Nutri Track – A Program that Helps Count Calories

Raunak Dani

LC5-13

ENGR 133

Professor Harrison-Smith

11/25/2024

1. Project Introduction

Currently, within America specifically, obesity is a very large and meaningful issue that needs to be paid attention to. Combatting obesity includes 2 different strategies – working out and eating right. Today, my project will help with the second solution – eating right. Therefore, the goal of my project is to create a nutrition tracker, which takes in a user's calorie and macronutrient goals, then takes in the food they have eaten during the day, and then outputs how much of each macronutrient they have consumed, and display graphs to pictorially represent their goals. This is all done by calling an API from the best food database that exists now – from MyFitness Pal.

This project is similar to my dream project idea, because my dream project idea was to create an app that exists on mobile devices, however, due to the limitations in my knowledge on the various coding languages needed for app development (JavaScript, CSS, HTML, etc.), I have just made a program that tracks the nutrition aspect. While these two projects are quite similar, the only limitation is the app development.

2. Project Overview of Inputs and Outputs

The first inputs that are required for this project are specific calorie and macronutrient (protein, carb, and fat) goals. These inputs are the baseline for what the user must reach by the end of the day. The second set of inputs for this project are the foods that the user ate that day. This list of foods is stored in an array, and then passed to the API, which then calculates and sends back the specific macronutrient amounts back to the program.

These combined inputs then help calculate the total amount of macronutrients eaten that day and help calculate the percentages on the charts that are output.

Some examples of inputs would be

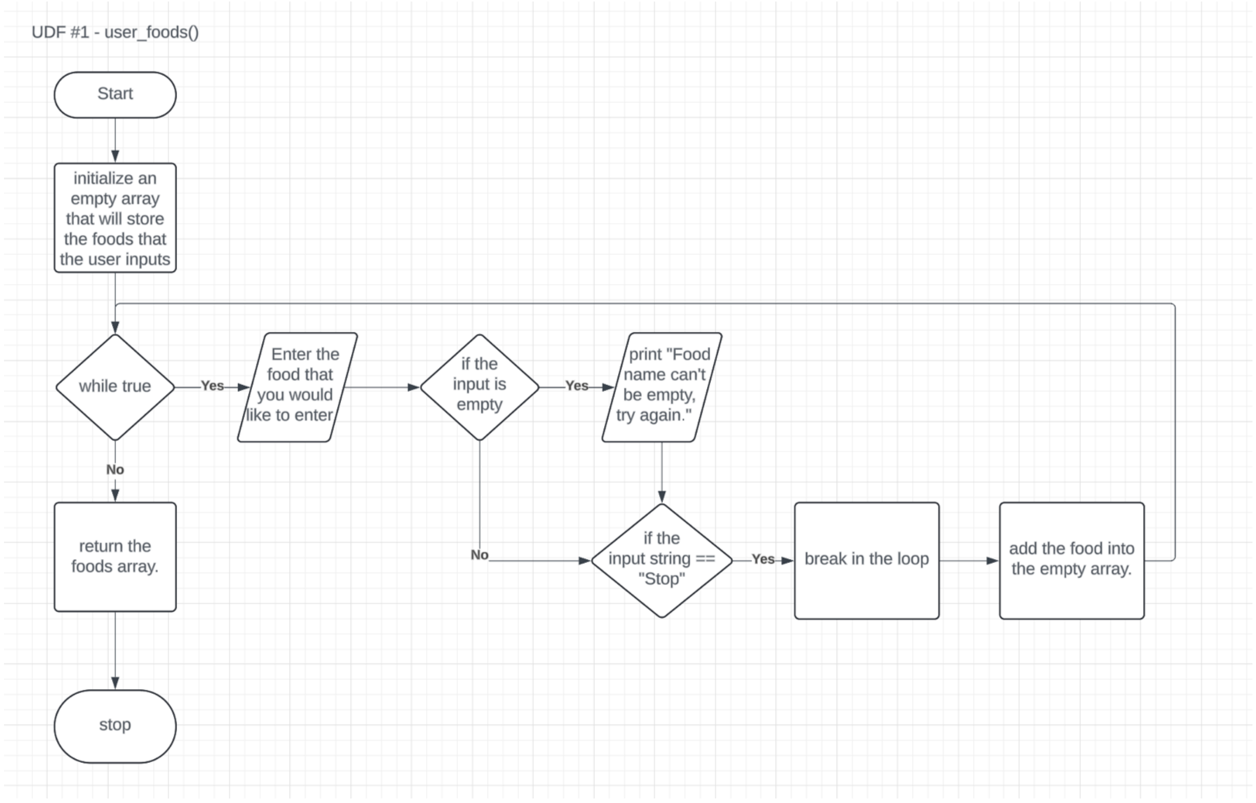
calories: 3000, protein: 200, carbs: 150, fats: 100

food: steak, food: rice, food: chicken, food: banana, food: STOP (this stops logging food and sends the list to the API).

The output would be the total calories and macronutrients consumed and a graph displaying it too.

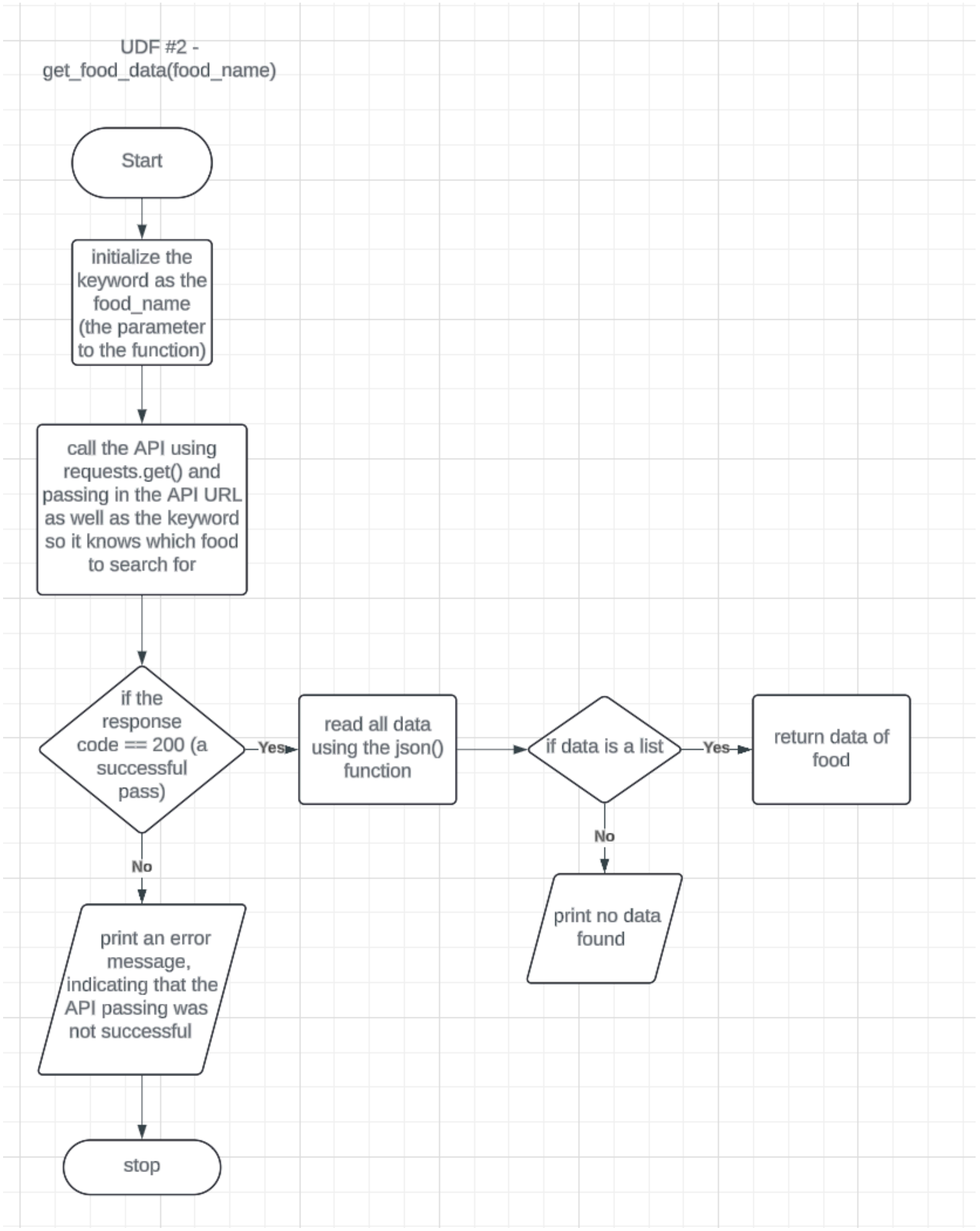
3. User-defined Functions

All UDFs are attached below where all code is attached.


UDF #1 - user_foods()
Start
initialize an empty array that will store the foods that the user inputs
while true
Yes
Enter the food that you would like to enter
if the input is empty
Yes
print "Food name can't be empty, try again."
No
No
if the input string == "Stop"
Yes
break in the loop
add the food into the empty array.
return the foods array.
stop

UDF #1 – user_foods()

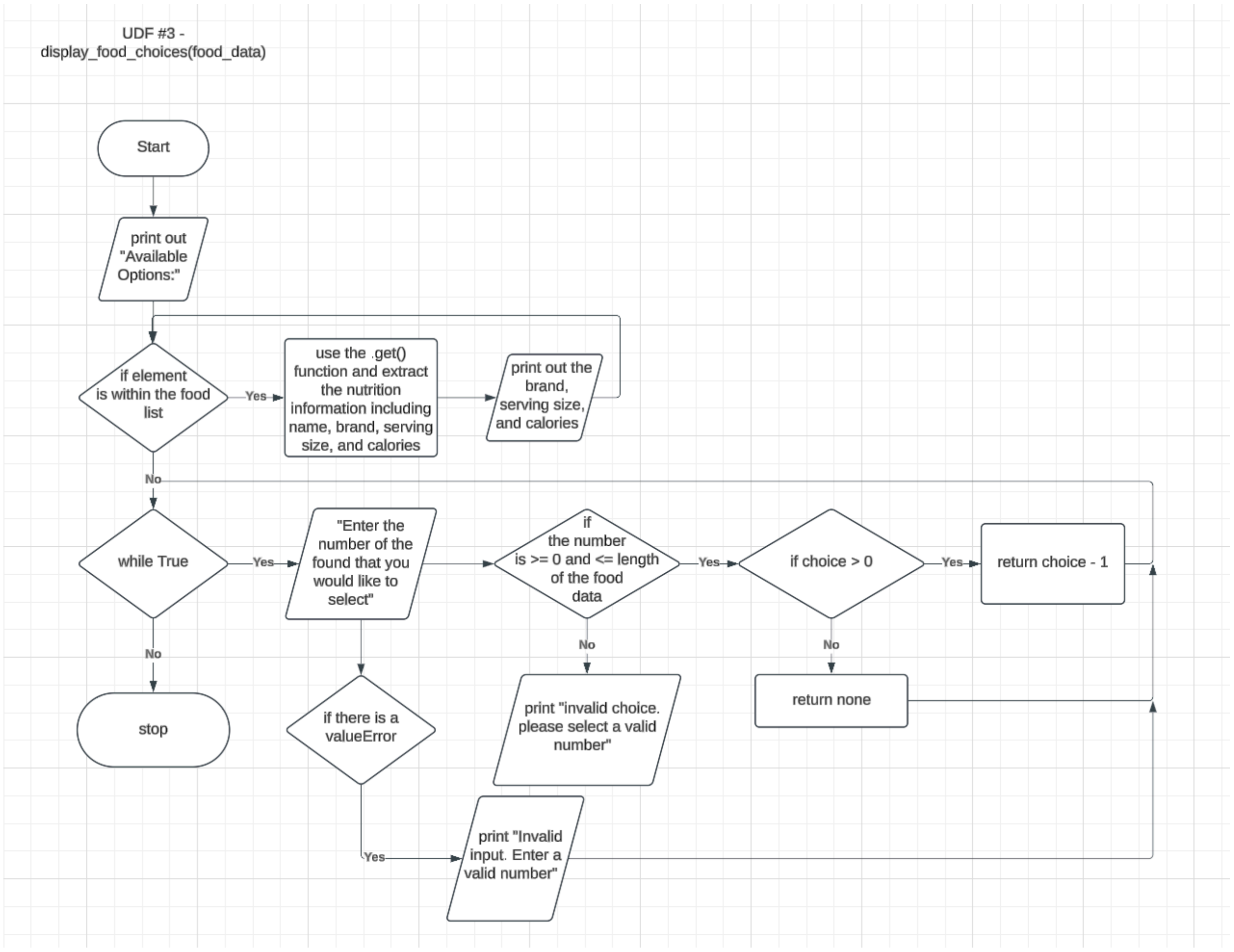
- The purpose of this UDF is to run a while loop to keep getting user input on what foods they would like to log. This while loop stops when ‘STOP’ is entered.
- This UDF has no inputs and returns a list of all the foods that were entered by the user as an output.
- There is no complex logic used in the UDF – it is just a while loop that keeps running as long as the entered food is not equal to ‘STOP’.


UDF #2 - get_food_data(food_name)
Start
initialize the keyword as the food_name (the parameter to the function)
call the API using requests.get() and passing in the API URL as well as the keyword so it knows which food to search for
if the response code == 200 (a successful pass)
Yes
read all data using the json() function
if data is a list
Yes
return data of food
No
print no data found
No
print an error message, indicating that the API passing was not successful
stop

UDF #2 – get_food_data(food_name)

- In main, each element of the list from user_foods() is parsed, and each food is individually passed into the get_food_data(food_name) UDF.
- The purpose of this UDF is to get all the macronutrients of the individual food that needs to be logged.

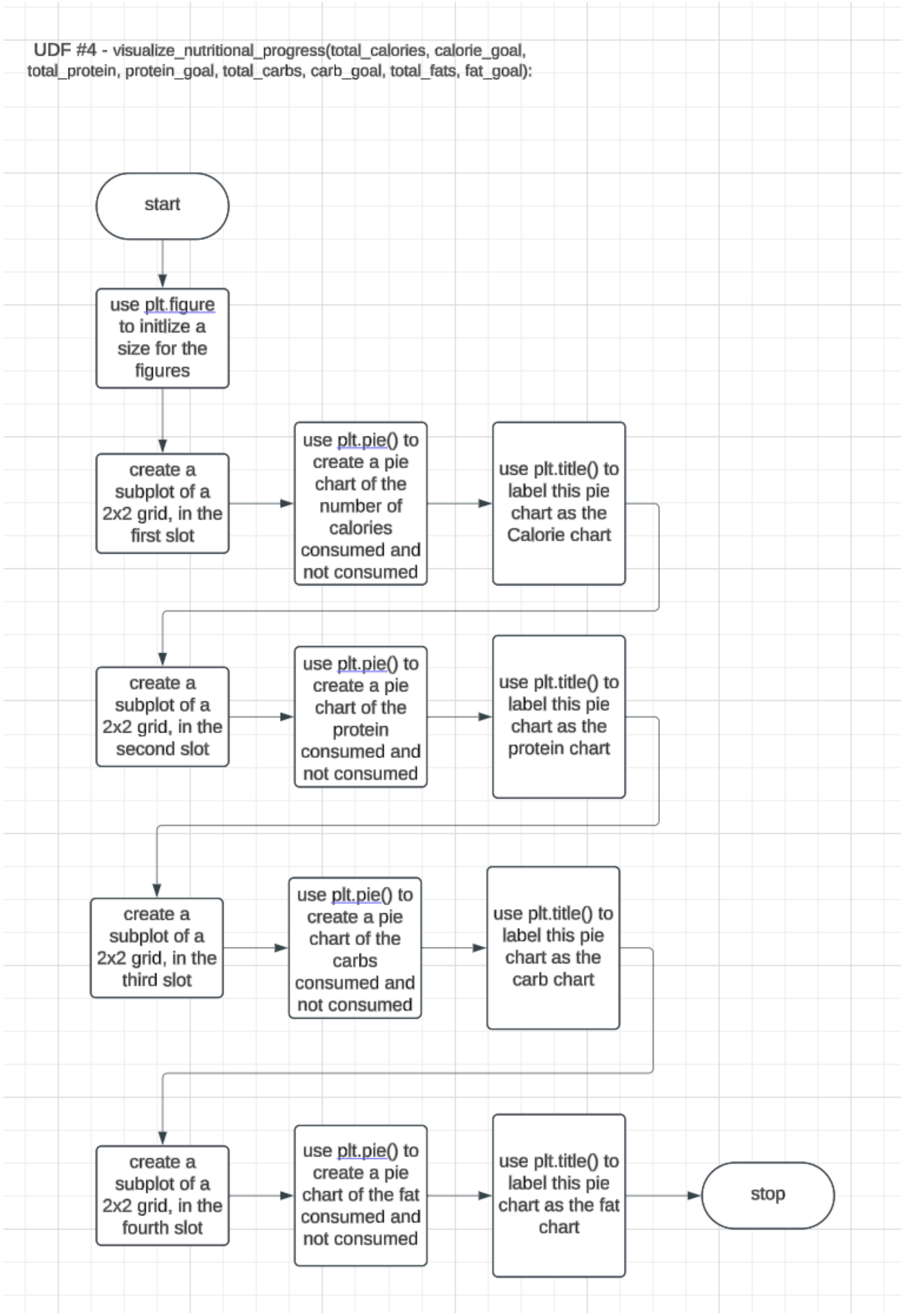
- This UDF has an input of the individual food (the method for getting this is detailed in the first bullet point under UDF #2) and has an output of a list which contains all the information about each macronutrient.
- The unique logic used in this function consists of calling the API to pass in the food name. I was able to learn this syntax directly from the link I got the API from (they had an example call of the API itself), so I was able to learn this syntax. Other than calling the API, there is no unique/complex logic in this UDF.


UDF #3 - display_food_choices(food_data)
Start
print out "Available Options:"
if element is within the food list
Yes
use the .get() function and extract the nutrition information including name, brand, serving size, and calories
print out the brand, serving size, and calories
No
while True
Yes
"Enter the number of the found that you would like to select"
if the number is >= 0 and <= length of the food data
Yes
if choice > 0
Yes
return choice - 1
No
return none
No
print "invalid choice. please select a valid number"
No
stop
if there is a valueError
Yes
print "Invalid input. Enter a valid number"

UDF #3 – display_food_choices(food_data)

- The purpose of this UDF is to get the individual data points for the macronutrients out of the list that is outputted out of get_food_data(food_name).
    - When the program calls the API, this API returns a list of the top 20 results of different foods, along with the corresponding serving sizes and calories
    - In this UDF, the user also picks which of the top 20 results the input matches
- The input for this function is the list of food_data that is outputted out of get_food_data(food_name) and the output for this function is to return a choice of which food the user would like to track.

- There is no unique/complex logic that is used in this function. This function runs a for loop to extract the values of every single element in the list food_data and then the function runs a while loop to get user input on which of the top 20 results the user wants.


UDF #4 - visualize_nutritional_progress(total_calories, calorie_goal, total_protein, protein_goal, total_carbs, carb_goal, total_fats, fat_goal):
start
use plt.figure to initlize a size for the figures
create a subplot of a 2x2 grid, in the first slot
use plt.pie() to create a pie chart of the number of calories consumed and not consumed
use plt.title() to label this pie chart as the Calorie chart
create a subplot of a 2x2 grid, in the second slot
use plt.pie() to create a pie chart of the protein consumed and not consumed
use plt.title() to label this pie chart as the protein chart
create a subplot of a 2x2 grid, in the third slot
use plt.pie() to create a pie chart of the carbs consumed and not consumed
use plt.title() to label this pie chart as the carb chart
create a subplot of a 2x2 grid, in the fourth slot
use plt.pie() to create a pie chart of the fat consumed and not consumed
use plt.title() to label this pie chart as the fat chart
stop
UDF #4 - visualize_nutritional_progress(total_calories, calorie_goal, total_protein, protein_goal, total_carbs, carb_goal, total_fats, fat_goal):

- The purpose of this function is to graph all of the calories, protein, carbs, and fats consumed in a day.
- The input for this function is detailed above (total_calories, calorie_goal, total_protein, protein_goal, total_carbs, carb_goal, total_fats, fat_goal) which are all values that have been calculated during the entirety of the program. The output of this function is 4 graphs that all display different amounts of macronutrients that are consumed.
- There is no unique/complex logic that is used in this function – it utilizes matplotlib.pyplot to allow for me to plot the graphs using the subplot command.

4. References

- *W3schools.com*. W3Schools Online Web Tutorials. (n.d.). https://www.w3schools.com/python/

- *Myfitnesspal*. API Hub - Free Public & Open Rest APIs. (n.d.).

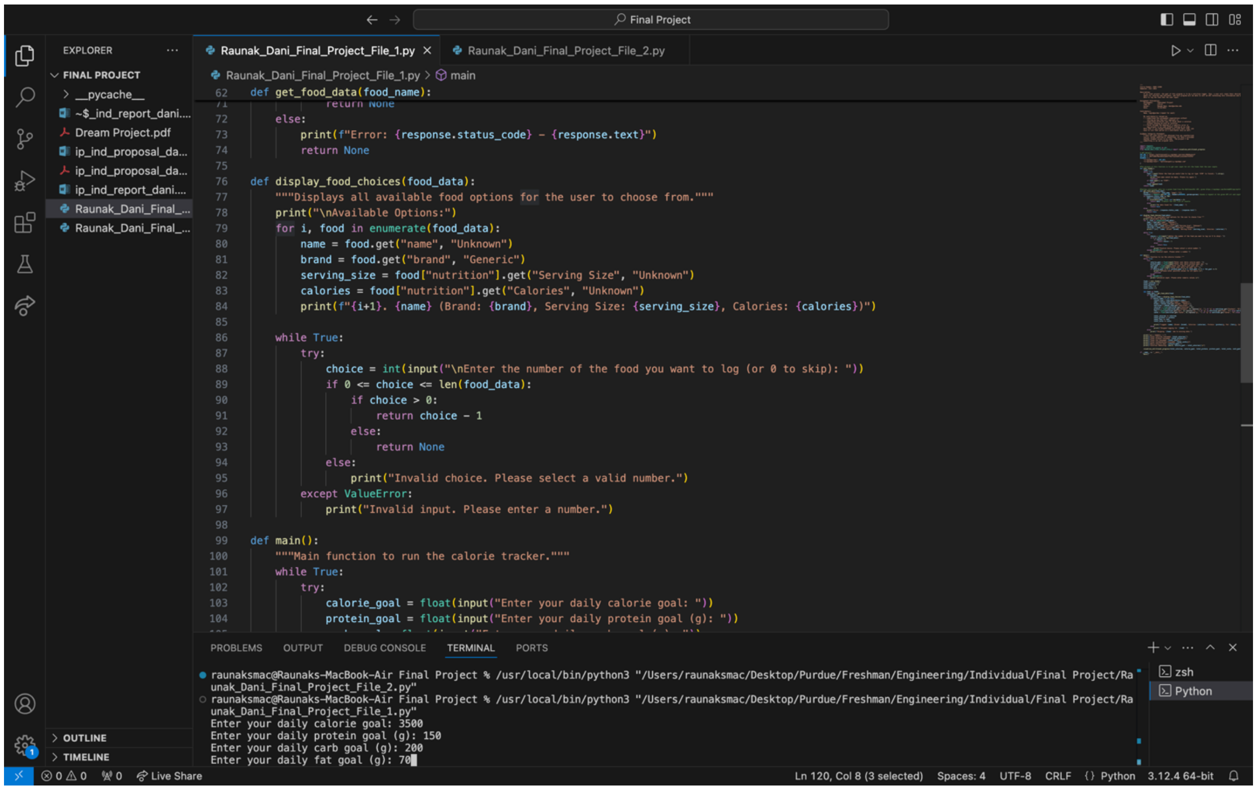
  https://rapidapi.com/UnitedAPI/api/myfitnesspal2/playground/apiendpoint_6366f956-a637-

  4915-bbf7-42243844259e

5. Appendix

1. User manual

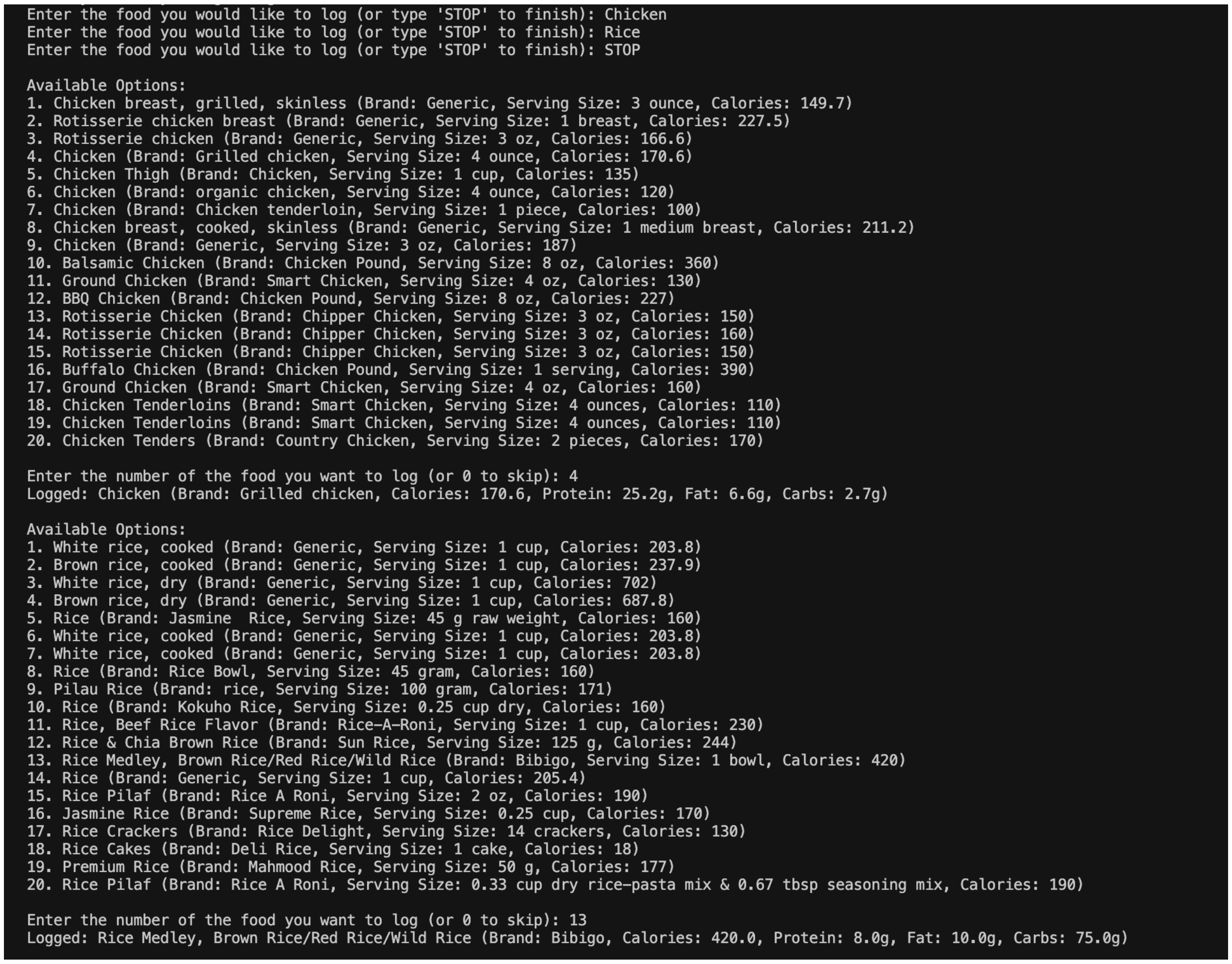
First, start by running the program, using the play button located in the top right corner of the VS Code page. After running the code, the terminal will prompt you to enter the desired calories, protein, carbs, and fats for the user as seen in the bottom of Figure 1.
Then, the user will input the food they consumed that day, for simplicity, I will pick the foods I ate to be chicken and rice (and I also enter 'STOP' to stop adding elements to the list). The result will be a list of the top 20 results that match the search. From here, the user will be prompted to select one of the top 20 choices to log their food. The user input for the food and the 20 results for each food can all be seen within Figure 2.
Finally, a summary is output at the end, along with a plot containing 4 graphs, pictorially displaying the consumption of macronutrients by the user as seen in Figures 3 and 4, respectively.



```python
def get_food_data(food_name):
            return None
    else:
        print(f"Error: {response.status_code} - {response.text}")
        return None

def display_food_choices(food_data):
    """Displays all available food options for the user to choose from."""
    print("\nAvailable Options:")
    for i, food in enumerate(food_data):
        name = food.get("name", "Unknown")
        brand = food.get("brand", "Generic")
        serving_size = food["nutrition"].get("Serving Size", "Unknown")
        calories = food["nutrition"].get("Calories", "Unknown")
        print(f"{i+1}. {name} (Brand: {brand}, Serving Size: {serving_size}, Calories: {calories})")

    while True:
        try:
            choice = int(input("\nEnter the number of the food you want to log (or 0 to skip): "))
            if 0 <= choice <= len(food_data):
                if choice > 0:
                    return choice - 1
                else:
                    return None
            else:
                print("Invalid choice. Please select a valid number.")
        except ValueError:
            print("Invalid input. Please enter a number.")

def main():
    """Main function to run the calorie tracker."""
    while True:
        try:
            calorie_goal = float(input("Enter your daily calorie goal: "))
            protein_goal = float(input("Enter your daily protein goal (g): "))
```

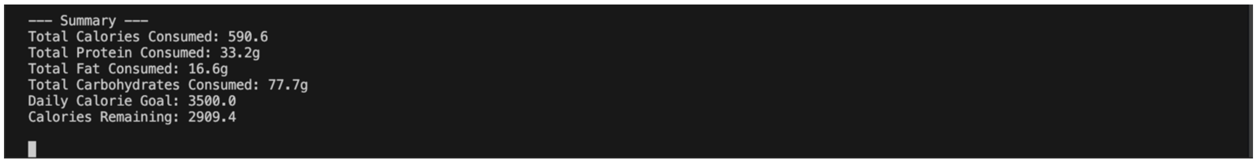
```
raunaksmac@Raunaks-MacBook-Air Final Project % /usr/local/bin/python3 "/Users/raunaksmac/Desktop/Purdue/Freshman/Engineering/Individual/Final Project/Ra
unak_Dani_Final_Project_File_2.py"
raunaksmac@Raunaks-MacBook-Air Final Project % /usr/local/bin/python3 "/Users/raunaksmac/Desktop/Purdue/Freshman/Engineering/Individual/Final Project/Ra
unak_Dani_Final_Project_File_1.py"
Enter your daily calorie goal: 3500
Enter your daily protein goal (g): 150
Enter your daily carb goal (g): 200
Enter your daily fat goal (g): 70
```

Figure 1: Initial user input for the daily calorie, protein, carb, and fat goals.



```
Enter the food you would like to log (or type 'STOP' to finish): Chicken
Enter the food you would like to log (or type 'STOP' to finish): Rice
Enter the food you would like to log (or type 'STOP' to finish): STOP

Available Options:
1. Chicken breast, grilled, skinless (Brand: Generic, Serving Size: 3 ounce, Calories: 149.7)
2. Rotisserie chicken breast (Brand: Generic, Serving Size: 1 breast, Calories: 227.5)
3. Rotisserie chicken (Brand: Generic, Serving Size: 3 oz, Calories: 166.6)
4. Chicken (Brand: Grilled chicken, Serving Size: 4 ounce, Calories: 170.6)
5. Chicken Thigh (Brand: Chicken, Serving Size: 1 cup, Calories: 135)
6. Chicken (Brand: organic chicken, Serving Size: 4 ounce, Calories: 120)
7. Chicken (Brand: Chicken tenderloin, Serving Size: 1 piece, Calories: 100)
8. Chicken breast, cooked, skinless (Brand: Generic, Serving Size: 1 medium breast, Calories: 211.2)
9. Chicken (Brand: Generic, Serving Size: 3 oz, Calories: 187)
10. Balsamic Chicken (Brand: Chicken Pound, Serving Size: 8 oz, Calories: 360)
11. Ground Chicken (Brand: Smart Chicken, Serving Size: 4 oz, Calories: 130)
12. BBQ Chicken (Brand: Chicken Pound, Serving Size: 8 oz, Calories: 227)
13. Rotisserie Chicken (Brand: Chipper Chicken, Serving Size: 3 oz, Calories: 150)
14. Rotisserie Chicken (Brand: Chipper Chicken, Serving Size: 3 oz, Calories: 160)
15. Rotisserie Chicken (Brand: Chipper Chicken, Serving Size: 3 oz, Calories: 150)
16. Buffalo Chicken (Brand: Chicken Pound, Serving Size: 1 serving, Calories: 390)
17. Ground Chicken (Brand: Smart Chicken, Serving Size: 4 oz, Calories: 160)
18. Chicken Tenderloins (Brand: Smart Chicken, Serving Size: 4 ounces, Calories: 110)
19. Chicken Tenderloins (Brand: Smart Chicken, Serving Size: 4 ounces, Calories: 110)
20. Chicken Tenders (Brand: Country Chicken, Serving Size: 2 pieces, Calories: 170)

Enter the number of the food you want to log (or 0 to skip): 4
Logged: Chicken (Brand: Grilled chicken, Calories: 170.6, Protein: 25.2g, Fat: 6.6g, Carbs: 2.7g)

Available Options:
1. White rice, cooked (Brand: Generic, Serving Size: 1 cup, Calories: 203.8)
2. Brown rice, cooked (Brand: Generic, Serving Size: 1 cup, Calories: 237.9)
3. White rice, dry (Brand: Generic, Serving Size: 1 cup, Calories: 702)
4. Brown rice, dry (Brand: Generic, Serving Size: 1 cup, Calories: 687.8)
5. Rice (Brand: Jasmine  Rice, Serving Size: 45 g raw weight, Calories: 160)
6. White rice, cooked (Brand: Generic, Serving Size: 1 cup, Calories: 203.8)
7. White rice, cooked (Brand: Generic, Serving Size: 1 cup, Calories: 203.8)
8. Rice (Brand: Rice Bowl, Serving Size: 45 gram, Calories: 160)
9. Pilau Rice (Brand: rice, Serving Size: 100 gram, Calories: 171)
10. Rice (Brand: Kokuho Rice, Serving Size: 0.25 cup dry, Calories: 160)
11. Rice, Beef Rice Flavor (Brand: Rice-A-Roni, Serving Size: 1 cup, Calories: 230)
12. Rice & Chia Brown Rice (Brand: Sun Rice, Serving Size: 125 g, Calories: 244)
13. Rice Medley, Brown Rice/Red Rice/Wild Rice (Brand: Bibigo, Serving Size: 1 bowl, Calories: 420)
14. Rice (Brand: Generic, Serving Size: 1 cup, Calories: 205.4)
15. Rice Pilaf (Brand: Rice A Roni, Serving Size: 2 oz, Calories: 190)
16. Jasmine Rice (Brand: Supreme Rice, Serving Size: 0.25 cup, Calories: 170)
17. Rice Crackers (Brand: Rice Delight, Serving Size: 14 crackers, Calories: 130)
18. Rice Cakes (Brand: Deli Rice, Serving Size: 1 cake, Calories: 18)
19. Premium Rice (Brand: Mahmood Rice, Serving Size: 50 g, Calories: 177)
20. Rice Pilaf (Brand: Rice A Roni, Serving Size: 0.33 cup dry rice-pasta mix & 0.67 tbsp seasoning mix, Calories: 190)

Enter the number of the food you want to log (or 0 to skip): 13
Logged: Rice Medley, Brown Rice/Red Rice/Wild Rice (Brand: Bibigo, Calories: 420.0, Protein: 8.0g, Fat: 10.0g, Carbs: 75.0g)
```

Figure 2: The top 20 results of the chicken and rice are output. From here, the user selects the specific chicken and rice recipes that match what they ate (in this case I picked recipe 4 for chicken and recipe 13 for the rice).



```
--- Summary ---
Total Calories Consumed: 590.6
Total Protein Consumed: 33.2g
Total Fat Consumed: 16.6g
Total Carbohydrates Consumed: 77.7g
Daily Calorie Goal: 3500.0
Calories Remaining: 2909.4
```

Figure 3: The summary of the total macronutrient that are eaten on that specific day.

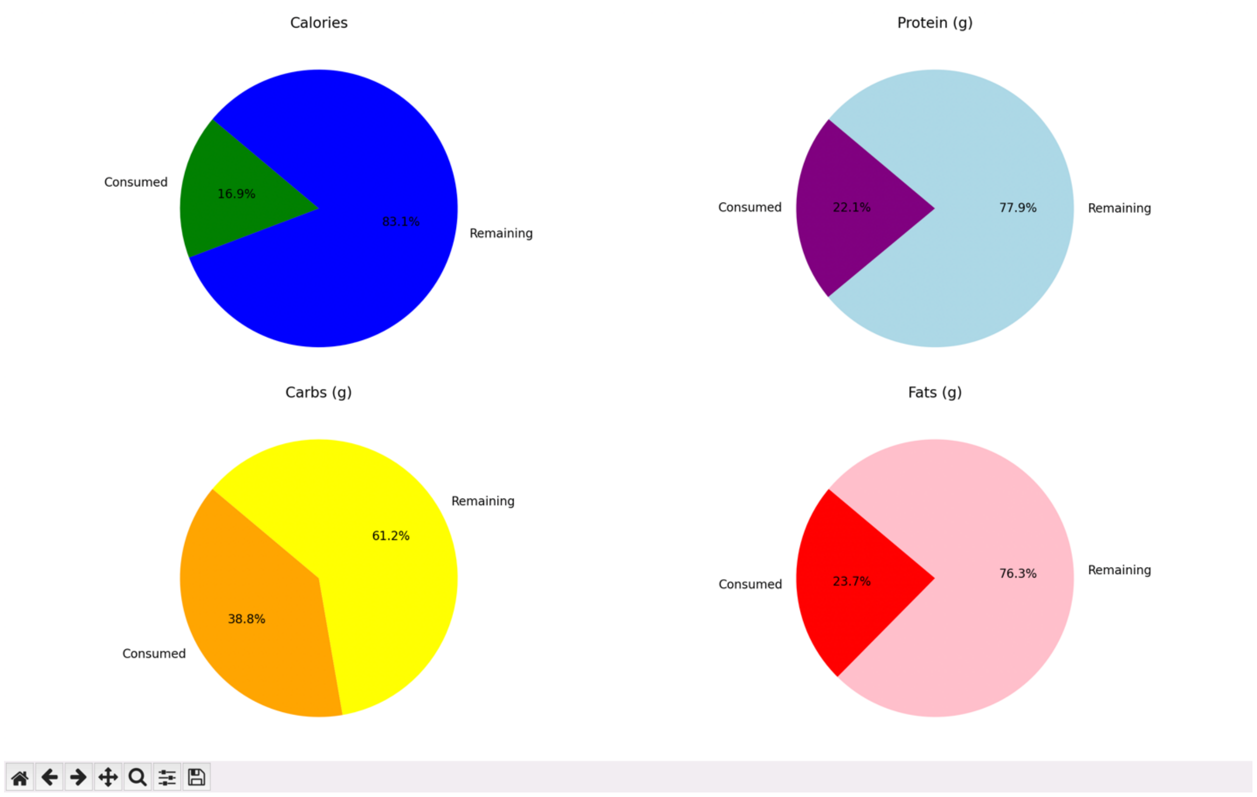
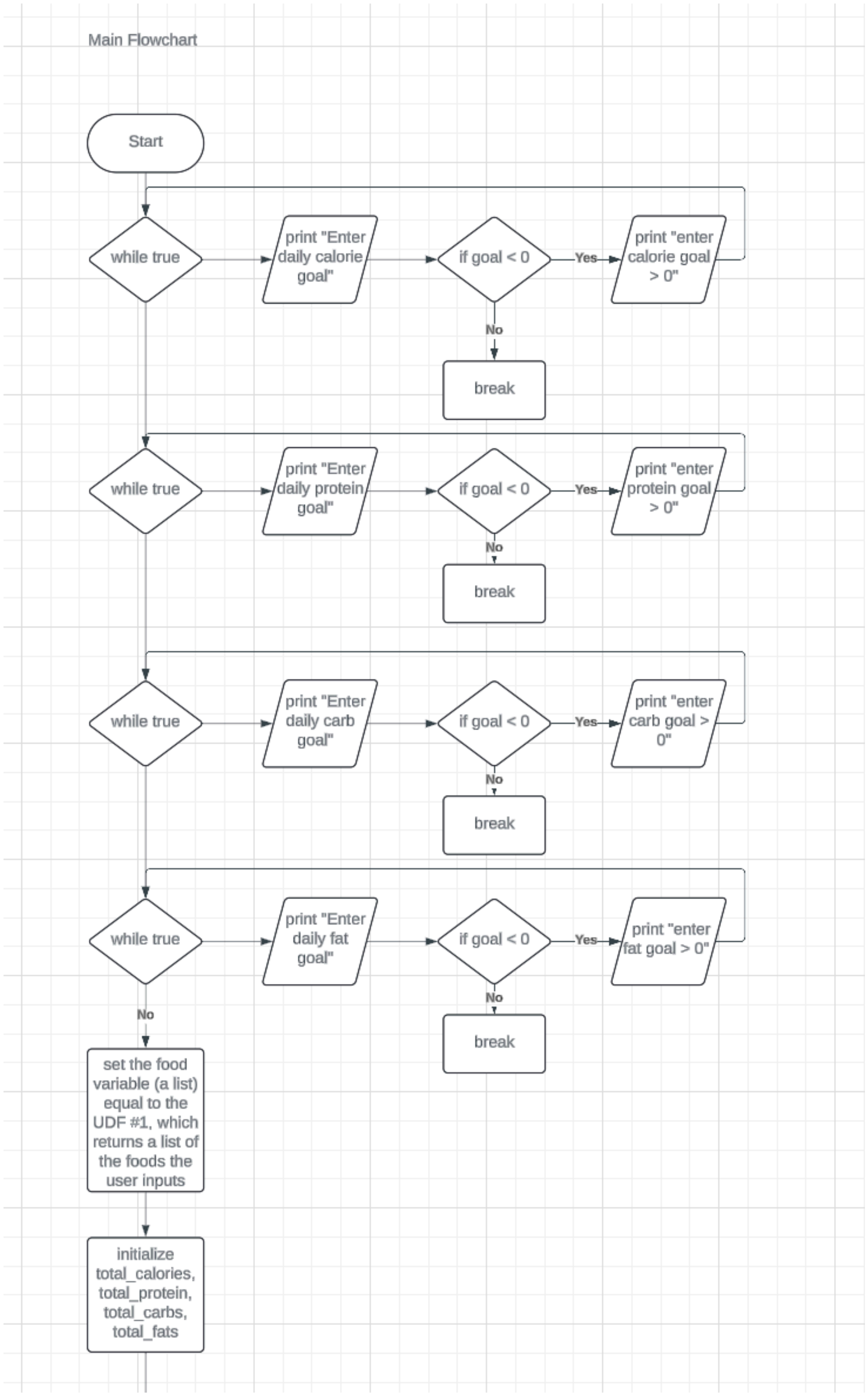
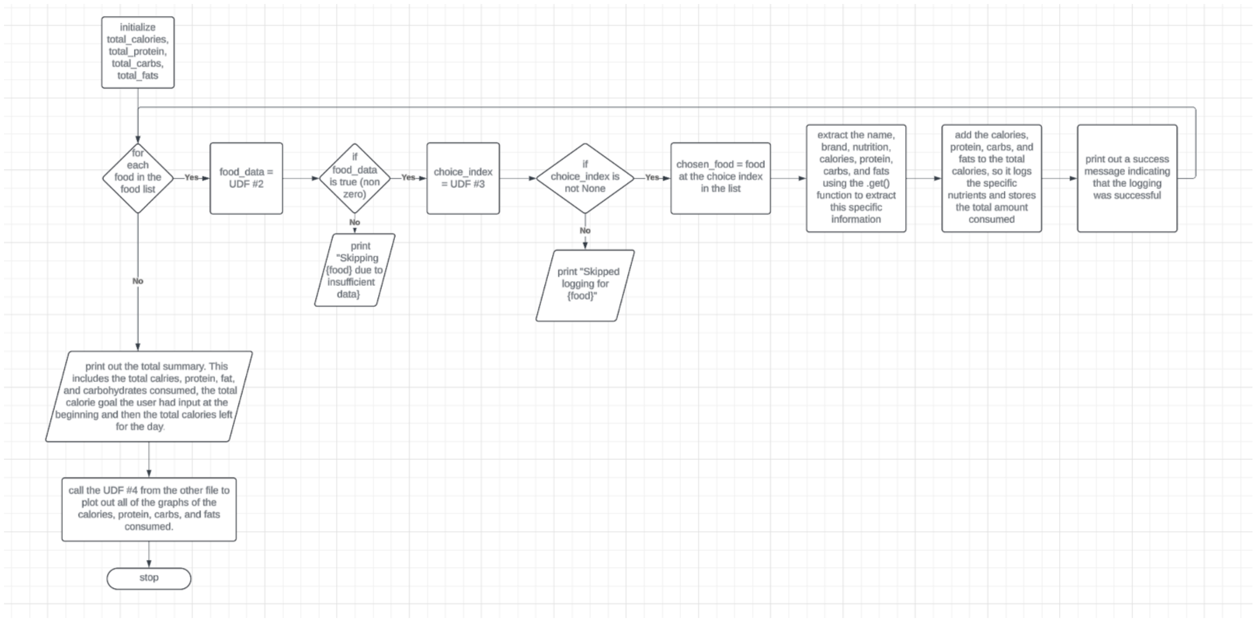


Figure 4: This is the visual representation of the specific macronutrients that are consumed by the user from the foods that have been logged.

2. Flowchart


Main Flowchart
Start
while true
print "Enter daily calorie goal"
if goal < 0
Yes
print "enter calorie goal > 0"
No
break
while true
print "Enter daily protein goal"
if goal < 0
Yes
print "enter protein goal > 0"
No
break
while true
print "Enter daily carb goal"
if goal < 0
Yes
print "enter carb goal > 0"
No
break
while true
print "Enter daily fat goal"
if goal < 0
Yes
print "enter fat goal > 0"
No
break
No
set the food variable (a list) equal to the UDF #1, which returns a list of the foods the user inputs
initialize total_calories, total_protein, total_carbs, total_fats


initialize total_calories, total_protein, total_carbs, total_fats
for each food in the food list
Yes
food_data = UDF #2
if food_data is true (non zero)
Yes
choice_index = UDF #3
if choice_index is not None
Yes
chosen_food = food at the choice index in the list
extract the name, brand, nutrition, calories, protein, carbs, and fats using the .get() function to extract this specific information
add the calories, protein, carbs, and fats to the total calories, so it logs the specific nutrients and stores the total amount consumed
print out a success message indicating that the logging was successful
No
print "Skipping {food} due to insufficient data}
No
print "Skipped logging for {food}"
No
print out the total summary. This includes the total calries, protein, fat, and carbohydrates consumed, the total calorie goal the user had input at the beginning and then the total calories left for the day.
call the UDF #4 from the other file to plot out all of the graphs of the calories, protein, carbs, and fats consumed.
stop

3. Code

File 1: Raunak_Dani_Final_Project_File_1.py

```
"""
Course Number: ENGR 13300
Semester: Fall 2024

Description:
    As my final project, the goal of this program is to be a nutrition logger. Thus, a
user will input their desired
    goals for calories and nutrients, and this program will be able to call an API
containing this information and be
    able to log the food that the user input.

Assignment Information:
    Assignment:     Individual Project
    Team ID:        LC5-13
    Author:         Raunak Dani, dani@purdue.edu
    Date:           11/19/2024

Contributors:
    Name, login@purdue [repeat for each]

    My contributor(s) helped me:
    [ ] understand the assignment expectations without
        telling me how they will approach it.
    [ ] understand different ways to think about a solution
        without helping me plan my solution.
    [ ] think through the meaning of a specific error or
        bug present in my code without looking at my code.
    Note that if you helped somebody else with their code, you
    have to list that person as a contributor here as well.

Academic Integrity Statement:
    I have not used source code obtained from any unauthorized
    source, either modified or unmodified; nor have I provided
    another student access to my code.  The project I am
    submitting is my own original work.
"""

import requests
from Raunak_Dani_Final_Project_File_2 import visualize_nutritional_progress

# API Details given from MyFitness Pal, given
https://rapidapi.com/UnitedAPI/api/myfitnesspal2/playground/apiendpoint_6366f956-a637-
4915-bbf7-42243844259e
API_URL = "https://myfitnesspal2.p.rapidapi.com/searchByKeyword"
API_KEY = "6cadefb81cmshabc116616c3cbd8p1a5715jsn1648a0f8d1e6"
HEADERS = {
    "x-rapidapi-key": API_KEY,
    "x-rapidapi-host": "myfitnesspal2.p.rapidapi.com"
}

#The purpose of this function is to get user input for all the foods that the user
inputs
def user_foods():
    foods = []
    while True:
        food = input("Enter the food you would like to log (or type 'STOP' to finish):
").strip()
        if not food:
            print("Food name cannot be empty. Please try again.")
            continue
        if food.upper() == "STOP":
            break
        foods.append(food)
    return foods

#Fetches nutritional data for a given food from the MyFitnessPal API
def get_food_data(food_name):
    #establishes the params and the response as the MyFitnessPal API code segment
demonstrated
    params = {"keyword": food_name, "page": "1"}
    response = requests.get(API_URL, headers=HEADERS, params=params) #sends a request
at the given API url and inputs the parameters
    if response.status_code == 200: #indicates that the request was a success
        data = response.json() #reads all data
        if isinstance(data, list) and len(data) > 0:
            return data  # return the full list of results if the data is a list and
is a length > 0
        else:
            print(f"No data found for '{food_name}'.")
            return None
    else: #this is an error code of 400
        print(f"Error: {response.status_code} - {response.text}")
        return None

#displays all available food options for the user to choose from.
def display_food_choices(food_data):
    print("\nAvailable Options:")
    #this following segment is responsible for extracting the speicifc information
based off of the list
    for i, food in enumerate(food_data):
        name = food.get("name", "Unknown")
        brand = food.get("brand", "Generic")
        serving_size = food["nutrition"].get("Serving Size", "Unknown") #this is a
subcategory as seen on the website
        calories = food["nutrition"].get("Calories", "Unknown")
        print(f"{i+1}. {name} (Brand: {brand}, Serving Size: {serving_size}, Calories:
{calories})")

    #this code is responsible for selecting the specific output (1-20) which is output
from the API
    while True:
        try:
            choice = int(input("\nEnter the number of the food you want to log (or 0
to skip): "))
            if 0 <= choice <= len(food_data):
                if choice > 0:
                    return choice - 1
                else:
                    return None
            else:
                print("Invalid choice. Please select a valid number.")
        except ValueError:
            print("Invalid input. Please enter a number.")

def main():
    #gets the daily goals for each macronutrient and has input validation
    while True:
        calorie_goal = float(input("Enter your daily calorie goal: "))
        if calorie_goal <= 0:
            print("\nEnter a calorie goal greater than 0\n")
            continue
        else:
            break

    while True:
        protein_goal = float(input("Enter your daily protein goal (g): "))
        if protein_goal <= 0:
            print("\nEnter a protein goal greater than 0\n")
            continue
        else:
            break

    while True:
        carb_goal = float(input("Enter your daily carbohydrate goal (g): "))
        if carb_goal <= 0:
            print("\nEnter a carb goal greater than 0\n")
            continue
        else:
            break

    while True:
        fat_goal = float(input("Enter your daily fat goal (g): "))
        if fat_goal <= 0:
            print("\nEnter a fat goal greater than 0\n")
            continue
        else:
            break

    #intializes the starting values of the total calories consumed
    foods = user_foods()
    total_calories = 0
    total_protein = 0
    total_fats = 0
    total_carbs = 0

    for food in foods: #iterates through each element within the user input list
        food_data = get_food_data(food)
        if food_data:
            choice_index = display_food_choices(food_data)
            if choice_index is not None: #takes out every single one of the 20
elements and is able to extract the name, brand, nutrition, calories, protein, fat,
and carbs, where these values are assigned to variables
```

```
                chosen_food = food_data[choice_index]
                name = chosen_food.get("name", "Unknown")
                brand = chosen_food.get("brand", "Generic")
                nutrition = chosen_food.get("nutrition", {})
                calories = float(nutrition.get("Calories", 0))
                protein = float(nutrition.get("Protein", 0).replace("g", "") if "g" in
nutrition.get("Protein", "0") else nutrition.get("Protein", 0))
                fats = float(nutrition.get("Fat", 0).replace("g", "") if "g" in
nutrition.get("Fat", "0") else nutrition.get("Fat", 0))
                carbs = float(nutrition.get("Carbs", 0).replace("g", "") if "g" in
nutrition.get("Carbs", "0") else nutrition.get("Carbs", 0))
                #these statements above remove the word 'grams' represented as 'g'
into a blank space to extract the values, replaces values w/ 0 if not present
                total_calories += calories #adds the corresponding macronutrients to
their previous values
                total_protein += protein
                total_fats += fats
                total_carbs += carbs

                print(f"Logged: {name} (Brand: {brand}, Calories: {calories}, Protein:
{protein}g, Fat: {fats}g, Carbs: {carbs}g)")
            else:
                print(f"Skipped logging for '{food}'.")
        else:
            print(f"Skipping '{food}' due to missing data.")

    print("\n--- Summary ---")
    print(f"Total Calories Consumed: {total_calories}")
    print(f"Total Protein Consumed: {total_protein}g")
    print(f"Total Fat Consumed: {total_fats}g")
    print(f"Total Carbohydrates Consumed: {total_carbs}g")
    print(f"Daily Calorie Goal: {calorie_goal}")
    print(f"Calories Remaining: {max(0, calorie_goal - total_calories)}\n")
    #calling the graphing function
    visualize_nutritional_progress(total_calories, calorie_goal, total_protein,
protein_goal, total_carbs, carb_goal, total_fats, fat_goal)

if __name__ == "__main__":
    main()
```

File 2: Raunak_Dani_Final_Project_File_2.py

```
"""
Course Number: ENGR 13300
Semester: Fall 2024

Description: The purpose of this UDF is to plot all of the various macronutrients that
are consumed.
This function creates a total of 4 graphs, plotting the calories, protein, carbs, and
fats consumed.

Assignment Information:
    Assignment:     Individual Project
    Team ID:        LC5-13
    Author:         Raunak Dani, dani@purdue.edu
    Date:           11/19/2024

Contributors:
    Name, login@purdue [repeat for each]

    My contributor(s) helped me:
    [ ] understand the assignment expectations without
        telling me how they will approach it.
    [ ] understand different ways to think about a solution
        without helping me plan my solution.
    [ ] think through the meaning of a specific error or
        bug present in my code without looking at my code.
    Note that if you helped somebody else with their code, you
    have to list that person as a contributor here as well.

Academic Integrity Statement:
    I have not used source code obtained from any unauthorized
    source, either modified or unmodified; nor have I provided
    another student access to my code.  The project I am
    submitting is my own original work.
"""

import matplotlib.pyplot as plt

def visualize_nutritional_progress(total_calories, calorie_goal, total_protein,
protein_goal, total_carbs, carb_goal, total_fats, fat_goal):
    """Visualizes calories, protein, carbs, and fats consumption vs goals using pie
charts."""
    plt.figure(figsize=(12, 12))

    # Calories
    plt.subplot(2, 2, 1)
    consumed = total_calories
    remaining = max(0, calorie_goal - total_calories)
    #The autopct helps display the percentage on the chart round to 1 decimal place.
    plt.pie([consumed, remaining], labels=["Consumed", "Remaining"],
autopct='%1.1f%%', colors=['green', 'blue'])
    plt.title("Calories")

    # Protein
    plt.subplot(2, 2, 2)
    consumed = total_protein
    remaining = max(0, protein_goal - total_protein)
    plt.pie([consumed, remaining], labels=["Consumed", "Remaining"],
autopct='%1.1f%%', colors=['purple', 'lightblue'])
    plt.title("Protein (g)")

    # Carbs
    plt.subplot(2, 2, 3)
    consumed = total_carbs
    remaining = max(0, carb_goal - total_carbs)
    plt.pie([consumed, remaining], labels=["Consumed", "Remaining"],
autopct='%1.1f%%', colors=['orange', 'yellow'])
    plt.title("Carbs (g)")

    # Fats
    plt.subplot(2, 2, 4)
    consumed = total_fats
    remaining = max(0, fat_goal - total_fats)
    plt.pie([consumed, remaining], labels=["Consumed", "Remaining"],
autopct='%1.1f%%', colors=['red', 'pink'])
    plt.title("Fats (g)")

    plt.tight_layout()
    plt.show()
```