GitHut repo link:

Data Sources:

Data Source 1: FoodData Central API
Documentation: https://fdc.nal.usda.gov/api-guide.html
The API provides information on food and nutrition, including macronutrients, micronutrients, and serving sizes.
Format: json

Access Techniques: The API provides information on food and nutrition, including macronutrients, micronutrients, and serving sizes. The API requests are made using the requests module in Python, with the query parameter specified in the API URL. Caching is also implemented using a dictionary to store the results of previous API requests, reducing the number of API calls and improving the efficiency of the program.

Data Summary:

- There is a total of 411,560 branded foods available
- Number of records retrieved will depend on the user's search input. For example, if the search item is “chicken”, the number of records retrieved is 18,014. If the search item is “celery”, the number of records retrieved is 155,39.
- Important properties include
    - fdcId: Unique ID of the food.
    - description: The description of the food.
    - foodCode: A unique ID identifying the food within FNDDS.
    - foodNutrients: nutritional information about the food
    - Ingredients: The list of ingredients - Only applies to Branded Foods.
    - score: Relative score indicating how well the food matches the search criteria

Data Source 2: Spoontacular
Documentation: https://spoonacular.com/food-api/docs
Format: json

Access Techniques: The access technique used is also an API. The Spoonacular API provides recipe search results based on user-specified query parameters. The API requests are made using the requests module in Python, with the API key and query parameters specified in the API URL. Similar to the USDA API, caching is implemented using a dictionary to store the results of previous API requests, reducing the number of API calls and improving the efficiency of the program.

Data Summary:

- There is over 5,000 recipes available
- Number of records retrieved will depend on the user's search input. For example, if the search term is “Chinese”, the number of records retrieved is 43. If the search term is “Italian”, the number of records retrieved is 275.
- Important properties include:
    - cuisine: The cuisine(s) of the recipes
    - diet: The diet(s) for which the recipes must be suitable
    - maxReadyTime: The maximum time in minutes it should take to prepare and cook the recipe
    - minCarbs & maxCarbs: The minimum and maximum amount of carbohydrates in grams the recipe must have per serving
    - minProtein & maxProtein: The minimum and maximum amount of protein in grams the recipe must have per serving
    - minCalories & maxCalories: The minimum and maximum amount of calories the recipe must have per serving
    - minFat & maxFat: The minimum and maximum amount of fat in grams the recipe must have per serving

Caching:

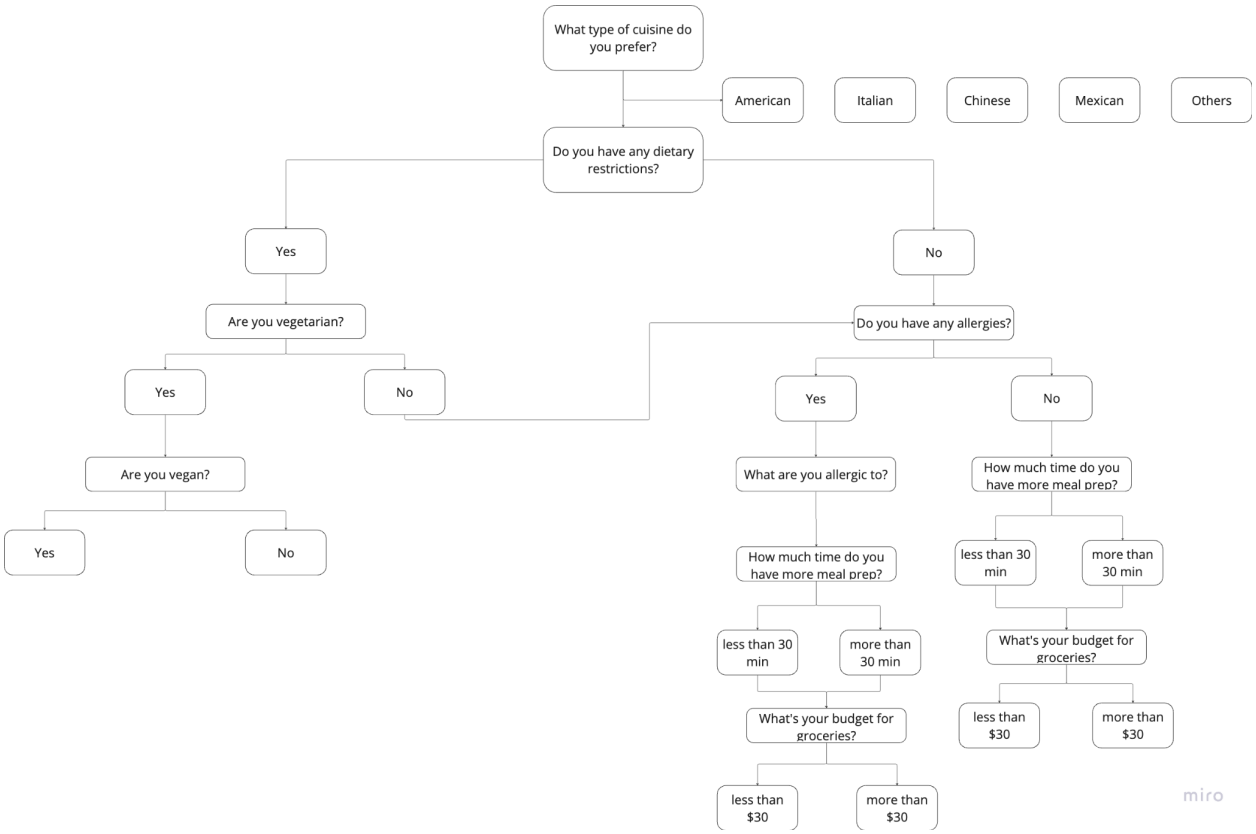
```
CACHE_FILE = "usda_cache.json"
CACHE_DICT = {}

def open_cache():
    ''' opens the cache file if it exists and loads the JSON into
    the CACHE_DICT dictionary.
    if the cache file doesn't exist, creates a new cache dictionary
    Parameters
    ----------
    None
    Returns
    -------
    The opened cache
    '''
    try:
        cache_file = open(CACHE_FILE, 'r')
        cache_contents = cache_file.read()
        cache_dict = json.loads(cache_contents)
        cache_file.close()
    except:
        cache_dict = {}
    return cache_dict


def save_cache(cache_dict):
    ''' saves the current state of the cache to disk
    Parameters
    ----------
    cache_dict: dict
        The dictionary to save
    Returns
    -------
    None
    '''
    dumped_json_cache = json.dumps(cache_dict)
    fw = open(CACHE_FILE,"w")
    fw.write(dumped_json_cache)
    fw.close()


def construct_unique_key(baseurl, params):
    ''' constructs a key that is guaranteed to uniquely and
    repeatably identify an API request by its baseurl and params
    Parameters
    ----------
    baseurl: string
        The URL for the API endpoint
```

Data Structure
Summary and is provided and makes sense given the description of data
Screenshots demonstrating progress or planning for organizing data into data structures

Here is the summary what how I plan to structure my data:
I will ask five main questions to get information from the user in order to generate a personalized meal plan:

1. What type of cuisine do you prefer?
2. Do you have any dietary restrictions?
3. Do you have any allergies?
4. How much time do you have for meal prep?
5. What's your budget?

Here is how I'm planning to structure it:



Interaction/Presentation

My plan is to use Plotly for visualization. Based on the type of data, I will determine which type of visualization to use for data representation. Here are some ideas:

- Establishing a scatter plot to show the relationship between budget spending and time spending on a meal to give users a better idea about the tradeoffs between these two variables.
- Line chart tracking calorie intake for a user each day
- Pie chart showing the breakdown of a recipe by ingredients and by nutritions including percentages of carbohydrates, proteins, and fats