**Recipe App**

Project submitted to the

SRM University – AP, Andhra Pradesh

for Partial Fulfilment of the requirements to award of the degree of

**Bachelor of Technology**

**In**

**Computer Science and Engineering**

Submitted by

**S. Judith Chrysolite – AP22110011267**



Under the Guidance of

**Syed Arshad**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[November, 2024]**

**Introduction**

Cooking is more than just a necessity; it’s an art, a science, and a way to bring people together. With **Recipe\_APP**, cooking becomes a creative and enjoyable journey. This modern recipe management platform empowers users to explore an extensive library of recipes, create and save their own culinary masterpieces, and organize their favourite dishes effortlessly.

Built using React, **Recipe\_APP** ensures a seamless, high-performance experience, providing an intuitive interface that caters to both beginner cooks and seasoned chefs. Whether you're experimenting with exotic cuisines, revisiting classic recipes, or just trying to make something quick and simple, **Recipe\_APP** is here to inspire and assist.

Its powerful features, such as ingredient-based searches and step-by-step guidance, make meal preparation easy and personalized. By integrating tools like shopping lists, dietary filters, and interactive timers, **Recipe\_APP** stands out as a comprehensive cooking companion for every home kitchen.

**Scenario-Based Introduction**

Imagine it’s a busy weekday evening, and you’re unsure what to cook. With Recipe\_APP, your decision-making process becomes stress-free:

1. **Browse Recipes:** Use the app to browse recipes tailored to the ingredients you already have.
2. **Select a Dish:** Choose a recipe that suits your taste, dietary needs, and available preparation time.
3. **Follow Step-by-Step Guidance:** Access detailed instructions, use built-in timers, and enjoy real-time tips for flawless results.

Hosting a dinner party? Impress your guests with international cuisines or decadent desserts. Need a quick snack? Find easy recipes that fit your schedule. Whether you’re cooking for yourself or a crowd, Recipe\_APP simplifies the process and enhances your culinary experience.

**Target Audience**

**Recipe\_APP** is designed for diverse user groups, including:

1. **Home Cooks:**
   * Passionate individuals who enjoy trying new recipes and improving their skills.
   * Families looking to explore more home-cooked meals.
2. **Professional Chefs:**
   * Culinary experts seeking a tool to manage and organize recipes efficiently.
   * Restaurants looking for a digital catalog of their signature dishes.
3. **Health Enthusiasts:**
   * Users focused on specific dietary goals, such as weight management or fitness.
   * People with allergies or preferences requiring customized recipes.
4. **Busy Professionals:**
   * Individuals with limited time but a desire for quick, healthy, and tasty meals.
   * Parents needing easy meal solutions for their children.
5. **Beginners:**
   * Novice cooks who need simple, guided recipes with clear instructions.

**Project Goals and Objectives**

**Goals:**

1. Provide a unified platform for discovering, managing, and sharing recipes.
2. Make cooking a fun, engaging, and stress-free activity for users of all skill levels.

**Objectives:**

1. **Efficient Recipe Management:** Enable users to add, edit, delete, and save recipes effortlessly.
2. **Search and Filters:** Allow users to search by name, ingredients, cuisine, and dietary preferences.
3. **Interactive Tools:** Integrate features like timers, shopping lists, and cooking tips to support users.
4. **Cross-Device Compatibility:** Ensure the app runs smoothly on desktops, tablets, and smartphones.

**Key Features**

1. **Recipe Management:**
   * Organize recipes into categories (e.g., breakfast, desserts, vegan).
   * Add notes or modify recipes to fit personal preferences.
2. **Search and Filters:**
   * Advanced filters for time, difficulty, cuisine, and allergens.
   * Ingredient-based searches to find recipes that minimize food waste.
3. **Interactive Cooking:**
   * Real-time guidance with step indicators and visual aids.
   * Built-in timers for multi-step cooking processes.
4. **Shopping List Integration:**
   * Automatically generate shopping lists based on selected recipes.
   * Share lists with family or sync them across devices.
5. **User Profiles:**
   * Save favourites, track cooking history, and customize app themes.
   * View personalized recipe recommendations.
6. **Responsive Design:**
   * Access the app on any device with a consistent and user-friendly interface.

**Pre-Requisites**

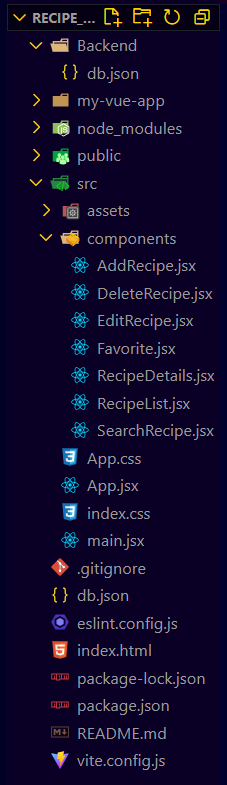
**Tools and Frameworks:**

1. **React.js**: To build dynamic and interactive interfaces.
2. **Node.js and npm**: For dependency management and development.
3. **React Router DOM**: To manage multi-page navigation within the app.

**Skills:**

* Familiarity with JavaScript, React, and CSS.
* Knowledge of REST APIs for integrating external recipe data sources.
* **Development Environment:**
* **Editor**: Visual Studio Code or any preferred code editor.
* **Version Control**: Git and GitHub for collaboration and tracking changes.

**Poject Structure**



* **src/**: Contains all source code for the app.
  + **App.jsx:** Main component that manages routing and application state.
  + **components/:** Contains reusable components like RecipeList, AddRecipe, and EditRecipe.
* **public/**: Static assets like index.html and app icons.
* **package.json**: Manages project metadata, dependencies, and scripts.

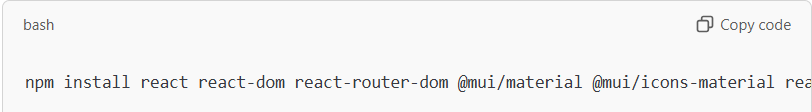
**Project Flow**

**Milestone 1: Project Setup and Configuration**

1. **Install Required Tools and Software:**

* Begin by setting up the development environment for the project. Ensure the following tools and libraries are installed and configured:
* **React Js**: For building the core user interface.
* **React Icons:** For adding visually appealing icons to enhance UI design.
* **Material-UI:** For building responsive and attractive components.
  + - **React-codemirror2:** For advanced code editing and syntax highlighting (if applicable).

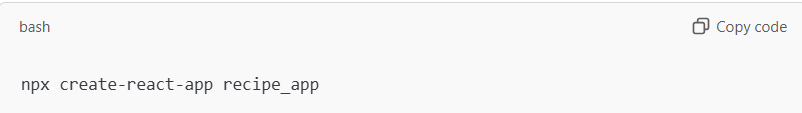
1. **Installation of Required Tools:**
   * Navigate to the project folder to begin the setup process.
   * Install dependencies with the following commands:



1. **References:**
   * Used the following official documentation for further assistance:
     + [React Installation Guide](https://react.dev/learn/installation)
     + [Material-UI Getting Started Guide](https://react-bootstrap-v4.netlify.app/getting-started/introduction/)

**Milestone 2: Web Development**

1. **Setup React Application**:
   * **Create React Application**:



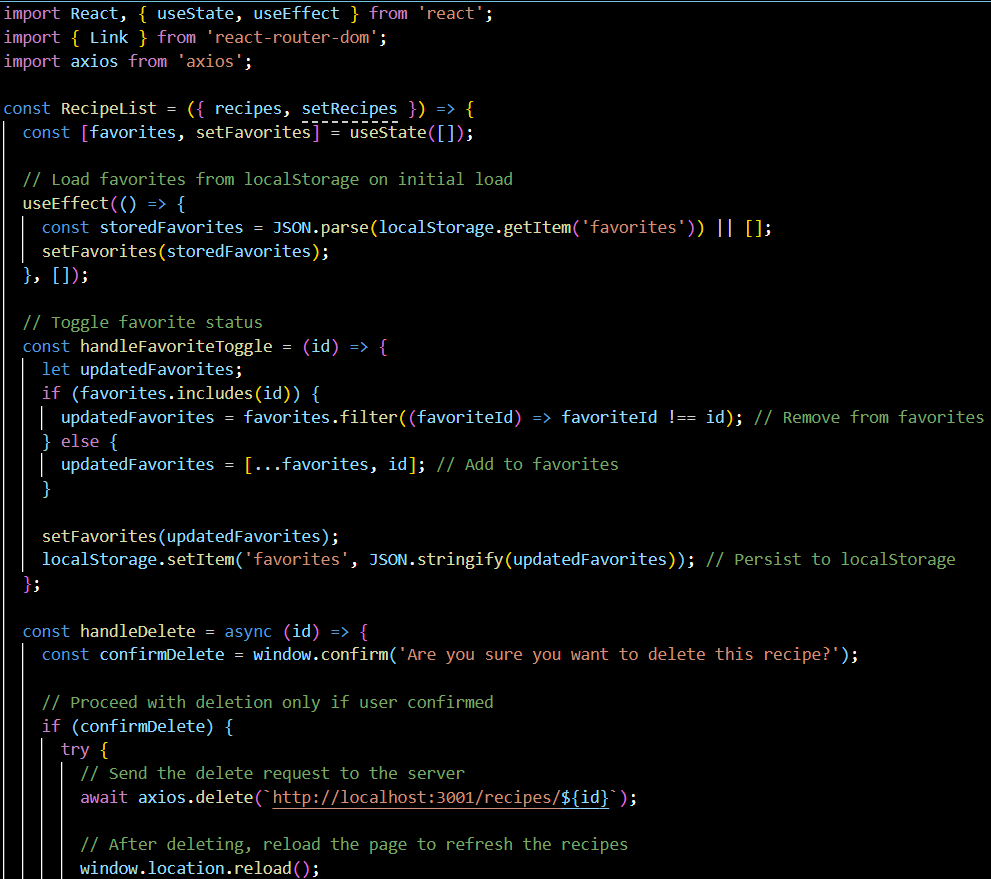
* + **Configure Routing**: Use react-router-dom to define navigation between pages.
  + **Install Required Libraries**: Ensure all essential libraries like Material-UI, Axios, and React Icons are installed.

**App.jsx Component**:



**Code Description**

1. **Component Imports**:
   1. The main App component imports React, React Router for routing, and axios for API calls.
   2. Components like RecipeList, AddRecipe, RecipeDetails, EditRecipe, SearchRecipe, and Favorites are lazily loaded using React.lazy to optimize performance.
2. **State Management**:
   1. The recipes state stores the list of recipes fetched from the backend API.
   2. The loading state indicates whether the recipes are still being loaded.
3. **API Integration**:
   1. The fetchRecipes function retrieves recipes from the backend at <http://localhost:3001/recipes> using axios and updates the recipes state.
4. **Lifecycle Hook**:
   1. The useEffect hook calls fetchRecipes once on component mount to fetch initial data.
5. **Navigation Bar**:
   1. Provides links for navigation: Home, Add Recipe, and Favorites.
   2. Includes a search bar that redirects to the /search route with the search term passed as a query parameter.
6. **Loading State**:
   1. Displays a loading message while recipes are being fetched.
7. **Suspense for Lazy Loading**:
   1. Wraps routes with React.Suspense, showing a fallback loading message while lazily loaded components are being fetched.
8. **Route Configuration**:
   1. /: Displays the RecipeList component with the fetched recipes.
   2. /add: Renders the AddRecipe component and updates the recipes after adding.
   3. /recipes/:id: Displays details of a specific recipe using the RecipeDetails component.
   4. /edit/:id: Allows editing of a recipe through the EditRecipe component and updates the recipes after editing.
   5. /search: Displays recipes matching a search term using the SearchRecipe component.
   6. /favorites: Displays a list of favorite recipes with the Favorites component.
9. **Search Functionality**:
   1. Captures user input from a search bar and redirects to the /search route with the search term in the URL.
10. **Export**:
    1. The App component is exported as the default export for use in the project.

**RecipeList.jsx component:**

**Code Description:**

1. **Component Imports**:
   1. Imports React and its hooks (useState, useEffect) for state management.
   2. Imports Link from react-router-dom for navigation to individual recipe details.
   3. Imports axios for making HTTP requests.
2. **Component Name**:

RecipeList is a functional component responsible for displaying a list of recipes.

1. **Props**:
   1. recipes: An array of recipe objects passed as a prop to display recipe data.
   2. setRecipes: A setter function (not used directly here) to update the recipes state in the parent component.
2. **State Variables**:
   1. favorites: Maintains a list of favorite recipe IDs.
   2. **Initialization**: Loads favorites from localStorage when the component mounts, ensuring persistence across sessions.
3. **Favorite Toggle Functionality**:
   1. **handleFavoriteToggle(id)**:
      1. Toggles the favorite status of a recipe.
      2. Updates the favorites state and stores the updated list in localStorage for persistence.
4. **Recipe Deletion**:
   1. **handleDelete(id)**:
      1. Prompts the user for confirmation before deleting a recipe.
      2. Sends a DELETE request to the backend API ([http://localhost:3001/recipes/${id](http://localhost:3001/recipes/$%7bid)}) to remove the recipe.
      3. Reloads the page upon successful deletion to refresh the displayed recipes.
5. **Rendering Recipes**:
   1. Iterates over the recipes array to render each recipe as a card with:
      1. **Image**: Displays the recipe's image or a placeholder if no image is provided.
      2. **Title**: The recipe's title.
      3. **Prep Time**: Preparation time for the recipe.
      4. **Action Buttons**:
         1. **View Button**: Navigates to the recipe's detailed page (/recipes/:id) using a Link.
         2. **Delete Button**: Deletes the recipe using the handleDelete function.
         3. **Favorite Button**: Toggles the favorite status using the handleFavoriteToggle function.
6. **Conditional Rendering**:
   1. Displays a "No recipes found" message if the recipes array is empty.
7. **Styling**:
   1. CSS classes like recipe-list-container, recipe-grid, recipe-card, view-btn, delete-btn, and favorite-btn are used for layout and styling (assumed to be defined in a CSS file).
8. **Export**:
   1. The RecipeList component is exported as the default export for use in other parts of the application.

**RecipeDetails.jsx component:**



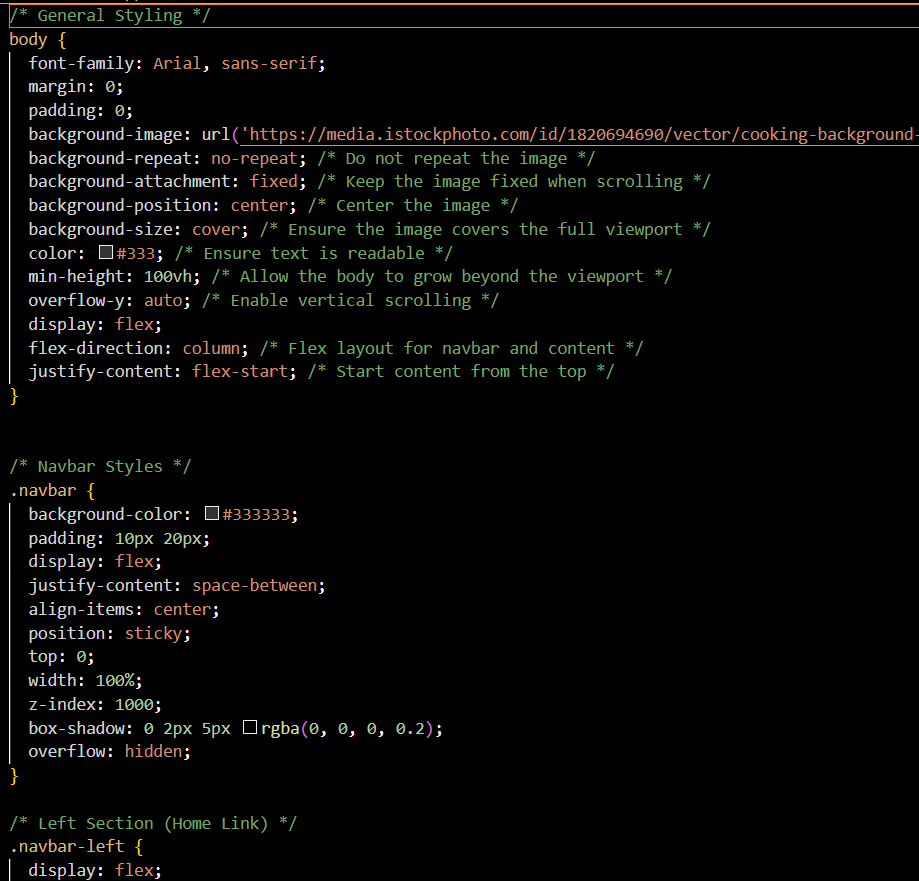
**Code description:**

1. **Component Imports**:
   1. React and hooks (useState, useEffect) for state management.
   2. axios for HTTP requests.
   3. useParams, Link, and useNavigate from react-router-dom for route handling, navigation, and redirection.
2. **Component Name**:

RecipeDetails is a functional component responsible for displaying the detailed view of a single recipe.

1. **Props**:
   1. fetchRecipes: A function passed from the parent component to refresh the recipe list after deletion.
2. **State Variables**:
   1. recipe: Stores the data of the selected recipe.
   2. error: Tracks errors during the API request.
3. **Route Parameter**:
   1. id: Extracted from the route using useParams to identify the specific recipe to display.
4. **Effect Hook**:
   1. **useEffect**:
      1. Triggers the fetchRecipe function whenever the id changes.
5. **API Interaction**:
   1. **fetchRecipe**:
      1. Sends a GET request to fetch recipe details from the backend (<http://localhost:3001/recipes/:id>).
      2. Updates the recipe state on success or sets an error message on failure.
   2. **handleDelete**:
      1. Sends a DELETE request to remove the recipe from the backend.
      2. Refreshes the recipe list using fetchRecipes and redirects to the home page upon successful deletion.
6. **Conditional Rendering**:
   1. Displays an error message if error is set.
   2. Shows a "Loading..." message if the recipe data is not yet available.
7. **Recipe Details Rendering**:
   1. Displays:
      1. **Title**: The recipe's title.
      2. **Preparation Time**: Displayed prominently.
      3. **Cuisine**: Type of cuisine associated with the recipe.
      4. **Ingredients**: Rendered as a list, split by commas, and trimmed for formatting.
      5. **Steps**: Rendered as an ordered list, split by newlines.
8. **Action Buttons**:
   1. **Edit Button**: Links to the recipe edit page (/edit/:id).
   2. **Back Button**: Navigates back to the recipe list.
9. **Image Display**:
   1. Displays the recipe image (if available) with styling:
      1. Flexible layout (flex: 1).
      2. Maximum width restricted to 300px with responsive height.
      3. Rounded corners and subtle shadow for aesthetics.
10. **Styling**:
    1. Inline styles are used for layout and appearance:
       1. flex properties for responsive design.
       2. borderRadius and boxShadow for polished visuals.
11. **Export**:
    1. The RecipeDetails component is exported as the default export for use in the application.

**App.css component:**



**Code description:**

This CSS code provides styling for a recipe application web interface. Here's a breakdown of its functionality:

### **General Styling**

* **Body Style**: Sets a responsive, centered background image with a food-related theme. Text is styled for readability.
* **Flex Layout**: Ensures alignment of page elements vertically, such as the navbar and content.

### **Navbar**

* **Sticky Navbar**: A navigation bar that remains at the top while scrolling.
* **Left and Right Sections**: Divide navbar elements (e.g., logo on the left, links, and search bar on the right).
* **Hover Effects**: Links change background color and scale on hover.
* **Responsive Design**: Adapts the navbar layout for smaller screens.

### **Favorites Section**

* **Card Layout**: Displays recipes in a grid with hover animations for interactivity.
* **Button Styling**: Includes buttons for actions like "View" and "Delete," with distinct hover effects.
* **No Favorites Message**: Styled to inform users when there are no saved recipes.

### **Recipe List**

* **Grid Display**: Recipes are shown in a responsive grid.
* **Hover Effect**: Cards slightly enlarge on hover for a dynamic feel.
* **Responsive Adjustments**: Adapt grid layout for smaller devices.

### **Add/Edit Recipe Forms**

* **Form Layout**: Includes input fields and buttons for adding or editing recipes.
* **Focus Effects**: Highlight inputs during interaction for user feedback.

### **Recipe Details**

* **Container Style**: Sets a clean layout for recipe details with a scrollable view.
* **Typography**: Adjusts text hierarchy for readability.

### **Search Component**

* **Search Box**: Styled input field with focus effects, placed within a centered container.

### **Error and Loading Messages**

* **Centered Text**: Styles for error and loading states, with prominent colors to draw attention.

### **Responsive Design**

* **Media Queries**: Adjust layouts and element properties for screens smaller than 768px.

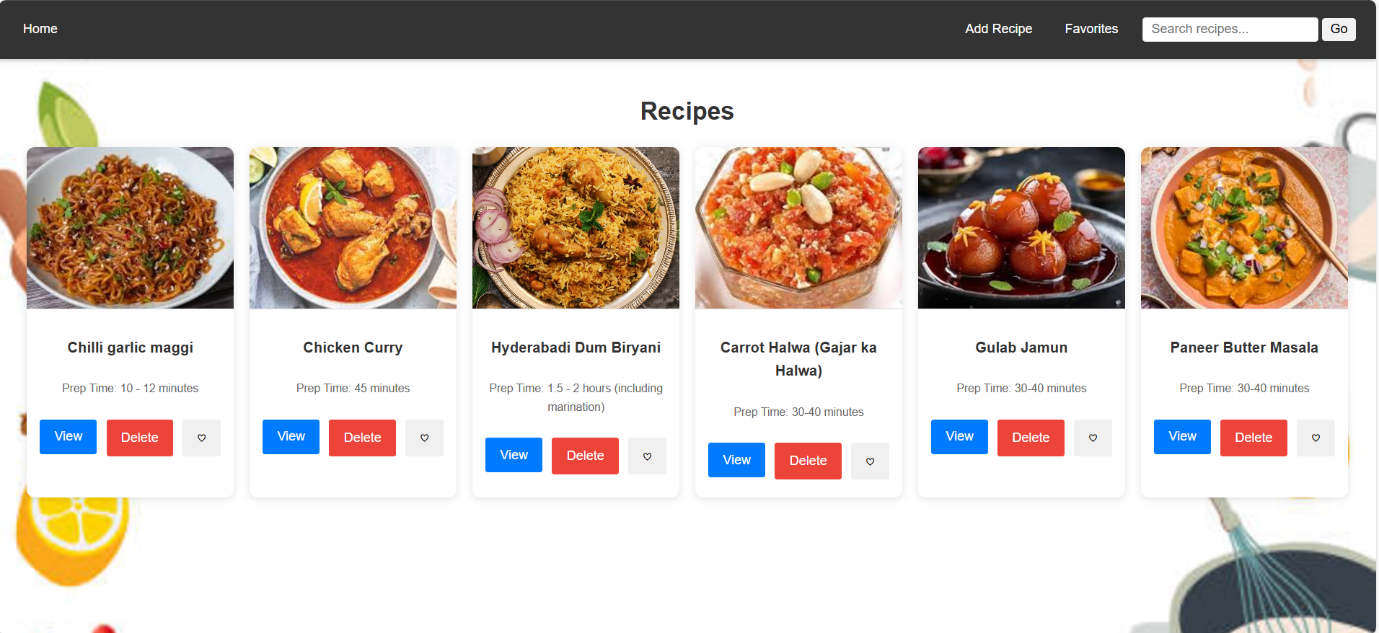
The code ensures a user-friendly, visually appealing, and responsive recipe application with interactive elements and a clean aesthetic.

Project Execution:

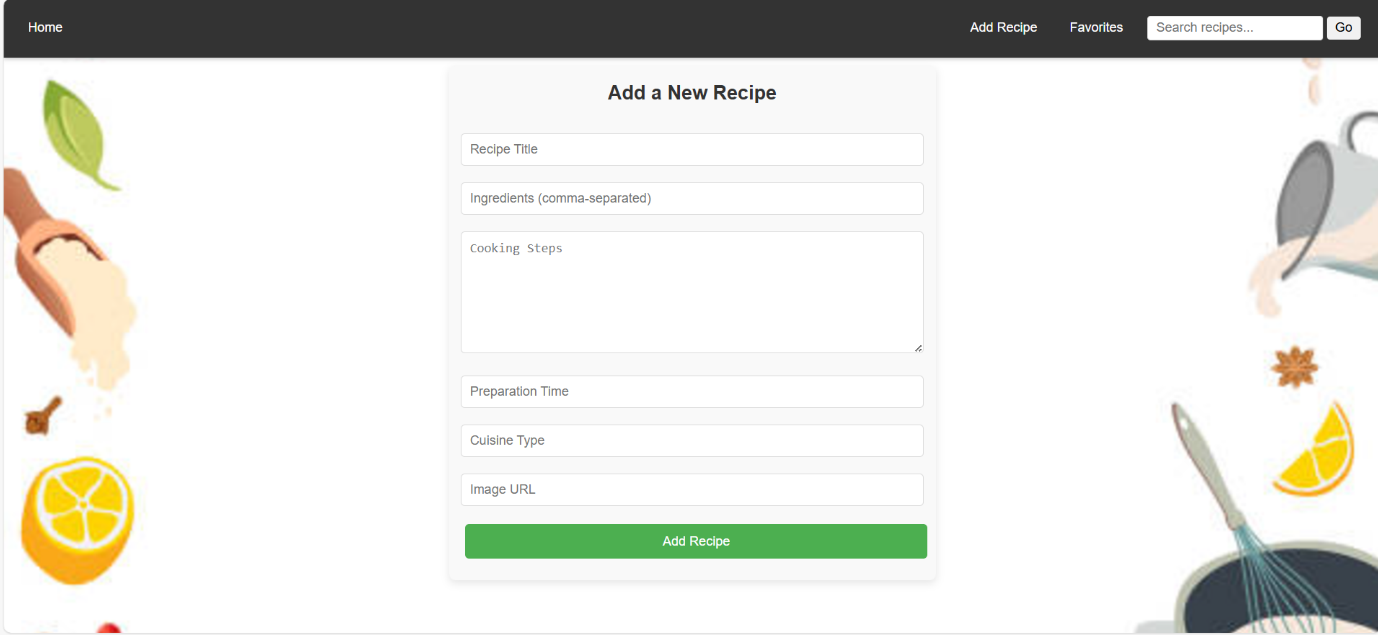
After completing the code, run the react application by using the command “npm run dev”

Here are some of the screenshots of the application.

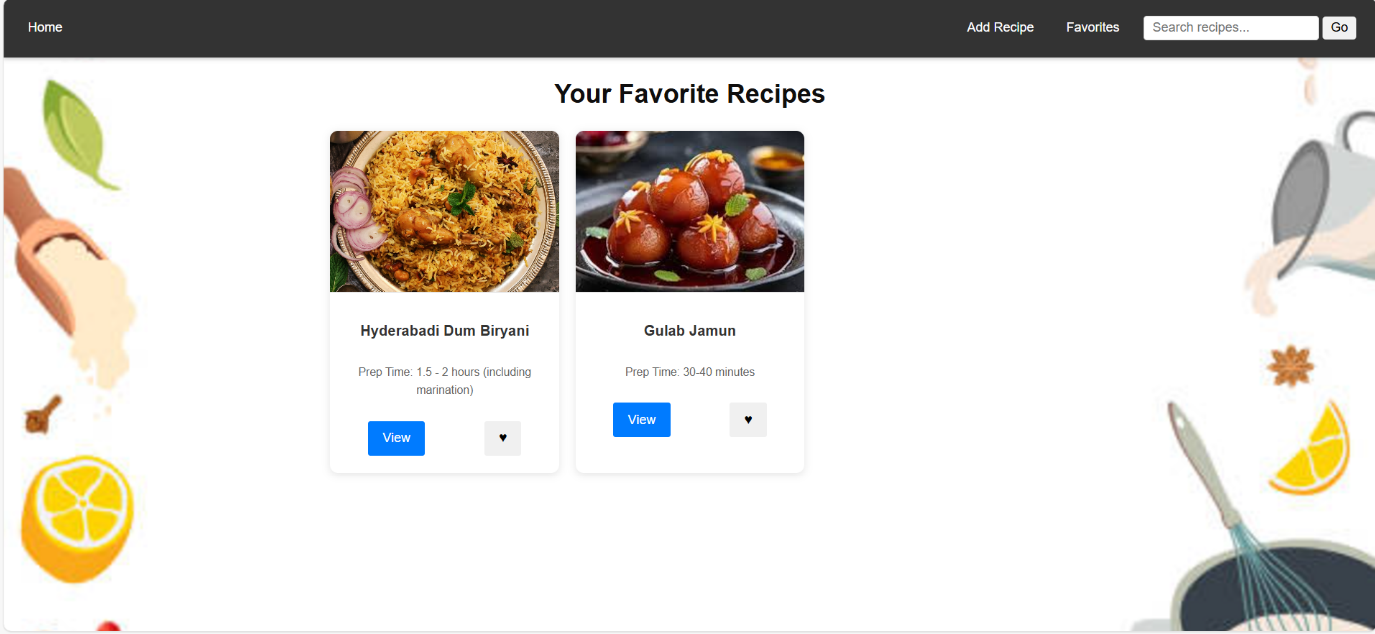
**Home page:**



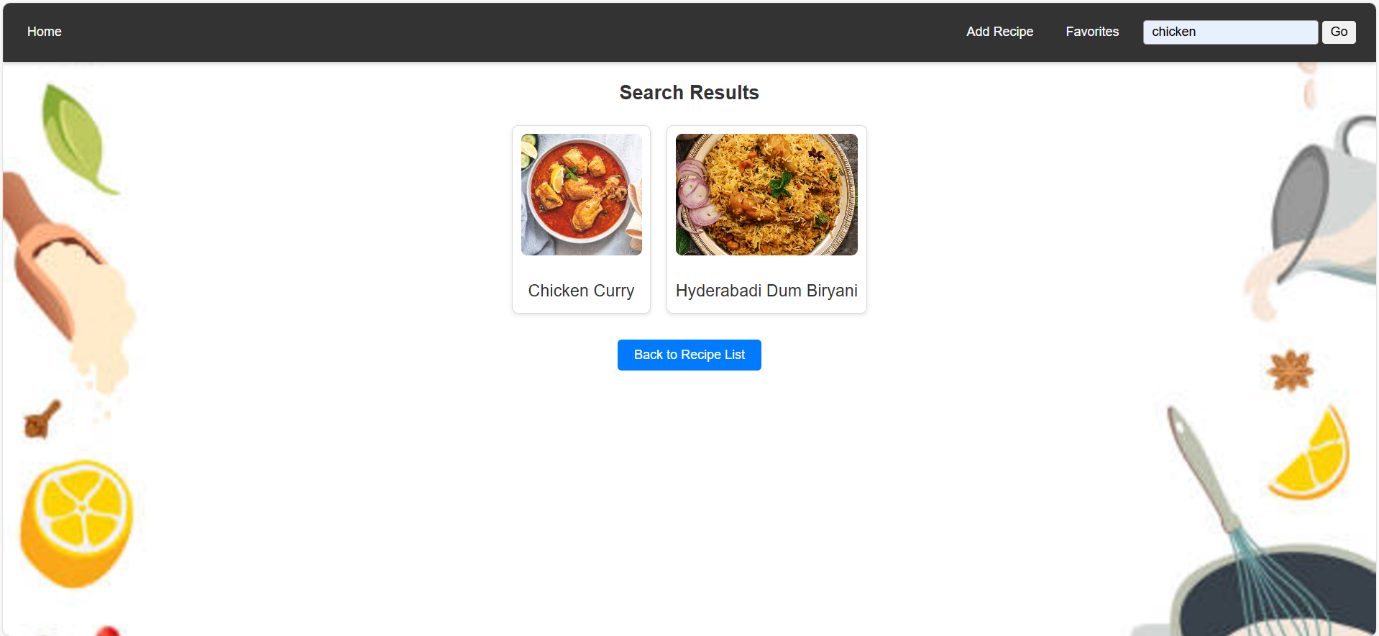
**Add Recipe:**



**Favorites page:**



**Search recipes:**



**Project Demo Link:**

[**https://drive.google.com/file/d/1ndOUKhM1qEqEC1HOOIWOp8voHhEZxyI-/view?usp=sharing**](https://drive.google.com/file/d/1ndOUKhM1qEqEC1HOOIWOp8voHhEZxyI-/view?usp=sharing)

**GitHub Repo:**

[**https://github.com/JudithChrysolite/Recipe-App**](https://github.com/JudithChrysolite/Recipe-App)