Leftovers Recipe Generator Project Spec

General Description of the Project

The Leftovers Recipe Generator is a web app that helps users turn leftover food ingredients into new recipes. It allows the user to reduce food waste and make cooking more convenient. Users type in the ingredients they already have at home, and the app generates recipe suggestions pulled from the Spoonacular Recipe API, using its Python SDK to handle data requests.

The app will allow users to filter recipes by cuisine, dietary restrictions, or cooking time. The interface will be a web-based GUI that displays recipes in a ranked order based on the ingredients the user provides. Each recipe result will include step-by-step instructions, ingredient lists, and images. **Users will also be able to bookmark recipes to allow them to come back to these recipes at a later date. The recipe ranking will be based on the number of ingredient matches, but alternative ranking criteria, such as calories, could be added later.

The first version could run as a command line interface (CLI), where users enter a list of ingredients and receive text-based recipe suggestions. Later versions would run as a web app with an interactive UI.

Revision: The application will transition to a React-based web GUI, which will display interactive recipe cards with images, and clickable Instacart links, filters.

Task Vignettes

Vignette 1: User inputs ingredients and gets recipes

Justin opens the app because he wants to cook dinner with the few items left in his refrigerator. On the home page, he sees a search box that asks him to “Enter your ingredients.” He types in “eggs, cheese, spinach” and clicks search. The app processes his input and makes a request to the Spoonacular API to ask for recipes that best match his list.

Justin is presented with a set of recipe cards, each showing a title, image, cooking time, and ingredient match. At the top of the list are omelets and spinach quiche, since they use most of the ingredients Justin entered. Justin clicks on spinach quiche. A recipe page opens, showing the full list of ingredients, cooking instructions, and a larger image of the finished dish.

Technical details:

- User input: ingredient string, sent to Spoonacular API through Python SDK
- API response: JSON containing recipe list (title, ID, image, ingredients, instructions)
- Recipes ranked by ingredient match percentage
- Front-end displays results as cards; CLI fallback displays as text

Vignette 2: Filtering and refining search
Alex is in a rush and only has about half an hour to cook. She opens the app and enters "chicken, rice, broccoli" as her ingredients. The app analyzes the recipes available for those ingredients and presents a list of cuisines that have matching recipes. *Alex selects Asian cuisine from this list and sets a maximum cooking time of 30 minutes.* She clicks search and waits for results.

(Revision: selecting from available cuisines rather than the user typing their own)

The app queries Spoonacular with Alex's filters included in the request. Instead of a long list of generic recipes, the results now focus on quick, Asian-inspired dishes. At the top of the list are chicken stir fry and teriyaki chicken rice bowls, both of which meet her cooking time requirement. **Alex decides to save the teriyaki recipe for later, so she clicks the bookmark button. The recipe is stored in her personal recipe list, which she can revisit anytime.

Technical details:

- Filter options included in API request (cuisine, cooking time, diet)
- Recipes are pre-filtered by API, results displayed in ranked order
- **Bookmark feature stored in local JSON file or lightweight database with user ID/recipe ID relationship
- CLI fallback: user provides filters as command-line arguments

****Vignette 3: Bookmarking and revisiting recipes**
After trying the spinach quiche he discovered earlier, Justin comes back to the app to look for more meal ideas. On his homepage, he sees a bookmarked recipes section where the spinach quiche is saved from his last session. He clicks on it and quickly pulls up the full recipe again, complete with ingredients and directions.

Justin appreciates that he doesn't have to re-enter his ingredients to find recipes he liked before. He then decides to enter a new set of leftovers, chicken, rice, and carrots, and is shown a fresh set of matching recipes. The bookmarking feature makes it easy for him to keep track of old favorites while he continues to explore new dishes.

Technical details:

- Bookmarked recipes stored locally (JSON or small database) keyed by user ID and recipe ID
- Data fetched again from Spoonacular when recipe is reopened, ensuring it's up to date
- Bookmark section displayed on user's home/dashboard page
- CLI fallback: displays a numbered list of bookmarked recipe titles to re-open

Technical Flow

The Leftovers Recipe Generator follows a structured data flow that connects user inputs with external recipe data, processes it, and returns recipes. The system can be divided into six main blocks: **User Input, Query Builder & API Request, API Response Handling, Ranking & Filtering, Storage & **Bookmarks, and Output & Display.**

1. **User Input**
   The process begins when the user enters a list of ingredients (e.g., "eggs, spinach, cheese") and optionally applies filters such as cuisine type, dietary restrictions, or maximum cooking time.

   ○ **Data Type:** string for ingredients, dictionary for filters.

   ○ **Flow:** This data is passed to the query builder module.

2. **Query Builder & API Request**
   The input is formatted into a valid API request using the Spoonacular Python SDK. This ensures ingredient lists and filter parameters are properly structured for the API call.

   ○ **Data Type:** formatted dictionary or URL string.

   ○ **Flow:** The request is sent to Spoonacular's servers.

3. **API Response Handling**
   Spoonacular returns a JSON response containing recipe information such as titles, images, cooking times, and ingredient match details. The app parses this JSON into Python dictionaries and lists for further processing.

   ○ **Data Type:** JSON object, parsed into lists/dictionaries.

   ○ **Flow:** Data passed to ranking and filtering logic.

4. **Ranking & Filtering**
   Recipes are ranked based on how many of the user's ingredients they match. Filters (such as cuisine) are applied before the results are displayed.

   ○ **Data Type:** list of recipe dictionaries with metadata (title, ID, match score, image, time).

   ○ **Flow:** Processed list passed to display module.

5. ****Storage & Bookmarks**
   When users bookmark a recipe, the recipe ID and user ID are stored locally in a JSON file or small database. This allows the app to fetch and display saved recipes later by querying Spoonacular again for up-to-date details.

   ○ **Data Type:** dictionary mapping

- **Flow:** Bookmark data saved locally, re-fetched when needed.

6. **Output & Display**
   In the web-based GUI version, recipes are displayed as cards showing the title, image, cooking time, and ingredient match. In a CLI fallback, results are displayed as plain text in a ranked list. Clicking or selecting a recipe leads to a full recipe page with instructions and ingredients.

   - **Data Type:** rendered HTML (GUI) or formatted text (CLI).

   - **Flow:** Results shown to the user, with options to explore or bookmark.

Possible future additions:

- Shopping list generator: create a list of missing ingredients for easier grocery planning.
  (Revision: The user will be given a link to instacart for the missing ingredients)

- Meal planning: schedule recipes across multiple days with combined shopping lists.
  (Revision: Likely will not be added to the final application)

- Personalized recommendations: tailor recipe suggestions based on past activity and **bookmarks.

Unknowns

- How to handle ambiguous ingredient inputs ("cheese" vs. "cheddar" vs. "parmesan")?

- What data storage method (JSON file vs. small database) works best for bookmarks?

*Revision: **Bookmarking feature will only be implemented if time allows*