# Recipe Finder - React + ReduxConverting

## 1. Key Differences Between Vanilla JS and React + Redux

### Vanilla JavaScript Implementation:

* Used **direct DOM manipulation** (document.getElementById(), innerHTML).
* Managed state using variables within functions.
* Event listeners were manually attached (addEventListener).
* API calls were handled directly in event handler functions.

### React + Redux Implementation:

* **Component-based architecture**: UI is divided into reusable React components.
* **State management with Redux**: Centralized store manages recipes and details.
* **Redux actions and reducers**: Handle API calls and state updates systematically.
* **Efficient re-rendering**: Components update only when necessary, improving performance.

## 2. Implementing the Recipe Finder in React + Redux

### ****State Management with Redux****

I used Redux Toolkit to manage the global state, including fetching and storing recipes.

#### ****Creating a Redux Slice (recipesSlice.js)****

import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';

// Async action to fetch recipes

export const fetchRecipes = createAsyncThunk('recipes/fetchRecipes', async (query) => {

const apiKey = 'YOUR\_API\_KEY';

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

const response = await fetch(url);

return await response.json();

});

// Async action to fetch recipe details

export const fetchRecipeDetails = createAsyncThunk('recipes/fetchRecipeDetails', async (recipeId) => {

const apiKey = 'YOUR\_API\_KEY';

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

const response = await fetch(url);

return await response.json();

});

const recipesSlice = createSlice({

name: 'recipes',

initialState: {

recipes: [],

selectedRecipe: null,

},

reducers: {},

extraReducers: (builder) => {

builder.addCase(fetchRecipes.fulfilled, (state, action) => {

state.recipes = action.payload.results;

});

builder.addCase(fetchRecipeDetails.fulfilled, (state, action) => {

state.selectedRecipe = action.payload;

});

}

});

export default recipesSlice.reducer;

### ****React Components****

#### ****SearchBar.js**** (Handles user input and triggers API calls)

import React, { useState } from 'react';

import { useDispatch } from 'react-redux';

import { fetchRecipes } from '../features/recipes/recipesSlice';

const SearchBar = () => {

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

const dispatch = useDispatch();

const handleSearch = () => {

if (query) dispatch(fetchRecipes(query));

};

return (

<div>

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

<button onClick={handleSearch}>Search</button>

</div>

);

};

export default SearchBar;

#### ****RecipeList.js**** (Displays the list of fetched recipes)

import React from 'react';

import { useSelector, useDispatch } from 'react-redux';

import { fetchRecipeDetails } from '../features/recipes/recipesSlice';

const RecipeList = () => {

const recipes = useSelector((state) => state.recipes.recipes);

const dispatch = useDispatch();

return (

<div className={`recipe-container ${recipes.length > 0 ? 'active' : ''}`}>

{recipes.map((recipe) => (

<div key={recipe.id} className="recipe" onClick={() => dispatch(fetchRecipeDetails(recipe.id))}>

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

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

</div>

))}

</div>

);

};

export default RecipeList;

RecipeDetails.js (Displays details of the selected recipe)

import React from 'react';

import { useSelector } from 'react-redux';

const RecipeDetails = () => {

const recipe = useSelector((state) => state.recipes.selectedRecipe);

if (!recipe) return null;

return (

<div className={`recipe-details-container ${recipe ? 'active' : ''}`}>

<h2>{recipe.title}</h2>

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

<p><strong>Ingredients:</strong></p>

<ul>

{recipe.extendedIngredients?.map((ing) => (

<li key={ing.id}>{ing.original}</li>

))}

</ul>

<p><strong>Instructions:</strong></p>

<p>{recipe.instructions || 'No instructions available.'}</p>

</div>

);

};

export default RecipeDetails;

## ****Benefits of React + Redux Over Vanilla JS****

| **Feature** | **Vanilla JavaScript** | **React + Redux** |
| --- | --- | --- |
| State Management | Local variables | Centralized Redux store |
| API Calls Handling | Inside event handlers | Async Thunks in Redux |
| UI Rendering | Manual DOM updates | Automatic re-rendering |
| Code Structure | Linear, hard to manage | Component-based & modular |
| Performance | Inefficient updates | Optimized state-driven UI |

## **SCREENSHOTS**

