

CalHealth: The Wellness Calorie Tracker Application

In Partial Fulfillment of the Requirements in

IT101 (Computer Programming Concepts 1 (Lab))

2nd Term, A.Y. 2023-2024

Submitted by:

Mayo, Jonathan Lance

Andulana, Lyca Marie A.

Submitted to:

Prof. Martzel Baste

Submitted on:

3 / 16 / 2024

**Introduction**

A. Problem Scenario

B. SDGs

C. Project Solution

D. Target Market

E. Similar Application

F. Conceptual Framework

G. Scope and Limitations

H. Project Definition

Project Solution

A. Project/System Prototyping

B. Sample Input/Output

C. Source Code

Appendix

A. References

B. Curriculum Vitae

C. Photo of the data gathering

**Introduction**

**Problem Scenario:**

For people in the IT Field who are stuck in their computers, health, and fitness are put on the sidelines as people prioritize their rather sedentary work and their busy schedules. This is coupled with the fact that despite the abundance of information at our fingertips, information on health and wellness is scattered around hundreds of sites that are often jumbled and complicated to understand. However, with CalHealth, we can mitigate both problems.

CalHealth is a practical tool that can help you manage your dietary goals. Whether you're focused on losing weight, maintaining a balanced diet, or just keeping aware of your food choices, this program is designed to promote self-awareness when it comes to the consumption of various foods that influence your eating habits.

**Sustainable Development Goals**:

Our program aims to promote one of the Sustainable Developmental Goals created by the United Nations to answer different problems existing. The third goal of ensuring healthy lives and promoting well-being for all at all ages is answered through the features of the application which brings awareness of daily consumption and being conscious of the amount people intake.

**Project Solution**:

This project aims to make healthy living more convenient for those willing to keep track of their eating habits. CalHealth is a tool to help you start your wellness journey. With its simplistic interface, this app can be your steppingstone to healthy living. This program alone cannot guarantee you a healthy lifestyle however as there is a weak connection between improvements in health with the use of applications (Iribarren et al., 2021). However, using this program along with discipline and actively improving your eating habits, you may achieve your goal.

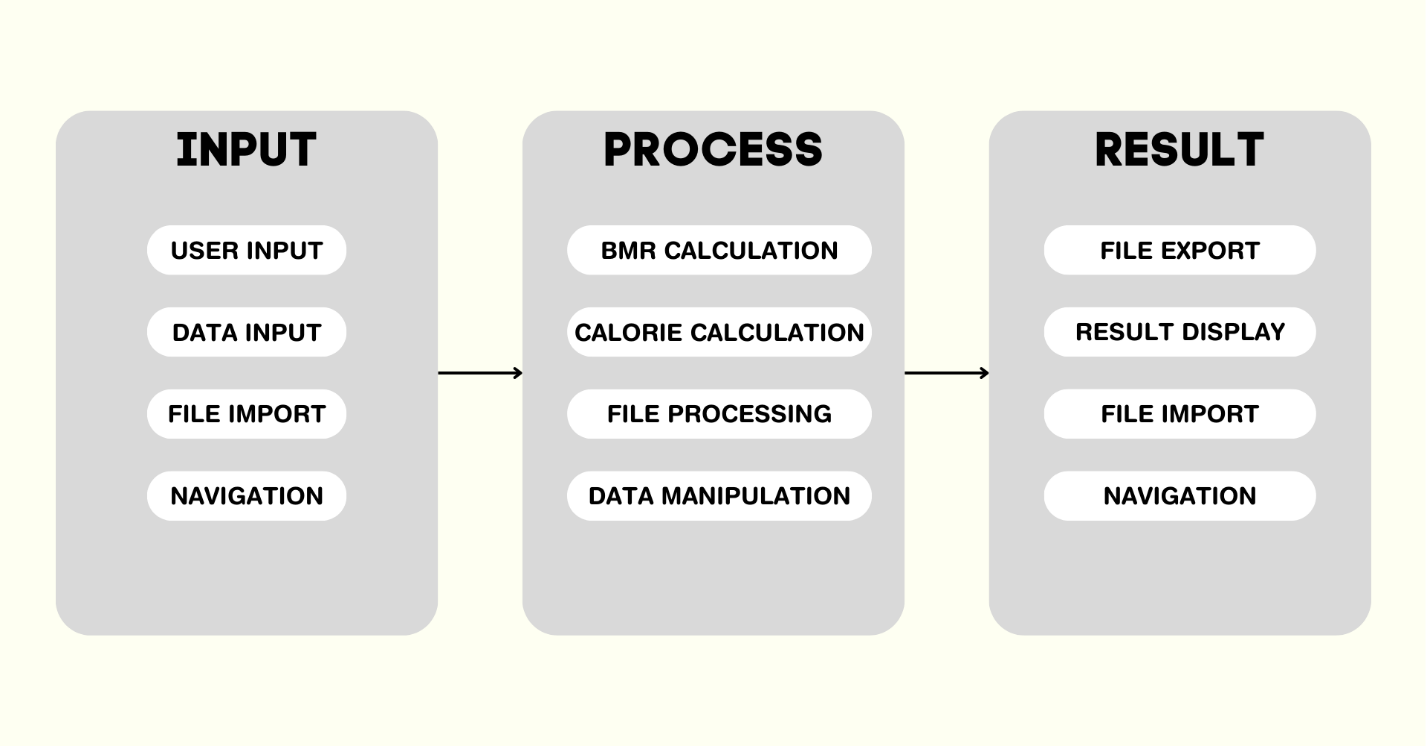
**Target Market**

The program aims to accommodate people who have trouble keeping track of their eating habits or are self-aware of their food consumption by providing an application that aims to give statistical results on their overall behavior. The health-conscious population is the targeted demographic of this application which is the most affected by the results of this application as people tend to do things repeatedly when given information which in this case is the present of caloric information of meals (Chen et al., 2015).

**Similar Application**

This application has similar concepts to WebMD food calculator where it provides an inventory of meals which you can calculate overall calories based on servings (Food Calorie Counter & Calculator, n.d.). Containing Databases of different consumed foods and adding up based on the number of servings.

**Conceptual Framework**



**Scope and Limitations**:

The program’s basic GUI and functions allow for coverage of different existing functions that can be found on similar apps. Features include the ability to calculate BMR based on formulas created by Harris-Benedict (Fletcher, 2020). However, the program does not include a much more accurate representation of Calories with regards to the work type that the user does every day, instead opting for the average value. The 2nd part meal planning allows for an extensive dictionary of different meals which are uncategorized and contained in a single list box. Furthermore, the lack of additional information about individual meals aside from their calories makes the prediction unrealistic in some cases as nutritional balance also comes into play when it comes to having a healthy lifestyle as the programmers opted for a large list instead of a more descriptive list.

The program's Multilayer approach to displaying features follows a more intuitive method, however, this comes at the cost of simplicity and ease of use for those who find having to press multiple buttons before reaching a certain feature inconvenient. Also, the application restriction to a locked 800x600 (aspect ratio) means that the application is not viable for all screen types. It also is limited due to the portability issue as this only is a Windows application which is limited to PCs/Laptops. While porting is doable, the redesign of the interface requires massive changes to the widget location system.

**Project Definition**

This section contains various words and items that are associated with this project.

**Calhealth-** CalHealth short for Calorie health is the title of this project as devised by the programmers working on it. It emphasizes the words calories and health as a hint of the features associated with the portmanteau words.

**BMR**-Basal Metabolic Rate is an estimation as to the minimum number of calories that a person requires to function during a 24-period of basic life functions.

**Calories-** is scientifically defined as the estimated amount of energy required to raise the temperature of exactly one gram of water by one degree Celsius at one atmospheric pressure. In food, this is described as kilocalories or (kcal) which are the estimated amount of energy that a food contains which is consumed by a person, and the energy is used to sustain life (Osilla et al., 2022).

**Python**- a high-level, object-oriented, interpretation-based programming language, one of the easiest to use but also one of the most widely used programming languages in the industry. Its basic syntax functions allow it for easy understanding, recommended for beginners who are new to the programming language space.

**Tkinter**- also known as TK interface, it is a standalone interface used to interact with the Tcl/Tk GUI toolkit. This along with Tk comes pre-packaged on widely used systems such as Unix platforms and Windows systems which makes this one of the most versatile and basic interface systems.

**TkinterModernThemes**- a collection of various modern themes based on the Tkinter interface, this library makes it easy to integrate into new or existing Tkinter projects.

**CSV** – also known as “Comma Separated Values” is a basic standard text format used to store spreadsheets with each line indicated as a row and each comma separated values into columns.

**Datetime-** a module used to import, change, and use dates and times with different formats.

**OS –** Operating system. In Python, importing an operating system allows for interaction between the python file and the operating system. This allows for multiple features mainly operating with directories for file manipulations.

**Pandas-** an open-source library that allows for data manipulation and analysis while also allowing for the creation of data structures and functions.

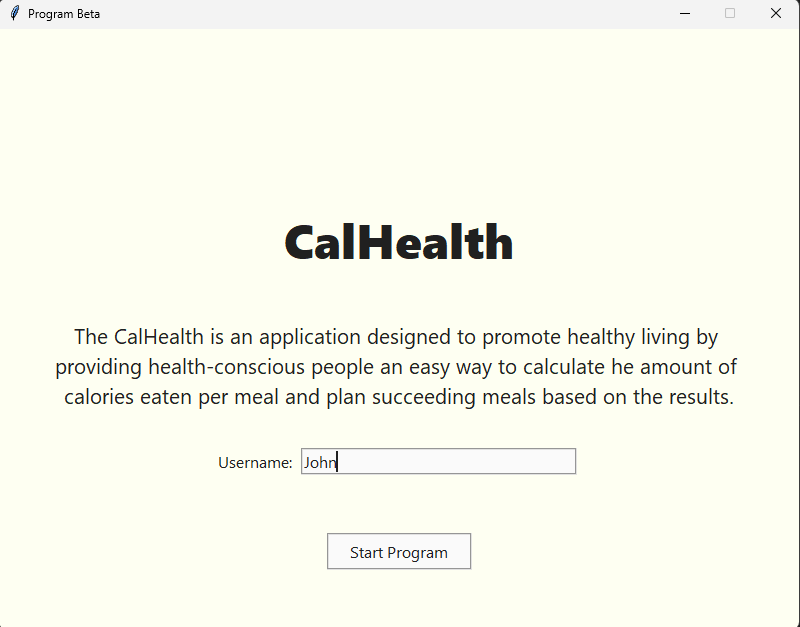
**Shutil-** a module that allows for different functionalities that include high-level manipulation on files and file directories.

**Project Solution**

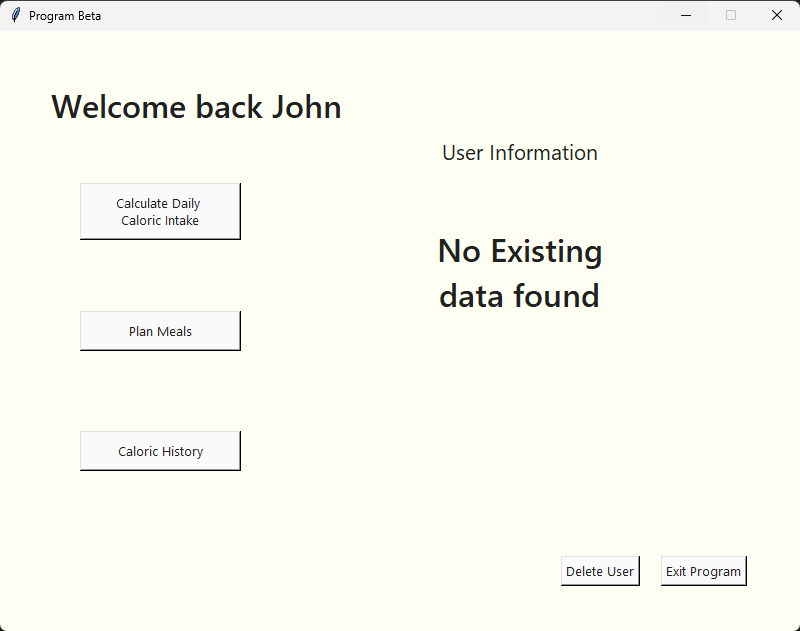
The **Calhealth** application includes various functionalities mainly using Tkinter as a basis for GUI functionality as well as different formulas to provide a statistical analysis of various data created by the user. It includes multiple functionalities which allow for a multitude of features to be included.

1. **Multilayer Graphical User Interface (Multilayer GUI) –** this contains multiple layers of interface with their unique functionalities to ensure a feature-rich application separated based on its intended features.
2. **Username-based history system** – the program saves previous sessions based on the inputted username by the user in the title screen as required.
3. **Directory system** – the program also includes a layered file system where user data is saved on dedicated folders.
4. **Extensive Database system** – the system provides multiple files that save data when a function is executed.
5. BMR Calculation – the system saves the user's BMR data in a CSV file when entries are entered, and the program presents results.
6. MEAL Plan – the system imports a Txt file that contains a dictionary of different items presented in the interface. It also has a feature to save selected meal plans which exports it to a CSV file.
7. User History – the system imports various files which are displayed on this interface for users to see.
8. **Title and Menu Interface** – this includes the tile interface which includes the title of the program, a required username entry box to be filled up with the user, and a start program button that directs the user to a menu interface with buttons for different features, and a brief information of previous session if user is existing.
9. **Feature interface** – contains all the intended features of this program which includes:
10. **BMR Calculation** – the first feature interface that the user can calculate their BMR and average estimated calories for different routines based on their age, gender, height, and weight.
11. **Meal Planning –** the second feature interface that the user can access to different functionalities such as: (1) being able to search and select meals, (2) being able to add meals to selected meals which calculates total calories, (3) being able to modify meal list, and lastly (4) being able to export selected meals.
12. **User History** – the third feature interface that the user can use to access existing history based on usage of the application.
13. **Exception handling** – the program includes exception handling on multiple instances where it occurs due to invalid/missing user entries as well as missing files. This ensure that the program ignores and continues to function.
14. **Navigation system** – the program includes navigation buttons on the bottom right of the interfaces to navigate to the next or previous interface.

**Program prototyping/Source Code**

****

**NEW USER**

****

**NEW USER NO DATA**

**A screenshot of a calorie calculator

Description automatically generated**

**BMR CALCULATE NO DATA**

**A screenshot of a calorie calculator

Description automatically generated**

**BMR CALCULATE IMPARTIAL DATA**

A screenshot of a calorie calculator

Description automatically generated

**BMR CALCULATE IMPARTIAL DATA**

**A screenshot of a calorie calculator

Description automatically generated**

**BMR CALCULATE**

**A screenshot of a computer

Description automatically generated**

**NEW USER UPDATED MENU**

**A screenshot of a calorie calculator

Description automatically generated**

**NEW USER LATEST BMR**

**A screenshot of a computer

Description automatically generated**

**NEW USER UPDATED LATEST BMR**

**A screenshot of a computer

Description automatically generated**

**NEW USER EXPORTED DATA**

A screenshot of a menu

Description automatically generated

**MEAL PLAN INTERFACE**

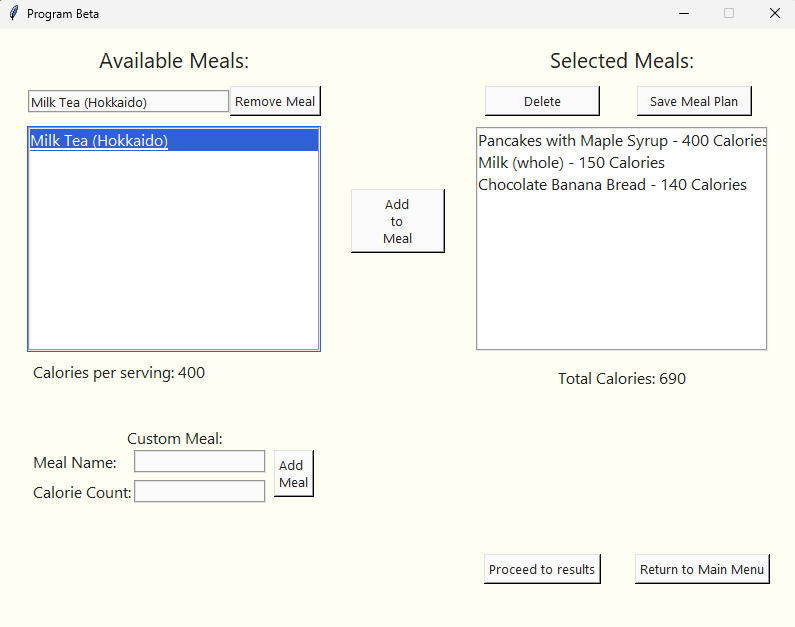
A screenshot of a menu

Description automatically generated

**MEALPLAN ADD**

****

**MEAL PLAN ADD CUSTOM MEAL**

****

**MEAL PLAN ADD CUSTOM MEAL RESULT**

A screenshot of a menu

Description automatically generated

**MEAL PLAN DELETE**

A screenshot of a menu

Description automatically generated

**MEAL PLAN DELETE RESULT**

A screenshot of a computer

Description automatically generated

**MEAL PLAN EXPORT**

A screenshot of a computer

Description automatically generated

**USER HISTORY NO DATA**

A screenshot of a computer

Description automatically generated

**USER HISTORY IMPARTIAL DATA**

A screenshot of a computer

Description automatically generated

**USER HISTORY DATA**

A screenshot of a computer

Description automatically generated

**DELETE USER**

A screenshot of a computer

Description automatically generated

**DELETE USER RESULT**

**Source code:**

|  |
| --- |
| #Code By Mayo and Andulana  #IMPORT LIBRARIES  import tkinter as tk  from tkinter import ttk  import TKinterModernThemes as TKMT  import csv  import datetime  import os  import pandas as pd  import shutil  from tkinter import \*  #CREATE MAIN CLASS  class App(TKMT.ThemedTKinterFrame):      title\_font = ("Segoe UI Black", 36)      heading\_font = ("Segoe UI Semibold", 24)      intro\_font = ("Segoe UI", 16)      text\_font = ("Segoe UI", 12)      color\_bg = '#FEFFF2'      #INITIALIZE THE MAIN CLASS      def \_\_init\_\_(self, theme, mode, usecommandlineargs=True, usethemeconfigfile=True):          super().\_\_init\_\_("Final Project", theme, mode, usecommandlineargs, usethemeconfigfile)          self.master.title("Program Beta")          self.master.geometry("800x600")          self.master.config(bg=self.color\_bg)          self.master.resizable(False, False)          self.startmenu()          self.run()          self.username = ""      def exit\_program(self):              self.master.destroy()      #START MENU CONTAINS THE FIRST LAYER OF THE PROGRAM WHIHC IS THE WELCOME SCREEN WITH USERNAME INPUT      def startmenu(self):          def check\_entry():              if self.username\_entry.get() == "":                  self.button.config(state="disabled")              else:                  self.username = self.username\_entry.get()                  self.button.config(state="normal")          def username\_get():              self.username = self.username\_entry.get()              self.mainmenu()          title = tk.Label(self.master, text="CalHealth", font=self.title\_font, bg= self.color\_bg)          title.place(relx=0.5, rely=0.35, anchor="center")          description = tk.Label(              self.master,              text="The CalHealth is an application designed to promote healthy living by \nproviding health-conscious people an easy way to calculate he amount of \ncalories eaten per meal and plan succeeding meals based on the results.",              font=self.intro\_font, bg= self.color\_bg          )          description.place(relx=0.5, rely=0.56, anchor="center")          Username = tk.Label(self.master, text="Username:", font=self.text\_font, bg= self.color\_bg)          Username.place(relx=0.32, rely=0.72, anchor="center")          self.username\_entry = tk.Entry(font=self.text\_font, width=30, borderwidth=2, relief="groove")          self.username\_entry.place(relx=0.55, rely=.72, anchor="center")          self.button = tk.Button(self.master, text="Start Program", command=username\_get, state='disabled', font=self.text\_font, width= 15, height=1, borderwidth=2, relief="groove")          self.button.place(relx=0.5, rely=0.87, anchor="center")          self.username\_entry.bind("<KeyRelease>", lambda \_: check\_entry())      #MAIN MENU FUNCTION OCNTAINING ALL NAVIGATION AND DISPLAY FUNCTION FOR 2ND LAYER      def mainmenu(self):          # Save the username to the user list          def save\_username():              with open('Datafile/USERS/User\_list.txt', 'r') as file:                  existing\_user\_list = [line.strip() for line in file]              if self.username not in existing\_user\_list:                  with open('Datafile/USERS/User\_list.txt', 'a') as file:                      file.write('\n'+self.username)          for widget in self.master.winfo\_children():              widget.destroy()          print("Main Menu launched!")          with open('Datafile/USERS/User\_list.txt', 'r') as file:              existing\_user\_list = [line.strip() for line in file]            def delete\_user():              user\_folder = f"Datafile/USERS/{self.username}"              if os.path.exists(user\_folder):                  for file\_name in os.listdir(user\_folder):                      file\_path = os.path.join(user\_folder, file\_name)                      if os.path.isfile(file\_path):                          os.remove(file\_path)                      elif os.path.isdir(file\_path):                          shutil.rmtree(file\_path)                  os.rmdir(user\_folder)                # Delete user from user list              with open('Datafile/USERS/User\_list.txt', 'r') as file:                  lines = file.readlines()              with open('Datafile/USERS/User\_list.txt', 'w') as file:                  for line in lines:                      if line.strip() != self.username:                          file.write(line)              for widget in self.master.winfo\_children():                  widget.destroy()              self.startmenu()          def display\_bmr\_data():                  username\_bmr\_file = f"Datafile/USERS/{self.username}/{self.username}\_BMR.csv"                  try:                      bmr\_data = pd.read\_csv(username\_bmr\_file)                        latest\_results = bmr\_data.tail(10)                      table = ttk.Treeview(self.master, columns=list(latest\_results.columns), show="headings")                      for column in latest\_results.columns:                          table.heading(column, text=column)                      for row in latest\_results.itertuples(index=False):                          table.insert("" , "end", values=row)                      table.place(relx=0.65, rely=0.5, anchor='center')                  # If the file is not found, display a message to the user                  except FileNotFoundError:                      print(f"File {username\_bmr\_file} not found.")                      Display\_val = tk.Label(self.master, text="No Existing\ndata found", font=self.heading\_font, bg= self.color\_bg)                      Display\_val.place(relx=0.65, rely=0.4, anchor="center")          # Checks if user has already entered the application          if self.username in existing\_user\_list:              welcome = tk.Label(self.master, text=f"Welcome back {self.username}", font=self.heading\_font, bg= self.color\_bg)              welcome.place(relx=0.06, rely=0.08)              display\_bmr\_data()          else:              welcome = tk.Label(self.master, text=f"Welcome to CalHealth {self.username}", font=self.heading\_font, bg= self.color\_bg)              save\_username()              display\_bmr\_data()              welcome.place(relx=0.06, rely=0.08)          # Main menu options          user\_information = tk.Label(self.master, text="User Information", font=self.intro\_font, bg= self.color\_bg)          user\_information.place(relx=0.65, rely=0.2, anchor="center")          option\_1 = tk.Button(self.master, text="Calculate Daily \nCaloric Intake", padx=24, pady=6, width=15, command=self.Calculate\_Calories)          option\_1.place(relx=0.2, rely=0.3, anchor="center")          option\_2 = tk.Button(self.master, text="Plan Meals", padx=24, pady=6, width=15, command=self.Plan\_Meals)          option\_2.place(relx=0.2, rely=0.5, anchor="center")          option\_3 = tk.Button(self.master, text="User History", padx=24, pady=6, width=15, command=self.user\_history)          option\_3.place(relx=0.2, rely=0.7, anchor="center")          return\_button = tk.Button(self.master, text="Exit Program", command=self.exit\_program)          return\_button.place(relx=0.88, rely=0.9, anchor="center")          delete\_button = tk.Button(self.master, text="Delete User", command=delete\_user)          delete\_button.place(relx=0.75, rely=0.9, anchor="center")      #FUNCTION TO CALCULATE CALORIES      def Calculate\_Calories(self):          for widget in self.master.winfo\_children():              widget.destroy()          print("Calculate Calories launched!")          # Function to check if the user has entered valid data after pressing the calculate button          def val\_check():              age = self.age\_entry.get()              gender = self.gender\_var.get()              height = self.height\_entry.get()              weight = self.weight\_entry.get()                if not age.isdigit():                  self.age\_entry.config(bg="red")              else:                  self.age\_entry.config(bg="white")              if not height.isdigit():                  self.height\_entry.config(bg="red")              else:                  self.height\_entry.config(bg="white")              if not weight.isdigit():                  self.weight\_entry.config(bg="red")              else:                  self.weight\_entry.config(bg="white")                if not gender:                  self.invalid\_label.config(text="Please indicate gender", font=self.text\_font, fg="red")              else:                  self.invalid\_label.config(text="")              # If all the data is valid, calculate the BMR              if age.isdigit() or height.isdigit() and weight.isdigit() and gender:                  calculate\_bmr()          # Function to calculate the Basal Metabolic Rate (BMR) and display the results          def calculate\_bmr():              age = int(self.age\_entry.get())              weight = int(self.weight\_entry.get())              height = int(self.height\_entry.get())              if self.gender\_var.get() == "M":                  bmr = (88.4 + 13.4 \* weight) + (4.8 \* height) - (5.68 \* age)              elif self.gender\_var.get() == "F":                  bmr = (447.6 + 9.25 \* weight) + (3.10 \* height) - (4.33 \* age)                  self.bmr\_value = tk.Label(self.master, text=f"{round(bmr)}", font=self.text\_font, bg=self.color\_bg)                  self.bmr\_value.place(relx=0.775, rely=0.2, anchor="w")              try:                  if bmr:                      proceed\_button.config(state="normal")                      bmr\_label = tk.Label(self.master, text="BMR:", font=self.text\_font, bg=self.color\_bg)                      bmr\_label.place(relx=0.775, rely=0.2, anchor="e")                      maintain\_weight\_label = tk.Label(self.master, text="Maintain Weight:", font=self.text\_font, bg=self.color\_bg)                      maintain\_weight\_label.place(relx=0.775, rely=0.3, anchor="e")                      maintain\_weight\_calories = round(bmr)                      maintain\_weight\_value = tk.Label(self.master, text=f"{maintain\_weight\_calories} calories/day", font=self.text\_font, bg=self.color\_bg)                      maintain\_weight\_value.place(relx=0.775, rely=0.3, anchor="w")                      mild\_weight\_loss\_label = tk.Label(self.master, text="Mild Weight Loss:", font=self.text\_font, bg=self.color\_bg)                      mild\_weight\_loss\_label.place(relx=0.775, rely=0.4, anchor="e")                      mild\_weight\_loss\_calories = round(bmr \* 0.91)                      mild\_weight\_loss\_value = tk.Label(self.master, text=f"{mild\_weight\_loss\_calories} calories/day", font=self.text\_font, bg=self.color\_bg)                      mild\_weight\_loss\_value.place(relx=0.775, rely=0.4, anchor="w")                      weight\_loss\_label = tk.Label(self.master, text="Weight Loss:", font=self.text\_font, bg=self.color\_bg)                      weight\_loss\_label.place(relx=0.775, rely=0.5, anchor="e")                      weight\_loss\_calories = round(bmr \* 0.82)                      weight\_loss\_value = tk.Label(self.master, text=f"{weight\_loss\_calories} calories/day", font=self.text\_font, bg=self.color\_bg)                      weight\_loss\_value.place(relx=0.775, rely=0.5, anchor="w")                      extreme\_weight\_loss\_label = tk.Label(self.master, text="Extreme Weight Loss:", font=self.text\_font, bg=self.color\_bg)                      extreme\_weight\_loss\_label.place(relx=0.775, rely=0.6, anchor="e")                      extreme\_weight\_loss\_calories = round(bmr \* 0.64)                      extreme\_weight\_loss\_value = tk.Label(self.master, text=f"{extreme\_weight\_loss\_calories} calories/day", font=self.text\_font, bg=self.color\_bg)                      extreme\_weight\_loss\_value.place(relx=0.775, rely=0.6, anchor="w")                      # Save the BMR data to a CSV file                      username = self.username                      folder\_path = f"Datafile/USERS/{username}"                      os.makedirs(folder\_path, exist\_ok=True)                      filename = f"{folder\_path}/{username}\_BMR.csv"                      with open(filename, 'a', newline='') as file:                          writer = csv.writer(file)                          if file.tell() == 0:                              writer.writerow(["Name", "Value"])                          writer.writerow(["Date", datetime.date.today()])                          writer.writerow(["User", username])                          writer.writerow(["Age", age])                          writer.writerow(["Weight", weight])                          writer.writerow(["Height", height])                          writer.writerow(["Gender", self.gender\_var.get()])                          writer.writerow(["BMR", round(bmr)])                          writer.writerow(["Maintain Weight", maintain\_weight\_calories])                          writer.writerow(["Mild Weight Loss", mild\_weight\_loss\_calories])                          writer.writerow(["Weight Loss", weight\_loss\_calories])                          writer.writerow(["Extreme Weight Loss", extreme\_weight\_loss\_calories])                          writer.writerow([])              except NameError:                  pass            # Creates entry fields for the user to enter          welcome = tk.Label(self.master, text=f"Calorie Calculator", font=self.intro\_font, bg=self.color\_bg)          welcome.place(relx=0.5, rely=0.08, anchor="center")            age\_label = tk.Label(self.master, text="Age:", font=self.text\_font, bg=self.color\_bg)          age\_label.place(relx=0.12, rely=0.2, anchor="center")          self.age\_entry = tk.Entry(font=self.text\_font, width=20, borderwidth=2, relief="groove")          self.age\_entry.place(relx=0.3, rely=0.2, anchor="center")          self.gender\_var = tk.StringVar()          gender\_label = tk.Label(self.master, text="Gender:", font=self.text\_font, bg=self.color\_bg)          gender\_label.place(relx=0.12, rely=0.3, anchor="center")            self.male\_checkbox = Checkbutton(self.master, variable=self.gender\_var, onvalue="M", offvalue="", text = "Male", bg= self.color\_bg)          self.male\_checkbox.place(relx=0.23, rely=0.3, anchor="center")          self.female\_checkbox = Checkbutton(self.master, variable=self.gender\_var, onvalue="F", offvalue="", text = "Female", bg= self.color\_bg)          self.female\_checkbox.place(relx=0.35, rely=0.3, anchor="center")          self.gender\_var.set("")          height\_label = tk.Label(self.master, text="Height (cm):", font=self.text\_font, bg=self.color\_bg)          height\_label.place(relx=0.12, rely=0.4, anchor="center")          self.height\_entry = tk.Entry(font=self.text\_font, width=20, borderwidth=2, relief="groove")          self.height\_entry.place(relx=0.3, rely=0.4, anchor="center")          weight\_label = tk.Label(self.master, text="Weight (kg):", font=self.text\_font, bg=self.color\_bg)          weight\_label.place(relx=0.12, rely=0.5, anchor="center")          self.weight\_entry = tk.Entry(font=self.text\_font, width=20, borderwidth=2, relief="groove")          self.weight\_entry.place(relx=0.3, rely=0.5, anchor="center")          calculate\_button = tk.Button(self.master, text="Calculate BMR", command=val\_check)          calculate\_button.place(relx=0.3, rely=0.6, anchor="center")            self.invalid\_label = tk.Label(self.master, text="", font=self.text\_font, bg=self.color\_bg)          self.invalid\_label.place(relx=0.295, rely=0.345, anchor="center")          proceed\_button = tk.Button(self.master, text="Proceed to Meal Plan", command=self.Plan\_Meals, state="disabled")          proceed\_button.place(relx=0.68, rely=0.9, anchor="center")            return\_button = tk.Button(self.master, text="Return to Main Menu", command=self.mainmenu)          return\_button.place(relx=0.88, rely=0.9, anchor="center")        #FUNCTION TO PLAN MEALS      def Plan\_Meals(self):          self.total\_calories = 0          for widget in self.master.winfo\_children():              widget.destroy()          print("Plan Meals launched!")          # Function to import the dictionary of meals and their calorie counts from a txt file          def import\_dictionary(filename):              dictionary = {}              with open(filename, 'r') as file:                  for line in file:                      key, value = line.strip().split(':')                      dictionary[key.strip()] = value.strip()              return dictionary          imported\_dict = import\_dictionary('Datafile/meals.txt')          # Create labels for the listbox and mealbox          listbox\_label = tk.Label(self.master, text="Available Meals:", font=self.intro\_font, bg=self.color\_bg)          listbox\_label.place(relx=0.22, rely=0.05, anchor="center")          mealbox\_label = tk.Label(self.master, text="Selected Meals:", font=self.intro\_font, bg=self.color\_bg)          mealbox\_label.place(relx=0.78, rely=0.05, anchor="center")          # Create a listbox to display the meals          listbox = tk.Listbox(self.master, width=32, height=10, font=self.text\_font, bg= "white", borderwidth=2, relief="groove")          listbox.place(relx=0.22, rely=0.35, anchor="center")          for key in imported\_dict.keys():              listbox.insert(tk.END, f"{key}")          # Create a label to display the calorie count of the selected meal          value\_label = tk.Label(self.master, text="Calories per serving: ", font=self.text\_font, bg=self.color\_bg)          value\_label.place(relx=0.04, rely=0.57, anchor="w")          def update\_calorie\_count():              try:                  selected\_item = listbox.get(listbox.curselection())                  calorie\_count = imported\_dict[selected\_item]                  value\_label.config(text=f"Calories per serving: {calorie\_count}")              except:                  value\_label.config(text="Calories per serving: N/A")          # Function to add a meal to the dictionary and the listbox          def add\_meal():              meal\_name = meal\_entry.get()              calorie\_count = calorie\_entry.get()              if not meal\_name or not calorie\_count.isdigit():                  return              imported\_dict[meal\_name] = calorie\_count              with open('Datafile/meals.txt', 'a') as file:                  file.write(f'{meal\_name}: {calorie\_count}\n')              meal\_entry.delete(0, tk.END)              calorie\_entry.delete(0, tk.END)              listbox.insert(tk.END, f"{meal\_name}")          # Function to add a meal to the mealbox and calculate the total calories          selected\_meal = {}          def add\_mealbox():              meal\_name = meal\_pick.get()              if meal\_name in imported\_dict:                  calorie\_count = imported\_dict[meal\_name]              mealbox.insert(tk.END, f"{meal\_name} - {calorie\_count} Calories")              self.total\_calories += int(calorie\_count)              total\_calories\_label.config(text=f"Total Calories: {self.total\_calories}")              selected\_meal[meal\_name] = int(calorie\_count)              selected\_meal.update({meal\_name: calorie\_count})          # Function to delete a meal from the mealbox and calculate the total calories          def delete\_from\_mealbox():              selected\_index = mealbox.curselection()              if selected\_index:                  selected\_item = mealbox.get(selected\_index)              meal\_name = selected\_item.split(" - ")[0]              calorie\_count = selected\_item.split(" - ")[1].split(" ")[0]              mealbox.delete(selected\_index)              self.total\_calories -= int(calorie\_count)              total\_calories\_label.config(text=f"Total Calories: {self.total\_calories}")              del selected\_meal[meal\_name]              try:                  selected\_meal.pop(meal\_name)              except KeyError:                  pass            # Function to save the meal history to a CSV file          def save\_meal\_history():              directory = f"Datafile/USERS/{self.username}"              os.makedirs(directory, exist\_ok=True)              file\_path = f"{directory}/{self.username}\_meals.csv"              current\_datetime = datetime.date.today()              # Write the data to the CSV file              with open(file\_path, mode='a', newline='') as file:                  writer = csv.writer(file)                  if file.tell() == 0:                      writer.writerow(["Name", "Calorie Count"])                  writer.writerow(["Date",current\_datetime])                  for key, value in selected\_meal.items():                      writer.writerow([key, value])                  writer.writerow(["Total Calories", self.total\_calories])                  writer.writerow([])                proceed\_button.config(state="normal")          # Function to remove a meal from the dictionary and the listbox          def remove\_meal():              selected\_index = listbox.curselection()              if selected\_index:                  selected\_item = listbox.get(selected\_index)                  del imported\_dict[selected\_item]                  listbox.delete(selected\_index)                  with open('Datafile/meals.txt', 'w') as file:                      for key, value in imported\_dict.items():                          file.write(f'{key}: {value}\n')            # Create the labels, entry fields, and buttons for the custom meal          custom\_meal\_label = tk.Label(self.master, text="Custom Meal:", font=self.text\_font, bg= self.color\_bg)          custom\_meal\_label.place(relx=0.22, rely=0.68, anchor="center")          meal\_label = tk.Label(self.master, text="Meal Name:", font=self.text\_font, bg= self.color\_bg)          meal\_label.place(relx=0.04, rely=0.72, anchor="w")          meal\_entry = tk.Entry(self.master, width=18, borderwidth=2, relief="groove")          meal\_entry.place(relx=0.17, rely=0.72, anchor="w")          calorie\_label = tk.Label(self.master, text="Calorie Count:", font=self.text\_font, bg= self.color\_bg)          calorie\_label.place(relx=0.04, rely=0.77, anchor="w")          calorie\_entry = tk.Entry(self.master, width=18, borderwidth=2, relief="groove")          calorie\_entry.place(relx=0.17, rely=0.77, anchor="w")          add\_button = tk.Button(self.master, text="Add \nMeal", command=add\_meal)          add\_button.place(relx=0.37, rely=0.74, anchor="center")              meal\_button = tk.Button(self.master, text="Add\nto\nMeal", width=12, height=3, command=add\_mealbox)          meal\_button.place(relx=0.5, rely=0.32, anchor="center")            meal\_pick = tk.Entry(self.master, width=28, borderwidth=2, relief="groove")          meal\_pick.place(relx=0.038, rely=0.12, anchor="w")          remove\_button = tk.Button(self.master, text="Remove Meal", command=remove\_meal, height=1)          remove\_button.place(relx=0.29, rely=0.12, anchor="w")          #removes unmatched items          def filter\_listbox(\_):              try:                  filter\_text = meal\_pick.get().lower()                  listbox.delete(0, tk.END)                  for key in imported\_dict.keys():                      if filter\_text in key.lower():                          listbox.insert(tk.END, f"{key}")              except:                  pass          meal\_pick.bind("<KeyRelease>", filter\_listbox)            # checks if user selected an item form the listbox          def select\_item(\_):              try:                  selected\_item = listbox.get(listbox.curselection())                  meal\_pick.delete(0, tk.END)                  meal\_pick.insert(tk.END, selected\_item)                  update\_calorie\_count()              except:                  pass          listbox.bind("<<ListboxSelect>>", select\_item)          # Create the mealbox to display the selected meals          mealbox = tk.Listbox(self.master, width=32, height=10, font=self.text\_font, bg="white", borderwidth=2, relief="groove")          mealbox.place(relx=0.78, rely=0.35, anchor="center")            total\_calories\_label = tk.Label(self.master, text=f"Total Calories: {self.total\_calories}", font=self.text\_font, bg=self.color\_bg)          total\_calories\_label.place(relx=0.78, rely=0.58, anchor="center")          delete\_button = tk.Button(self.master, text="Delete", command=delete\_from\_mealbox, width=15, height=1, borderwidth=2)          delete\_button.place(relx=0.68, rely=0.12, anchor="center")          save\_button = tk.Button(self.master, text="Save Meal Plan", command=save\_meal\_history, width=15, height=1, borderwidth=2)          save\_button.place(relx=0.87, rely=0.12, anchor="center")          proceed\_button = tk.Button(self.master, text="Proceed to results", command=self.user\_history)          proceed\_button.place(relx=0.68, rely=0.9, anchor="center")            return\_button = tk.Button(self.master, text="Return to Main Menu", command=self.mainmenu)          return\_button.place(relx=0.88, rely=0.9, anchor="center")      #FUNCTION TO DISPLAY USER HISTORY      def user\_history(self):          for widget in self.master.winfo\_children():              widget.destroy()          print("User History launched!")          title\_history = tk.Label(self.master, text="User History", font=self.heading\_font, bg= self.color\_bg)          title\_history.place(relx=0.5, rely=0.1, anchor="center")          # Display the BMR history in a table          bmr\_filename = f"Datafile/USERS/{self.username}/{self.username}\_BMR.csv"          if os.path.isfile(bmr\_filename):              bmr\_data = pd.read\_csv(bmr\_filename)              bmr\_table = ttk.Treeview(self.master, height = 4)              bmr\_table["columns"] = tuple(bmr\_data.columns)              bmr\_table["show"] = "headings"              for column in bmr\_table["columns"]:                  bmr\_table.heading(column, text=column)              for index, row in bmr\_data.iterrows():                  bmr\_table.insert("", "end", values=tuple(row))              bmr\_table.place(relx=0.5, rely=0.3, anchor="center")          else:              Missing\_bmr = tk.Label(self.master, text="No BMR data found", font=self.heading\_font, bg= self.color\_bg)              Missing\_bmr.place(relx=0.5, rely=0.3, anchor="center")          # Display the meal history in a table          meals\_filename = f"Datafile/USERS/{self.username}/{self.username}\_meals.csv"          if os.path.isfile(meals\_filename):              meals\_data = pd.read\_csv(meals\_filename)              meals\_table = ttk.Treeview(self.master, height = 4)              meals\_table["columns"] = tuple(meals\_data.columns)              meals\_table["show"] = "headings"              for index, row in meals\_data.iterrows():                  meals\_table.insert("", "end", values=tuple(row))              meals\_table.place(relx=0.5, rely=0.6, anchor="center")          else:              Missing\_meals = tk.Label(self.master, text="No meal data found", font=self.heading\_font, bg= self.color\_bg)              Missing\_meals.place(relx=0.5, rely=0.5, anchor="center")            proceed\_button = tk.Button(self.master, text="Return to Main Menu", command=self.mainmenu)          proceed\_button.place(relx=0.71, rely=0.9, anchor="center")            return\_button = tk.Button(self.master, text="Exit Program", command=self.exit\_program)          return\_button.place(relx=0.88, rely=0.9, anchor="center")  if \_\_name\_\_ == "\_\_main\_\_":      App("sun-valley", "light") |

**Appendix**

**References**:

Iribarren, S. J., Akande, T. O., Kamp, K., Barry, D., Kader, Y. G., & Suelzer, E. (2021). Effectiveness of mobile apps to promote health and manage disease: systematic review and meta-analysis of randomized controlled trials. *Jmir Mhealth and Uhealth*, *9*(1), e21563. https://doi.org/10.2196/21563

Chen, R., Smyser, M., Chan, N., Ta, M., Saelens, B. E., & Krieger, J. (2015). Changes in awareness and use of calorie information after mandatory menu labeling in restaurants in King County, Washington. *American Journal of Public Health*, *105*(3), 546–553. https://doi.org/10.2105/ajph.2014.302262

Fletcher, J. (2020, March 9). What to know about basal metabolic rate. https://www.medicalnewstoday.com/articles/basal-metabolic-rate#how-to-calculate-it

Osilla, E. V., Safadi, A. O., & Sharma, S. (2022, September 12). *Calories*. StatPearls – NCBI Bookshelf. https://www.ncbi.nlm.nih.gov/books/NBK499909/

**CURRICULUM VITAE**

**PERSONAL INFORMATION**

Name: Mayo, Jonathan Lance S.

Birthday: May 16, 2004

Age: 19

Year and Section: 1st Year, IT101L.A124

Program: BSIS

Current Address: Unit 105, Casa Graciana, Camia Street Dao Corner, Juna Subdivision, Matina Crossing, Davao, Davao Del Sur

Contact Number/s: 09205696575

Email Address: nathanmayo15@gmail.com

**EDUCATIONAL BACKGROUND**

Senior High School:                         Malayan Colleges Mindanao

Gen. Douglas MacArthur Hwy, Talomo, Davao

                                                           City, 8000 Davao del Sur (2021-Present)

Junior High School:                             St. John Early Learning Center Inc.

08 Molave Street, Brgy. New

Isabela, Tacurong City, Sultan Kudarat (2017-2021)

Elementary:             St. John Learning Center Inc.

08 Molave Street, Brgy. New

Isabela, Tacurong City, Sultan Kudarat (2011-2017)

**AWARDS/ACHIEVEMENTS**

List your awards/achievements in your extracurricular activities from elementary up to present in reverse chronological order (present to past).

|  |  |
| --- | --- |
| MALAYAN DEBATE CUP (PARTIC) | February 2022 |
| Dean’s Lister (MMCM 1st sem) | 2023-2024 |
| 2nd Place, Pasikatay Short Film | April 2023 |
| 2nd Place, E-Sports (Minecraft Mascot Building), Pasikatay 2022 | March 2022 |
| With High Honors (MMCM) | (SY 2021-2022) |
| 3rd Place, Division Level Technolympics (Technical Drafting) | October 2019 |
| With Honors/With High Honors (SJELCI) | March 2022 (SY 2007-2021) |

**SKILLS AND INTEREST**

**Programming Knowledge:** Ren-py (VN creator based on Python), HPL (Hoi4-Programming Language based on C++), basic C++, Java, and Python knowledge.

**Skill/s:** Cognitive and Critical Thinking Skills.

**Talents:** Multi-lingualise, Mathematical Thinking, Strategic Thinking, Teaching, and Writing.

**Hobbies:** Chess, cooking, cycling, game modding (programming), fixing electronics, PC hobbyist, and Audiovisual Enthusiast.

**CURRICULUM VITAE**

A person with long hair smiling

Description automatically generated**PERSONAL INFORMATION**

Name: Andulana, Lyca Marie A.

Birthday: November 23, 2004

Age: 19

Year and Section: 1st Year, IT101L.A124

Program: BSIS

Current Address: 101-1 Juan Luna Street Davao City

Contact Number/s: 09618563993

Email Address: andulanalycamarie1123@gmail.com

**EDUCATIONAL BACKGROUND**

Senior High School:                         San Pedro College

12 Guzman St, Davao City,

8000 Davao del Sur (2021-2023)

Junior High School:                              Stella Maris Academy of Davao

Nicasio Torres Street Barrio Obrero Davao City, 8000 Davao del Sur (2017-2021)

Junior High School:                              Stella Maris Academy of Davao

Nicasio Torres Street Barrio Obrero Davao City, 8000 Davao del Sur (2017-2021)

**AWARDS/ACHIEVEMENTS**

List your awards/achievements in your extracurricular activities from elementary up to present in reverse chronological order (present to past).

|  |  |
| --- | --- |
| Interschool Short Film Festival 2023, Best Actress (PSITS) | December 2023 |
| Deans’ Lister (MMCM 1st Sem) | 2023 |
| 1st Runner-Up Impersonation SPC-SHS Intramurals 2023 | March 2023 |
| SPC Director’s Cut Short Film Competition. Best Leading Actress | May 2022 |
| With Honors (San Pedro College) | 2022 |
| 3rd Place SPC Global Entrepreneurship Days | May 2021 |

**SKILLS AND INTEREST**

**Programming Knowledge:** Basic Python Skill

**Skill/s:** Organizational Skills, Managing, Resourcefulness

**Talents:** Multi-lingual, Creative Thinking, Adaptable, Communicative

**Hobbies:** Crocheting, Sewing, Crafting, Acting, Cinema and Video Game Enthusiast