PERI COLLEGE OF ARTS AND SCIENCE

NAAN MUDHALVAN – GROUP PROJECT

FRONTEND DEVELOPMENT WITH REACT.JS

COOK BOOK : YOUR VIRTUAL KITCHEN ASSISTANT

2025

PROJECT TEAM INFORMATION

TEAM ID	SWTID1741244635155376
TEAM SIZE	5
TEAM LEADER	Tharini T (Project architecture)
TEAM MEMBER	Vinothini p (*Frontend Developer*)
TEAM MEMBER	Swetha T (*Backend Developer & API Integration Specialist*)
TEAM MEMBER	Sona R (*Testing & Quality Assurance Engineer*)
TEAM MEMBER	Priyanka V *(Database & Backend Optimization)*

INDEX

INTRODUCTION

The **Kitchen Assistant in Cook Book** is a web application designed to assist users in meal preparation by providing recipe recommendations, ingredient tracking, and cooking timers. This app aims to simplify the cooking process by integrating a user-friendly interface with real-time guidance.

Built using React.js, the application efficiently retrieves data from an external recipe API and manages state using Redux. It provides an intuitive platform for both novice and experienced cooks to explore new recipes and manage their kitchen inventory effectively.

This documentation outlines the application's architecture, features, and setup instructions for developers and users.

PROJECT OVERVIEW

PURPOSE

The **Kitchen Assistant in Cook Book** is developed to help users organize their cooking process efficiently. The app offers:

- **Recipe suggestions** based on available ingredients.
- **Step-by-step cooking guidance** with timers.
- **Ingredient inventory tracking** for better meal planning.
- **Nutritional insights** for informed meal choices.

FEATURES

- **Smart Recipe Recommendations:** Fetches recipes based on user-input ingredients.
- **Interactive Cooking Steps:** Provides real-time guidance with timers.

- **Ingredient Management:** Tracks pantry stock and notifies users about shortages.
- **Personalized Favorites List:** Allows users to save their favorite recipes.
- **Responsive UI:** Ensures seamless experience across devices.

ARCHITECTURE

COMPONENT STRUCTURE

The application follows a modular and scalable component-based architecture, ensuring maintainability and ease of development. The core components include:

- **App.jsx:** This serves as the main entry point of the application, handling overall state management and routing mechanisms.
- **RecipeList.jsx:** Responsible for fetching and displaying a list of available recipes based on user preferences and search queries.
- **RecipeDetails.jsx:** Provides in-depth cooking instructions, including step-by-step guidance, ingredient lists, and cooking durations.
- **IngredientTracker.jsx:** Helps users manage their kitchen inventory by tracking available ingredients and alerting shortages.
- **CookingTimer.jsx:** Enables users to set timers for different cooking steps, ensuring an organized cooking process.

STATE MANAGEMENT

Efficient state management is crucial for seamless application performance. The project employs the following methods:

- **Global State (Redux):** Manages key aspects like recipe data, user preferences, and ingredient availability, ensuring synchronization across components.
- **Local State (useState):** Handles UI-based interactions such as toggling modals, updating timers, and user interface responsiveness.

ROUTING

- The application utilizes **React Router** to enable smooth navigation between different views. Users can easily switch between the recipe list, detailed recipe views, and the ingredient tracker.
- Dynamic URL parameters are used to fetch specific recipe details, enhancing the user experience with deep linking support.

SETUP INSTRUCTIONS

PREREQUISITES

To run this application, ensure you have the latest **Node.js (16.x or later)** installed, along with **npm or yarn** for managing dependencies. These tools provide the necessary runtime environment for executing JavaScript code and handling project packages effectively.

Before proceeding, verify the installation using:

```
node -v  # Should return 16.x or later
npm -v   # Should return the installed npm version
```

If Node.js is not installed, download it from Node.js official site and follow the installation instructions.

INSTALLATION

Follow these steps to set up the project on your local machine:

1. **Clone the Repository**: This copies the project files to your system.

   ```
   git clone https://github.com/your-repo/kitchen-assistant.git
   ```

2. **Navigate to the Project Directory**: Move into the project folder to execute further commands.

   ```
   cd kitchen-assistant
   ```

3. **Install Dependencies**: Fetch all necessary libraries and modules required to run the application.

   ```
   npm install
   ```

4. **Run the Application**: Start the development server to preview the application in your browser.

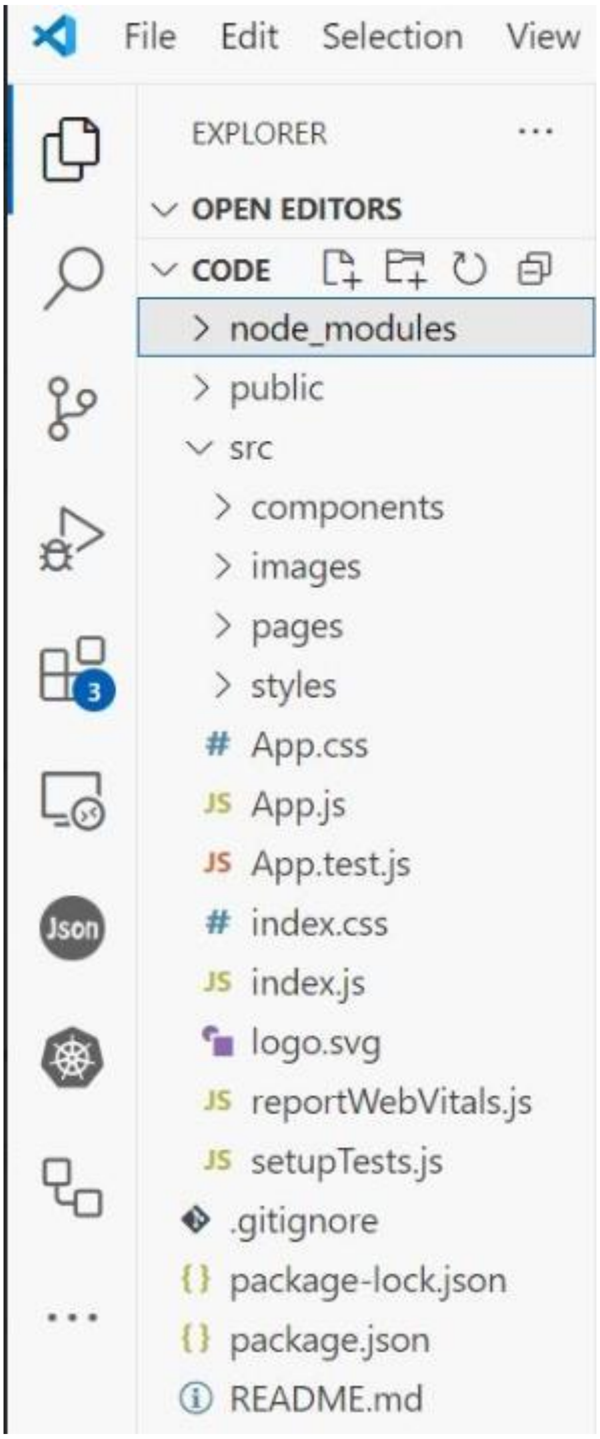
   ```
   npm start
   ```

   Once started, open your browser and navigate to http://localhost:3000 to access the app.

If any issues arise during installation, ensure that all dependencies are correctly installed by running:

```
npm audit fix --force
```

This command resolves any potential conflicts in package dependencies.



```
> node_modules
> public
v src
  > components
  > images
  > pages
  > styles
  App.css
  App.js
  App.test.js
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  setupTests.js
.gitignore
package-lock.json
package.json
README.md
```

FOLDER STRUCTURE

RUNNING THE APPLICATION

Start the application:

```
npm start
```

Then, open the browser and go to:

```
http://localhost:3000
```

COMPONENT DOCUMENTATION

RecipeList.jsx

The **RecipeList** component is responsible for fetching and displaying recipe options based on user preferences. It connects to an API, retrieves available recipes, and presents them in an organized manner, making it easy for users to explore different meal options. Users can apply filters, search recipes by name, and sort results based on cooking time or ingredients.

RecipeDetails.jsx

The **RecipeDetails** component provides users with step-by-step cooking instructions. It includes detailed recipe descriptions, ingredient lists, and interactive timers to help users follow the cooking process efficiently. Additionally, users can adjust portion sizes dynamically, and the app recalculates ingredient measurements accordingly.

IngredientTracker.jsx

The **IngredientTracker** component enables users to manage their pantry stock. It helps in tracking available ingredients, notifying users when supplies are running low, and ensuring they have the necessary items before starting a recipe. It also provides suggestions on recipes that can be made with existing ingredients.

STATE MANAGEMENT

Global State (Redux)

The application uses **Redux** to handle global state management efficiently. It manages essential data like recipe information, user preferences, and ingredient inventory, ensuring consistency across different components of the app. This approach enables seamless synchronization across multiple sessions and devices, providing users with a persistent and reliable experience.

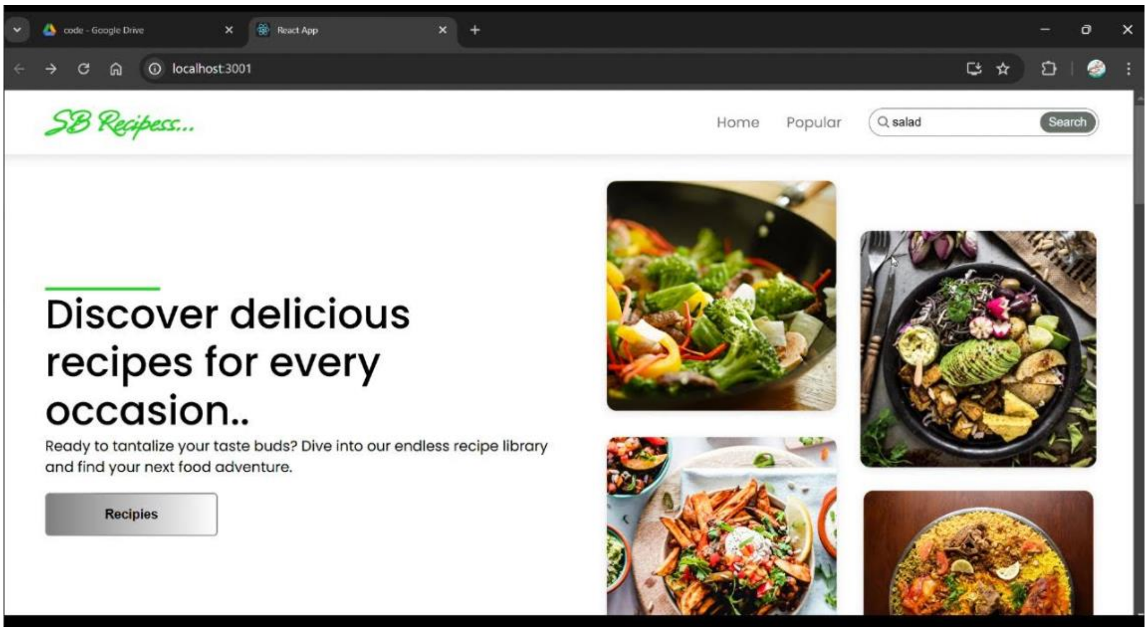
Local State (useState)

For UI-specific interactions, **useState** is used to manage component-level state. This is especially helpful for toggling UI elements, handling user inputs, and managing temporary values that do not need to be stored globally. Modal windows, dropdown selections, and live search features utilize local state for instant updates without requiring global state overhead.

USER INTERFACE

The application is designed with a **clean and intuitive UI** to provide a seamless experience. Users can easily navigate through:

- **Recipe Cards:** Thumbnail images and brief descriptions of various recipes. Each card dynamically updates based on filters



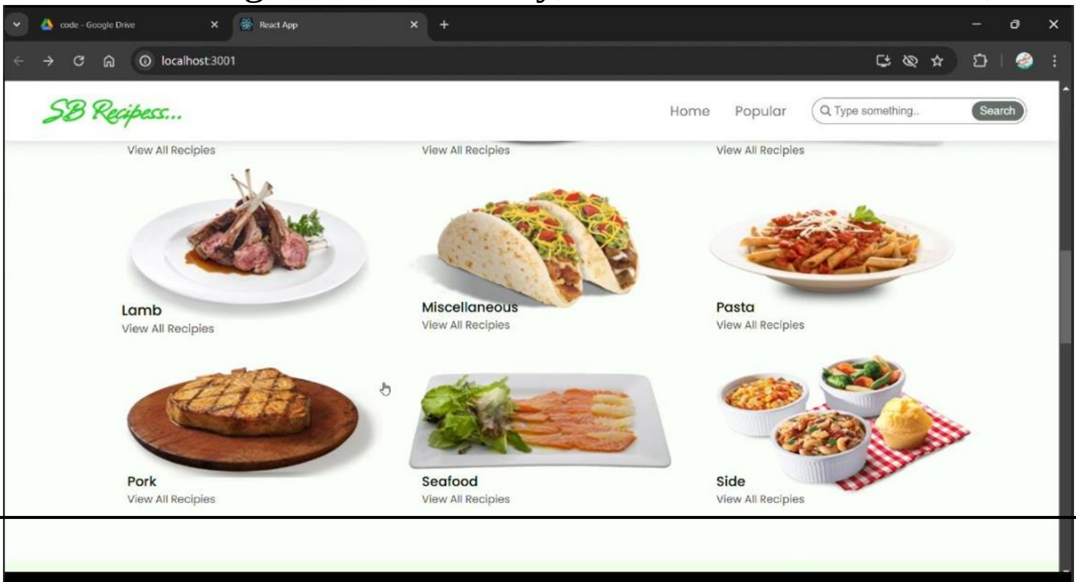
Discover delicious recipes for every occasion..

Ready to tantalize your taste buds? Dive into our endless recipe library and find your next food adventure.

and preferences.

- **Recipe Details Page:** A structured view with step-by-step cooking guidance, ingredient lists, and embedded cooking timers.

- **Pantry Management Dashboard:** A section where users can track their ingredient inventory, receive low-stock alerts, and get



recipe suggestions based on available ingredients.

STYLING

CSS Framework: Tailwind CSS

The application is styled using **Tailwind CSS**, ensuring a responsive and modern design that adapts well to different screen sizes. The utility-first approach allows for flexible customization and rapid development of the UI.

Themes: Dark and Light Modes

The application supports **dark mode and light mode**, allowing users to switch between themes for a comfortable viewing experience. This enhances accessibility, making it easier to use in different lighting conditions.

Animations: Smooth Transitions

Smooth animations and transitions are implemented using **Framer Motion**, providing a more engaging user experience by making interactions fluid and visually appealing. Button clicks, modal openings, and page transitions feature elegant animations to improve user engagement.

TESTING

Unit Testing

Unit tests are written using **Jest and React Testing Library** to validate individual components and ensure they function correctly. Each component undergoes rigorous testing to confirm correct rendering, user interactions, and state updates.

Integration Testing

Integration testing is performed to verify that different components work together seamlessly. API calls and data flows are tested to ensure smooth interactions between services. Mock API data is used to simulate real-world scenarios and edge cases.

E2E Testing

End-to-End (E2E) testing is conducted using **Cypress** to simulate real-world user interactions, including navigating through pages, adding ingredients, and following recipe instructions. Automated tests validate that the entire application functions correctly from start to finish.

KNOWN ISSUES

Frequent API requests may trigger rate limits, causing intermittent failures in fetching recipe data. Implementing request caching and local data storage can mitigate these issues. Ingredient tracking might not always sync correctly across multiple devices, requiring improvements in real-time data updates. Future updates will introduce cloud-based synchronization for a more seamless experience. Minor user interface inconsistencies may appear on certain screen sizes, especially for very small mobile devices. Ongoing UI testing and responsive design improvements will address these issues.

FUTURE ENHANCEMENTS

A planned feature is **voice command integration**, enabling users to interact with the application hands-free while cooking. Users will be able to navigate through recipes, start cooking timers, and check ingredient availability using voice commands. An **AI-powered meal planner** will suggest weekly meal plans based on user preferences and available ingredients. Users can receive personalized meal recommendations, set dietary restrictions, and generate shopping lists based on planned meals. Future updates will include **video tutorials** for each recipe, offering a more interactive and visually guided cooking experience. Users will be able to watch step-by-step instructional videos within the app, improving their confidence and cooking skills.