RecipEat
Technology Review: Spoonacular, Edamam

Corina Geier, Jeffrey Lai, Edward Lou, Sai Muktevi & Andrew Zhou

Outline

- Project Overview
- Use Case
- APIs: Spoonacular and Edamam
- API Outputs
- API Pros and Cons
- Decision
- Tech Stack

Project Overview

- Goals:
    - Fast and simple to find recipes based on the ingredients you have at home
    - Easy to incorporate a balanced diet based on nutrient targets
- Requirements:
    - User can login
    - User can keep a running list of available ingredients
    - User can input ingredient-based & nutrient-based inputs
    - RecipEat will return some recommended recipes
    - RecipEat will provide a visual comparison of recipes

Use Case
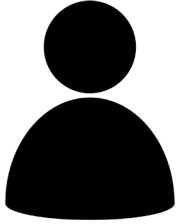
User: Max, a working dad

Objective: Max wants to make healthy recipes for his kids with the ingredients he has on hand so he doesn't have to run to the grocery store

Action: Max logs into RecipEat and enters the ingredients he has at home, along with some nutritional targets he has for him and his kids

Result: Max receives a list of the most relevant recipes and can view a visual comparison of the two

APIs: Spoonacular vs. Edamam

Spoonacular

- Free version allows for 150 points per day (Roughly 150 searches)
- Allow for filtering based on food intolerances and desired dietary goals
- Returns recipes based on ingredients based on "what's in your fridge"

Edamam

- Free version allows for 200 recipes per month
- Allows for analyzing nutrients of a meal given ingredients
- Can return recipes based on nutrient or ingredient inputs

Spoonacular Output

- Returns a JSON output that is easily converted into a pandas dataframe
- Based on ingredients and nutrient information will receive back a recipe with protein, calories, carbs, and fat.
- A .jpg file path is also returned for a visual of the recipe

Spoonacular Response

Search Recipes by Nutrients

```python
request_url = 'https://api.spoonacular.com/recipes/findByNutrients?'
apikey =
filterby = '&minCarbs=10&maxCarbs=50&number=2000'

url = request_url + apikey + filterby
payload={}
headers = {
  'Cookie': '__cfduid=dff952ebbf9c020c4f07c314e6bcb9c711613423774'
}

response = requests.request("GET", url, headers=headers, data=payload)

pd.DataFrame(response.json())
```

	calories	carbs	fat	id	image	imageType	protein	title
0	114	12g	6g	157109	https://spoonacular.com/recipeImages/157109-31...	jpg	4g	Raw Vegan Peanut Butter Pumpkin Bites
1	218	25g	10g	631763	https://spoonacular.com/recipeImages/631763-31...	jpg	8g	Warm and Luscious Sipping Chocolate with Xocai...
2	359	38g	21g	631769	https://spoonacular.com/recipeImages/631769-31...	jpg	6g	Bad Boy" Giant Double Chocolate Cookies
3	125	14g	7g	632168	https://spoonacular.com/recipeImages/632168-31...	jpg	2g	Almond Pistachio Cookied With Saffron Icing
4	279	33g	10g	632944	https://spoonacular.com/recipeImages/632944-31...	jpg	19g	Asparagus Soup
5	128	17g	6g	632952	https://spoonacular.com/recipeImages/632952-31...	jpg	6g	Asparagus Stir-Fry With Black Bean Sauce
6	358	47g	19g	633091	https://spoonacular.com/recipeImages/633091-31...	jpg	4g	Autumn Cheesecake

Edamam Output

- Returns a heavily nested JSON output that is not easily converted into a pandas dataframe
  - The JSON requires a lot of processing before converting to a DataFrame (JSON Normalization)
- Displays link to recipes with given ingredients
- Nutrition information is also returned, such as calories, sugars, and fats

Edamam Response

```json
1    {
2      "uri" : "http://www.edamam.com/ontologies/edamam.owl#recipe_ec290b2568b7d745a0762f6d5db4a042",
3      "yield" : 15.0,
4      "calories" : 21814,
5      "dietLabels" : [ "LOW_CARB" ],
6      "healthLabels" : [ "DAIRY_FREE", "MILK_FREE", "PEANUT_FREE", "TREE_NUT_FREE", "SOY_FREE", "FISH_FREE", "SHELLFISH_FREE" ],
7      "cautions" : [ ],
8      "totalNutrients" : {
9        "ENERC_KCAL" : {
10         "label" : "Energy",
11         "quantity" : 21814.306200000003,
12         "unit" : "kcal"
13       },
14       "FAT" : {
15         "label" : "Fat",
16         "quantity" : 1562.5579472000002,
17         "unit" : "g"
18       },
19       "FASAT" : {
20         "label" : "Saturated",
21         "quantity" : 538.5602623999998,
22         "unit" : "g"
23       },
```

Spoonacular: Pros and Cons

Pros:

- Well documented with extensive tutorials for support and maintenance
- API is not limited to recipes, includes grocery products and menu items
- Ability to search for recipes based on ingredients and nutrients using one endpoint
- Extensive list of nutrition filters available to filter on
- Output is easily converted into a pandas dataframe
- Includes over 365,000 recipes and 86,000 food products

Cons:

- Can only return a maximum of 100 recipes per search
- Some API functionalities are not listed out in detail as compared to Edamam.

Edamam: Pros and Cons

Pros:

- Has built in visualization tools for comparing recipes
- Edamam extracts nutrition and ingredient data from ingredient text
- Edamam returns information for calories, fats, carbohydrates, protein, cholesterol, and sodium for each ingredient or food entered

Cons:

- The API is split into 3 API's
  - Each has their own documentation
- Documentation is not very clean or easy to read
- Less ingredient and nutrient filters than Spoonacular
- No commercial use
- Output is a heavily nested JSON object, requiring a lot of processing to convert to a dataframe

Decision: Spoonacular

- Edamam and Spoonacular are similar solutions but overall Spoonacular is better suited for our purposes
- Spoonacular is better documented, only requires use of one API, and output is easily converted to pandas dataframe
- Spoonacular fulfills all of our requirements
  - Keep a running list of available ingredients - "what's in your fridge?" search
  - Input ingredient-based & nutrient-based inputs
  - Returns some recommended recipes - find similar recipes
  - RecipEat will provide a visual comparison of recipes

Tech Stack

- Python
- Flask
- Firebase - Realtime Database
- Pyrebase
- HTML, CSS, Bootstrap

Thank you!