

Report

1. The goal of our project

- a. Getting food/dish information from API mainly including its recipe and cost.
Analyze the data
- b. When user types in one type of food such as pork, beef, the program return many recipes.
- c. Analyze the cost of food in the past few years, taking into account inflation.

2. What we achieved

- a. Getting food/dish information from API mainly including its recipe, calculate cost of the recipe.
- b. When user types in one type of food such as pork, beef, the program return possible recipes.
- c. Analyze cost, calories, and nutrients of each recipe
- d. Provide a minor language translation feature

3. Problems we faced:

As Walmart closed its API during the time, we were unable to get enough information about many products. Tried the API Kroger, but the authentication process was very complex. Also, we could not find an API that could help us calculation the inflation. Hence, we changed our plan slightly: We did not analyze the price of food in the past years, instead, we dive into nutrients of the food, while also providing a language translation feature.

One major problem we faced with language translation is that matplotlib graph has a hard time recognizing many languages (possibly due to encoding problems), therefore, the places where we can put the translated text is very limited.

4. Calculation:

Calculate.py: calculate the cost of recipes using join command.

Data stored in cost.txt

5. Visualization

Upon different user input, the program creates different types of graph. Stored in "top_10_calories.png", "top_10_cost.png", "nutrients.png". "nutrients.png" uses more complex visualization than others. Running the program each time will produce different graphs, as more data are inserted in the database.

6. Instructions

Run project_main.py. All programs are intergraded in this main file.

- a. Input a language such as English, Spanish, or Chinese. Invalid input will make the program default to English
- b. Input what type of food you are interested in
- c. Input the matrix you care about (will result in different graph, as well as different data inserted into the database)

7. Documentation

- a. Calculate.py:

- i. **Calculate_cost_and_write()**

Calculate the cost of recipes in the database relating to the specific product, write it to the file.

1. Input:

a. language_abr – language of the program calculation output

b. product – the product that the user is interested in

2. output: Two files.

a. “cost.txt” contains the English version, which is subsequently used to graph the output.

b. “cost_translated.txt” contains the translated version

b. translation.py:

i. translateText()

Send request to the API and translate the given text based on the language

1. input:

a. language_abr – language of the program calculation output

b. text – the text needs to be translated

2. output: The translated text

ii. lang_abr()

Convert user input language to acceptable language abbreviation

1. input: language – input language

2. output: language abbreviation

c. graph.py:

i. graphTop10Calories():

Graph the recipes with the most calories in the database

1. Input: product – the product that the user is interested in
2. Output: Saved as “top_10_calories.png”

ii. graphTop10Cost():

Graph the recipes with the least cost in the database

1. Input: costs - Dictionary of the costs of recipes
2. Output: Saved as “top_10_cost.png”

iii. graphNutrients():

Graph random 20 recipes from the database with its nutrients information.

1. Input: product – the product that the user is interested in
2. Output: Saved as “nutrients.png”

d. Request_and_store.py:

i. Create_request_url():

Create urls needed for API request

1. Input: Product – the product that the user is interested in
2. Output: crafted URL for requesting API

ii. Request_recipe():

Request recipes from API

1. Input: food_search_url – url needed to request recipe from API

2. Output: A list of recipes

iii. SetupDatabase():

Set up, Stored data in the database

1. Input:
 - a. Product- product the user is interested in
 - b. Recipe – A list of recipes of the product
2. Output: Set up the database at “Food Data.db”

iv. RequestCost():

Request API the cost of each ingredient of each recipe, store them in the database

1. Input:
 - a. Product – product the user is interested in
 - b. Ingredient-url – the URL needed to request the ingredient API
2. Output: Produce new table “Cost” in the new database, storing the ingredient costs of each recipe

8. Resources Used

	Description	Location of Resources	
	Recipes	https://developer.edamam.com/	

	finding cost ingredients	spoonacular.com/food-api	
	translation	cloud.google.com/translate/docs	
	join + Where	mode.com/sql-tutorial/sql-joins-where-vs-on/	Successfully implemented
	matplotlib create a bar graph plot	matplotlib.org/3.1.1/gallery/lines_bars_and_markers/bt.html	Used and used methods to implement
	to integrate the Python and run a file one on another	www.tutorialspoint.com/How-can-I-make-one-Python-file-run-another	Used and implemented

9. Important Notes:

- a. The program limits number of queries to each API to at most 20. For the ingredient-cost API, the program uses a counter in the loop to keep track of number of queries, data already stored in the database will be directly retrieved and will not cost a query. For the recipes-search api, each search returns at most 20 recipes. On each run of the program, the starting index of each search is tracked via the max query id of the database.
- b. The ingredient API <https://developer.edamam.com/> limits Request 150 points/day, (approximately running the program 5 – 6 times/day). If ran out, I have commented out several API keys("ingredient_key") associated with different

gmail accounts I created for this API, so each one could be used to run another sets of test.

c. The google translation api <https://cloud.google.com/translate/docs> is essentially not free. We will have to limit the time used for the API.

d. Some ingredients cannot be found on the API. For those, a message will be printed, and the recipe with any ingredients not in the API will not be put in the "Cost" database, as the cost will not be accurate without any one of the ingredients information.

e. Each run of program will modify the database as well as the graph produced