The Bananaboat Cookbook

Aditya, Geoffrey, Isaac, Kim, Kirk, Nirja

June 14, 2025

Abstract

This document provides a comprehensive overview of the Custom Recipe Manager web application. It details the features, functionality, and technical implementation of the project, including the database schema, user features, and API integration. This report serves as a guide to the application's usability and design.

1 Introduction

The Bananaboat Cookbook is our recipe manager app. It is designed to help organize and share cooking recipes with other users. This was made using py4web, Vue.js, and integrates TheMealDB API. It also utilizes SQL databases to store all the user, ingredient, and recipe information.

2 Application Setup and Installation

Users should run our web app the same as other py4web-based web apps in the class. First, enter the Group 8 project repository (ucsc2025-cse183/project-8). Once there, within the terminal, enter `py4web run apps`. Then, go to recipe_manager to access the app.

3 User Features and Functionality

This section details the core features available to users of the Recipe Manager, including account creation, adding ingredients, and creating recipes.

3.1 User Accounts

Once users have an account set up, they can simply log in by entering their username and password. Otherwise, users can sign up by including a unique username, email, and password, as well as their first and last name.

3.2 Ingredient Management

The application provides a shared database of ingredients that all users can access. All added ingredients may also be viewed in the "All Ingredients" tab.

3.2.1 Searching for Ingredients

Users can search for ingredients by name. To do this, enter a string of letters or word(s) in the search bar located above the grid of ingredients and hit the 'Search' button. To make a new search, users may hit the 'Clear' button and enter an entirely new input, or enter a new input and hit the 'Search' button again.

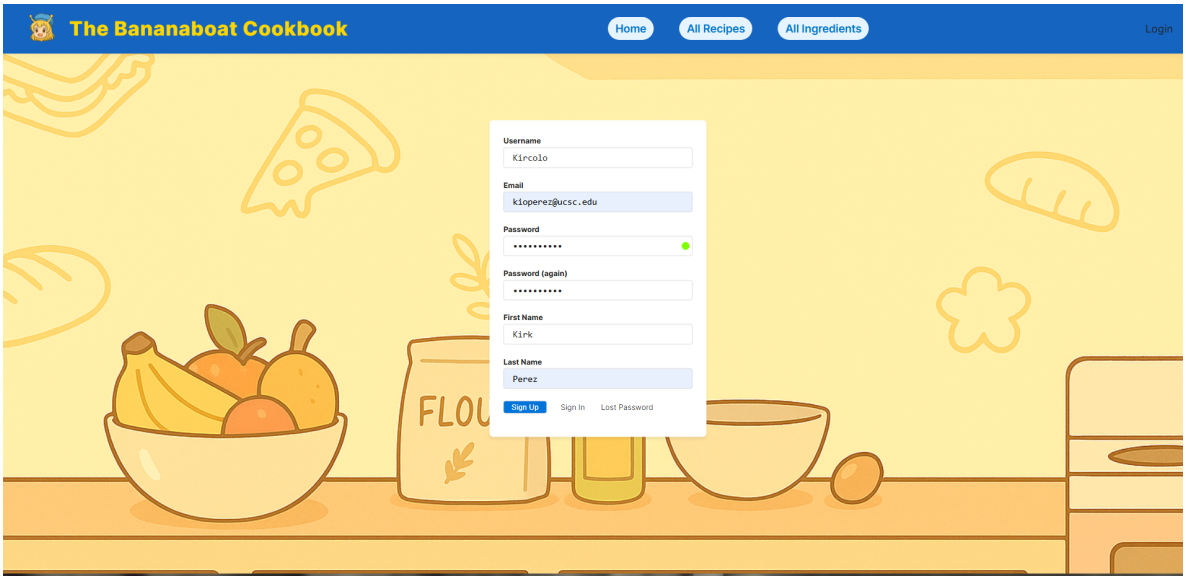

Figure 1: Screenshot of the login.

3.2.2 Adding New Ingredients

Users can contribute to the shared ingredient library by adding new ingredients. These ingredients are then available for all users to use in their recipes, and cannot be edited once created. Users can add a new ingredient by going into the "Add New Ingredient" box and specifying the Name, Unit, Calories Per Unit, and providing a simple description for the ingredient. Only an ingredient 'Name' is required to successfully add a new ingredient.

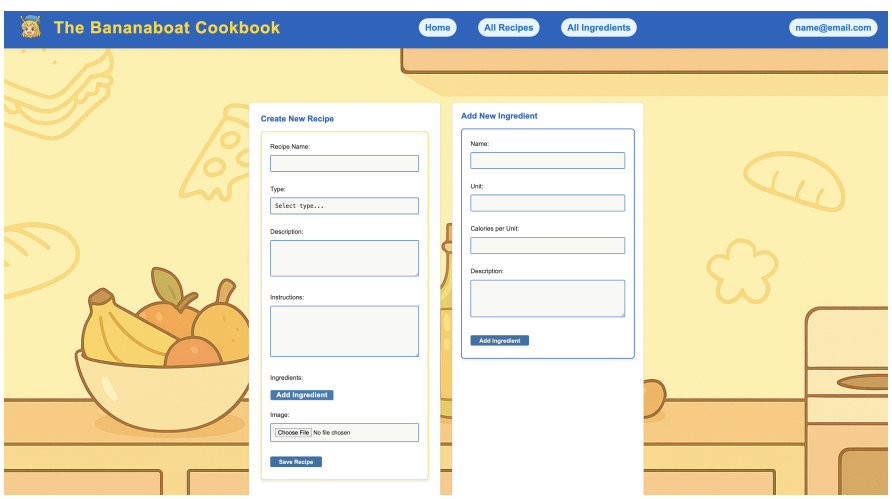

Figure 2: The form for adding a new recipe or ingredient.

3.3 Recipe Management

All recipes in the application are public and can be viewed by anyone. To view recipes, users should click the "All Recipes" button in the app's navigation bar.

3.3.1 Searching and Browse Recipes

The application allows for searching recipes by name and type. Users can do this by going into the "All Recipes" tab at the top, which will then show a search bar as well as the option for type of recipe.

The two other options next to it will be to confirm the search or clear.

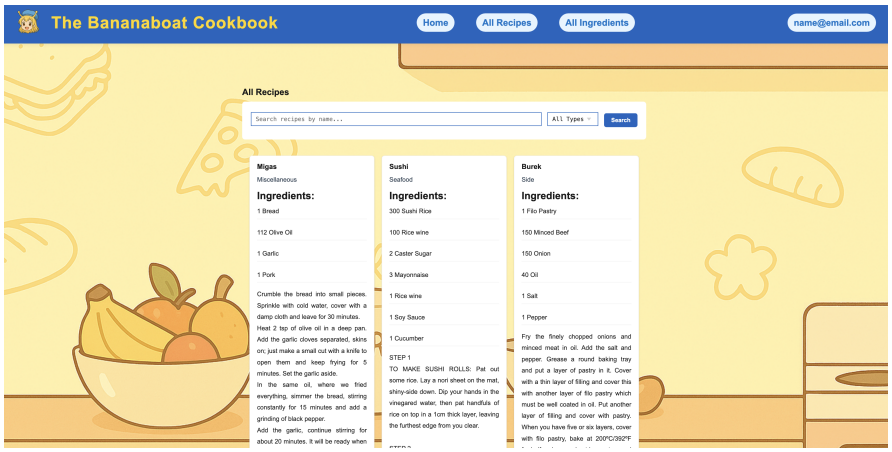

Figure 3: The public recipe browse and searching interface.

3.3.2 Creating a New Recipe

Users can create their own recipes. To do so, enter a name, type, ingredients, and a description for the recipe; only a "Recipe Name" and a "Type" are required. Once the necessary fields have been filled, hitting the "Save Recipe" button will trigger a pop-up window that confirms a recipe has been saved successfully.

Figure 4: The public ingredient browse and searching interface.

3.3.3 Editing a Recipe

Users can only edit the recipes they have authored. To do so, click on the "Edit" button and update the information as desired. Once the desired recipe has been edited, clicking the "Save Recipe" button will display the pop-up window to confirm that the edited recipe was successfully saved.

Security Logic The interface ensures that only the recipe's author sees the edit button, using a conditional check:

```
<button v-if="recipe.author === currentUser" @click="editRecipe(recipe)">Edit Recipe</button>
```

This means that the "Edit Recipe" and "Delete Recipe" button are only visible to the user if they authored the post. On the backend-side, the PUT and DELETE endpoints validate that the current user is the recipe author, further preventing any unauthorized updates even if the client is tampered with.

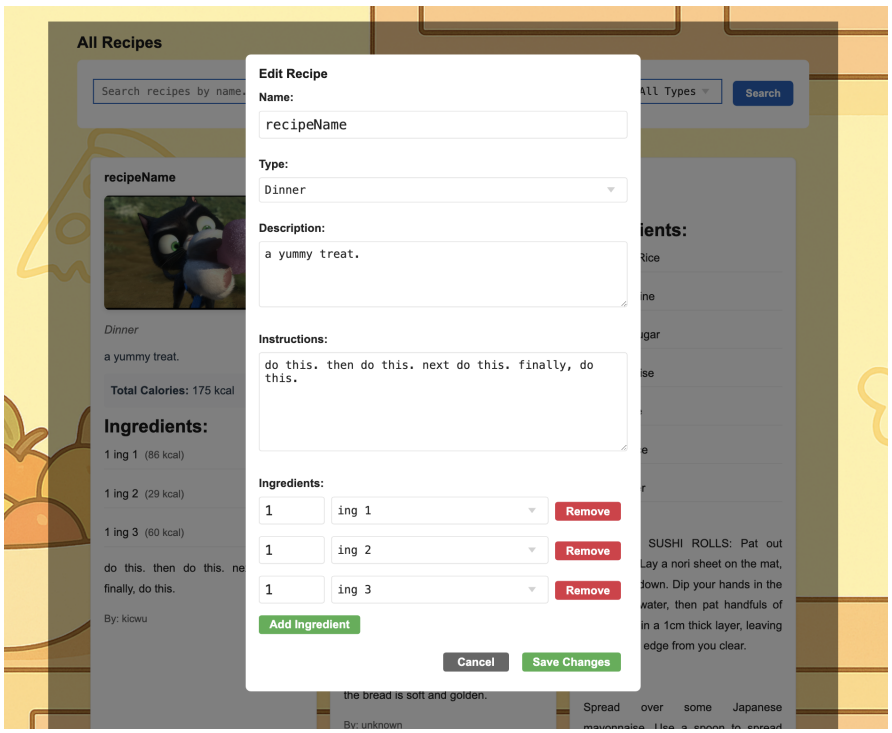

Figure 5: The recipe editing view, available only to the author.

4 Technical Implementation

This section covers the technical details of the project.

4.1 Database Schema

The application is a database-driven application with support for user accounts, ingredients, and recipes. The schema consists of three main tables.

- **Ingredients Table:** Stores ingredient data with fields for name, unit, calories_per_unit, and description.
- **Recipes Table:** Stores recipe data with fields for name, type, description, image, author, instruction_steps, and servings.
- **Linking Table:** Connects recipes and ingredients, with fields for recipe_id, ingredient_id, and quantity_per_serving.

4.2 Automated Calorie Calculation

The application automatically computes the total calories for each recipe by summing the calories contributed by each linked ingredient. This is done by multiplying the ingredient's `quantity` (as specified per recipe) by its `calories_per_unit` value.

$$\text{Total Calories} = \sum (\text{quantity} \times \text{calories_per_unit})$$

If any ingredient does not have calorie information (shown as `N/A`), it is excluded from the calculation. As a result, recipes containing such ingredients will not display total calorie information.

4.3 External API Integration

4.3.1 TheMealDB Data Import

A one-time import was implemented to populate the database with recipes and ingredients from `TheMealDB` API. This import fetches publicly available recipes and their associated data using the endpoint:

`https://www.themealdb.com/api/json/v1/1/search.php?s=`

The import process follows these steps:

1. Sends a `GET` request to the API endpoint.
2. Parses the JSON response and extracts a list of meals.
3. For each meal:
   - A new recipe is inserted into the database using the meal's name, category as description, and instructions.
   - Up to 20 ingredients are extracted by checking keys like `strIngredient1` through `strIngredient20`.
   - Each ingredient is either reused (if it already exists in the database by name) or inserted as a new record with an empty unit and `None` for calories.
   - A mapping between the recipe and each ingredient is created using the `recipe_ingredients` table. Quantities are parsed from the provided string using a helper function.
4. After all meals are processed, the transaction is committed.

The quantity parser (`parse_quantity`) attempts to extract a numeric value from the measurement string. If parsing fails or yields no digits, it defaults to 1.0.

This import ensures that users have a base set of recipes and ingredients to interact with immediately after setup.

4.3.2 Public Recipe Search API

The application provides a public, JSON-based API for searching recipes.
Endpoint:

`/recipe_manager/api/recipes`

Usage:
This endpoint returns a list of all available recipes. It accepts optional query parameters to filter recipes by name and/or type. It is accessible without authentication.
Optional Query Parameters:

- `name` (string): Filters recipes whose names contain the specified substring (case-insensitive).
- `type` (string): Filters recipes by the specified type (e.g., "Breakfast", "Lunch", "Dinner", "Dessert").

Example Request:

```
GET /recipe_manager/api/recipes?name=avocado&type=Breakfast
```

Example JSON Response:

```
{
  "recipes": [
    {
      "id": 3,
      "name": "Avocado Toast",
      "type": "Breakfast",
      "description": "A simple and healthy avocado toast recipe.",
      "instructions": "1.Toast bread.\n2.Mash avocado.\n3.Spread on toast.\n4.Add salt and pepper.",
      "total_calories": 210,
      "image": "avocado_toast.jpg",
      "author": "julia_dev",
      "ingredients": [
        {
          "ingredient_id": 5,
          "name": "Avocado",
          "quantity": 1,
          "unit": "whole",
          "calories": 160
        },
        {
          "ingredient_id": 7,
          "name": "Bread",
          "quantity": 2,
          "unit": "slices",
          "calories": 50
        }
      ]
    }
  ]
}
```


Response Details:

- `recipes`: A list of recipe objects matching the filters (or all recipes if no filters are applied).
- Each recipe includes:
  - `id`: Unique recipe ID.
  - `name`: Recipe title.
  - `type`: Recipe category.
  - `description`: Short summary of the recipe.
  - `instructions`: Preparation steps.
  - `total_calories`: Total calorie count.
  - `image`: Filename of the associated image.
  - `author`: Username of the author.
  - `ingredients`: List of ingredient objects, each with:
    - `ingredient_id`: Ingredient identifier.
    - `name`: Ingredient name.
    - `quantity`: Quantity used in the recipe.
    - `unit`: Measurement unit (e.g., slices, grams).
    - `calories`: Number of calories for the specified quantity.

5 Extra Credit Features

5.1 Optional Feature: Public Ingredient Search API

This feature allows users to search for ingredients by name using a public interface. When the page loads, it sends a GET request to:

```
/recipe_manager/api/ingredients
```

Users can then enter a name into the search input. Clicking the `Search` button sends a filtered request:

```
/recipe_manager/api/ingredients?name=<search_term>
```

The server responds with a list of matching ingredients, which are rendered in a responsive grid format using Vue.js. Each item displays the ingredient's name, unit, calories per unit, and description if applicable. If no ingredients match the search, a message is shown prompting the user to add new ones.

The `Clear` button resets the search input and reloads all available ingredients.

- A public search API for ingredients. `GET /api/ingredients?name=x`.

6 Conclusion

To summarize, we've created a web-based recipe manager that allows users to create, browse, and share recipes. This application includes an initial sign-in/sign-up process, leading into the main site, which includes boxes for users to add a new ingredient or create a new recipe. This data is then stored in two tabs at the very top of the site, where users can see everything listed altogether. Additionally, there is also a home screen to lead you back to the original screen. Lastly, there is also an option on the top right of the screen for users to edit their personal information.