_CORRECT\_L,

box\_size=10,

border=4,

)

qr.add\_data(bill\_content)

qr.make(fit=True)

img = qr.make\_image(fill\_color="black", back\_color="white")

# Save the QR code as an image file

img.save("bill\_qr.png")

# Display the QR code in a new window

qr\_window = Toplevel()

qr\_window.title("Bill QR Code")

qr\_image = Image.open("bill\_qr.png")

qr\_photo = ImageTk.PhotoImage(qr\_image)

qr\_label = Label(qr\_window, image=qr\_photo)

qr\_label.image = qr\_photo

qr\_label.pack()

# Remove the temporary image file after displaying

os.remove("bill\_qr.png")

def GenerateBill():

global randomRef

bill\_window = Toplevel(root)

bill\_window.title("Bill")

# Create the bill layout

bill\_text = Text(bill\_window)

bill\_text.pack()

# Generate and format the bill

bill = "-------- Cafe Bill --------\n\n"

bill += f"Order No.: {randomRef}\n"

bill += f"Date & Time: {localtime}\n\n"

bill += "Items:\n"

items = [

("French Fries", Fries.get(), 25),

("Lunch", Largefries.get(), 40),

("Burger", Burger.get(), 35),

("Pizza", Filet.get(), 50),

("Cheese Burger", Cheese\_burger.get(), 30),

("Drinks", Drinks.get(), 35),

]

total\_price = 0

for item, quantity, price in items:

if quantity:

total\_price += float(quantity) \* price

bill += f"{item}: {quantity} x {price} = {float(quantity) \* price}\n"

bill += f"\nTotal Cost: Rs. {total\_price:.2f}\n"

bill += f"Service Charge: Rs. {float(total\_price) / 99:.2f}\n"

bill += f"Tax: Rs. {total\_price \* 0.33:.2f}\n"

bill += f"Subtotal: Rs. {total\_price:.2f}\n"

bill += f"Overall Total: Rs. {total\_price + float(total\_price) / 99 + total\_price \* 0.33:.2f}\n"

bill\_text.insert(INSERT, bill)

def Ref():

try:

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 \* 30

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)

except ValueError:

messagebox.showerror("Error", "Invalid Input. Please enter numeric values.")

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("")

#---------------------------------------------------------------------------------------

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="green",bd=20,anchor='w')

lblreference.grid(row=0,column=0)

txtreference = Entry(f1,font=('ariel' ,16,'bold'), textvariable=rand , bd=6,insertwidth=6,bg="blue" ,justify='right')

txtreference.grid(row=0,column=1)

lblfries = Label(f1, font=( 'aria' ,16, 'bold' ),text=" French Fries ",fg="blue",bd=10,anchor='w')

lblfries.grid(row=2,column=0)

txtfries = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Fries , bd=6,insertwidth=4,bg="gray" ,justify='right')

txtfries.grid(row=2,column=1)

lblLargefries = Label(f1, font=( 'aria' ,16, 'bold' ),text="Lunch ",fg="blue",bd=10,anchor='w')

lblLargefries.grid(row=3,column=0)

txtLargefries = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Largefries , bd=6,insertwidth=4,bg="gray" ,justify='right')

txtLargefries.grid(row=3,column=1)

lblburger = Label(f1, font=( 'aria' ,16, 'bold' ),text="Burger ",fg="blue",bd=10,anchor='w')

lblburger.grid(row=4,column=0)

txtburger = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Burger , bd=6,insertwidth=4,bg="gray" ,justify='right')

txtburger.grid(row=4,column=1)

lblFilet = Label(f1, font=( 'aria' ,16, 'bold' ),text="Pizza ",fg="blue",bd=10,anchor='w')

lblFilet.grid(row=5,column=0)

txtFilet = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Filet , bd=6,insertwidth=4,bg="gray" ,justify='right')

txtFilet.grid(row=5,column=1)

lblCheese\_burger = Label(f1, font=( 'aria' ,16, 'bold' ),text="Cheese burger",fg="blue",bd=10,anchor='w')

lblCheese\_burger.grid(row=6,column=0)

txtCheese\_burger = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Cheese\_burger , bd=6,insertwidth=4,bg="gray" ,justify='right')

txtCheese\_burger.grid(row=6,column=1)

#--------------------------------------------------------------------------------------

lblDrinks = Label(f1, font=( 'aria' ,16, 'bold' ),text="Drinks",fg="blue",bd=10,anchor='w')

lblDrinks.grid(row=1,column=0)

txtDrinks = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Drinks , bd=6,insertwidth=4,bg="gray" ,justify='right')

txtDrinks.grid(row=1,column=1)

lblcost = Label(f1, font=( 'aria' ,16, 'bold' ),text="Cost",fg="black",bd=10,anchor='w')

lblcost.grid(row=2,column=2)

txtcost = Entry(f1,font=('ariel' ,16,'bold'), textvariable=cost , bd=6,insertwidth=4,bg="white" ,justify='right')

txtcost.grid(row=2,column=3)

lblService\_Charge = Label(f1, font=( 'aria' ,16, 'bold' ),text="Service Charge",fg="black",bd=10,anchor='w')

lblService\_Charge.grid(row=3,column=2)

txtService\_Charge = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Service\_Charge , bd=6,insertwidth=4,bg="white" ,justify='right')

txtService\_Charge.grid(row=3,column=3)

lblTax = Label(f1, font=( 'aria' ,16, 'bold' ),text="Tax",fg="black",bd=10,anchor='w')

lblTax.grid(row=4,column=2)

txtTax = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Tax , bd=6,insertwidth=4,bg="white" ,justify='right')

txtTax.grid(row=4,column=3)

lblSubtotal = Label(f1, font=( 'aria' ,16, 'bold' ),text="Subtotal",fg="black",bd=10,anchor='w')

lblSubtotal.grid(row=5,column=2)

txtSubtotal = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Subtotal , bd=6,insertwidth=4,bg="white" ,justify='right')

txtSubtotal.grid(row=5,column=3)

lblTotal = Label(f1, font=( 'aria' ,16, 'bold' ),text="Total",fg="green",bd=10,anchor='w')

lblTotal.grid(row=6,column=2)

txtTotal = Entry(f1,font=('ariel' ,16,'bold'), textvariable=Total , bd=6,insertwidth=4,bg="grey" ,justify='right')

txtTotal.grid(row=6,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="French Fries", 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 ", 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 ", 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 ", 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()

def open\_main\_application():

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("")

root.deiconify()

def login():

username = username\_entry.get()

password = password\_entry.get()

if username == "admin" and password == "0000": # Replace with your desired login credentials

login\_window.destroy()

open\_main\_application()

else:

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

# Create a Treeview for Order History

order\_tree = ttk.Treeview(root, columns=("Order ID", "Order Date", "Total Amount"))

order\_tree.heading("#0", text="Order ID")

order\_tree.heading("Order ID", text="Order ID")

order\_tree.heading("Order Date", text="Order Date")

order\_tree.heading("Total Amount", text="Total Amount")

order\_tree.pack()

saved\_orders = []

# Function to save order to the list

def SaveOrderHistory():

order\_data = {

"Order ID": rand.get(),

"Order Date": localtime,

"Order Time": time.strftime("%H:%M:%S"),

"French Fries": Fries.get(),

"Lunch": Largefries.get(),

"Burger": Burger.get(),

"Pizza": Filet.get(),

"Cheese Burger": Cheese\_burger.get(),

"Drinks": Drinks.get(),

"Subtotal": Subtotal.get(),

"Tax": Tax.get(),

"Service Charge": Service\_Charge.get(),

"Total": Total.get()

}

saved\_orders.append(order\_data)

# Function to show saved orders

def show\_saved\_orders():

orders\_window = Toplevel(root)

orders\_window.title("Order History")

order\_tree = ttk.Treeview(orders\_window, columns=list(saved\_orders[0].keys()), show="headings")

for column in saved\_orders[0].keys():

order\_tree.heading(column, text=column)

order\_tree.column(column, width=100, anchor=CENTER)

for i, order\_data in enumerate(saved\_orders, start=1):

values = [order\_data[column] for column in saved\_orders[0].keys()]

order\_tree.insert("", i, values=values)

order\_tree.pack(expand=YES, fill=BOTH)

# Create a login window

login\_window = Toplevel(root)

login\_window.title("Login")

login\_window.geometry("400x300+300+200")

username\_label = Label(login\_window, text="Username:")

username\_label.grid(row=0, column=0, padx=10, pady=10)

username\_entry = Entry(login\_window)

username\_entry.grid(row=0, column=1, padx=10, pady=10)

password\_label = Label(login\_window, text="Password:")

password\_label.grid(row=1, column=0, padx=10, pady=10)

password\_entry = Entry(login\_window, show="\*") # Use show="\*" to hide the password

password\_entry.grid(row=1, column=1, padx=10, pady=10)

login\_button = Button(login\_window, text="Login", command=login)

login\_button.grid(row=2, columnspan=2, pady=10)

#-----------------------------------------buttons------------------------------------------

lblTotal = Label(f1,text="---------------------",fg="white")

lblTotal.grid(row=7,columnspan=3)

# Button to calculate total cost

btnTotal = Button(f1, padx=10, pady=6, bd=6, fg="black", font=('ariel', 16, 'bold'), width=8, text="TOTAL", bg="green", command=Ref)

btnTotal.grid(row=8, column=0, padx=2, pady=2)

# Button to reset the form

btnReset = Button(f1, padx=10, pady=6, bd=6, fg="black", font=('ariel', 16, 'bold'), width=8, text="RESET", bg="yellow", command=reset)

btnReset.grid(row=8, column=1, padx=2, pady=2)

# Button to exit the application

btnExit = Button(f1, padx=10, pady=6, bd=6, fg="black", font=('ariel', 16, 'bold'), width=8, text="EXIT", bg="red", command=qexit)

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

# Button to show price list

btnPrice = Button(f1, padx=10, pady=6, bd=6, fg="black", font=('ariel', 16, 'bold'), width=8, text="PRICE", bg="green", command=price)

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

# Button to generate bill

btnGenerateBill = Button(f1, padx=10, pady=6, bd=6, fg="black", font=('ariel', 12, 'bold'), width=8, text="Bill", bg="purple", command=GenerateBill)

btnGenerateBill.grid(row=8, column=4, padx=2, pady=2, sticky="e")

# Button to generate QR code

btnGenerateQR = Button(f1, padx=10, pady=6, bd=6, fg="black", font=('ariel', 12, 'bold'), width=8, text="QR Code", bg="purple", command=GenerateBill0)

btnGenerateQR.grid(row=8, column=5, padx=2, pady=2, sticky="e")

# Button to save order and view order history

btnSaveOrder = Button(f1, padx=10, pady=6, bd=6, fg="black", font=('ariel', 12, 'bold'), width=8, text="Save", bg="orange", command=SaveOrderHistory)

btnSaveOrder.grid(row=8, column=6, padx=2, pady=2, sticky="e")

# Button to view saved orders

btnViewOrders = Button(f1, padx=10, pady=6, bd=6, fg="black", font=('ariel', 12, 'bold'), width=8, text="Orders", bg="purple", command=show\_saved\_orders)

btnViewOrders.grid(row=8, column=7, padx=2, pady=2, sticky="e")

root.iconify()

root.mainloop()

**5. Modules of the proposed work**

The proposed Cafe Billing System can be organized into several modules to manage different aspects of the application efficiently. Here are the suggested modules for the proposed work:

1. **User Authentication Module:**
   * Responsible for user login and authentication.
   * Ensures that only authorized users have access to the system.
2. **Order Entry Module:**
   * Allows users to input customer orders.
   * Validates and processes the order details.
3. **Total Calculation Module:**
   * Computes the total cost of the order based on the quantity and prices of selected items.
   * Calculates tax, service charge, and overall total.
4. **Bill Generation Module:**
   * Generates a detailed bill for the customer.
   * Includes information such as order number, items, quantity, prices, and total amount.
5. **QR Code Generation Module:**
   * Creates a QR code containing essential order information.
   * Enhances the customer experience for quick and easy order tracking.
6. **Order History Module:**
   * Saves and manages a history of customer orders.
   * Allows users to view and retrieve past orders.
7. **User Interface Module:**
   * Manages the graphical user interface (GUI) components.
   * Handles the layout, design, and user interactions for a seamless user experience.
8. **Database Integration Module:**
   * Integrates with a database to store and retrieve order history data.
   * Enables data persistence and efficient management of customer records.
9. **Reporting Module:**
   * Generates reports on sales, order trends, and other relevant data.
   * Provides insights for business analysis and decision-making.
10. **Error Handling Module:**
    * Manages error handling and validation of user inputs.
    * Ensures data integrity and prevents unexpected issues.
11. **Documentation Module:**
    * Facilitates the creation and maintenance of project documentation.
    * Includes user manuals, code documentation, and project reports.
12. **Security Module:**
    * Implements security measures to protect user data and sensitive information.
    * Ensures secure user authentication and data transmission.

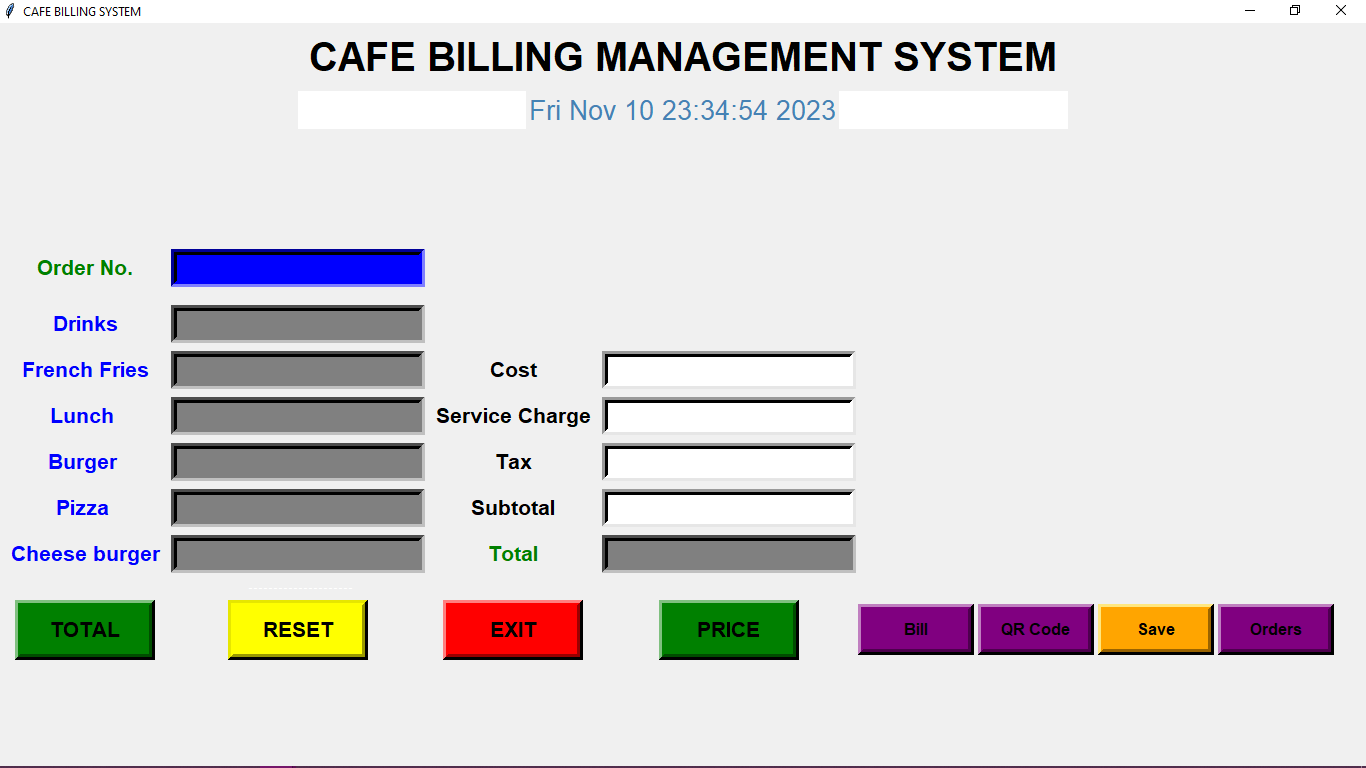
These modules collectively contribute to the functionality, efficiency, and maintainability of the Cafe Billing System. Each module can be developed, tested, and maintained independently, facilitating a modular and scalable approach to the software development process.

**6. Results/Screenshots**

**A screenshot of a computer

Description automatically generated**

**First the login page will ask the user to enter the username and password, after entering it will proceed to the billing window.**

****

**In the billing window we can enter the billing details to proceed.**

**A screenshot of a computer

Description automatically generated**

**After entering the billing details we need to click on Total button to get the total amount displayed.**

**A screenshot of a computer

Description automatically generated**

**By clicking on the Price button we can see the price of all the items.**

**A computer screen shot of a computer

Description automatically generated**

**By clicking on the button we can this window which shows the actual bill.**

**A qr code on a screen

Description automatically generated**

**If we click on the QR Code button it will display a QR code through which we can get the billing details.**

**A screenshot of a computer

Description automatically generated**

**If we click on the Orders button after saving the order, this window will be displayed which has the database of the order details.**

**7. Conclusion**

The Cafe Billing System project demonstrates the successful integration of various features to create an efficient and user-friendly application. The implementation of security measures, intuitive order management, and advanced features like QR code generation contribute to a robust system that can benefit cafes in their daily operations.

**8. References**

Tkinter Documentation:

qrcode Documentation

Pillow (PIL) Documentation