Project Overview:

This project is a web application that allows users to search for food items and retrieve nutritional information using the Edamam API for recipe results. The app has a Python Flask backend and a simple HTML frontend. The core functionality includes:

- Allowing users to input a food item and select dietary restrictions (e.g., Vegan, Gluten-Free).
- Fetching recipes using the Edamam API based on the user's input and displaying the top 10 results.
- Displaying the results in a clean and responsive format for better user experience.

Parts Implemented So Far:

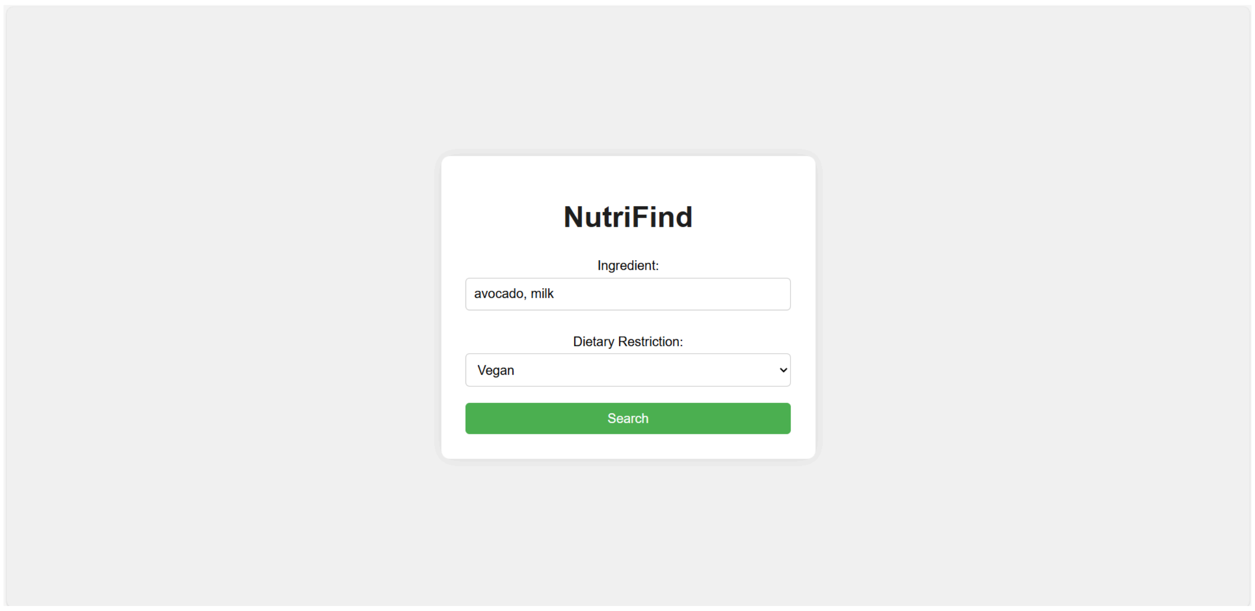
- API Integration: The Edamam API has been successfully integrated into the Flask backend. The backend now handles user queries for food items, retrieves the corresponding nutritional data and recipes, and sends the results to the frontend.
- UI Enhancements: The user interface has been improved to allow users to input a food item and choose dietary preferences (Vegan, Vegetarian, Gluten-Free, etc.). A results page (results.html) has been created to display the top 10 results in a structured and user-friendly layout.
- Dynamic Results Display: The app is designed to show the search results seamlessly, providing a responsive experience as the results are displayed on the results page without requiring a full page reload.

Demonstration of How the App Works:

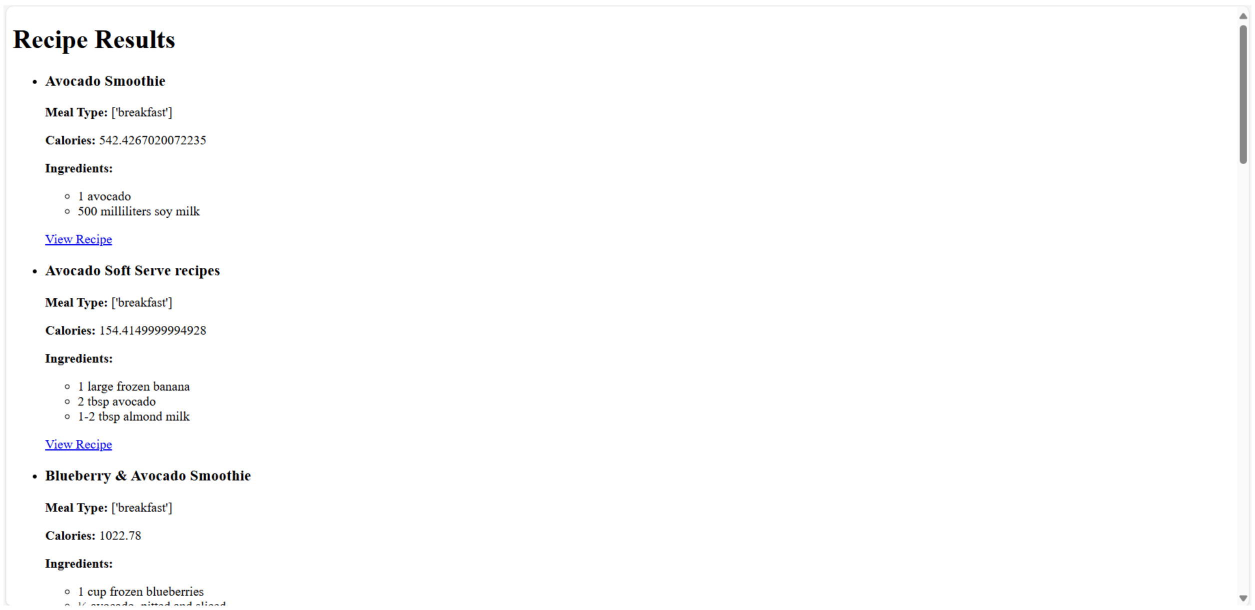
- Step 1: The user visits the home page and sees the title "NutriFind" centered on the page with a search form below it.
- Step 2: The user inputs a food item (e.g., "apple") and selects a dietary restriction (e.g., "Vegan"). They then click "Search."
- Step 3: The search request is sent to the backend, which queries the Edamam API for both nutritional data and recipe suggestions.
- Step 4: The results (nutritional information and recipes) are displayed on the results page, providing the user with recipe details in an organized manner.

Screenshots:

Home Page: Shows the title "NutriFind," the search bar, and the diet preference dropdown. When the user selects search, it goes to the next page that shows the results.



Results Page: A table with the top 10 recipes and nutritional information for the food item.



Recipe Results

- **Avocado Smoothie**

  Meal Type: ['breakfast']

  Calories: 542.4267020072235

  Ingredients:

  - 1 avocado
  - 500 milliliters soy milk

  View Recipe
- **Avocado Soft Serve recipes**

  Meal Type: ['breakfast']

  Calories: 154.4149999994928

  Ingredients:

  - 1 large frozen banana
  - 2 tbsp avocado
  - 1-2 tbsp almond milk

  View Recipe
- **Blueberry & Avocado Smoothie**

  Meal Type: ['breakfast']

  Calories: 1022.78

  Ingredients:

  - 1 cup frozen blueberries

Issues Encountered & Solutions:

1. Flask could not find index.html due to improper file placement. To resolve, I created the templates folder and placed index.html and results.html there, as Flask expects HTML files to be inside the templates directory.
2. Displaying results dynamically on the results page was initially tricky due to difficulties in handling the data flow between the backend and frontend. Adjusted how the backend processes and sends the data, ensuring that the results are properly formatted and displayed in the table on the frontend without reloading the page.

Unsolved Issues & Proposals:

1. Some API queries return inconsistent data (e.g., some recipes do not have complete nutritional information).
    - Proposal: Implement error handling on the backend to filter out incomplete data before sending it to the frontend. This would prevent partial results from being displayed. However, I am not too sure on how this can be done.
2. Styling of the results page could be improved for better user experience.
    - Proposal: Allocate more time for frontend development, potentially using a CSS framework like Bootstrap to enhance the UI.

Help Needed:

1. The current method of handling table updates on the frontend feels cumbersome, and I'm looking for guidance on improving this process to make it more efficient and user-friendly.
2. I might need help with frameworks like Bootstrap to implement better UI and UX.

Milestones for the Next Weeks:

1. Clarify Search Input Options

- **Task:** Update the placeholder text and/or label for the input field to clearly indicate that users can search for multiple ingredients. For example, enter ingredients (avocado, apple, chicken).
- **Goal:** Improve user understanding of the search functionality by clarifying that they can enter multiple search terms.

2. Add "Search Again" or "Go Back" Button

- **Task:** Implement a "Go Back" or "Search Again" button on the results page that takes the user back to the main search page, or rethink the layout so that results appear below the search box on the same page.
- **Goal:** Simplify the navigation for users and allow them to search multiple times without needing to reload or manually go back.

3. Refactor Search Results Display

- **Task:** Modify the results page to show just the recipe titles in a list since showing the full recipe details for all 10 results may be overwhelming. When a title is clicked, display the detailed information for that recipe (e.g., calories, ingredients).
- **Goal:** Improve the user experience by reducing clutter and allowing users to focus on the recipes they are interested in.

4. **UI/UX Improvements**

- **Task:** Rework the results page to potentially display results on the same page below the search box, with options to filter or sort the recipes by calories or other criteria. I am considering implementing these UI changes using a CSS framework like Bootstrap.
- **Goal:** Improve the app's overall look and feel, make the interface more intuitive, and enhance the user experience with better navigation and layout.

Self Reflection:

- **Progress:** So far, I'm satisfied with the progress, especially with the successful API integration and the ability to return multiple recipes. However, the UI/UX aspects definitely need improvement.
- **Timeline:** With the new milestones in mind, I believe I can complete the next version in the coming weeks.
- **Expectations vs. Reality:** I initially thought integrating the API would be the hardest part, but improving the UI/UX and making the user flow smooth has proven to be more challenging than expected.
- **Light Bulb Moments:** I realized how important it is to guide users with clear input labels and provide easy navigation options. UI clarity and a seamless user experience are just as crucial as backend functionality.