Review of Version 1

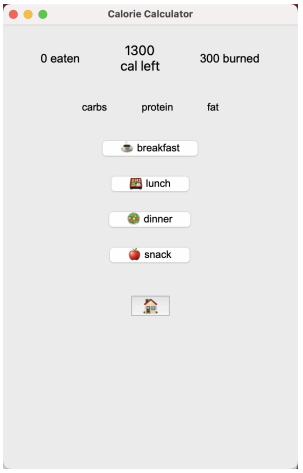
1. Project Overview

The goal of my project is to develop a calorie-tracking app that allows users to search for food items, input their weight, and calculate total calories based on the provided data. In version 1-A, I created a command-line interface (CLI) tool that used fuzzy matching to find food items in a dataset (calories.csv) and calculate calories accordingly. In version 1-B, I expanded this project into a graphical user interface (GUI) with Tkinter, making it more accessible and user-friendly.

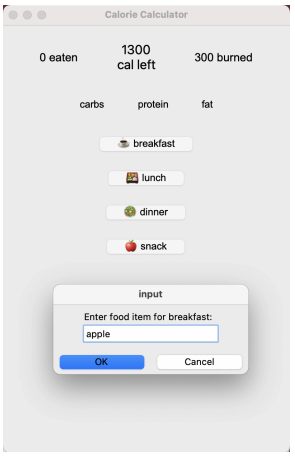
2. GUI Demonstration

Users can enter food items for each meal by using fuzzy searching to match food items from a database. Then the tool allows the users to enter the weight of the food to calculate the exact calories. After that, the homepage will show the total calories eaten, burned calories, and remaining calories against a set goal.

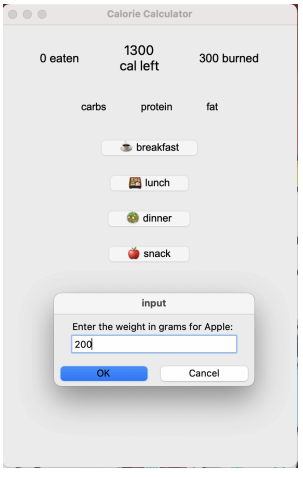
-Home page with different zones: calorie data, Macronutrients, and several meal buttons to input food items.



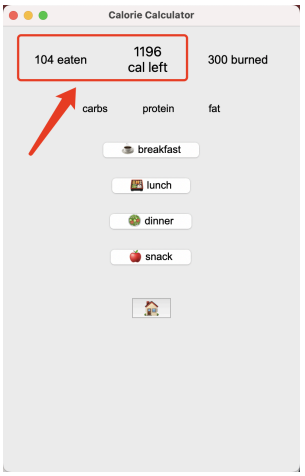
-Textbox to input food items.



-Textbox to input food weight.



-Homepage will show the data changes.



3. Challenges Encountered

-OOP: Using OOP to create a Class is quite a difficult part for me, and I spent a lot of time debugging.
--Solution: I reviewed the example project several times to understand and practice OOP. Then be patient, patient, and patient.

-GUI Layout: Organizing the layout in Tkinter was more difficult than expected. I had to revise the way the widgets were positioned to ensure a clean, user-friendly interface.
--Solution: I learned to use `grid()` more effectively for positioning, making the interface look cleaner and more intuitive.

4. Unsolved Issues

- Macronutrient Tracking: I originally planned to track macronutrients (carbs, protein, fat), but this feature hasn't been fully implemented. I plan to add this functionality in the next weeks.

- Burned Calories: Currently, the calories burned are set to a static value (300). I want to allow users to enter exercise data to dynamically calculate calories burned.

- API Integration: This is pending. The plan is to ensure the core functionality using the static data source is fully stable. Next, revisit API integration by simplifying API calls, focusing only on essential nutrition data.

5. Milestones for the Next Weeks (Version 2)

(1) User Profile Form
- Create a form to input user personal data (age, weight, gender, health goals), which will affect personalized calorie recommendations.

(2) Complete API Integration
- Integrate Edamam API to pull nutritional data for food items and allow fuzzy searching with real-time data.

(3) Macronutrient Tracking:
- Implement tracking for carbs, protein, and fat.
- I don't know if Edamam API has this kind of database. I need some help.

(4) Add a food list
 - When users add a food item, weight, or calories, this information will be shown underneath the button making intaken food information obvious.

(5) Circular Progress Ring and Calorie Limit Alert
 - Implement a circular progress ring to visually represent the user's remaining daily calorie allowance, and trigger an alert if the user exceeds their calorie limit.

(6) Polished GUI

6. Self-Reflection:

- Progress Satisfaction: I'm satisfied with my progress so far. The basic functionality is implemented, and the app is working as intended.

- Project Completion: I believe I can finish the project within the remaining weeks, although the API integration and additional features (like personalized plans) may need to be simplified depending on time constraints.

- Expectations vs. Reality: The transition from CLI to GUI was more complex than expected, but fuzzy searching and calorie tracking were easier to implement with pandas and rapidfuzz. The API integration is proving to be more difficult than expected.

- Light Bulb Moments: I discovered that setting a match threshold significantly improved the fuzzy matching results, making the app more user-friendly.