# Recipe Finder - Modern React (Functional Components)

## Overview

This **Recipe Finder** web application is built using **React (functional components with hooks)**. It allows users to search for recipes based on ingredients and view detailed information about selected recipes.

## JavaScript Concepts Used

* **React Functional Components**: Used for structuring the UI.
* **useState Hook**: Manages state for user input, recipe list, and selected recipe.
* **Fetch API**: Handles asynchronous API calls to Spoonacular.
* **Event Handling**: Handles user interactions like searching and clicking on recipes.
* **Dynamic Rendering**: Updates UI dynamically without reloading the page.

## Key JavaScript Functions

### 1. Handling User Input

const [query, setQuery] = useState("");

<input type="text" value={query} onChange={(e) => setQuery(e.target.value)} />

* useState stores the input value.
* onChange updates the query state as the user types.

### 2. Fetching Recipes

const searchRecipes = async () => {

if (!query) {

alert("Please enter an ingredient");

return;

}

const url = `https://api.spoonacular.com/recipes/complexSearch?query=${query}&number=5&apiKey=${apiKey}`;

try {

const response = await fetch(url);

const data = await response.json();

setRecipes(data.results);

} catch (error) {

alert("Failed to fetch recipes. Try again!");

}

};

* Uses fetch() to retrieve data from Spoonacular API.
* Converts response to JSON and updates recipes state.
* Handles errors with try-catch.

### 3. Fetching Recipe Details

const showRecipeDetails = async (recipeId) => {

const url = `https://api.spoonacular.com/recipes/${recipeId}/information?apiKey=${apiKey}`;

try {

const response = await fetch(url);

const data = await response.json();

setSelectedRecipe(data);

} catch (error) {

alert("Failed to fetch recipe details. Try again!");

}

};

* Fetches detailed recipe data based on the recipeId.
* Updates selectedRecipe state.

### 4. Displaying Recipes Dynamically

{recipes.map(recipe => (

<div key={recipe.id} onClick={() => showRecipeDetails(recipe.id)}>

<h3>{recipe.title}</h3>

<img src={recipe.image} alt={recipe.title} />

</div>

))}

* Uses .map() to dynamically create elements for each recipe.
* Clicking on a recipe triggers showRecipeDetails().

## How to Run the Project

1. **Install Node.js**: [Download here](https://nodejs.org/)
2. Create a React app:

npx create-react-app recipe-finder

cd recipe-finder

1. Replace src/App.js with the provided React code.
2. Start the development server:

npm start

1. Open http://localhost:3000/ in your browser.

## API Integration

### Fetch Recipe List

fetch(`https://api.spoonacular.com/recipes/complexSearch?query=${query}&number=5&apiKey=${apiKey}`)

.then(response => response.json())

.then(data => setRecipes(data.results))

.catch(error => alert("Error fetching recipes!"));

### Fetch Recipe Details

fetch(`https://api.spoonacular.com/recipes/${recipeId}/information?apiKey=${apiKey}`)

.then(response => response.json())

.then(data => setSelectedRecipe(data))

.catch(error => alert("Error fetching details!"));

**Recipe Finder** app showcases core JavaScript techniques in React, making it an excellent learning project.

**SCREENSHOTS**

