WDD 330 Final project Proposal

Recipe finder

# BYU – Web Frontend Development

Mozart Armstrong Bandeira Soares

Trello board: [https://trello.com/invite/b/67e6132aa3a870adb9dc4727/ATTIb17a1673ef21893640499b5fce29b04f4EB709C2/recipe-finder]

# **Overview**

This project aims to solve the common problem of deciding what to cook with the ingredients you already have. Many people struggle to find recipes that utilize the ingredients they have on hand, leading to food waste and meal planning frustration. The "Recipe Finder" addresses this issue by allowing users to input a list of ingredients and quickly discover suitable recipes. Furthermore, it provides immediate nutritional information for those ingredients, allowing users to make informed decisions about their meals based on nutritional content.

My motivation for working on this project stems from a desire to create a practical and useful tool that I myself and my friends can use, it simplifies meal planning and promote healthier eating habits. I am also interested in learning how to effectively integrate and utilize multiple APIs to create a richer user experience.

# **Target Audience**

The target audience for the project is broad, encompassing anyone who cooks at home and seeks inspiration for meal planning. Specifically, it will serve:

* Home cooks of all skill levels: From beginners to experienced chefs, anyone can benefit from quickly finding recipes based on available ingredients.
* Health-conscious individuals: The inclusion of nutritional information for ingredients allows users to make informed choices based on dietary needs and preferences.
* Budget-conscious individuals: By using ingredients they already have, users can reduce food waste and save money.
* Busy individuals: The app provides a quick and easy way to find recipes, saving time on meal planning.

# **Major Functions**

The web app will provide the following major functions:

1. **Ingredient Input:** The user will be able to enter a list of ingredients using a text field. The application will accept individual input fields for each ingredient.
2. **Recipe Search:** The application will send a request to the Spoonacular API, using the user's inputted ingredients as the query parameters.
3. **Recipe Display:** The application will display a list of recipes returned by the Spoonacular API, showing the recipe title, a thumbnail image, and a brief description (if available).
4. **Nutritional Information Retrieval:** For each ingredient entered by the user, the application will send a separate request to the Edamam Food Database API to retrieve its nutritional information.
5. **Nutritional Information Display:** The application will display the retrieved nutritional information (calories, protein, carbohydrates, fat, etc.) for each ingredient *alongside* the recipe results. This allows the user to see the nutritional impact of their ingredient choices.
6. **Recipe Details Display (Optional):** When the user clicks on a specific recipe, a detailed view will be displayed, showing the full recipe instructions and a complete list of ingredients. (Implementation depends on time constraints).
7. **Error Handling (Recipe Search):** If no recipes are found for the given ingredients, a user-friendly message will be displayed, suggesting alternative ingredient combinations or search terms.
8. **Error Handling (Nutritional Info):** If nutritional information is not found for a specific ingredient, a message will be displayed indicating that the information is unavailable.
9. **Loading Indicators:** While the data is being retrieved from the APIs, a loading indicator will show, giving users feedback about the application's current state.
10. **Clear Ingredients:** A button to clear all ingredients, so users can start over.

# External Data

This project will utilize two external APIs:

1. Spoonacular API (https://spoonacular.com/):

This API will be used to search for recipes based on user-provided ingredients. The API returns a JSON object containing an array of recipes. Each recipe object includes attributes such as:

- id: Unique recipe ID.

- title: Recipe title.

- image: URL of the recipe image.

- imageType: Type of the image.

- usedIngredients: An array of ingredients used in the recipe.

- missedIngredients: An array of ingredients the user is missing.

- unusedIngredients: An array of ingredients the user did not use.

- likes: Number of likes the recipe has.

2. Edamam Food Database API (https://www.edamam.com/):

This API will be used to retrieve nutritional information for individual ingredients it returns a JSON object containing detailed nutritional information, including:

- calories: Total calories.

- totalNutrients: An object containing detailed nutrient information (protein, fat, carbohydrates, vitamins, minerals, etc.). This object contains many nested attributes.

- totalDaily: An object containing the recommended daily intake for nutrients.

- label: Name of the ingredient.

- image: URL of the ingredient image.

# Initial Module List

The project will be structured with the following modules

* index.html: The main HTML file, serving as the entry point for the web application and containing the basic structure (header, main content area, footer).
* css/: A directory to contain all CSS files

style.css: The primary CSS file for global styling and layout.

components.css: (Optional) A CSS file dedicated to styling individual components (e.g., recipe cards, input fields). This promotes modularity.

* js/: A directory to contain all JavaScript files

app.js: The main JavaScript file, responsible for initializing the application, handling user events, and coordinating data flow between modules.

api.js: A JavaScript module dedicated to handling all API calls to Spoonacular and Edamam. This module encapsulates the API interaction logic, making it reusable and maintainable

it should contain functions for fetching recipes and nutritional information.

components.js: A JavaScript module responsible for creating and managing UI components dynamically (e.g., creating recipe cards, displaying nutritional information). This promotes reusability and separation of concerns.

utils.js: A utility module for helper functions that are used across multiple modules (e.g., formatting data, error handling, local storage management).

* assets/: A directory to store static assets.

images/: A directory to store images used in the application (e.g., the application icon, placeholder images).

# Graphic Identity

Color Scheme:

* Primary Color: #4CAF50 (Green) - Represents freshness, health, and natural ingredients.
* Secondary Color: #FF9800 (Orange) - Represents energy, warmth, and appetite.
* Accent Color: #F0F0F0 (Light Gray) - Provides a clean and neutral background.

Typography:

* Headings: Montserrat - A modern and bold sans-serif font for clear headings.
* Body Text: Open Sans - A highly readable and versatile sans-serif font for body text.

Application Icon:

The application icon will be a simple green leaf with a stylized cooking pot silhouette inside. This visually represents both the natural ingredients and the cooking aspect of the application.

Specific Styling Elements:

* Rounded corners will be used for buttons and input fields to create a softer and more modern look.
* Subtle shadows will be added to elements to create depth and visual interest.
* A consistent button style will be used throughout the application to maintain a unified look and feel.

I plan on using bootstrap as a CSS framwork for this assignment to help with responsiveness.

# Timeline (Weeks 5-7)

Week 5:

\* Deliverable: Set up project structure (HTML, CSS, JavaScript files and folders).

\* Deliverable: Implement ingredient input form and basic styling.

\* Deliverable: Implement API call to Spoonacular based on ingredient input.

Tasks:

* [ ] Create folders
* [ ] Create boilerplate
* [ ] Read user input
* [ ] Fetch api information

Week 6:

\* Deliverable: Display recipe results (titles and images).

\* Deliverable: Implement API call to Edamam for nutritional information.

\* Deliverable: Display nutritional information for ingredients alongside recipe results.

Tasks:

* [ ] Display fetched API information
* [ ] Fetch second API
* [ ] Display second API information

Week 7:

\* Deliverable: Implement error handling and loading indicators.

\* Deliverable: Implement recipe details view.

\* Deliverable: Finalize styling and ensure responsiveness.

Tasks:

* [ ] Fix bugs
* [ ] Implement second results page
* [ ] Improve CSS and webpage looks

# Challenges

I anticipate the following challenges in building this application:

1. API Rate Limits: Both Spoonacular and Edamam APIs may have rate limits on the number of requests I can make per day or per minute. So for testing purposes I will use mocked data

2. Data Parsing and Formatting: The data returned by the APIs may be in different formats and may require parsing and formatting before it can be displayed in the application. I will need to write code to handle this data transformation.

3. Asynchronous JavaScript: Making multiple API calls and handling the asynchronous responses can be challenging. I will need to use Promises or async/await to manage the asynchronous operations effectively.

4. Responsiveness: Ensuring that the application is responsive and works well on different screen sizes (desktop, tablet, mobile) will require careful planning and testing.