Evan Buckley, Chris Nielson, Zachary Sandifer

Special Topics in CS: Python

Shifat Mithila

20 Nov 2024

Group 20 Project Report

Create\_Meal\_Plan() Algorithm:

The create\_meal\_plan() algorithm within the create\_meal\_plan.py class creates a random list of meals for each day to ensure meal diversity using the shuffle() function. The function checks to make sure no two meals are the same on any given day to ensure diversity.

Dietary Plans:

| **Diet** | **Calories** | **Protein (g)** | **Carbs (g)** | **Fat (g)** |
| --- | --- | --- | --- | --- |
| Vegetarian | 2500 | 50 | 250 | 65 |
| Vegan | 2000 | 60 | 220 | 60 |
| Keto | 2800 | 120 | 25 | 140 |
| Mediterranean | 3000 | 70 | 275 | 73 |
| Low-Carb | 3000 | 95 | 100 | 110 |
| Bulking | 5000 | 180 | 350 | 85 |
| Cutting | 3500 | 160 | 140 | 55 |

Within the settings tab, the user pics a dietary plan based on the plans shown above. This plan decides their preferred nutrition, to help decide which meals to use. Included in the recipe-app-backend, there is a file, ‘recipes.csv’, which contains 136 recipes, each with a different category to fit the different meal plans. Our meal planning algorithm creates a list of each of the recipes with the same category as the user’s preference, then shuffles the list to find 3 recipes for each day that stay under the daily recommended calories and within 80% of the recommended protein. The total nutritional values of the three lists is calculated by the get\_total\_nutrition() function created in nutrition.py. We understand that with a rudimentary algorithm such as this, it would be very difficult to obtain the correct calories, while also maintaining the recommended protein count, which is why we want to ensure the protein is at least 80% of the daily recommended. If needed, our algorithm will double the portion of a meal to meet the daily protein goal, as long as it stays within the daily caloric intake as well.

This algorithm generates a dictionary of each day of the week, which each store data for breakfast, lunch and dinner, along with the recipes and all of their nutritional information. It returns this dictionary, which is accessed by a route and called from the front end to display to the user. When creating the list of recipes, it also takes into account the available groceries stored in user\_available\_ingredients.csv and then generates both a csv and txt file of the grocery list needed to cook for the week, using the functions created in ingredients.py.