**Frontend Development with React.js**

**Project Documentation format**

1. **Introduction**
   * **Project Title**: [CookBook: Your Virtual Kitchen Assistant]
   * **Team Members**: 1)Team leader-ABDUL WAHID.A
   * **2)**SARAVANAN.K
   * **3)**GOWRI SANKAR.P
   * **4)**TAMIL SELVAN
2. **Project Overview**
   * **Purpose**: The **CookBook: Your Virtual Kitchen Assistant** project is designed to provide users with an AI-driven tool that enhances their cooking experience. Its primary purpose is to assist in meal planning, cooking, and learning about new recipes. Here's a breakdown of the project's purpose and goals:
   * **Features**: In Our project we have following key features they are;
   * 1. Recipe Management, 2. Smart Ingredient Management, 3. AI-Powered Suggestions, 4. Interactive Cooking Mode, 5. Nutrition & Dietary Preferences, 6. Social & Community Features, 7. Smart Integrations, etc,.
3. **Architecture**

**Component Structure**: Outline the structure of major React components and how they interact**.**

**App Component (App.js),**

**Interactions:**

* Wraps RecipeList, RecipeDetail, Pantry, MealPlanner, etc.
* Provides global state using Context API or Redux.
* **State Management**: Describe the state management approach used (e.g., Context API, Redux). Since this app involves **multiple global states** (recipes, pantry, shopping list, user authentication, etc.), we can use **a hybrid approach**:

1. **Context API** – For lightweight, localized state (e.g., authentication, UI state, theme).

Here’s how you can **use Context API** to manage **lightweight and localized state** in your **CookBook: Virtual Kitchen Assistant** project.

1. **Redux (or Zustand)** – For managing complex, shared state across multiple components (e.g., recipes, pantry, meal planner).

Since **CookBook: Virtual Kitchen Assistant** requires managing complex state across multiple components (**recipes, pantry, meal planner, shopping list**), **Redux** (or **Zustand**) is the best choice.I'll provide **Redux Toolkit** setup first (since it's more widely used and structured), followed by a **Zustand** alternative.

1. **Routing**: Explain the routing structure if using react-router or another routing library.

## **Routing in CookBook: Your Virtual Kitchen Assistant**

Since this is a **multi-page app** with different sections (Recipes, Pantry, Shopping List, etc.), we’ll use **React Router** for navigation.

1. **Setup Instructions**
   * **Prerequisites**: List software dependencies (e.g., Node.js).

**Core Dependencies**

| **Dependency** | **Version (Recommended)** | **Description** |
| --- | --- | --- |
| **Node.js** | >=16.x | Required to run the project. |
| **npm** / **yarn** | >=8.x / >=1.x | Package manager for dependencies. |

**Installation**: Provide a step-by-step guide to clone the repository, install dependencies, and configure environment variables.

**Clone the Repository**

Ensure you have **Git installed**. If not, download it from [git-scm.com](https://git-scm.com/).

📌 **Run this command in your terminal:**

sh

CopyEdit

git clone https://github.com/your-username/cookbook-virtual-kitchen.git

cd cookbook-virtual-kitchen

1. **Folder Structure**
   * **Client**: Describe the organization of the React application, including folders like components, pages, assets, etc.

**1️⃣ public/ - Static Files**

* Stores **static assets** like the index.html, favicon, and images that don’t change dynamically.
* Used for public files accessible without authentication.

**2️⃣ src/assets/ - Images, Icons, and Styles**

* Stores **static assets** like images, icons, and stylesheets.
* **3️⃣ src/components/ - Reusable UI Components**
* Contains **reusable, small UI components** that can be used across multiple pages.
* **Utilities**: Explain any helper functions, utility classes, or custom hooks used in the project.

## **🛠️ Utilities in CookBook: Your Virtual Kitchen Assistant**

The project includes **utility functions, helper classes, and custom hooks** to streamline operations like formatting, API calls, validation, and state management.

1. **Running the Application**
   * Provide commands to start the frontend server locally.
     + **Frontend**: npm start in the client directory.

## **🚀 Running the Application**

To start the **CookBook: Your Virtual Kitchen Assistant** frontend locally, follow these steps:

### ****1️⃣ Install Dependencies****

Ensure you have **Node.js** (v16 or later) installed. If not, download it from [Node.js official website](https://nodejs.org/).

Then, navigate to the project directory and install dependencies:

sh

CopyEdit

cd cookbook-virtual-kitchen

npm install

### ****2️⃣ Start the Frontend Server****

Run the following command in the project root or client directory:

sh

CopyEdit

npm start

This will start the **React development server** and open the application in your default web browser.

* By default, the app runs on **http://localhost:3000**.
* The server **automatically reloads** when you make changes to the code.

### ****3️⃣ Environment Variables Setup (Optional)****

If the project uses **environment variables**, create a .env file in the root directory and add values like:

ini

CopyEdit

REACT\_APP\_API\_URL=http://localhost:5000

REACT\_APP\_THEME=light

Restart the app if you modify the .env file.

### ****4️⃣ Stop the Server****

To stop the development server, press:

sh

CopyEdit

CTRL + C

## **✅ Summary of Commands**

| **Command** | **Description** |
| --- | --- |
| npm install | Installs project dependencies. |
| npm start | Starts the frontend React development server. |
| CTRL + C | Stops the running server. |

1. **Component Documentation**
   * **Key Components**: Document major components, their purpose, and any props they receive.

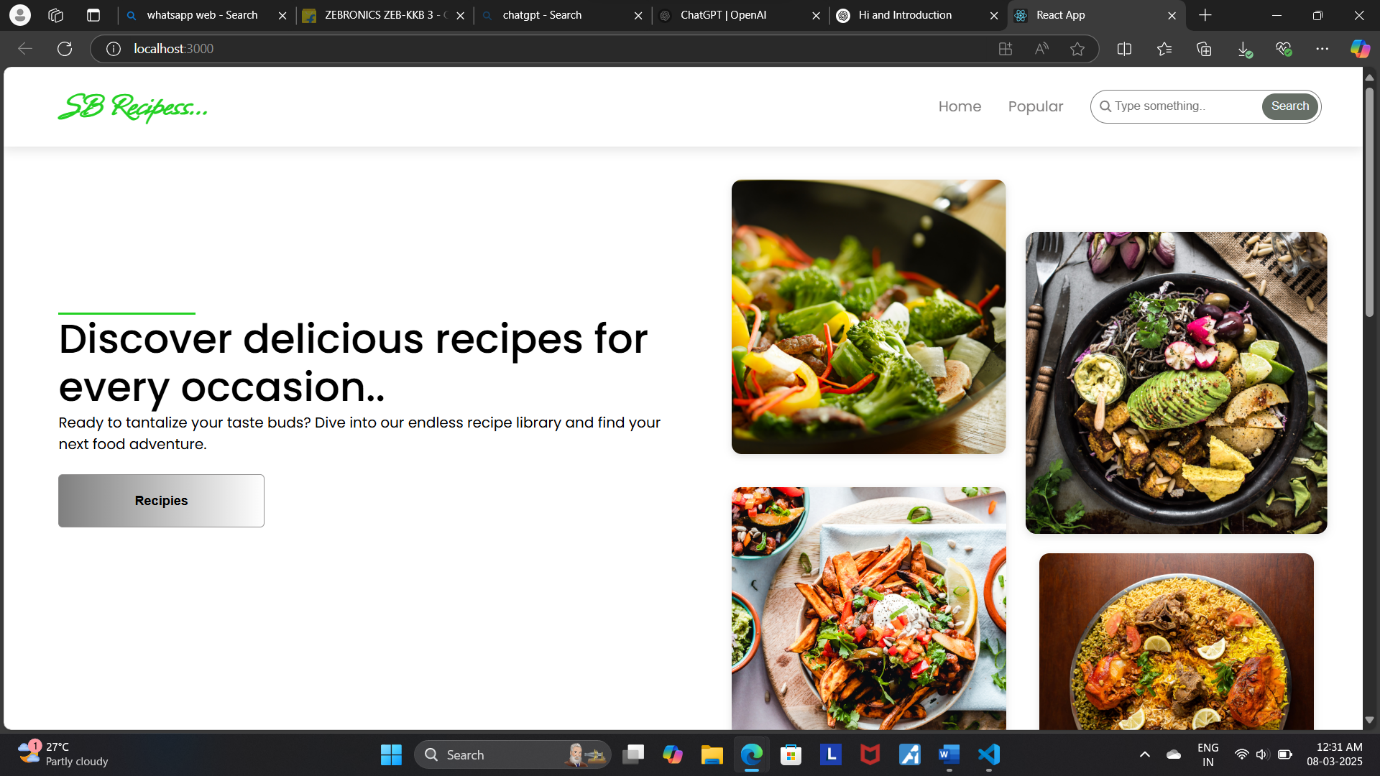
| **Component** | **Purpose** | **Props** |
| --- | --- | --- |
| Navbar | Displays the navigation menu. | user (object), onLogout (function) |
| RecipeCard | Displays a recipe preview. | title, image, description, onClick |
| RecipeList | Displays a list of RecipeCard components. | recipes (array) |
| RecipeDetail | Shows full details of a recipe. | recipeId |
| PantryItem | Represents a single pantry ingredient. | name, quantity |
| PantryList | Displays a list of pantry items. | items (array) |
| MealPlanner | Allows users to schedule meals. | meals (array) |
| Button | Reusable button component. | text, onClick, variant |
| Modal | Generic modal popup component. | isOpen, onClose, children |

* + **Reusable Components**: Detail any reusable components and their configurations.

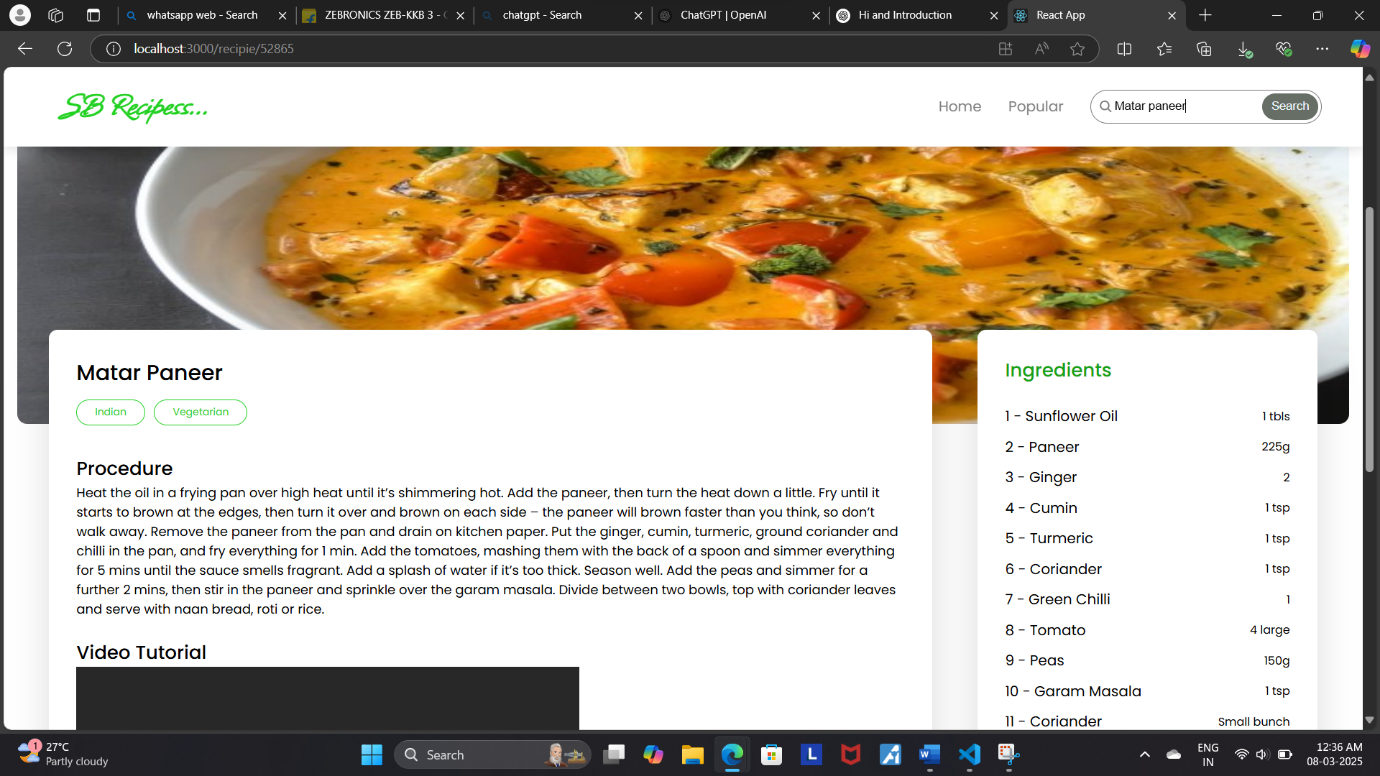
| **Component** | **Purpose** | **Props** |
| --- | --- | --- |
| Button | A customizable button with different styles. | text, onClick, variant, disabled |
| InputField | A reusable input field with validation. | type, placeholder, value, onChange, error |
| Modal | A generic modal for displaying popups. | isOpen, onClose, children |
| Card | A flexible card for displaying content. | title, image, children |
| Spinner | A loading indicator. | size, color |
| Tooltip | A hover tooltip for extra information. | text, position |
| Toast | A notification message. | message, type, onClose |

1. **State Management**
   * **Global State**: Describe global state management and how state flows across the application.
   * State management in the CookBook application ensures that data flows smoothly between components, providing a seamless user experience. We use a **combination of Context API and Redux (or Zustand)** to manage both **localized** and **global** states effectively.
   * **Local State**: Explain the handling of local states within components.
   * Local state is used within individual components to **manage UI interactions and temporary data**. In the CookBook application, **React's useState hook** is primarily used for handling local state efficiently.
2. **User Interface**
   * Provide screenshots or GIFs showcasing different UI features, such as pages, forms, or interactions.

**SCREENSHOTS OF WEB PAGE:**

****

**USER INTERFACE:**

****

1. **Styling**

* **CSS Frameworks/Libraries**: Describe any CSS frameworks, libraries, or pre-processors (e.g., Sass, Styled-Components) used.

# **🎨 CSS Frameworks & Libraries in CookBook**

The CookBook application uses a combination of **modern CSS tools** to create a visually appealing and responsive UI. Below are the key CSS frameworks, libraries, and techniques used:

## **📌 1️⃣ CSS Frameworks & Libraries**

### ****🔹 Tailwind CSS**** (Primary Styling Framework)

✅ Utility-first framework for rapid styling.  
✅ Highly customizable via tailwind.config.js.  
✅ Reduces the need for writing custom CSS classes.

* **Theming**: Explain if theming or custom design systems are implemented.

# **🎨 Theming & Custom Design System in CookBook**

The CookBook application supports **dynamic theming** to enhance **user experience, accessibility, and brand consistency**. Below is an overview of how theming is implemented.

## **📌 1️⃣ Theming Implementation**

### ****🔹 Dark Mode & Light Mode Support****

* **Tailwind’s dark: classes** are used for conditional styling.
* Users can switch between themes using local storage or system preferences.

### ****🌙 Theming with Tailwind & Local Storage in CookBook****

CookBook supports **light & dark mode** using **Tailwind’s dark: classes** and **local storage** for user preferences. The theme can automatically adapt to the **user’s system settings** or be manually toggled.

* **ShadCN’s useTheme hook** manages the theme state.

### ****🎨 Theming with ShadCN’s**** useTheme ****Hook in CookBook****

CookBook leverages **ShadCN’s useTheme hook** to efficiently manage **light and dark mode**. This ensures:  
✅ **Automatic system detection** (follows OS preference).  
✅ **User-controlled theme switching** via a toggle.  
✅ **Persistent theme selection** using **local storage**.

1. **Testing**

* **Testing Strategy**: Describe the testing approach for components, including unit, integration, and end-to-end testing (e.g., using Jest, React Testing Library).

# **🛠 Testing Strategy for CookBook**

CookBook follows a **comprehensive testing strategy** to ensure **reliability, maintainability, and performance** across the application. The testing approach includes:  
✅ **Unit Testing** → Individual components & functions  
✅ **Integration Testing** → Interaction between components  
✅ **End-to-End (E2E) Testing** → Full user flows

* **Code Coverage**: Explain any tools or techniques used for ensuring adequate test coverage.

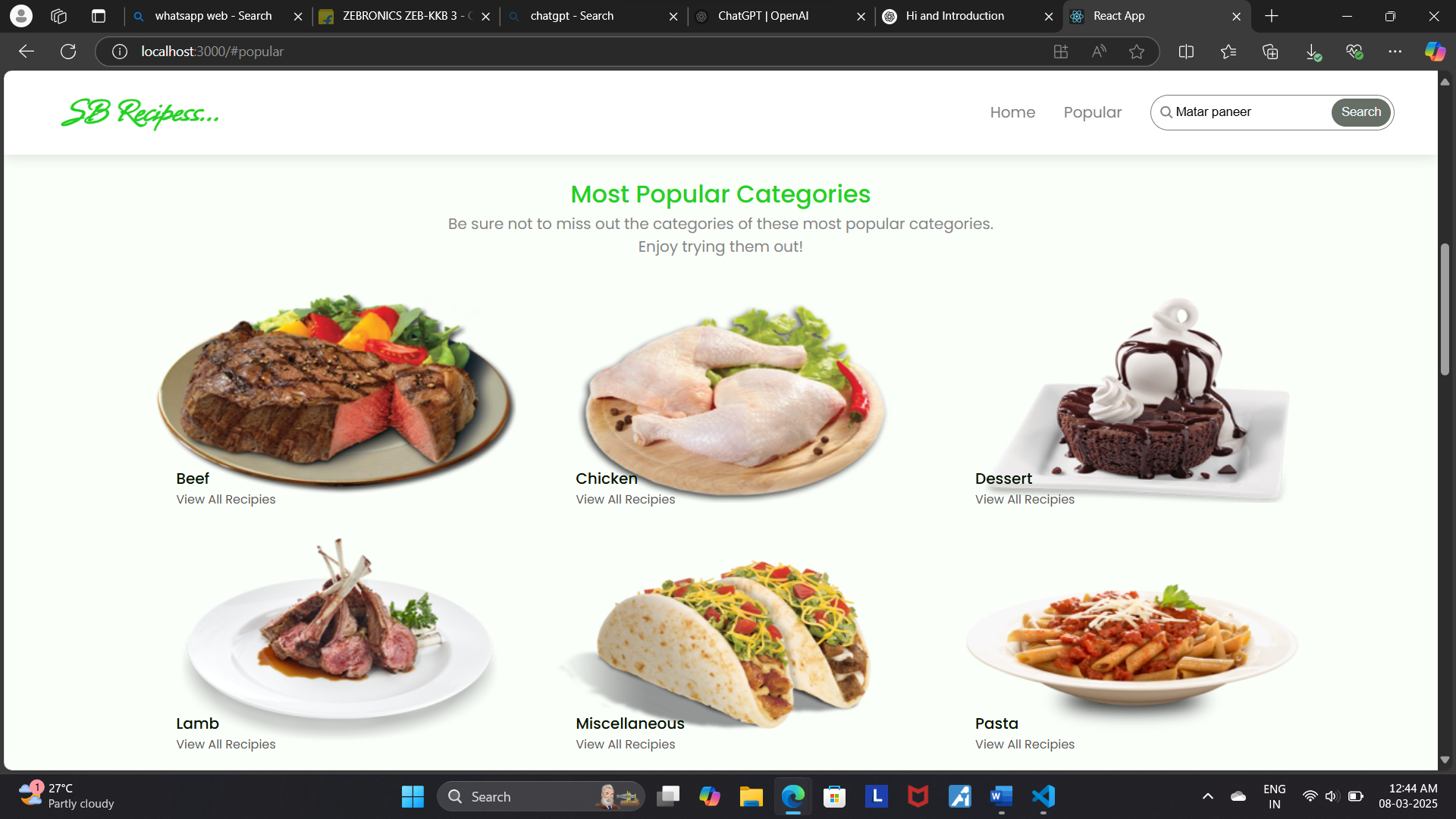
# **📊 Code Coverage in CookBook**

CookBook ensures **high test coverage** using **Jest’s built-in coverage reporting**. This helps:  
✅ Identify **untested code**  
✅ Improve **reliability**  
✅ Maintain a **robust testing strategy**

1. **Screenshots or Demo**

* Provide screenshots or a link to a demo showcasing the application’s features and design.

[React App](http://localhost:3000/)



1. **Known Issues**

# Document any known bugs or issues that users or developers should be aware of. **🐛 Known Bugs & Issues in CookBook**

This section documents **known issues** in CookBook, including **workarounds** where applicable.

## **📌 1️⃣ Dark Mode Flicker on Page Load**

### ****🔹 Issue:****

* When switching themes, the UI briefly **flashes in the wrong theme** before applying the correct one.
* Occurs due to **hydration mismatch in Next.js** (useTheme needs time to determine the theme).

### ****🔹 Workaround:****

* Ensure useTheme is only rendered **after mounting**:

1. **Future Enhancements**

* Outline potential future features or improvements, such as new components, animations, or enhanced styling.

# **🚀 Future Enhancements for CookBook**

CookBook is designed to **evolve** with **new features, optimizations, and better user experiences**. Below are **planned future enhancements** to make the platform even more powerful and user-friendly.

## **📌 1️⃣ AI-Powered Recipe Suggestions 🤖**

✅ **Enhancement:**

* Use AI to **suggest recipes** based on ingredients in the pantry.
* Recommend **meal plans** based on user preferences & dietary restrictions.

✅ **Implementation Plan:**

* Integrate **OpenAI’s GPT model** or **Spoonacular API** for recipe generation.
* Users enter available ingredients → AI suggests **recipes that can be made**.

✅ **Example Feature:**

js

CopyEdit

// AI-powered recipe suggestion (Mock API)

fetch(`/api/recipe-suggestions?ingredients=${userIngredients}`)

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

.then(data => setSuggestedRecipes(data));

✅ **Status:**  
🟡 **In Research Phase** (Exploring AI APIs).