**Recipe Organizer-Plan, Cook, and Organize with Ease!**

**Design Document**

**Author:** Ansh Gangola

**Scope:** Need to know

**Last Modified on:** 19 February 2025

**Table of Contents:**

| **S.No** | **Contents** |
| --- | --- |
| 1. | Overview |
| 2. | Overall Architecture   * Technology Stack  1. Frontend 2. Backend |
| 3. | Features   * Add/Delete Recipe * Search Recipe * View Recipe List * Meal Planner * Shopping List Generation |
| 4. | Schema   * Recipe Schema * Ingredients Schema |
| 5. | Future Scope |

**Overview:**

The **Recipe Organizer** is a web-based application designed to help users efficiently manage their recipes by allowing them to add, search, and organize recipes with categorized ingredients, dietary tags, and cooking steps. It also includes a **Meal Planner** to plan meals for the week and generate a **shopping list** based on selected recipes, ensuring a seamless cooking experience.

Key Features:

* Add new recipes with ingredients, steps, and dietary tags.
* Filter recipes by category, ingredients, or tags.
* Browse, view, and delete saved recipes.
* Plan weekly meals and generate a shopping list.

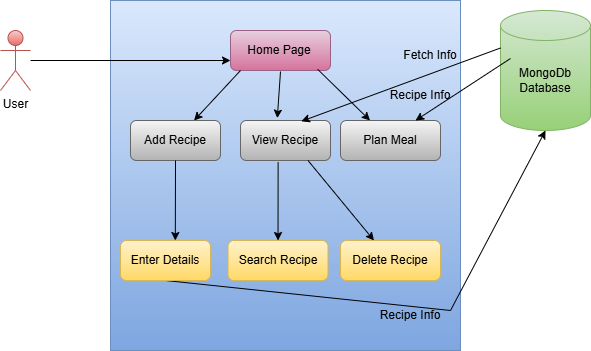
This design document provides a detailed overview of Recipe Organizer’s architecture, functionality, and implementation. It serves as a comprehensive guide for developers, stakeholders, and future members to understand the application's structure and design decisions.

The document is structured as follows:

1. **System Architecture:** An overview of the application's technical stack, database design, and overall structure.
2. **User Interface Design:** Detailed wireframes and descriptions of the user interface, focusing on user experience and interaction flow.
3. **Core Functionality:** In-depth explanations of key features such as Adding/Viewing Recipe, Meal Planning and Shopping list Generation.
4. **Data Management:** Description of data models, storage solutions, and data flow within the application.

This document aims to provide a clear and comprehensive understanding of Recipe Organizer’s design and functionality, serving as a roadmap for development and a reference for future enhancements.

**Overall Architecture:**



**Technology Stack -**

**Frontend:**

* **Framework:** ReactJs
* **Styling:** CSS

**Backend:**

* **Framework:** Node.js
* **Database:** MongoDB

**Features**

1. **Recipe Addition and Validation:**

Users can add new recipes by specifying the name, category , ingredients (with proper units and quantity), cooking steps, and dietary tags.

**Backend Implementation**

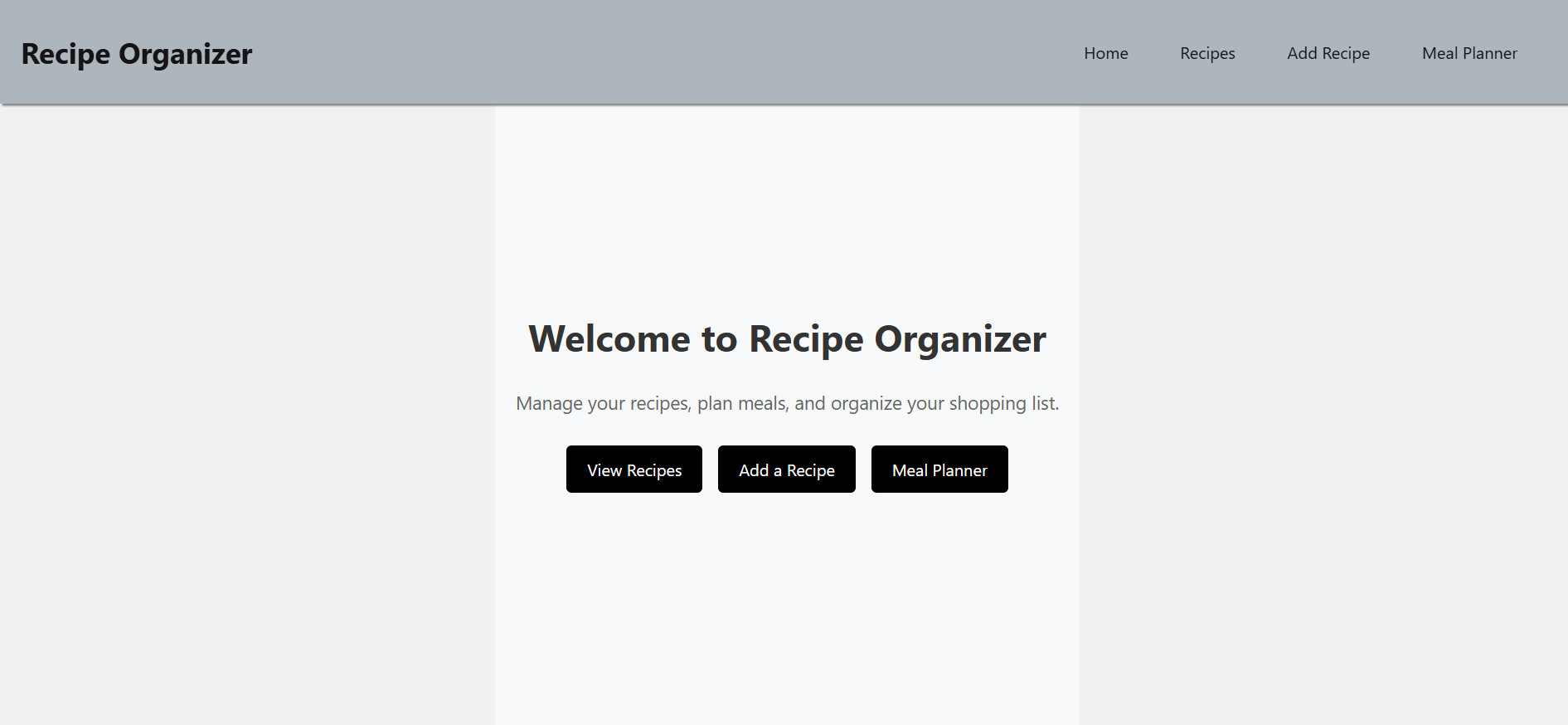
* Add Recipe:
  + Receives a POST request with recipe details.
  + Stores the data in MongoDB using Mongoose schema.
  + Responds with a success message or proper error.
* Recipe Validation:
  + Checks for all the necessary fields to be non-empty.
  + Validates the Name given from the database for duplicates.

**Frontend Implementation**

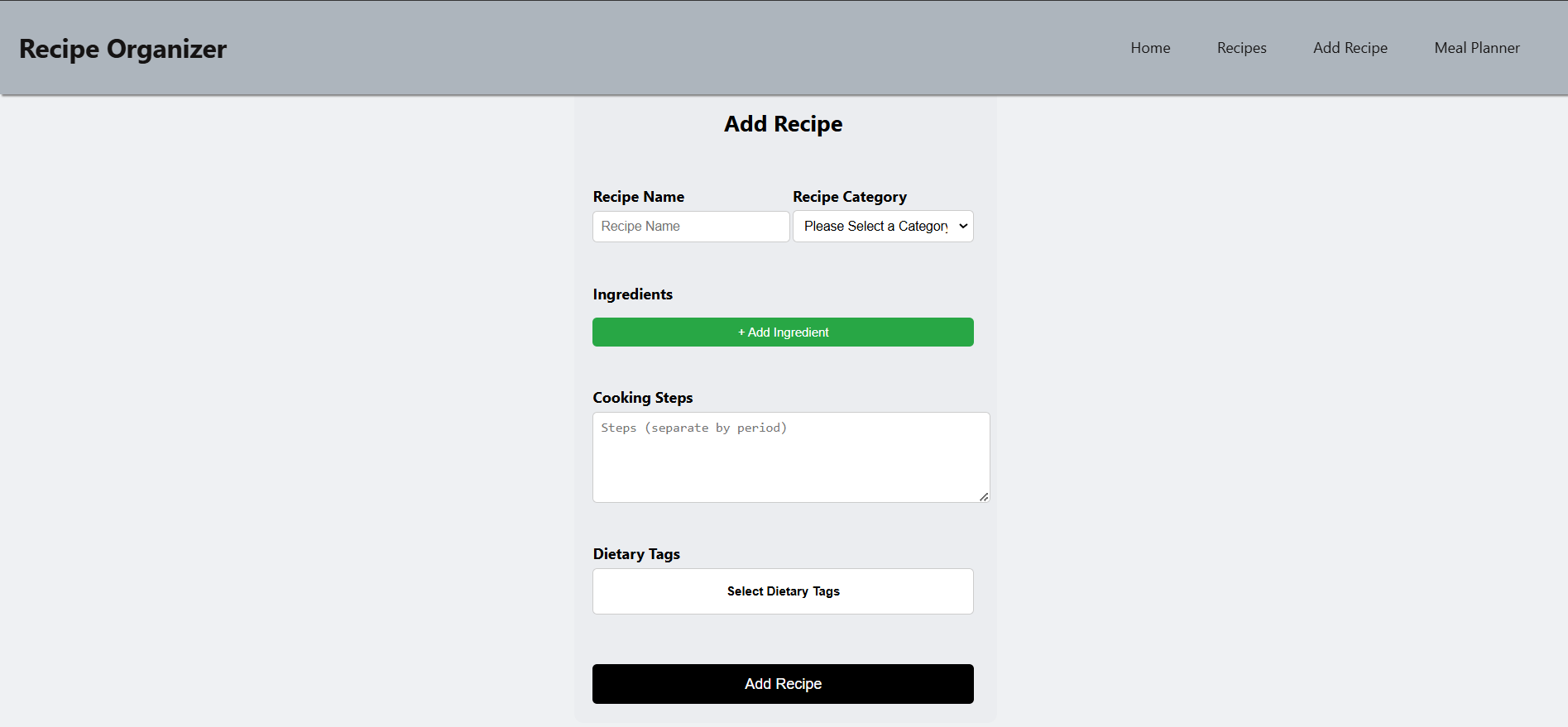
* Add Recipe:
  + Displays a form where users enter recipe details.
  + Click on add ingredients to add ingredients.
  + On submission, send a POST request to the backend.
  + Updates the UI to reflect the new recipe in the list.

**Screenshot:**

**Landing Page (Home Page):**

****

**Add Recipe Page:**

****

1. **View ,Search and Delete Recipe:**

Users can view all the recipes added. They can search recipes by Category (Breakfast, Lunch, Dinner, Snacks), Dietary tags ,or by ingredients. They can also delete any recipe by clicking on the delete button on the recipe card.

**Backend Implementation**

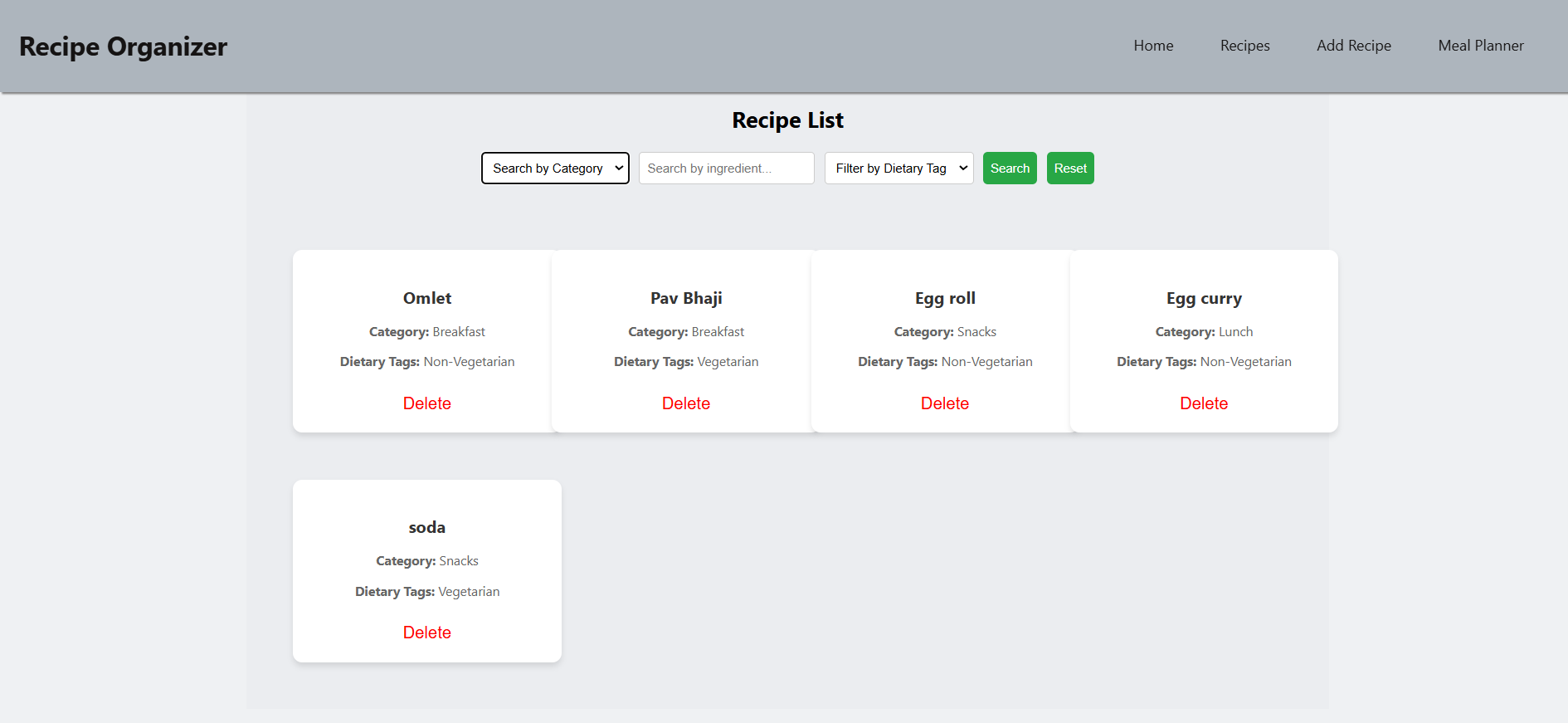
* **View Recipe Cards:**
  + A GET request is made to the database.
  + All the data from the database is received.
  + Appropriate success or error message is shown.
* **Searching a Recipe:**
  + Receives a GET request with a search query (category, tag ,or ingredient/s).
  + Searches the MongoDB database using regex or indexed search.
  + Returns the matching recipes as a response.
* **Delete Recipe:**
  + Receives a delete request.
  + Finds and removes the recipe from the database.
  + Responds with a success message or error.

**Frontend Implementations:**

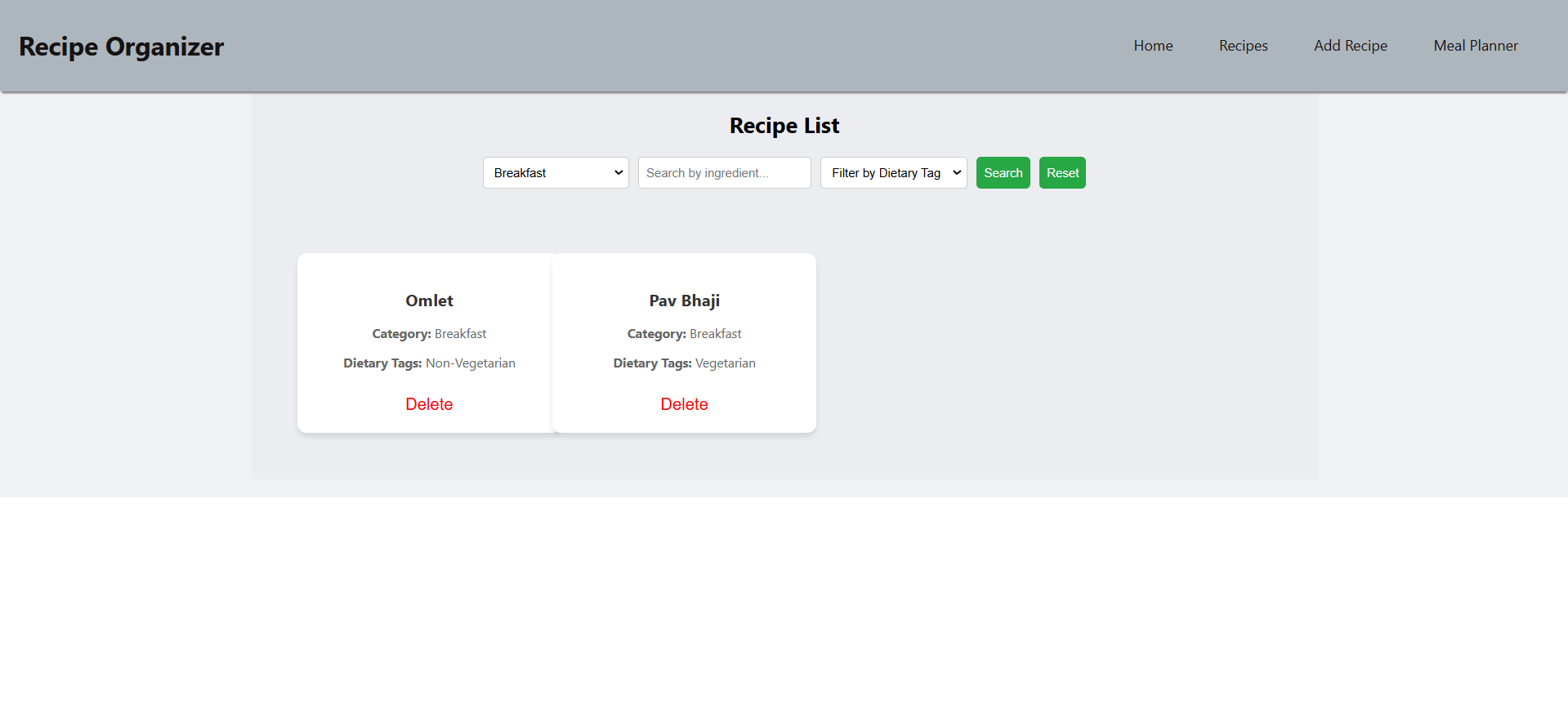
* **View Recipe Cards:**
  + Every newly added recipe is added .
  + A card view is maintained.
  + Dynamic card additions are done for every new recipe.
* **Searching a Recipe:**
  + Provides a search bar/dropdown menu for input.
  + On search, sends a request to the backend with the search term.
  + Displays the filtered results dynamically.
  + On clicking Reset the page is reset.
* **Delete Recipe:**
  + Users click on the delete button.
  + A delete request is triggered.
  + UI updates by removing the deleted recipe from the list.

**Screenshots:**

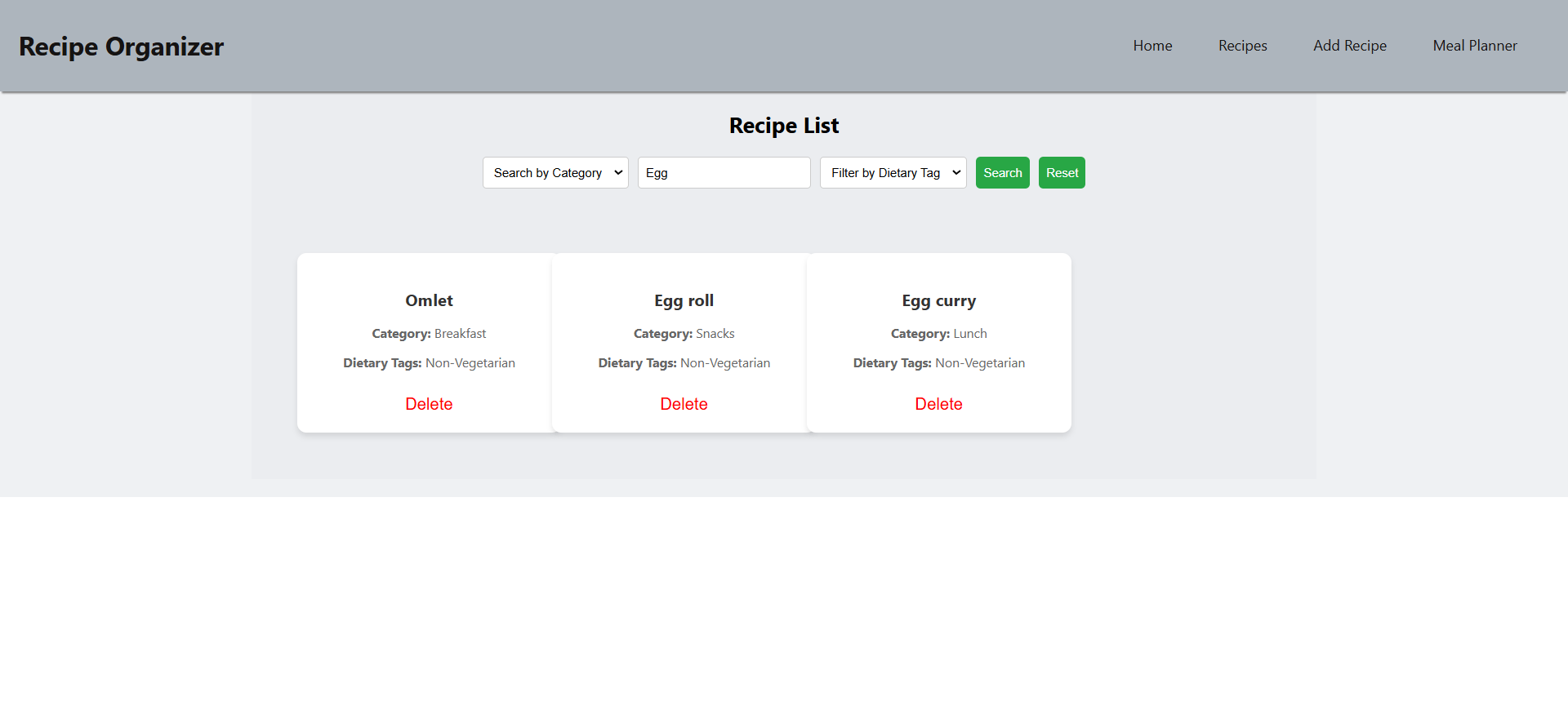
**View Recipe Cards:**



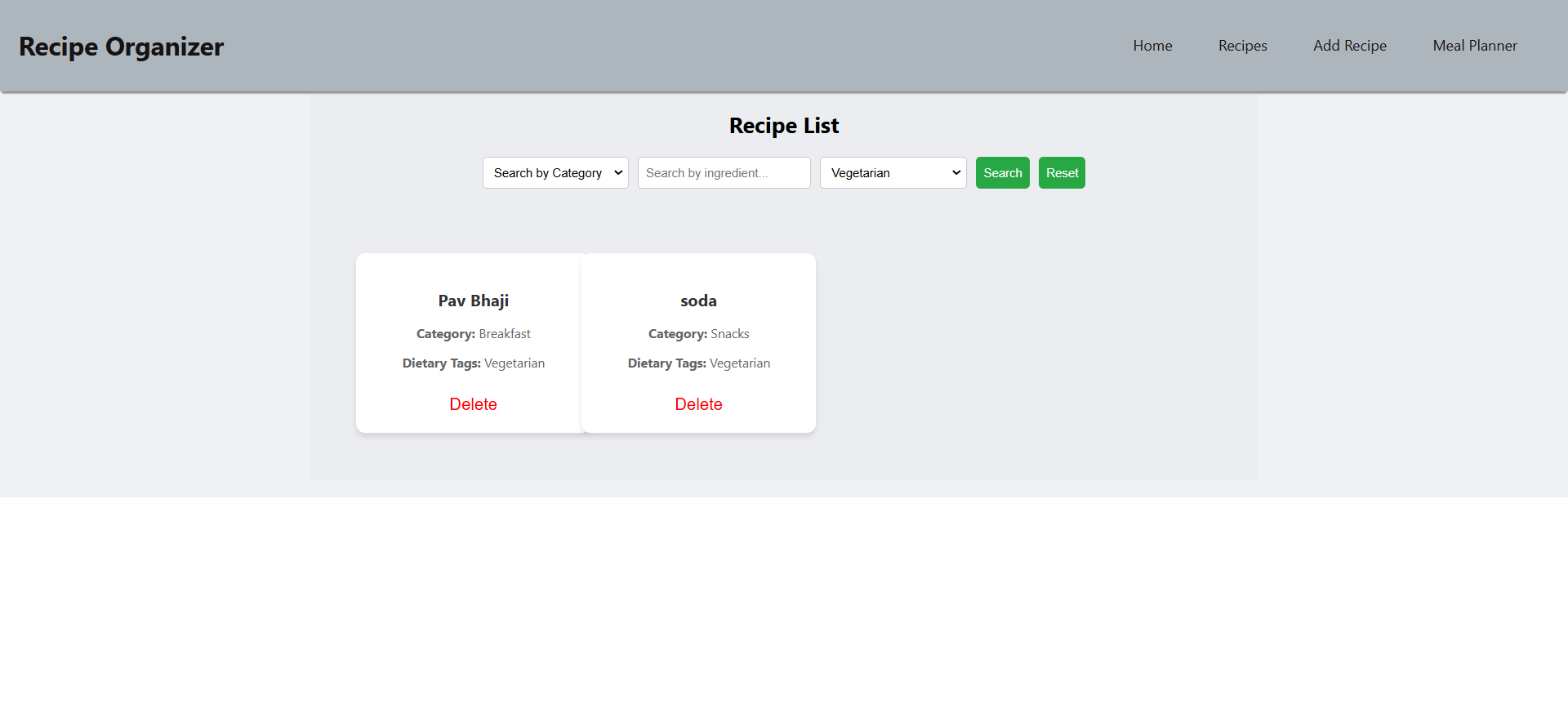
**Search by Category:**

****

**Search by Ingredient:**

****

**Search by Tag:**

****

1. **Meal Planner:**

Helps users assign recipes to different days of the week. Users can select any recipe from the available ones and add any recipe to any day of the week. They can also change the recipe assigned to any particular day.

**Backend Implementation**

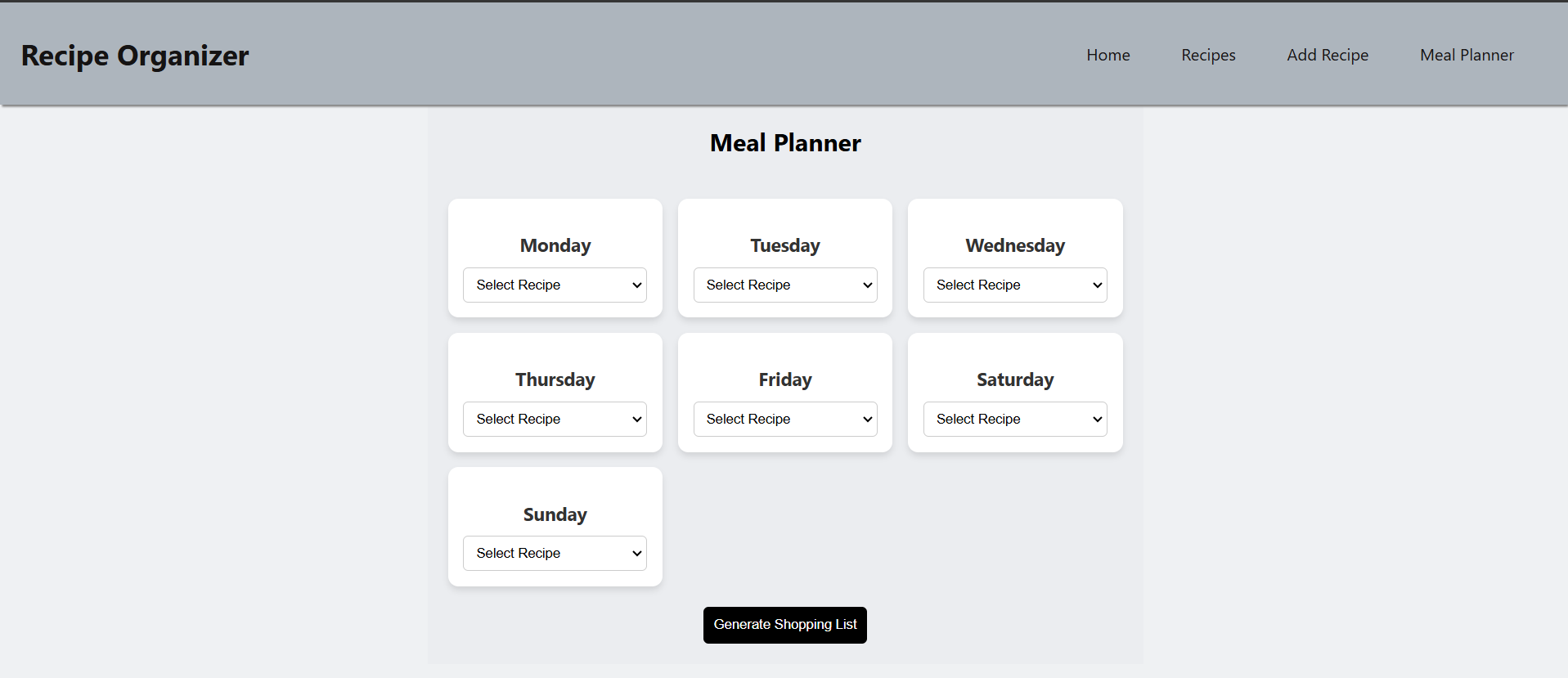
* Receives a POST request with selected recipes and dates.
* Stores meal plans in MongoDB under a separate collection.
* Returns a confirmation response.

**Frontend Implementation**

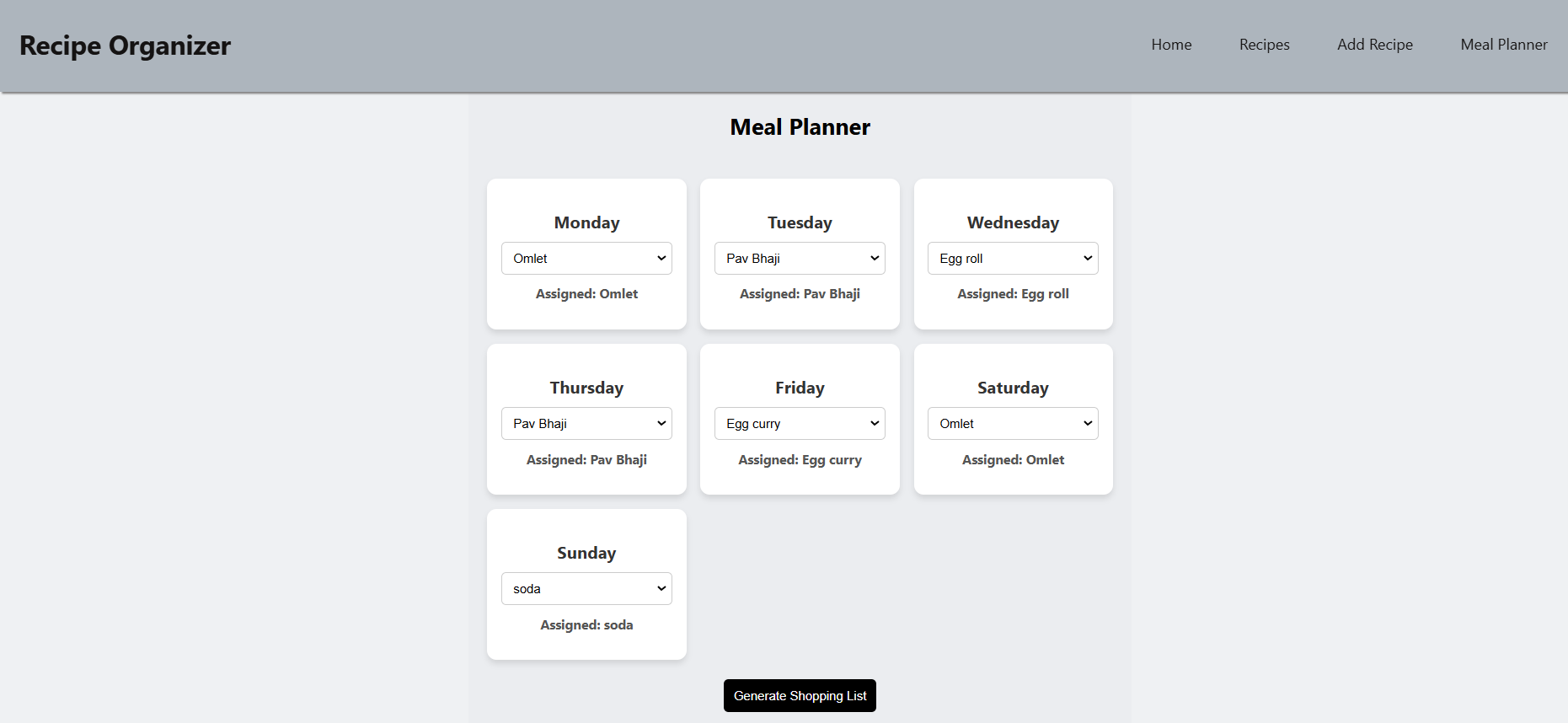
* Lets users select recipes and assign them to days.
* On submission, send a POST request to the backend.
* Displays recipe under the assigned day.

**Screenshots:**

**Meal Planner Page:**



**Recipe Assignment to days:**



1. **Shopping List Generation:**

Users can generate an aggregated shopping list for all the ingredients required for an entire week based on the recipes assigned to different days.

**Backend Implementation**

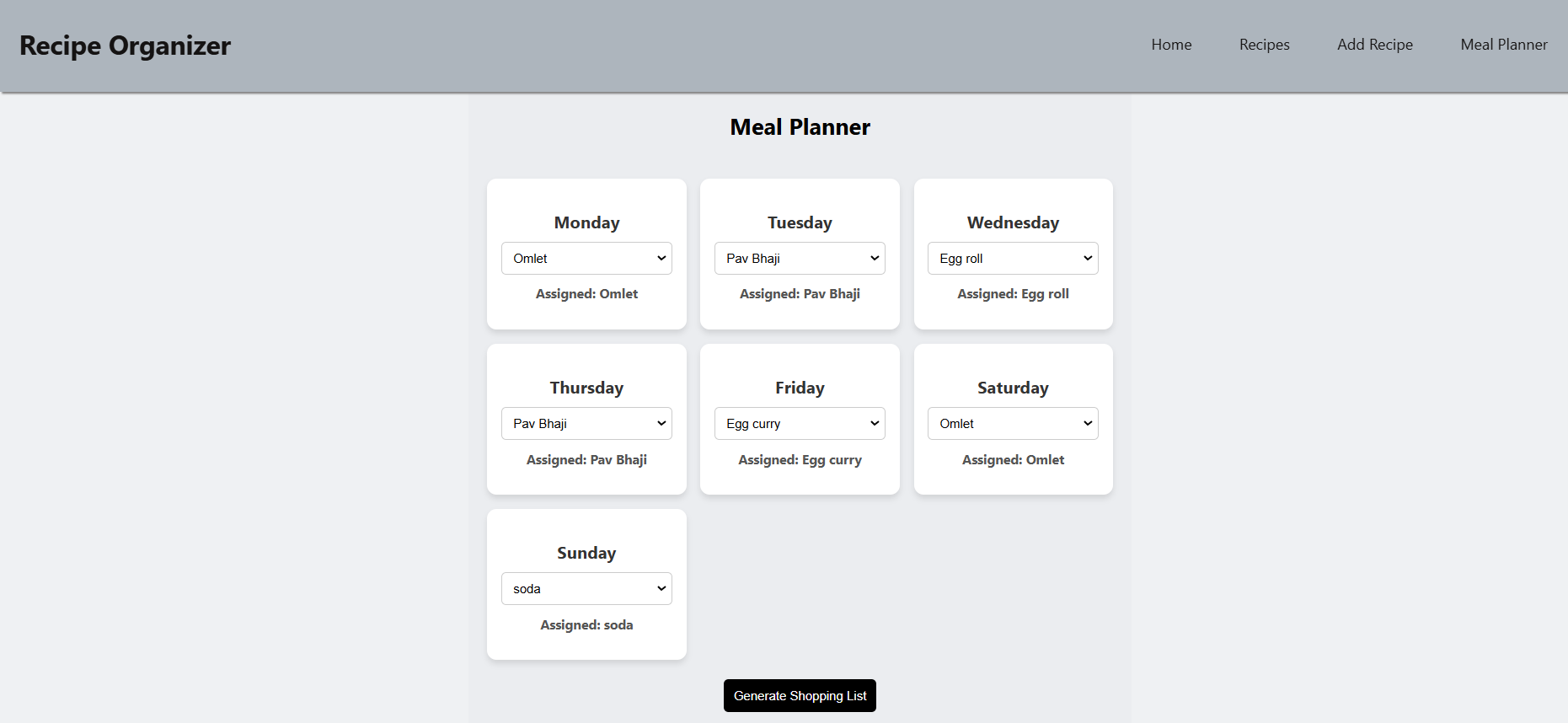
* Receives a GET request with the selected recipes.
* Extracts required ingredients from the recipes.
* Returns a compiled shopping list.

**Frontend Implementation**

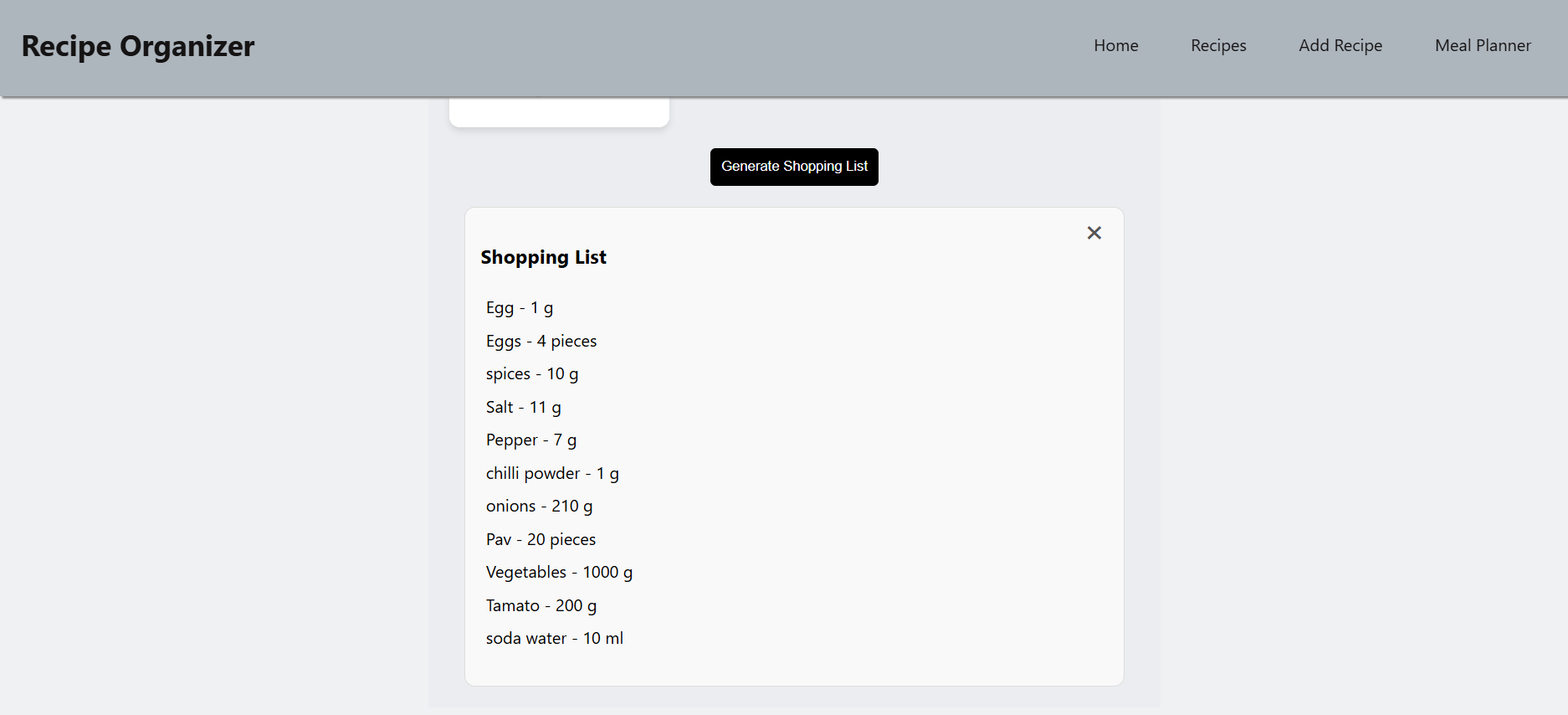
* Displays the ingredients in the form of a list.
* The cross button closes the list display.

**Screenshot:**

**Recipes Assigned to days:**



**Shopping list generated:**



**Schema**

**Recipe Schema:**

| **Field Name** | **Type** | **Attributes** | **Description** |
| --- | --- | --- | --- |
| name | String | required:true | Stores the name of the recipe. |
| category | String | required:true | Stores the category of the recipe (Breakfast, Lunch, Dinner, Snacks). |
| ingredients | Array of objects | - | Stores the list of ingredients used in the recipes along with their quantity and units. |
| steps | String | required:true | Stores the cooking steps. |
| tags | String | - | Stores the dietary tags for the recipes. |

**Ingredient Schema:**

| **Field Name** | **Type** | **Attributes** | **Description** |
| --- | --- | --- | --- |
| name | String | required:true | Name of the ingredient used in the recipe. |
| quantity | String | required:true | Stores the quantity used in the recipe. |
| unit | String | required:true | Stores the unit for the quantity of ingredient used(“g”,”ml”,”pieces”). |

**Future Scopes:**

* **Calorie Tracking:** Add nutritional information for each ingredient to calculate total recipe calories.
* **Portion Scaling:** Adjust ingredient quantities dynamically based on the desired serving size.
* **Image URLs:** Store and display images of recipes for better visualization.