The Story Behind My Application

The idea for this application stems from when we first moved onto our property seven years ago. It was a hectic time—our business was just beginning to take off, and life was full of challenges. My wife, being the meticulous planner that she is, was firm about sticking to a budget and planned weekly grocery trips. Each week, she would ask my son and me what meals we wanted for dinner. Without fail, our answer was always, "We don’t know."

I can only imagine how frustrating it must have been for her to not only handle the grocery planning but also decide on meals and cook for us. Meanwhile, we "kitchen-challenged" boys were of no help. After weeks of this, I started thinking about how much easier it would be if an app could generate meal plans and recipes automatically. That’s where the inspiration for this application began.

How the Application Works

This app is a starting point for something great. It begins with the user entering their name—while it doesn’t serve a significant purpose yet, it personalizes the experience. After that, the user selects how many meals they want to plan for the week. Once the “Generate Meal Plan” button is clicked, the app displays the specified number of meals.

For each meal, users can:

1. View ingredients and cooking instructions.
2. Generate a PDF download, which they can print and add to their cookbook.

The data is pulled from [**TheMealDB**](https://www.themealdb.com/), a free, unlicensed API. Specifically, I am using the random meal API endpoint. Each time the “Generate Meal Plan” button is clicked, the app makes an asynchronous request to fetch a random meal. In the future, I plan to integrate additional API endpoints offered by TheMealDB to expand the app’s features.

Challenges I Anticipate

As with any development project, I expect to encounter many “What the hell am I trying to do?” moments. Debugging will likely be a frequent companion. My favorite challenge is when I discover that I misplaced a comma or bracket, fix it, and then realize that the app looks or behaves completely differently. I foresee spending hours figuring out why such changes occur, but that’s all part of the journey.

How My Application Meets the Project Requirements

Requirement 1: Single HTML/CSS/JS Frontend with Public API

* My app is a frontend that accesses data from TheMealDB API.
* The API returns JSON data containing meal information, such as:
  + strMeal (meal name)
  + strInstructions (cooking instructions)
  + strMealThumb (meal image)
* All interactions with the API are asynchronous using fetch() and handle JSON responses.

Requirement 2: No Redirects or Reloads

* The entire app runs on a single HTML page without any redirects or reloads.
* Dynamic updates to the DOM (e.g., #recipe-list, #instructions) provide a seamless, single-page user experience.

Requirement 3: At Least 3 Distinct Event Listeners

1. Click Event on .selectable-box Elements:
   * Allows users to select up to 7 meals.
   * Callback: Toggles the .selected class and limits the maximum number of selected items to 7.
2. Click Event on #generate-plan Button:
   * Fetches random meals based on the user’s selection and displays them in the meal plan.
   * Callback: generateMealPlan.
3. Click Event on detailsButton:
   * Fetches and displays the cooking instructions and ingredients for the selected meal.
   * Callback: displayInstructionsAndIngredients.
4. Click Event on printButton:
   * Generates a PDF of the selected recipe, including the image, ingredients, and instructions.
   * Callback: generatePDF.

Requirement 4: Array Iteration

* The code uses forEach multiple times to iterate over arrays:
  + Displaying meal instructions
  + Adding ingredients to a list

Requirement 5: Good Coding Practices

1. DRY (Don’t Repeat Yourself) Principle:
   * Functions like getIngredients, displayInstructionsAndIngredients, and fetchRecipeDetails abstract repetitive tasks.
2. Error Handling:
   * Functions like fetchMealPlan and fetchRecipeDetails use try...catch blocks to handle errors and display user-friendly alerts.
   * My experience with programming PLCs in water and sewer treatment plants taught me the importance of providing clear error messages to help with troubleshooting.
3. Code Readability:
   * The code is well-organized, with clear function names like displayMealPlan, generatePDF, and getIngredients, making it easy to understand and maintain.

This application is just the beginning of my journey to create a seamless meal planning experience for users. With future updates, I plan to expand its features and functionality further.