Project Checkpoint

* Preksha Kamleshkumar Shah
* 8645 1278

Upload on GitHub

The project uses data retrieved from an API called Spoonacular API (<https://spoonacular.com/food-api/docs>). This is a recipe search tool which suggestss recipes based on various inputs and configurations. The project right now, uses a simple search query to search the recipe (e.g Pasta) but the plan is to incorporate trees to gather user preferences for main ingredients, cuisine and dietary preferences. Furthermore, the project uses their /analyzedInstructions extension to retrieve recipe steps and information about the ingredients used in the recipe. The spoonacular API requires pricing and subscription plans to access their data. I have a free subscription that provides me with an API key and I am allowed 150 requests every day with a max speed of 1 request/sec.

All data is retrieved from the API in JSON and cached appropriately. Right now, the code caches only the top recipe suggestions for a certain query. But the final project will cache the recipe instructions for each recipe and also a list of required ingredients. The final cached file would have {‘pasta’: recipes}, {‘recipe\_id’: [ingredients, steps]} or something similar.

Examples of data retrieval:

1. Top recipe recommendations:
   1. baseurl = "https://api.spoonacular.com/recipes/complexSearch?apiKey=" + API\_key
   2. parameters: A lot of them can be found at (<https://spoonacular.com/food-api/docs>).
   3. Use requests.get to get the json data and json.loads to load everything in a usable python data type (dictionary).
   4. Once we receive this dictionary, we store each individual recipe as an object of the “Recipe” class that we defined previously. Refer to Figure 4 for this.

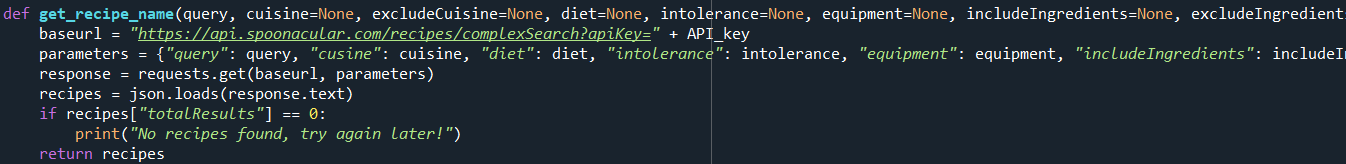


Figure 1: Retrieving top recipe recommendations.

1. Recipe information:
   1. We use the /analyzedInstructions field to gather the output (Figure 3) which contains information on the recipe steps as well as the required ingredients that we will use.

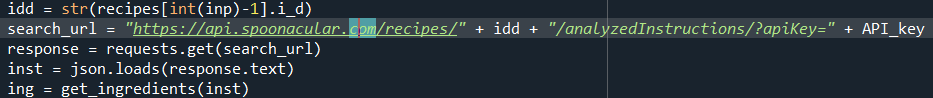


Figure 2: Retrieving Recipe Information

Graphical user interface, text, application, email

Description automatically generated

Figure 3: Output of the /analyzedIngredients Step

Next, we collect the ingredient information in a list for the searched recipe and then we print out all the necessary steps using a separate function for user readability. As mentioned, we will be caching these data points in the final project. Images of caching the recipe name with the recipe recommendations are shown below (Figure 5, Figure 6). As mentioned before, each recipe is stored as an object of the recipe class that contains all the necessary attributes for defining a recipe.

Text

Description automatically generated

Figure 4: Recipe class

Graphical user interface

Description automatically generated

Figure 5: Saving recipes as objects or importing it from the cached file

Letter

Description automatically generated with medium confidence

Figure 6: Snippet of cache file showing entries for “pasta” and some for “pizza”

Finally to summarize, here is a flow of the code:

1. Use a tree to gather input from the user for their main food ingredient, example Tofu, Chicken, Beef, Broccoli etc.
2. Use a tree to gather input from the user for their dietary preference, example Vegan, vegetarian etc.
3. Use a tree to gather input from the user for their preferred cuisine, example Italian, Mexican, Chinese, Indian etc.
4. A string input from the user regarding the recipe they would like to search for.
5. We retrieve data from the API for the top recommendations for this query/search string and save each of these recipes in a list of objects of the Recipe class previously defined.
6. Using this class and its attributes, we print the recipe name and ask user to select any one to learn more.
7. Once the user selects a number, we retrieve the recipe instructions and ingredients from the API and save the ingredients in a list.
8. We will cache the input query with the suggested recipes, the ingredients, and the recipe steps for future use.
9. Finally, after the user selects a number/recipe we will play a small tree with them to determine if they have all the ingredients required to make the meal or not. Check Figure 7 for an example of this tree.

Text

Description automatically generated

Text

Description automatically generated

Figure 7: Snippet of the ingredients “check” tree

The user will be interacting with this program using command line prompts as shown above, they will be occasionally asked to input their top ingredients, cuisine and dietary preferences while also selecting recipes, “yes” or “no” inputs to the tree and reading the displayed recipe information from the command window. Finally, the user can type in “exit” to exit from the program.

Final Output Apr 16, 2023.

Text

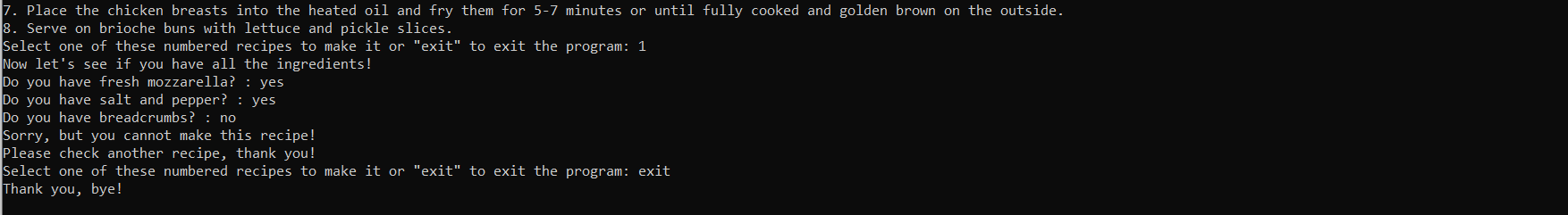
Description automatically generated  


Figure 8: Final output snipped as of Apr 16, 2023