**COOK BOOK: YOUR VIRTUAL KITCHEN ASSISTANT – PROJECT DOCUMENTATION**

1**. INTRODUCTION**:

* Project Title : Cookbook: your virtual kitchen assistant
* Team ID : NM2025TMID41379
* TREM LEADER : MAINSH S (jamburani4024@gmail.com)
* ROLE : CODING AND DEVALOPMENT
* TEAM MEMBER : KAVIPRIYAN K (kavipriyankumar19@gmail.com)
* ROLE : CODING AND DEVALOPMENT
* TEAM MEMBER : DHARANIVELAN D (velandharani03@gmail.com)
* ROLE : DEMO VIDEO
* TEAM MEMBER : HARISUDHAN S(harisudhans29@gmail.com)
* ROLE : DOCUMENT CREATER

**2. PROJECT OVERVIEW**

**Purpose:**

Cookbook is a React.Js-based web app designed to be your personal kitchen helper. It lets you explore, add, and manage recipes easily, plan your meals, and create shopping lists — all in one place. The goal is to make cooking simpler and more enjoyable by providing a clean, user-friendly interface.

**Features:**

* Browse a variety of recipes with search and filter options
* Add your own recipes and edit existing ones
* Plan your meals using an interactive calendar
* Generate shopping lists based on your meal plans
* User authentication to save and manage your profile

3**. ARCHITECTURE**

**Component Structure:**

The app is built with React components organized to keep things modular and easy to maintain:

* `App.js`: The root component that sets up routing and global state
* `Recipe List`: Displays all recipes with search and filter capabilities
* `Recipe Detail`: Shows detailed information about a selected recipe
* `Meal Planner`: A calendar interface for scheduling meals
* `Shopping List`: Generates shopping lists from planned meals
* `User Profile`: Allows users to manage their account and preferences

Components communicate through props and shared global state.

**State Management:**

The project uses Reacts Context API combined with `use Reducer` to manage global state efficiently without adding extra dependencies.

**Routing:**

React Router is used to handle navigation between pages:

* `/` – Home page with recipe listings
* `/recipe/:id` – Detailed recipe view
* `/planner` – Meal planner calendar
* `/shopping-list` – Shopping list page
* `/profile` – User profile and settings

4**. SETUP INSTRUCTIONS**

**Prerequisites**:

* Node.js (v14 or higher)
* NPM (v6 or higher)

**Installation Steps:**

1. Clone the project repository:

```bash

git clone https://github.com/mounish1815-lead/Cook-house-the-virtual-Kitchen.git ```

**2. Navigate to the client directory:**

```bash

Cd COOK-BOOK-/client

```

**3. Install dependencies:**

```bash

npm install

```

4. Create a `. en v` file in the `client` folder and add any necessary environment variables (e.g., API URLs).

**5. FOLDER STRUCTURE**

Client:

The React app is organized as follows:

```

client/

├── public/ # Static files like index.html and images

├── src/

│ ├── assets/ # Images, icons, fonts

│ ├── components/ # Reusable UI components (buttons, modals, inputs)

│ ├── pages/ # Page components (Recipe List, Recipe Detail, etc.)

│ ├── context/ # Context providers and reducers for state management

│ ├── hooks/ # Custom React hooks

│ ├── utils/ # Helper functions

│ ├── styles/ # CSS and styled-components

│ └── App.js # Root component

└── package. Json

```

**Utilities:**

* API helper functions for fetching data
* Custom hooks for local Storage syncing and other reusable logic

6**. RUNNING THE APPLICATION**

* To start the app locally, run this inside the `client` folder:

```bash

npm start

```

* Open your browser and go to `http://localhost:3000` to see the app in action.

**7. COMPONENT DOCUMENTATION**

**Key Components**:

`Recipe List`

Purpose: Displays a searchable list of recipes

Props:

* `recipes` (array) – List of recipe objects
* `on Select Recipe` (function) – Call back when a recipe is clicked

`Recipe Detail`

* Purpose: Shows detailed recipe info including ingredients and instructions
* Props:
* `Recipeid ` (string) – ID of the recipe to display

`Meal Planner`

* + Purpose: Lets users schedule meals on a calendar
  + Props: None (uses global state)

**Reusable Components:**

* `Button` – Customizable buttons for various actions
* `Modal` – For dialogs and confirmations
* `Input` – Form inputs with validation

**8. STATE MANAGEMENT**

Global State:

Managed with React Context and `use Reducer`, global state includes user info, recipes, meal plans, and shopping lists. This keeps the app data consistent and easy to update.

Local State:

Components use `use State` for temporary UI states like form inputs and modal visibility.

**9. USER INTERFACE**

* The UI is clean, modern, and responsive, designed to work well on both desktop and mobile devices.
* Highlights include:
* Recipe browsing with search and filters
* Detailed recipe pages with clear instructions
* Meal planner calendar for easy scheduling
* Shopping list with checkable items

**10. STYLING**

**CSS Frameworks/Libraries:**

The app uses Styled-Components for styling, allowing CSS to be written directly in JavaScript with support for dynamic Thaming

**Thaming:**

Supports light and dark modes, with user preferences saved for a personalized experience.

**11. TESTING**

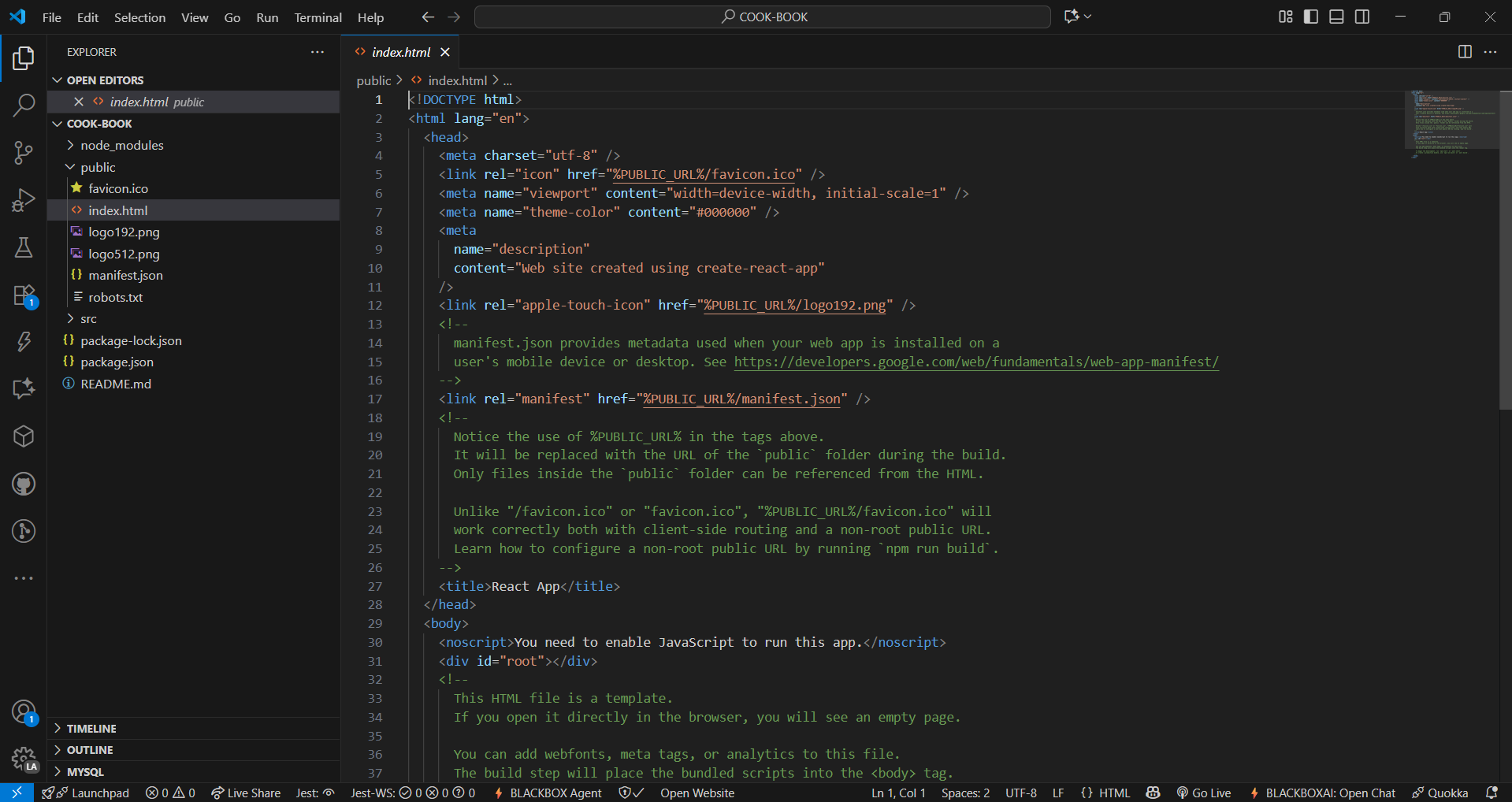
Testing Strategy:

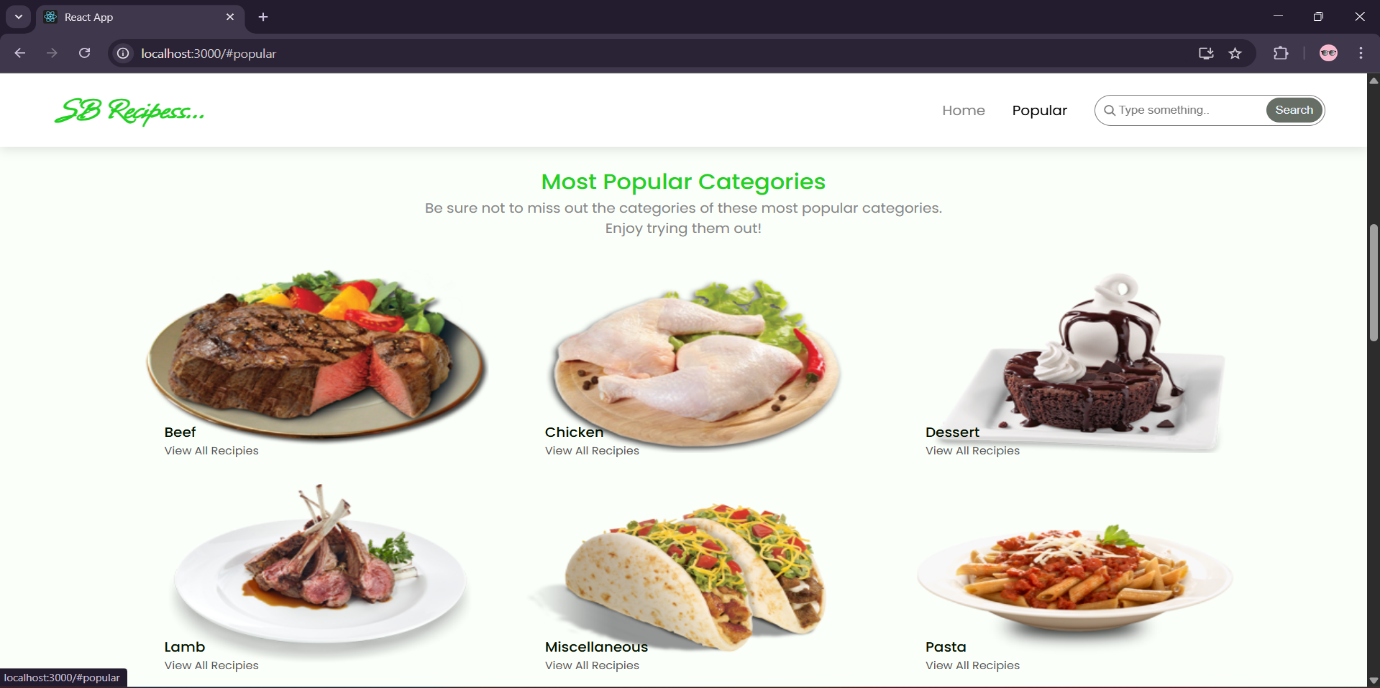
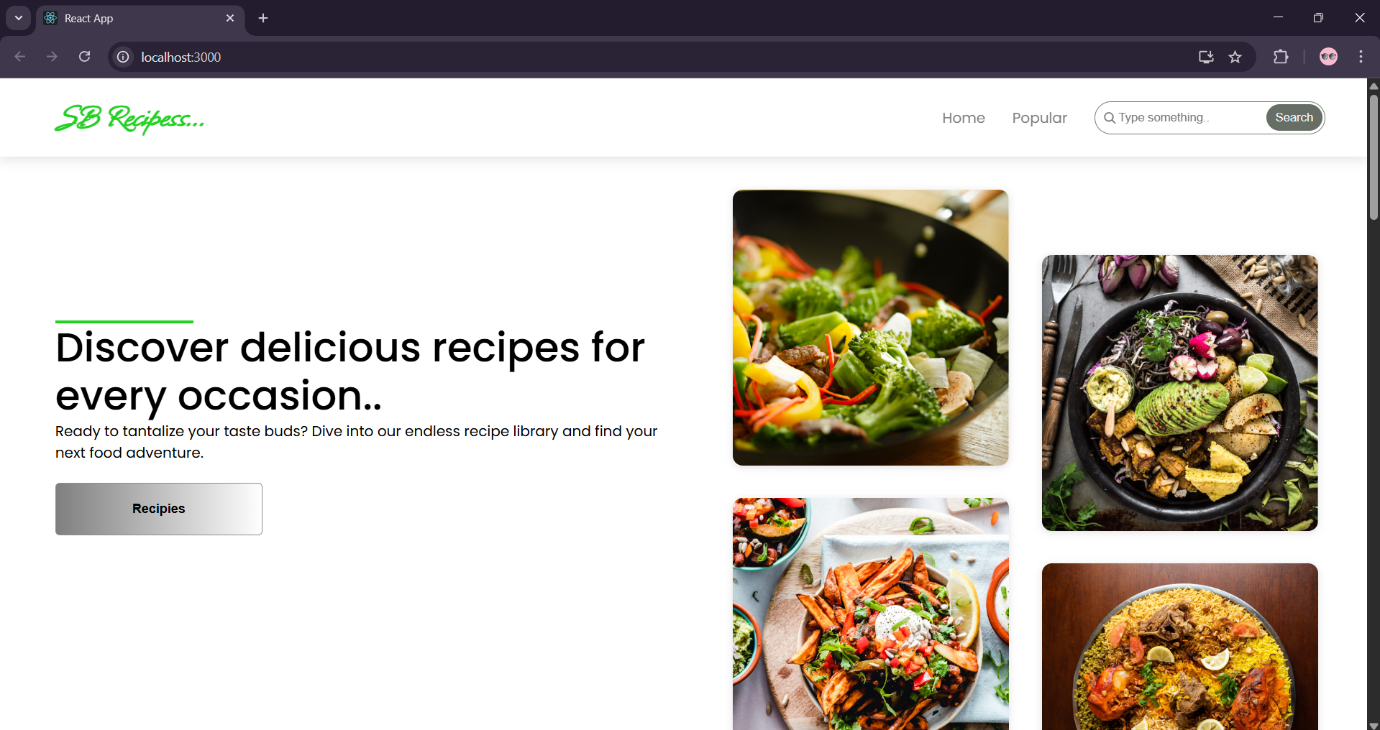
* Unit tests for components using Jest and React Testing Library
* Integration tests to verify component interactions
* Plans to add end-to-end tests with Cypress in the future

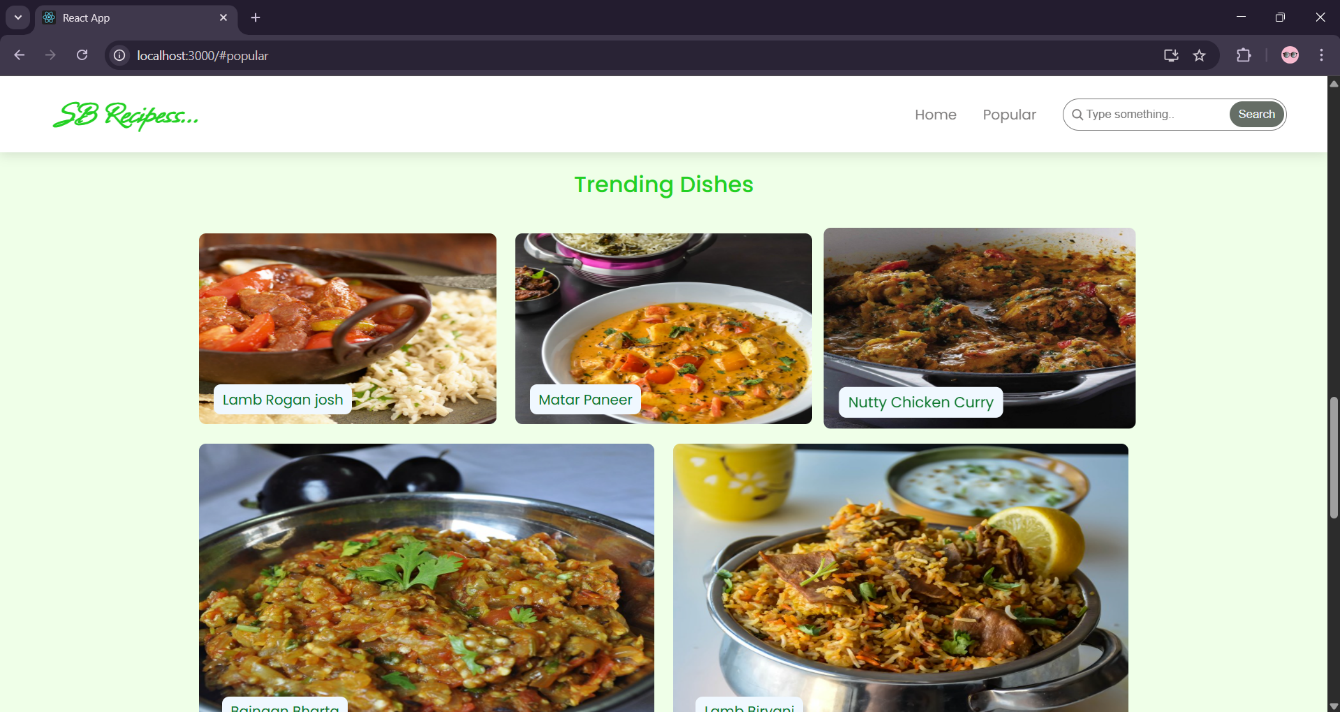
**Code Coverage:**

Jest’s coverage reports help ensure important parts of the app are well tested.

**12. SCREENSHOTS OR DEMO**

Coding:

Output: 



**13. KNOWN ISSUES**

* The meal planner calendar may sometimes not update immediately after adding a meal — refreshing the page fixes this.
* Some recipe images might fail to load if the source URL is broken.
* User profile updates can occasionally take a moment to reflect due to API response delays.

**14. FUTURE ENHANCEMENTS**

* Add social login options like Google and Facebook
* Implement offline support with service workers
* Add drag-and-drop functionality to the meal planner
* Include smooth animations for better user experience
* Expand testing with full end-to-end coverage using Cypress
* Add voice command support for hands-free navigation