NutriFind Project Specification

General Description of the Project

NutriFind is a web-based application that enables users to discover foods that align with their specific dietary needs and preferences. The app provides a comprehensive nutrition database, allowing users to filter foods based on criteria like low-carb, high-protein, gluten-free, or vegan. NutriFind simplifies the process of finding suitable foods, making it easier for users to plan meals that adhere to their health goals and dietary restrictions.

The application will be built using Python and Flask to handle backend processing and user input. A nutrition database will be sourced from publicly available datasets such as the USDA FoodData Central to ensure accurate and comprehensive nutritional information. The app's frontend will be developed using HTML and CSS for a clean and user-friendly interface, with JavaScript enhancing interactive elements. This combination of technologies allows NutriFind to be both scalable and efficient in processing user requests.

The ideal GUI will feature an intuitive layout where users select their dietary preferences using interactive dropdowns, checkboxes, and sliders. While the primary version is web-based, the app can also function with a Command Line Interface (CLI) in a minimal form, where users specify their dietary preferences via text input. In the future, the app could be extended to include a REST API, enabling third-party applications to access the database and filtering functionalities remotely.

Task Vignettes

1. **Setting Dietary Preferences:**
   Sarah is looking to maintain a gluten-free and high-protein diet. She opens NutriFind and sees a form with a variety of dietary preferences such as gluten-free, low-carb, dairy-free, etc. Sarah selects "Gluten-Free" and "High-Protein" from the options and clicks the "Filter Foods" button.

   Technical Details:
   - User input is captured through a series of checkboxes and dropdowns.
   - The selected preferences are sent to the Flask backend via an HTTP request.
   - The backend uses the filtering logic to process the input against the nutrition database.

2. **Viewing Filtered Results:**

After Sarah submits her dietary preferences, NutriFind processes the data and presents her with a list of foods that match her criteria. Each item on the list shows detailed nutritional information, including calories, protein content, vitamins, and minerals. Sarah clicks on "Quinoa" to see more details.

Technical Details:

- The backend retrieves data from the nutrition database using the user's input criteria.
- Filtered results are returned as a response to the frontend.
- The frontend dynamically generates a table displaying the filtered foods.

3. **Refining Search Criteria:**
   Sarah decides she also wants to see only foods with low sugar content. She updates her search by selecting "Low-Sugar" from the options and clicks "Update Results." The list automatically refreshes, displaying foods that are gluten-free, high-protein, and low-sugar.

   Technical Details:

   - The filtering process is triggered again, incorporating the new dietary restriction.
   - The updated results replace the previous list without reloading the entire page.

4. **Saving Dietary Preferences:**
   Sarah wants to save her dietary preferences for future use. She clicks the "Save Preferences" button, and the app prompts her to create an account or log in. Once she logs in, her preferences are saved, allowing for easy retrieval next time she uses the app.

   Technical Details:

   - User preferences are stored in a database linked to their account.
   - Flask manages the user authentication and data storage.
   - The frontend offers an option to load saved preferences upon login.

Technical Flow

1. User Input: Users select dietary preferences using the GUI. These inputs (lists of selected criteria) are sent to the Flask server via HTTP requests.

2. Data Retrieval & Processing: Flask processes the request and queries the comprehensive nutrition database (likely stored in a SQL or NoSQL database) to retrieve foods that match the selected criteria. The data is filtered using a Python algorithm that compares each food item against the user's preferences.

3. Data Presentation: The filtered data is converted to JSON format and sent back to the frontend. The GUI displays the results in a clean, tabular format using HTML/CSS, and JavaScript enables dynamic interaction.

Data Types:

- Input: User-selected preferences (dictionaries/lists).
- Database: A structured table of foods (SQL database), each entry containing fields like food name, calories, protein, carbs, fats, vitamins, minerals, etc.
- Output: Filtered list of food items in JSON format.

Data Flow Diagram (Conceptual)
User Interface → User Preferences → Flask Backend → Filtering Logic → Nutrition Database

Data types:

- User preferences: List of dietary restrictions (e.g., ["gluten-free", "high-protein"])
- Database entries: Dictionary with keys for nutrient values (e.g., {"name": "quinoa", "protein": 14g, "gluten-free": True})
- Output: JSON array of matched foods

Additional Features and Considerations:

- Remote Control/API: A REST API could be developed for third-party integration, allowing external apps to query the NutriFind database with specific dietary preferences.
- Localization: The application could adapt to different units (e.g., metric vs. imperial) based on user location or preferences.

Biggest Implementation Challenges:
I can work on designing and developing the user interface, including creating the HTML/CSS layout and the filtering form. With guidance, I can also help implement the filtering logic using Python and Flask, ensuring that the application accurately processes user inputs and displays relevant results. Deploying the application on a web platform like GitHub Pages will finalize the project, making it available to users.

This spec provides a clear blueprint for NutriFind, a user-friendly app that offers personalized food recommendations based on specific dietary needs and preferences.

Github: https://github.com/uma01234/nutrifind