COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS,

UNIVERSITI TEKNOLOGI MARA,

CAWANGAN KEDAH

DIPLOMA IN LIBRARY INFORMATICS (CDIM144)

PROGRAMMING FOR LIBRARIES (IML208)

"FOOD ORDERING SYSTEM: HUH? CAFE ORDERING SYSTEM"

PREPARED BY:

NAME	STUDENT ID
PUTERI AYU NADHIRAH BINTI ROSLAN	2022453888
SHADATUL AMALIN BINTI AZMI	2022807952
SITI AISYAH NATASHA BINTI SOPHIAN	2022677088
SYAZA AFIFAH BINTI AHMAD FISOL	2022491862

CLASS: KCDIM1443F

PREPARED FOR:

SIR AIRUL SHAZWAN BIN NORSHAHIMI

SUBMISSION DATE:

17TH JANUARY 2024

“FOOD ORDERING SYSTEM: HUH? CAFE ORDERING SYSTEM”

PREPARED BY:

NAME	STUDENT ID
PUTERI AYU NADHIRAH BINTI ROSLAN	2022453888
SHADATUL AMALIN BINTI AZMI	2022807952
SITI AISYAH NATASHA BINTI SOPHIAN	2022677088
SYAZA AFIFAH BINTI AHMAD FISOL	2022491862

DIPLOMA IN LIBRARY INFORMATICS (CDIM144)

COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS,

UNIVERSITI TEKNOLOGI MARA,

MERBOK, KEDAH

SUBMISSION DATE:

17TH JANUARY 2024

TABLE OF CONTENT

STUDENT PLEDGE OF ACADEMIC INTEGRITY

As a student of Universiti Teknologi MARA (UiTM), it is my responsibility to act in accordance with UiTM's academic assessment and evaluation policy. I hereby pledge to act and uphold academic integrity and pursue scholarly activities in UiTM with honesty and responsible manner. I will not engage or tolerate acts of academic dishonesty, academic misconduct, or academic fraud including but not limited to:

a. Cheating: Using or attempt to use any unauthorized device, assistance, sources, practice, or materials while completing academic assessments. This include but not limited to copying from another, allowing others to copy, unauthorized collaboration on an assignment or open book tests, or engaging in any act or conduct that can be construed as cheating.
b. Plagiarism: Using or attempts to use the work of others (ideas, design, words, art, music, etc.) without acknowledging the source; using or purchasing materials prepared by another person or agency or engaging in other behavior that a reasonable person would consider as plagiarism.
c. Fabrication: Falsifying data, information, or citations in any academic assessment and evaluation.
d. Deception: Providing false information with intend to deceive an instructor concerning any academic assessment and evaluation. e. Furnishing false information: Providing false information or false representation to any UiTM official, instructor, or office.

With this pledge, I am fully aware that I am obliged to conduct myself with utmost honesty and integrity. I fully understand that a disciplinary action can be taken against me if I, in any manner, violate this pledge.

Name : PUTERI AYU NADHIRAH BINTI ROSLAN

Matric Number : 2022453888

Course Code : IML208

Programme Code : CDIM144

Faculty / Campus : UiTM Kampus Sungai Petani

STUDENT PLEDGE OF ACADEMIC INTEGRITY

As a student of Universiti Teknologi MARA (UiTM), it is my responsibility to act in accordance with UiTM's academic assessment and evaluation policy. I hereby pledge to act and uphold academic integrity and pursue scholarly activities in UiTM with honesty and responsible manner. I will not engage or tolerate acts of academic dishonesty, academic misconduct, or academic fraud including but not limited to:

a. Cheating: Using or attempt to use any unauthorized device, assistance, sources, practice, or materials while completing academic assessments. This include but not limited to copying from another, allowing others to copy, unauthorized collaboration on an assignment or open book tests, or engaging in any act or conduct that can be construed as cheating.
b. Plagiarism: Using or attempts to use the work of others (ideas, design, words, art, music, etc.) without acknowledging the source; using or purchasing materials prepared by another person or agency or engaging in other behavior that a reasonable person would consider as plagiarism.
c. Fabrication: Falsifying data, information, or citations in any academic assessment and evaluation.
d. Deception: Providing false information with intend to deceive an instructor concerning any academic assessment and evaluation. e. Furnishing false information: Providing false information or false representation to any UiTM official, instructor, or office.

With this pledge, I am fully aware that I am obliged to conduct myself with utmost honesty and integrity. I fully understand that a disciplinary action can be taken against me if I, in any manner, violate this pledge.

Name : SHADATUL AMALIN BINTI AZMI

Matric Number : 2022807952

Course Code : IML208

Programme Code : CDIM144

Faculty / Campus : UiTM Kampus Sungai Petani

STUDENT PLEDGE OF ACADEMIC INTEGRITY

As a student of Universiti Teknologi MARA (UiTM), it is my responsibility to act in accordance with UiTM's academic assessment and evaluation policy. I hereby pledge to act and uphold academic integrity and pursue scholarly activities in UiTM with honesty and responsible manner. I will not engage or tolerate acts of academic dishonesty, academic misconduct, or academic fraud including but not limited to:

a. Cheating: Using or attempt to use any unauthorized device, assistance, sources, practice, or materials while completing academic assessments. This include but not limited to copying from another, allowing others to copy, unauthorized collaboration on an assignment or open book tests, or engaging in any act or conduct that can be construed as cheating.
b. Plagiarism: Using or attempts to use the work of others (ideas, design, words, art, music, etc.) without acknowledging the source; using or purchasing materials prepared by another person or agency or engaging in other behavior that a reasonable person would consider as plagiarism.
c. Fabrication: Falsifying data, information, or citations in any academic assessment and evaluation.
d. Deception: Providing false information with intend to deceive an instructor concerning any academic assessment and evaluation. e. Furnishing false information: Providing false information or false representation to any UiTM official, instructor, or office.

With this pledge, I am fully aware that I am obliged to conduct myself with utmost honesty and integrity. I fully understand that a disciplinary action can be taken against me if I, in any manner, violate this pledge.

Name : SITI AISYAH NATASHA BINTI SOPHIAN

Matric Number : 2022677088

Course Code : IML208

Programme Code : CDIM144

Faculty / Campus : UiTM Kampus Sungai Petani

STUDENT PLEDGE OF ACADEMIC INTEGRITY

As a student of Universiti Teknologi MARA (UiTM), it is my responsibility to act in accordance with UiTM's academic assessment and evaluation policy. I hereby pledge to act and uphold academic integrity and pursue scholarly activities in UiTM with honesty and responsible manner. I will not engage or tolerate acts of academic dishonesty, academic misconduct, or academic fraud including but not limited to:

a. Cheating: Using or attempt to use any unauthorized device, assistance, sources, practice, or materials while completing academic assessments. This include but not limited to copying from another, allowing others to copy, unauthorized collaboration on an assignment or open book tests, or engaging in any act or conduct that can be construed as cheating.
b. Plagiarism: Using or attempts to use the work of others (ideas, design, words, art, music, etc.) without acknowledging the source; using or purchasing materials prepared by another person or agency or engaging in other behavior that a reasonable person would consider as plagiarism.
c. Fabrication: Falsifying data, information, or citations in any academic assessment and evaluation.
d. Deception: Providing false information with intend to deceive an instructor concerning any academic assessment and evaluation. e. Furnishing false information: Providing false information or false representation to any UiTM official, instructor, or office.

With this pledge, I am fully aware that I am obliged to conduct myself with utmost honesty and integrity. I fully understand that a disciplinary action can be taken against me if I, in any manner, violate this pledge.

Name : SYAZA AFIFAH BINTI AHMAD FISOL

Matric Number : 2022491862

Course Code : IML208

Programme Code : CDIM144

Faculty / Campus : UiTM Kampus Sungai Petani

ACKNOWLEDGEMENTS

First and foremost, we would love to express our deepest appreciation and respect to the individuals who provided us the possibility and gave us the chance to complete this final group project. A special gratitude to our respected lecturer for Programming for Libraries (IML208) subject, Sir Airul Shazwan bin Norshahimi for giving a clear and concise instructions to make it easier for us to proceed this assignment smoothly, also assisted us on encouragement and guidance to finally make this assignment succeed without exceeding its due date.

A very huge thank you to our classmates in KCDIM1443F class for giving each other cooperation and support while we were working on our very last assignment for this subject. Special appreciation to each group mate for striving while doing an abundance of practice while adding on to our own skills and gaining a bunch of new priceless knowledge on how to execute coding using Python program and a database using XAMPP while relating them to our developed system.

Lastly, we would like to thank our loved ones and our family especially our beloved parents for always being supportive and keep motivating us to continue our work until the end of our studies.

1.0 INTRODUCTION

For the last group project of IML208 subject in our third semester, we are required to design and develop a python program and computer interface which consists of Creation, Read, Update and Delete (CRUD) operations. Apart from that, we are also required to include calculation operations which are necessary for our system. We also had included looping requirement in our system to ensure the system execute a block of statements or commands repeatedly until a given condition is satisfied. The purpose of a database system is to assist various individuals in ensuring their collected data is stored, systematically arranged, and manage a large amount of data at once according to the capacity of the database. A computer interface database is highly crucial as it performs as a boundary that allows an interaction or communication between a computer or a program and a user.

For this very final project, the system that we created is a cafe ordering system which functions to ease a company's work to record customer's orders from various set of menus in the cafe. The name of our system is Huh? Cafe Ordering System which is an ordering type of system that consists of three submodules which are the customer's profile, finance page and menu page. The system works by starting with the customers registering or setting up their profiles by filling up their first and last names, phone numbers and e-mails. This section allows the customers to update their information such as updating their old e-mails to a new one. This section also allows the users to delete their profile.

Next, the system would redirect them to the finance page. This page requires user to fill in the amount of top-ups which is not less than RM20. Customers can also choose their top-up methods such as online banking, TouchNGo and debit cards. Moreover, customers would be redirected to menu section and they can fill up their names, addresses, phone numbers and selected menus with quantity of their chosen set of foods. This is essential to let the system calculate the total price of their chosen set of menus per quantity. The calculation in coding is total_price = sum (price [menu_item] * quantities [menu_item] for menu_item in selected_menu). Lastly, the system would output to the customers the total price in RM of their orders.

2.0 PROBLEM STATEMENTS

- Limited choices of menus and lack of varieties in a food ordering system would lead to less customers being attracted to make use of the database or system.
- A complex system with no options to update any information or edit any mistakes and delete no longer in use information would be difficult for users and the system manager to keep up with user's latest information.
- A system with no requirements of filling up exact certain amount of money would trouble the users to keep topping-up their wallets again as they would be entering a small amount for the first time.

3.0 OBJECTIVES

- **TO CREATE A SYSTEM WITH VARIETY OF CHOICES**
  The main objectives of our food ordering system are to provide an informative platform that offers online ordering capabilities, which allows customers to browse the menu, choose a tempting and mouth-watering set of food completed with a main course, a refreshing drink, and a delightful pastry.
- **TO CREATE A USER-FRIENDLY SYSTEM**
  This function as a convenient system by allowing users to edit and delete their information details in the system. The system also enables users to update any mistakes of their information.
- **TO CREATE A FIXED AND STEADY SYSTEM**
  To perform as a fixed system as our system requires users to enter an amount that is not less than RM20 to decide an adequate amount for users to do food ordering in our café.

4.0 FLOWCHART

We have prepared the flowcharts that represent three submodules which are Profile, Finance, and Menu. These flowcharts guide the flow of customers inputting their data and choosing their set of orders to be inserted into the database. These flowcharts also show the calculation to get the total price of the customer's chosen set of menus, the looping process and Update and Delete process.

4.1 MAIN PAGE FLOWCHART

This flowchart shows the main page of the system which being the first page of our ordering system to customers. It would display our café name which is Huh? Café and users are allowed to click on Profile page to fill their information, Finance page to choose the payment method and Menu page to choose their selected set of menus.

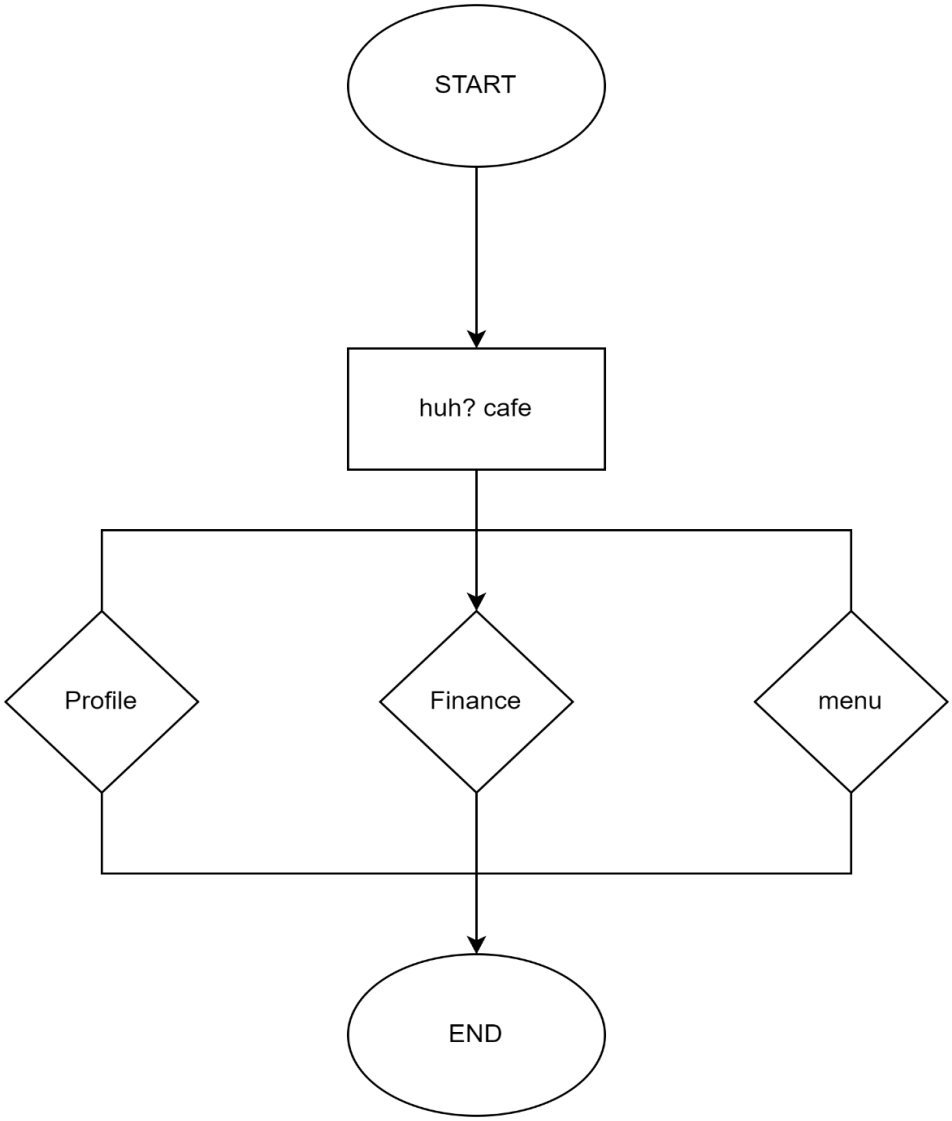


Figure 1: Flowchart of Main Page of Huh? Café Ordering System

4.2 PROFILE FLOWCHART

This flowchart shows the flow of users registering their names and phone numbers into the system. This section also allows users to delete their data which would input their profile has been deactivated. Moreover, users can also update any data such as substituting an old e-mail to a new one.

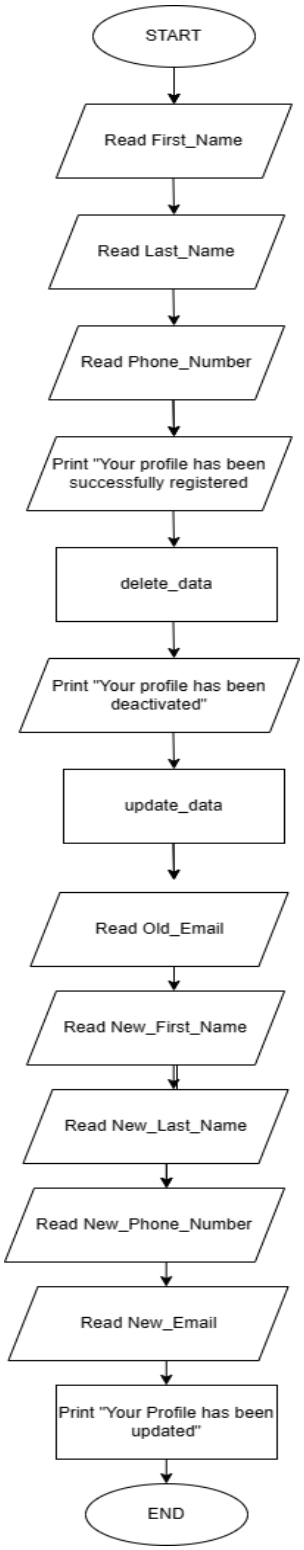


Figure 2: Flowchart of Profile Page of Huh? Café Ordering System

4.3 FINANCE FLOWCHART

This flowchart guides the process of customers entering the amount of RM they want to top-up and the minimum amount is RM20. This is the section where the looping happens where the system would command users to repeatedly enter an amount if the amount is less than RM20. If the amount is RM20 and above, the system would direct users to choose the payment method which are Online Banking, TouchNGo and Debit or Credit Card. This section would output to users that they successfully have topped-up their wallet.

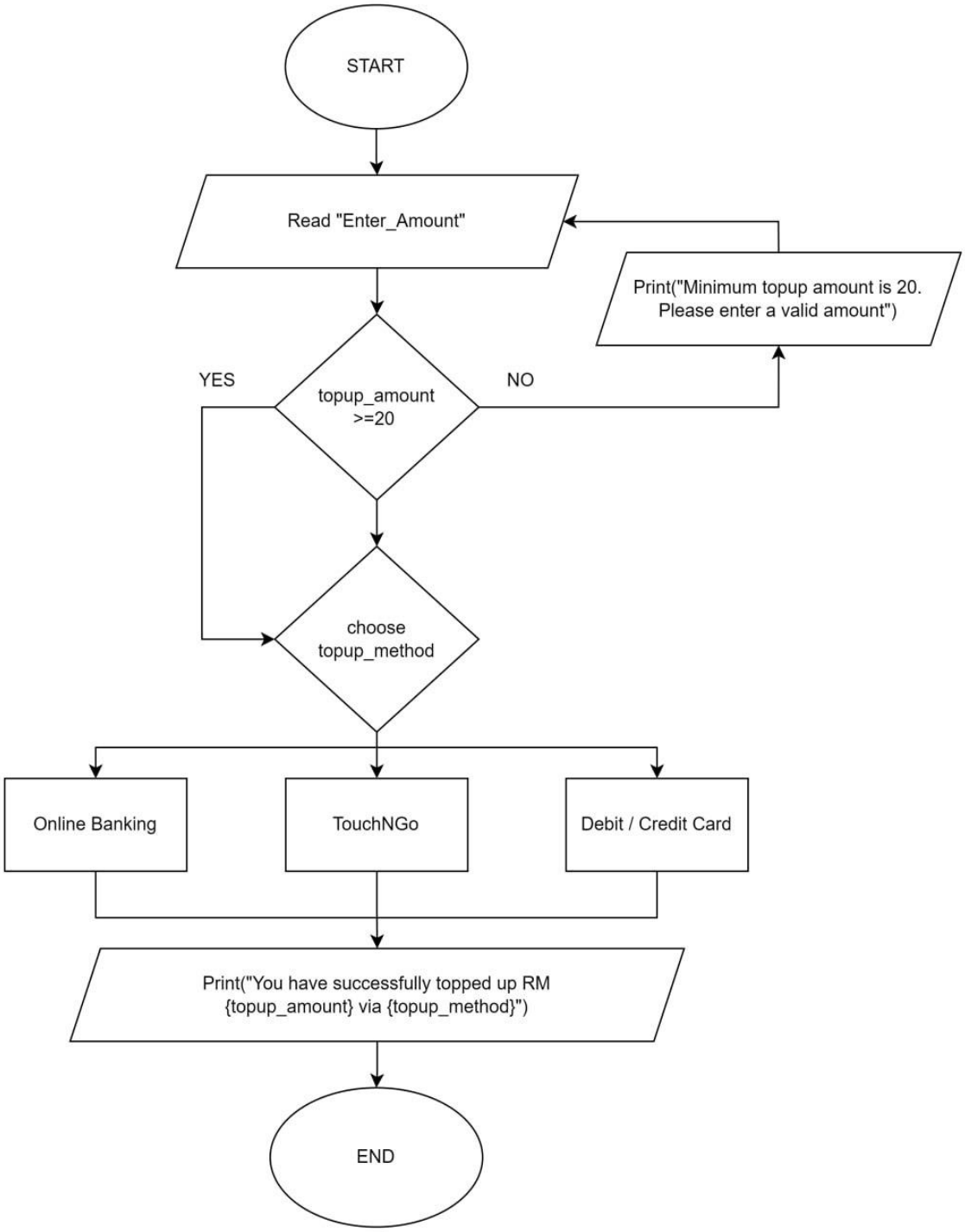


Figure 3: Flowchart of Finance Page of Huh? Café Ordering System

4.4 MENU FLOWCHART

This flowchart shows that the system requires users to fill up their names, addresses and phone numbers to ease the delivery person to deliver the orders to the customers. This section also allows the customers to select their chosen set of menus with their quantities. This is the part where the system calculates the total price of customer's orders multiplied by the quantity of orders. This section would output to the customers the total price (RM) of their chosen set of menus per quantity.

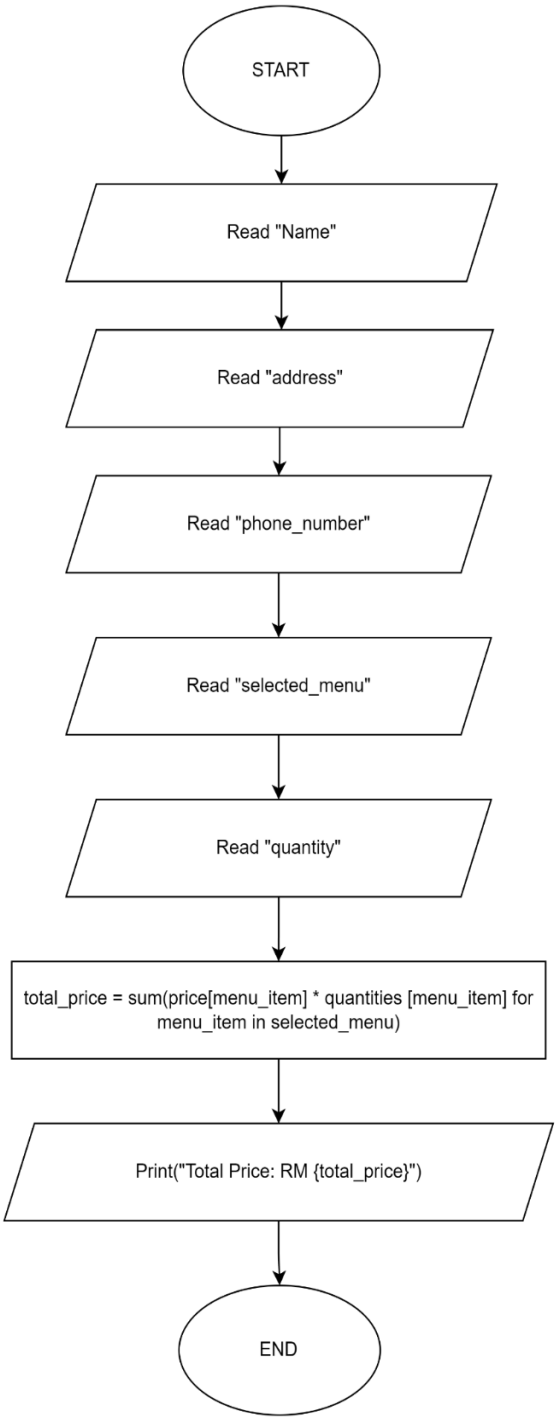
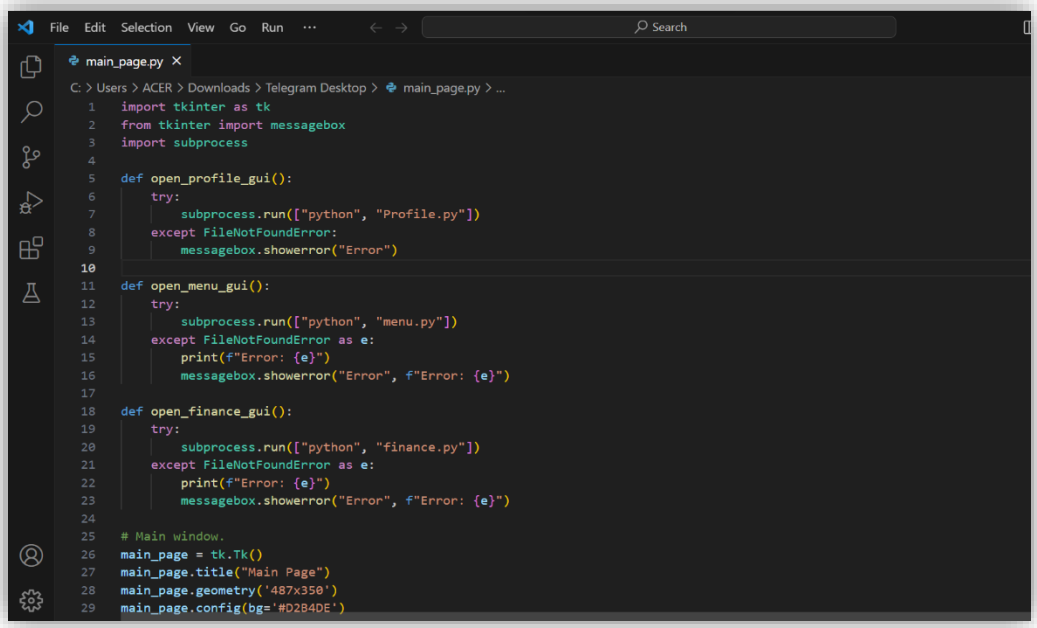


Figure 4: Flowchart of Menu Page of Huh? Café Ordering System

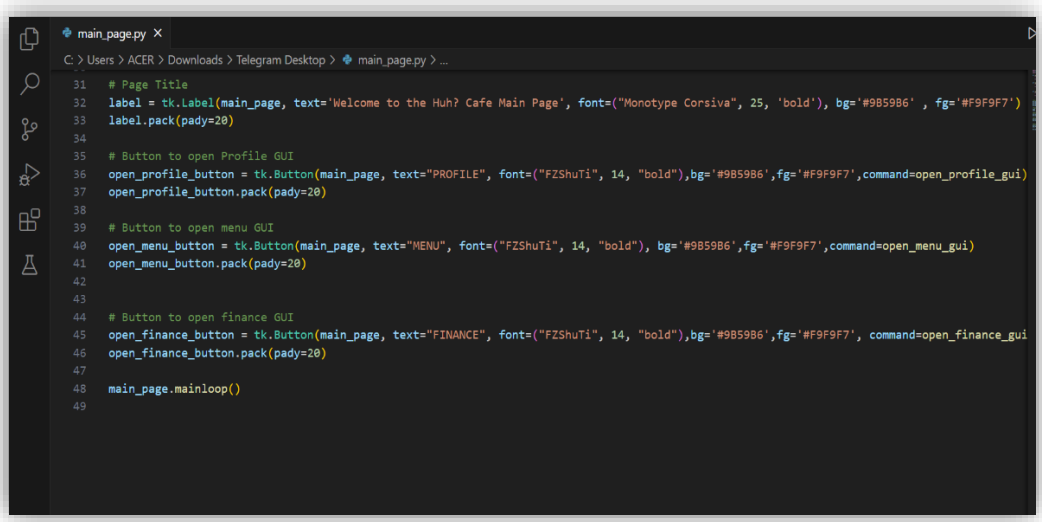
5.0 SNAPSHOT OF CODE

This is the coding process to create a Main Page for our system which would display a GUI consisting of Profile page, Finance page and Menu Page for users to choose before they proceed to do food ordering process.



```python
import tkinter as tk
from tkinter import messagebox
import subprocess

def open_profile_gui():
    try:
        subprocess.run(["python", "Profile.py"])
    except FileNotFoundError:
        messagebox.showerror("Error")

def open_menu_gui():
    try:
        subprocess.run(["python", "menu.py"])
    except FileNotFoundError as e:
        print(f"Error: {e}")
        messagebox.showerror("Error", f"Error: {e}")

def open_finance_gui():
    try:
        subprocess.run(["python", "finance.py"])
    except FileNotFoundError as e:
        print(f"Error: {e}")
        messagebox.showerror("Error", f"Error: {e}")

# Main window.
main_page = tk.Tk()
main_page.title("Main Page")
main_page.geometry('487x350')
main_page.config(bg='#D2B4DE')
```

Figure 5: Codes for Main Page of Huh? Café System



```python
# Page Title
label = tk.Label(main_page, text='Welcome to the Huh? Cafe Main Page', font=("Monotype Corsiva", 25, 'bold'), bg='#9B59B6' , fg='#F9F9F7')
label.pack(pady=20)

# Button to open Profile GUI
open_profile_button = tk.Button(main_page, text="PROFILE", font=("FZShuTi", 14, "bold"),bg='#9B59B6',fg='#F9F9F7',command=open_profile_gui)
open_profile_button.pack(pady=20)

# Button to open menu GUI
open_menu_button = tk.Button(main_page, text="MENU", font=("FZShuTi", 14, "bold"), bg='#9B59B6',fg='#F9F9F7',command=open_menu_gui)
open_menu_button.pack(pady=20)


# Button to open finance GUI
open_finance_button = tk.Button(main_page, text="FINANCE", font=("FZShuTi", 14, "bold"),bg='#9B59B6',fg='#F9F9F7', command=open_finance_gui
open_finance_button.pack(pady=20)

main_page.mainloop()
```

Figure 6: Codes for Main Page of Huh? Café System

This is the coding process of Profile Page that allows users to register their data such as first name, last name, phone number and email.

```python
import tkinter as tk
import mysql.connector

# Connect MySQL database
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="huh_cafe"
)

mycursor = mydb.cursor()

#Register Data
def collect_data():
    First_Name = str(First_Name_entry.get())
    Last_Name = str(Last_Name_entry.get())
    Phone_Number = Phone_Number_entry.get()
    Email = str(Email_entry.get())

    # Check if any field is empty
    if not all([First_Name, Last_Name, Phone_Number, Email]):
        output_label.config(text="Please fill in all the fields", bg='#F1948A')
        return


    # Insert Data to Database
    sql = "INSERT INTO `profile` (First_Name , Last_Name , Phone_Number, Email) VALUES (%s, %s, %s, %s)"
    val = (First_Name, Last_Name, Phone_Number, Email)
```

Figure 7: Codes for Profile Page of Huh? Café System: Registering Information

```python
    mycursor.execute(sql, val)
    mydb.commit()

    # To Print back the output.
    output_label.config(text="Your profile has been successfully registered", bg=('#82E0AA'))


  #To limit the number phone
def validate_Phone_Number_callback(value):
    return value.isdigit() and len(value) <= 11 or value == ''
```

Figure 8: Codes for Profile Page of Huh? Café System: Registering Information

This is the code in allowing users to delete their profiles or deactivate it.

```python
#Delete Data
def delete_data():
    Email_to_delete = str(Email_delete_entry.get())

    # Check if Email field is empty
    if not Email_to_delete:
        delete_output_label.config(text="Please enter your email", bg='#F1948A')
        return

    # Delete Data from Database
    sql = "DELETE FROM `profile` WHERE Email = %s"
    val = (Email_to_delete,)
    mycursor.execute(sql, val)
    mydb.commit()

    # To print back the output.
    delete_output_label.config(text="Your profile has been deactivated", bg=('#82E0AA'))
```

Figure 9: Codes for Profile Page of Huh? Café System: Deactivating Profile

This is the code in allowing users to update their data such as changing an old email to a new one and to edit any mistakes.

```python

#Update Data
def update_data():
    Old_Email = str(Old_Email_entry.get())
    New_First_Name = str(New_First_Name_entry.get())
    New_Last_Name = str(New_Last_Name_entry.get())
    New_Phone_Number = str(New_Phone_Number_entry.get())
    New_Email = str(New_Email_entry.get())

    # Start building the SQL query
    sql = "UPDATE `profile` SET "
    update_values = []
```

Figure 10: Codes for Profile Page of Huh? Café System: Updating Profile

```python

    if New_First_Name:
        sql += "First_Name=%s, "
        update_values.append(New_First_Name)
    if New_Last_Name:
        sql += "Last_Name=%s, "
        update_values.append(New_Last_Name)
    if New_Phone_Number:
        sql += "Phone_Number=%s, "
        update_values.append(New_Phone_Number)
    if New_Email:
        sql += "Email=%s, "
        update_values.append(New_Email)


    sql = sql.rstrip(', ')
    sql += " WHERE Email=%s"
```

Figure 11: Updating Profile

```python
    update_values.append(Old_Email)

    # Execute the update query
    mycursor.execute(sql, tuple(update_values))
    mydb.commit()

    # To print back the output
    update_output_label.config(text=f"Your profile has been updated", bg=('#82E0AA'))

#To limit the number for new number phone
def validate_New_Phone_Number_callback(value):
        return value.isdigit() and len(value) <= 11 or value == ''

# Main window.
root = tk.Tk()
root.title("Profile")
root.geometry('1350x400')
root.config(bg='#D2B4DE')

# Registration Section on the Left
registration_frame = tk.Frame(root, bg='#D2B4DE')
registration_frame.grid(row=0, column=0, padx=20, pady=10, sticky="nsew")

# Page Title
label = tk.Label(registration_frame, text='PROFILE', font=("FZShuTi", 30, "bold"),bg='#D2B4DE')
label.grid(row=0, column=0, columnspan=2, pady=10)

# First Name Entry
First_Name_label = tk.Label(registration_frame, text="Insert your first name:", font=("Cascadia Code", 10), bg='#D2B4DE')
```

Figure 12: Updating Profile

```python
First_Name_label.grid(row=1, column=0, padx=10, pady=5, sticky="e")
First_Name_entry = tk.Entry(registration_frame)
First_Name_entry.grid(row=1, column=1, padx=10, pady=5)

# Last Name Entry
Last_Name_label = tk.Label(registration_frame, text="Insert your last name:", font=("Cascadia Code", 10), bg='#D2B4DE')
Last_Name_label.grid(row=2, column=0, padx=10, pady=5, sticky="e")
Last_Name_entry = tk.Entry(registration_frame)
Last_Name_entry.grid(row=2, column=1, padx=10, pady=5)

# Phone Number Entry
Phone_Number_label = tk.Label(registration_frame, text="Insert your phone number:", font=("Cascadia Code", 10), bg='#D2B4DE')
Phone_Number_label.grid(row=3, column=0, padx=10, pady=5, sticky="e")
validate_phone_number = registration_frame.register(validate_Phone_Number_callback)
Phone_Number_entry = tk.Entry(registration_frame, validate="key", validatecommand=(validate_phone_number, '%P'))
Phone_Number_entry.grid(row=3, column=1, padx=10, pady=5)

# Email Entry
Email_label = tk.Label(registration_frame, text="Insert your email:", font=("Cascadia Code", 10), bg='#D2B4DE')
Email_label.grid(row=4, column=0, padx=10, pady=5, sticky="e")
Email_entry = tk.Entry(registration_frame)
Email_entry.grid(row=4, column=1, padx=10, pady=5)

# Save Button
save_button = tk.Button(registration_frame, text="Register", bg='#BB8FCE', command=collect_data)
save_button.grid(row=5, column=1, pady=10)

# Output Label & Result
output_label = tk.Label(registration_frame, text='Registration status:', font=("Imprint MT Shadow", 12), bg=('#D2B4DE'))
```

Figure 13: Updating Profile

```python
output_label.grid(row=6, column=0, columnspan=2, pady=10)
output_result_label = tk.Label(registration_frame, text="")
output_result_label.grid(row=7, column=0, columnspan=2)

# Deletion Section in the Middle
deletion_frame = tk.Frame(root, bg='#D2B4DE')
deletion_frame.grid(row=0, column=1, padx=20, pady=10, sticky="nsew")

# Email Entry for DELETE
Email_delete_label = tk.Label(deletion_frame, text="Account deactivation.\nEnter email to delete your profile:", font=("Cascadia Code", 10)
Email_delete_label.grid(row=1, column=0, padx=10, pady=5, sticky="e")
Email_delete_entry = tk.Entry(deletion_frame)
Email_delete_entry.grid(row=1, column=1, padx=10, pady=5)

# Delete Button
delete_button = tk.Button(deletion_frame, text="Deactivate", bg='#BB8FCE', command=delete_data)
delete_button.grid(row=2, column=1, pady=10)

# Delete Output Label & Result
delete_output_label = tk.Label(deletion_frame, text='Deactivation status:', font=("Imprint MT Shadow", 12), bg=('#D2B4DE'))
delete_output_label.grid(row=3, column=0, columnspan=2, pady=10)
delete_result_label = tk.Label(deletion_frame, text="")
delete_result_label.grid(row=4, column=0, columnspan=2)

# Update Section on the Right
update_frame = tk.Frame(root, bg='#D2B4DE')
update_frame.grid(row=0, column=2, padx=20, pady=10, sticky="nsew")

Old_Email_label = tk.Label(update_frame, text="Insert old email to update:", font=("Cascadia Code", 10), bg='#D2B4DE')
```

Figure 14: Updating Profile

```python
Old_Email_label.grid(row=1, column=0, padx=10, pady=5, sticky="e")
Old_Email_entry = tk.Entry(update_frame)
Old_Email_entry.grid(row=1, column=1, padx=10, pady=5)

New_First_Name_label = tk.Label(update_frame, text="Insert new first name:", font=("Cascadia Code", 10), bg='#D2B4DE')
New_First_Name_label.grid(row=2, column=0, padx=10, pady=5, sticky="e")
New_First_Name_entry = tk.Entry(update_frame)
New_First_Name_entry.grid(row=2, column=1, padx=10, pady=5)

New_Last_Name_label = tk.Label(update_frame, text="Insert new last name:", font=("Cascadia Code", 10), bg='#D2B4DE')
New_Last_Name_label.grid(row=3, column=0, padx=10, pady=5, sticky="e")
New_Last_Name_entry = tk.Entry(update_frame)
New_Last_Name_entry.grid(row=3, column=1, padx=10, pady=5)

New_Phone_Number_label = tk.Label(update_frame, text="Insert new phone number:", font=("Cascadia Code", 10), bg='#D2B4DE')
New_Phone_Number_label.grid(row=4, column=0, padx=10, pady=5, sticky="e")
validate_New_Phone_Number = update_frame.register(validate_New_Phone_Number_callback)
New_Phone_Number_entry = tk.Entry(update_frame, validate="key", validatecommand=(validate_New_Phone_Number, '%P'))
New_Phone_Number_entry.grid(row=4, column=1, padx=10, pady=5)

New_Email_label = tk.Label(update_frame, text="Insert new email:", font=("Cascadia Code", 10), bg='#D2B4DE')
New_Email_label.grid(row=5, column=0, padx=10, pady=5, sticky="e")
New_Email_entry = tk.Entry(update_frame)
New_Email_entry.grid(row=5, column=1, padx=10, pady=5)

# Update Button
update_button = tk.Button(update_frame, text="Update", bg='#BB8FCE', command=update_data)
update_button.grid(row=6, column=1, pady=10)
```

Figure 15: Updating Profile

```python
# Update Output Label & Result
update_output_label = tk.Label(update_frame, text='Update status:', font=("Imprint MT Shadow", 12), bg='#D2B4DE'))
update_output_label.grid(row=7, column=0, columnspan=2, pady=10)
update_result_label = tk.Label(update_frame, text="")
update_result_label.grid(row=8, column=0, columnspan=2)

root.mainloop()
```

Figure 16: Updating Profile

This is the code to create Finance page which allows users to select payment method and top-up their wallet. This code also shows where the looping happens by using if, else statements.

```python
import tkinter as tk
import mysql.connector

# Connect to your MySQL database
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="huh_cafe"
)

# Create a cursor object to execute SQL queries
mycursor = mydb.cursor()

# Function to handle the calculation and database saving
def collect_data():
    topup_amount_str = topup_amount_entry.get()
    topup_method = topup_method_var.get()

    # Validate topup_amount as an integer value
    try:
        topup_amount = int(topup_amount_str)
    except ValueError:
        output_label.config(text="Invalid top up amount. Please enter a valid amount.", bg='#F1948A', font=10)
        return

    # Check if the top-up amount is less than 20
    if topup_amount < 20:
        output_label.config(text="Minimum top up amount is 20. Please enter a valid amount.", bg='#F1948A', font=9)
```

Figure 17: Codes for Finance Page of Huh? Café System

```python
        topup_amount_entry.delete(0, tk.END)  # Clear the topup_amount_entry
    else:
        # Proceed with database insertion
        sql = "INSERT INTO `finance` (topup_amount, topup_method) VALUES (%s, %s)"
        val = (topup_amount, topup_method)
        mycursor.execute(sql, val)
        mydb.commit()

        # Clear entry fields after successful top-up
        topup_amount_entry.delete(0, tk.END)
        topup_method_var.set("Select Topup Method")

        # Update the output label
        output_label.config(text=f"You have successfully topped up RM{topup_amount} via {topup_method}.", bg='#82E0AA', font=10)

# Your Main window, You need to have the title, geometry (MUST)
root = tk.Tk()
root.title("FINANCE")
root.geometry('500x400')
root.config(bg='#D2B4DE')

# Page Title
label = tk.Label(root, text='FINANCE', font=("FZShuTi", 50, "bold"), bg='#D2B4DE')
label.pack(ipadx=5, ipady=5)

# Topup Method Dropdown (Label)
topup_method_label = tk.Label(root, text="Choose topup method:",font=("Cascadia Code", 15), bg='#D2B4DE')
topup_method_label.pack()
```

Figure 18: Looping for Finance Page

```python
# Topup Method Dropdown
topup_method_var = tk.StringVar(root)
topup_method_var.set("Select Topup Method")
topup_method_dropdown = tk.OptionMenu(root, topup_method_var, "Online Banking", "TouchNGo", "Debit / Credit Card")
topup_method_dropdown.pack(pady=10)
dropdown_menu = topup_method_dropdown.nametowidget(topup_method_dropdown.menuname)
dropdown_menu.configure(bg='#D2B4DE', activebackground='#D2B4DE')


# Topup Amount Entry. Label and user can insert data thru entry
topup_amount_label = tk.Label(root, text="Enter Amount:",font=("Cascadia Code", 15), bg='#D2B4DE')
topup_amount_label.pack()
topup_amount_entry = tk.Entry(root)
topup_amount_entry.pack()

# Save Button
save_button = tk.Button(root, text="Topup",bg='#BB8FCE', font=("FZShuTi", 13, "bold"), command=collect_data)
save_button.pack(pady=5)

# Output Label & result
label = tk.Label(root, text='Top-Up Status', font=("Cascadia Code", 13),bg='#D2B4DE')
label.pack(ipadx=10, ipady=5)
output_label = tk.Label(root, text="")
output_label.pack()

root.mainloop()
```

Figure 19: Codes for Top-up Methods

This is the code to create various menu selection and its prices. This code also shows the calculation of set of orders per its quantity.

menu.py 1 ×

C: > Users > ACER > Downloads > Telegram Desktop > menu.py > collect_data

```python
import tkinter as tk
import mysql.connector

# Connect MySQL database
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="huh_cafe"
)

mycursor = mydb.cursor()

# Coding and Calculation
def collect_data():
    # List to store selected stages
    selected_menu = []

    # Check which menu are selected
    if menu_type_var_Puteri_Favorite.get():
        selected_menu.append("Puteri's Favorite")
    if menu_type_var_Tasha_Favorite.get():
        selected_menu.append("Tasha's Favorite")
    if menu_type_var_Syaza_Favorite.get():
        selected_menu.append("Syaza's Favorite")
    if menu_type_var_Amalin_Favorite.get():
        selected_menu.append("Amalin's Favorite")
    if menu_type_var_Best_Seller.get():
        selected_menu.append("Best Seller")
```

Figure 20: Codes for Menu Page of Huh? Café System

```python
    # Dictionary to store quantities
    quantities = {
        "Puteri's Favorite": int(puteri_quantity_var.get()) if puteri_quantity_var.get() else 0,
        "Tasha's Favorite": int(tasha_quantity_var.get()) if tasha_quantity_var.get() else 0,
        "Syaza's Favorite": int(syaza_quantity_var.get()) if syaza_quantity_var.get() else 0,
        "Amalin's Favorite": int(amalin_quantity_var.get()) if amalin_quantity_var.get() else 0,
        "Best Seller": int(best_seller_quantity_var.get()) if best_seller_quantity_var.get() else 0,
    }

    name = str(name_entry.get())
    address = str(address_entry.get())
    phone_number_str = phone_number_entry.get()
    phone_number = int(phone_number_str) if phone_number_str else 0

    menu_type = selected_menu

    # The price
    price = {
        "Puteri's Favorite": 30,
        "Tasha's Favorite": 29,
        "Syaza's Favorite": 28,
        "Amalin's Favorite": 26,
        "Best Seller": 27,
    }

    # Calculate the total price.
    total_price = sum(price[menu_item] * quantities[menu_item] for menu_item in selected_menu)
```

Figure 21: Codes for Menu Page of Huh? Café System

```python
    # Calculate the total price.
    total_price = sum(price[menu_item] * quantities[menu_item] for menu_item in selected_menu)

    # To insert Data into the database
    sql = "INSERT INTO `menu` (name, address, phone_number, menu_type, quantity, price) VALUES (%s, %s, %s, %s, %s, %s)"
    val = (name, address, phone_number, ",".join(selected_menu), ",".join(str(quantities[menu_item]) for menu_item in selected_menu), total
    mycursor.execute(sql, val)
    mydb.commit()

    # To Print back the output.
    output_label.config(text=f"Total Price: RM{total_price}", bg='#D2B4DE', font=10)


# Main window.
root = tk.Tk()
root.title("HUH? CAFE")
root.geometry('1000x1000')
root.config(bg='#D2B4DE')

# Page Title
label = tk.Label(root, text='HUH? CAFE ORDERING SYSTEM', font=("FZShuTi", 30, "bold"), bg='#D2B4DE')
label.pack(ipadx=20, ipady=20)

# Prices List
prices_text = tk.Text(root, height=12, width=100, font=("Arial Black", 9), bg='#EBDEF0')
prices_text.pack(pady=20)

# The defined list
prices_text.insert(tk.END, "Menu & Prices:\n\n")
```

Figure 22: Codes to calculate total price per quantity

```
C: > Users > ACER > Downloads > Telegram Desktop > menu.py > collect_data
prices_text.insert(tk.END, "Puteri's Favorite: Creamy Carbonara Pasta, Cromboloni, Earl Grey Tea \nPrice: RM30\n\n")
prices_text.insert(tk.END, "Tasha's Favorite: Pesto Pasta, Croissant, Honey Lemon \nPrice: RM29\n\n")
prices_text.insert(tk.END, "Syaza's Favorite: Beef Bolognese Pasta, Pain au Chocolat, Iced Chocolate \nPrice: RM28\n\n")
prices_text.insert(tk.END, "Amalin's Favorite: Tomyum Thai Pasta, Egg Tart, Hazelnut Latte \nPrice: RM26\n\n")
prices_text.insert(tk.END, "Best Seller: Gigi Hadid Pasta, Choux Creme, Lychee Strawberry \nPrice: RM27\n\n")
prices_text.configure(state='disabled')

# User Information Frame
user_info_frame = tk.LabelFrame(root, text="User Information", bg='#EBDEF0')
user_info_frame.pack(side=tk.LEFT, padx=20, pady=10, fill="both", expand=True)

# User name entry
name_label = tk.Label(user_info_frame, text="Enter your name:", bg='#EBDEF0', font= 'Century')
name_entry = tk.Entry(user_info_frame)

# User address entry
address_label = tk.Label(user_info_frame, text="Enter your address:",bg='#EBDEF0',font= 'Century')
address_entry = tk.Entry(user_info_frame)

# User phone number entry
phone_number_label = tk.Label(user_info_frame, text="Enter your phone number:",bg='#EBDEF0',font= 'Century')
phone_number_entry = tk.Entry(user_info_frame)

# Organize widgets using grid
name_label.grid(row=0, column=0, sticky="w", pady=(0, 5))
name_entry.grid(row=0, column=1, pady=(0, 5))

address_label.grid(row=1, column=0, sticky="w", pady=(0, 5))
address_entry.grid(row=1, column=1, pady=(0, 5))
```

Figure 23: Menu page: Entering data

```python
phone_number_label.grid(row=2, column=0, sticky="w", pady=(0, 5))
phone_number_entry.grid(row=2, column=1, pady=(0, 5))

# Order Summary Frame
order_summary_frame = tk.LabelFrame(root, text="Order Summary", bg='#EBDEF0')
order_summary_frame.pack(side=tk.LEFT, padx=20, pady=10, fill="both", expand=True)

# Menu Type Checkboxes
menu_type_var_Puteri_Favorite = tk.BooleanVar(order_summary_frame)
menu_type_var_Tasha_Favorite = tk.BooleanVar(order_summary_frame)
menu_type_var_Syaza_Favorite = tk.BooleanVar(order_summary_frame)
menu_type_var_Amalin_Favorite = tk.BooleanVar(order_summary_frame)
menu_type_var_Best_Seller = tk.BooleanVar(order_summary_frame)

Puteri_Favorite_checkbox = tk.Checkbutton(order_summary_frame, text="Puteri's Favorite",bg='#EBDEF0',font= 'Century', variable=menu_type_va
Puteri_Favorite_checkbox.grid(row=0, column=0, sticky="w", pady=(0, 5))

# Quantity entry for Puteri's Favorite
puteri_quantity_label = tk.Label(order_summary_frame, text="Quantity:", bg='#EBDEF0', font=('Century', 9))
puteri_quantity_var = tk.Entry(order_summary_frame, font=('Century', 9))

puteri_quantity_label.grid(row=1, column=0, sticky="w", pady=(0, 5))
puteri_quantity_var.grid(row=1, column=1, pady=(0, 5))

Tasha_Favorite_checkbox = tk.Checkbutton(order_summary_frame, text="Tasha's Favorite",bg='#EBDEF0',font= 'Century', variable=menu_type_var_
Tasha_Favorite_checkbox.grid(row=2, column=0, sticky="w", pady=(0, 5))

# Quantity entry for Tasha's Favorite
tasha_quantity_label = tk.Label(order_summary_frame, text="Quantity:",bg='#EBDEF0',font= ('Century',9))
```

Figure 24: Codes to create frames and boxes in GUI

```python
tasha_quantity_var = tk.Entry(order_summary_frame, font=('Century',9))

tasha_quantity_label.grid(row=3, column=0, sticky="w", pady=(0, 5))
tasha_quantity_var.grid(row=3, column=1, pady=(0, 5))

Syaza_Favorite_checkbox = tk.Checkbutton(order_summary_frame, text="Syaza's Favorite",bg='#EBDEF0',font= 'Century', variable=menu_type_var_
Syaza_Favorite_checkbox.grid(row=4, column=0, sticky="w", pady=(0, 5))

# Quantity entry for Syaza's Favorite
syaza_quantity_label = tk.Label(order_summary_frame, text="Quantity:",bg='#EBDEF0',font= ('Century',9))
syaza_quantity_var = tk.Entry(order_summary_frame, font=('Century',9))

syaza_quantity_label.grid(row=5, column=0, sticky="w", pady=(0, 5))
syaza_quantity_var.grid(row=5, column=1, pady=(0, 5))

Amalin_Favorite_checkbox = tk.Checkbutton(order_summary_frame, text="Amalin's Favorite",bg='#EBDEF0',font= 'Century', variable=menu_type_va
Amalin_Favorite_checkbox.grid(row=6, column=0, sticky="w", pady=(0, 5))

# Quantity entry for Amalin's Favorite
amalin_quantity_label = tk.Label(order_summary_frame, text="Quantity:",bg='#EBDEF0',font= ('Century',9))
amalin_quantity_var = tk.Entry(order_summary_frame, font=('Century',9))

amalin_quantity_label.grid(row=7, column=0, sticky="w", pady=(0, 5))
amalin_quantity_var.grid(row=7, column=1, pady=(0, 5))

Best_Seller_checkbox = tk.Checkbutton(order_summary_frame, text="Best Seller",bg='#EBDEF0',font= 'Century', variable=menu_type_var_Best_Sel
Best_Seller_checkbox.grid(row=8, column=0, sticky="w", pady=(0, 5))
```

Figure 25: Codes for GUI design

```python
# Quantity entry for Best Seller
best_seller_quantity_label = tk.Label(order_summary_frame, text="Quantity:",bg='#EBDEF0',font= 'Century')
best_seller_quantity_var = tk.Entry(order_summary_frame)

best_seller_quantity_label.grid(row=9, column=0, sticky="w", pady=(0, 5))
best_seller_quantity_var.grid(row=9, column=1, pady=(0, 5))

# Save Button
save_button = tk.Button(root, text="Calculate", font=("FZShuTi", 12, "bold"), bg='#9B59B6',fg='#F9F9F7', command=collect_data)
save_button.pack(pady=10)

# Output Label & Result
total_price_label = tk.Label(root, text='Total Price:', font=("Imprint MT Shadow", 12), bg='#D2B4DE')
total_price_label.pack(ipadx=10, ipady=10)
output_label = tk.Label(root, text="")
output_label.pack()

root.mainloop()
```

Figure 26: Codes for GUI design

6.0 SNAPSHOT OF GRAPHICAL USER INTERFACE (GUI)

This is the main page of our café ordering system which allows user to choose to fill up their data.

Figure 27: GUI for Main Page

This is the GUI of Profile Page for users to fill their data. Users are also able to update and delete their data.

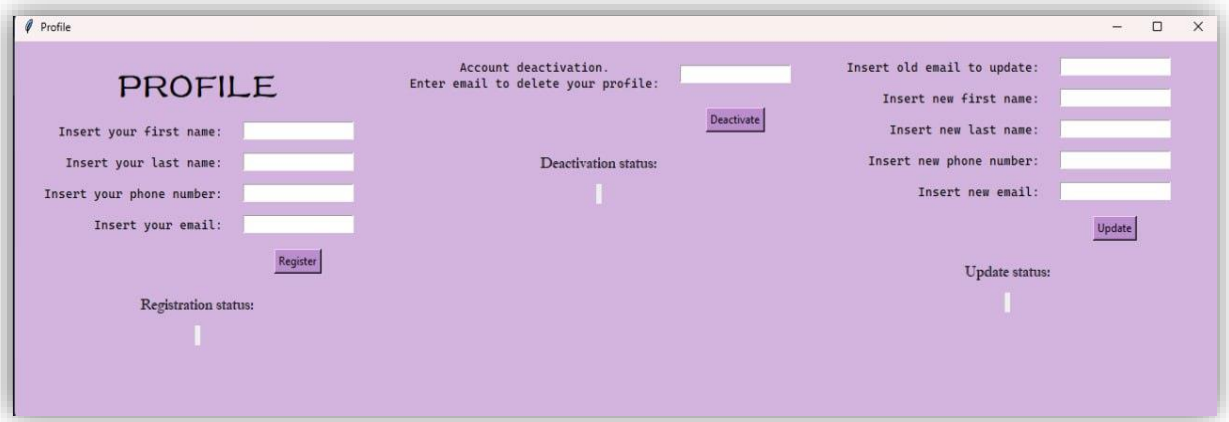


Figure 28: GUI for Profile Page

This is the GUI of Finance Page that allows users to choose their payment methods and enter their desired amount to top-up into their wallet as long as the amount is more than RM20. If users enter an amount that is less than RM20, the system will repeatedly ask users to enter amount until it meets the minimum amount required.

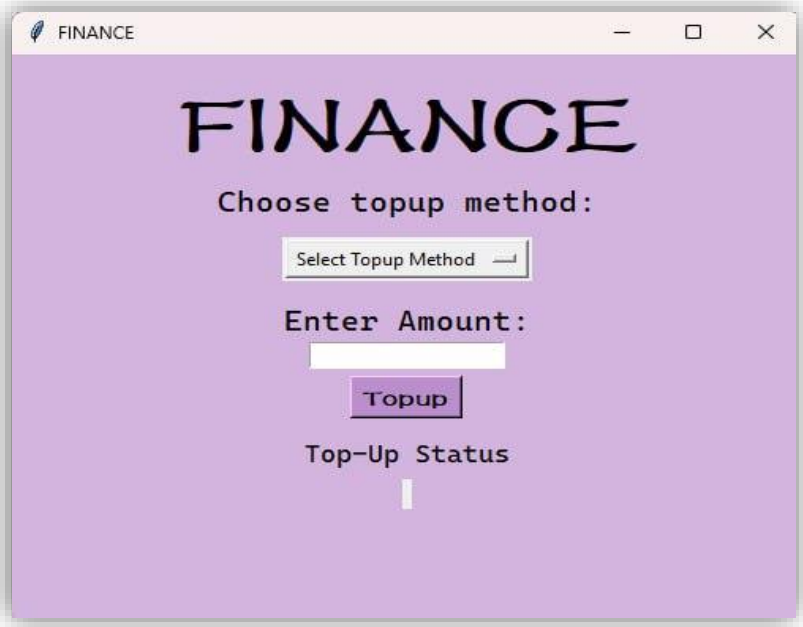


Figure 29: GUI for Finance Page

This is the Menu page that displays various set of menus for customers to choose and enter the order quantity to allow the system to calculate the order total. Users are also required to enter their names, addresses and phone numbers for delivery purposes.

Figure 30: GUI for Menu Selections and its quantity

7.0 SNAPSHOT OF DATABASE

This is the structure and the database of Profile submodule.

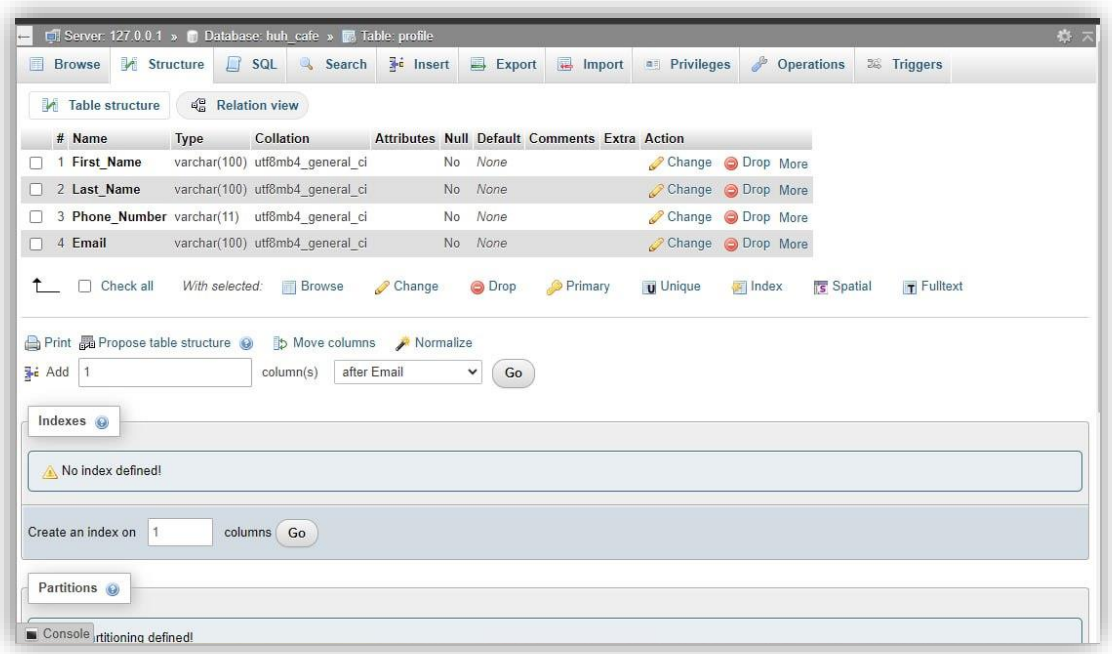


Figure 31: Structure

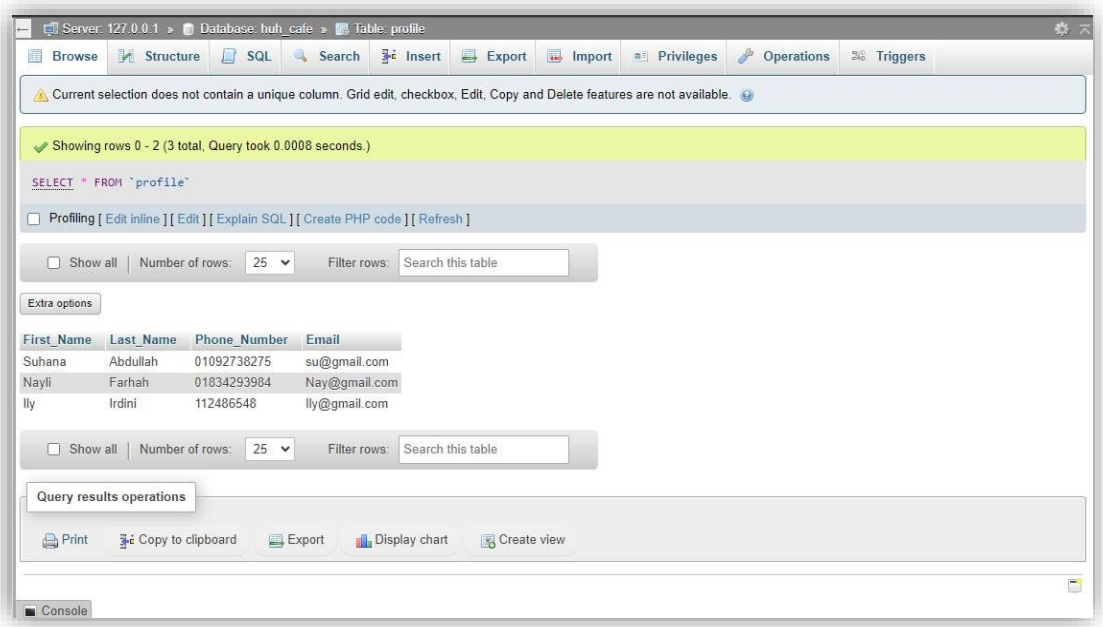


Figure 32: Customer's Data for Profile

This is the structure and database of Finance submodule.

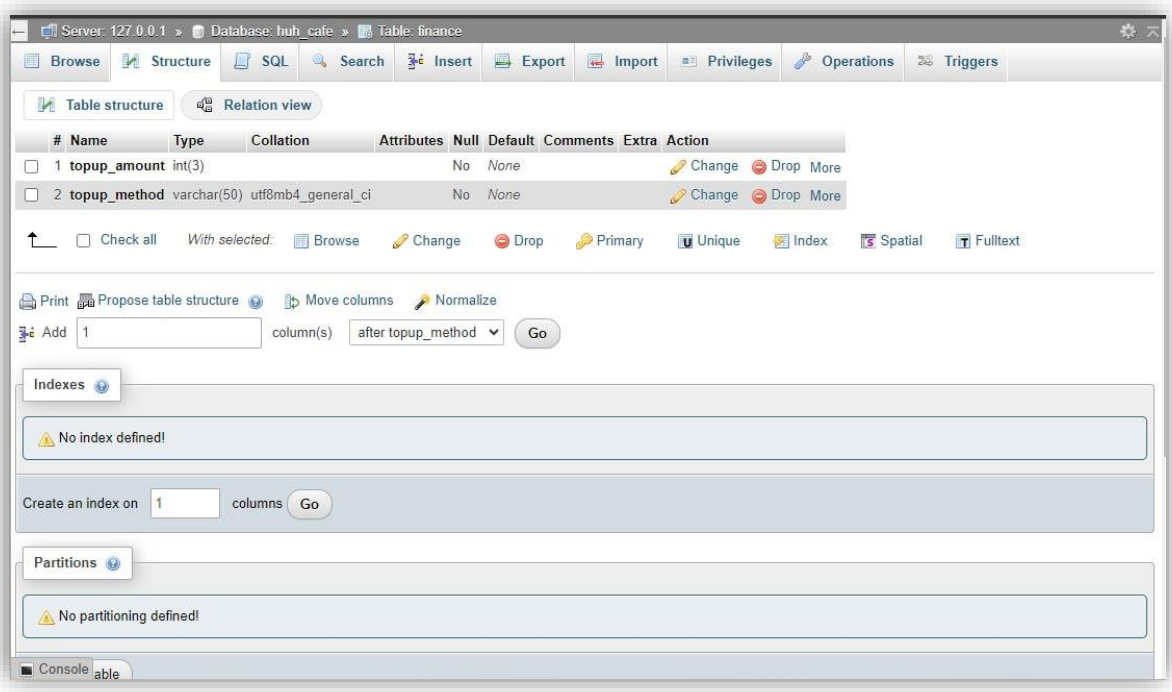


Figure 33: Structure

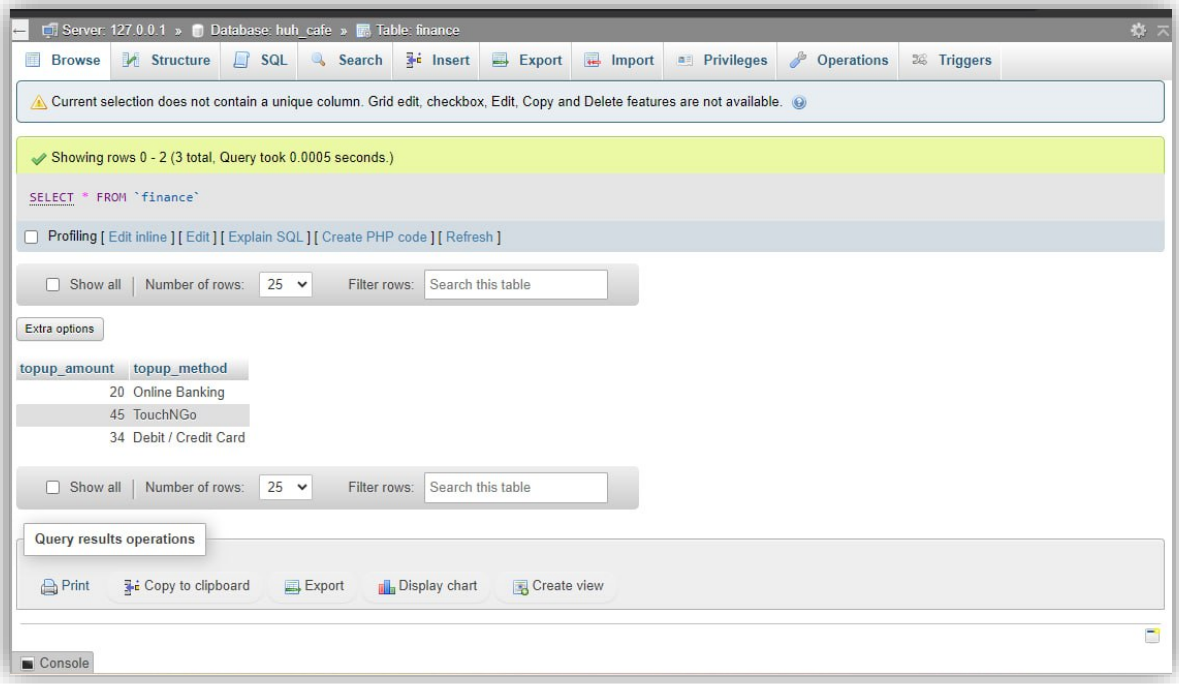


Figure 34: Customer's Data for Finance

This is the structure and database of Menu submodule.

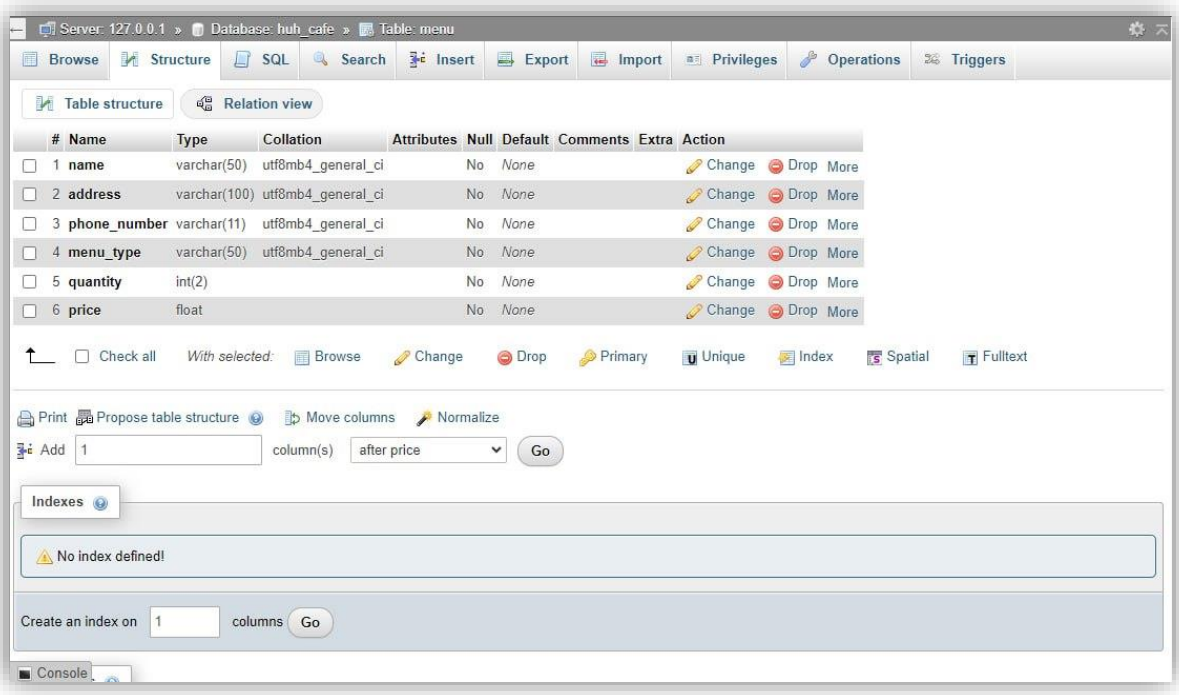


Figure 35: Structure

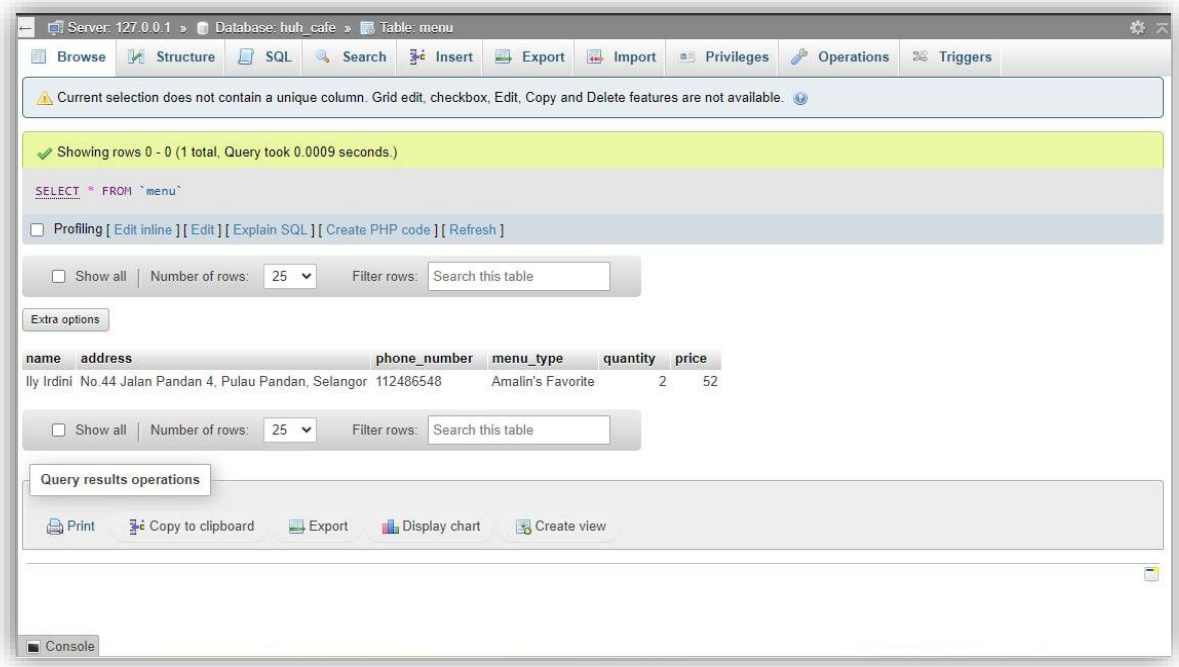


Figure 36: Customer's Data and their Menu Selection and its quantity

8.0 CONCLUSION

Completing this technical group project has made us come to conclusions and lead us to a deeper understanding on the importance of a flowchart to let us see the flow of the system works in a clearer vision. Things that we learned from executing codes in Python is that a lot of things need to be explored to perform and be able to do coding according to its multiple rules and formats. For instance, by exploring various types of coding methods and trying and error until we succeed to improve our coding skills instead of copying and pasting from other websites. Moreover, this group project had also made us learned how to create a database to ensure a platform for users to record and store their data through Graphical User Interface (GUI).

Apart from that, we also learned to create three submodules to relate them to our coding and generate the calculation results. Finally, we do think that programming is a complex subject to understand as it requires an abundance of technical skills and knowledge by self-exploring. We also learned the importance of teamwork and dividing tasks in succeeding this technical project according to each other's knowledge. However, we are glad that we got this opportunity and chance to learn this programming subject and understand the bright side of it which is to assist people to ease their burdens by simplifying and instantly providing people with solutions to their problems.

REFERENCES

GeeksforGeeks. (2023, July 14). *Purpose of database system in DBMS*. GeeksforGeeks. https://www.geeksforgeeks.org/purpose-of-database-system-in-dbms/

Shah, R. (2021, August 28). *What is interfaces in computer*. Bench Partner. https://benchpartner.com/what-is-interfaces-in-computer