**\*\*Frontend Development with React.js\*\***

**\*\*Project Documentation for CookBook\*\***

### 1. Introduction

- \*\*Project Title:\*\* CookBook

- \*\*Team Members:\*\*

- Vijay K (Team Leader) [Email Id: [vijaynithilan282004@gmail.com](mailto:vijaynithilan282004@gmail.com) ]

- Hariprasath P [Email Id: [pcharidon1817@gmail.com](mailto:pcharidon1817@gmail.com) ]

- Ravi M [Email Id: [ravi987865@gmail.com](mailto:ravi987865@gmail.com) ]

- Anish Kumar A [Email Id: [anishkmr1204@gmail.com](mailto:anishkmr1204@gmail.com) ]

### 2. Project Overview

- \*\*Purpose:\*\*

CookBook is a web application designed to help users discover, save, and manage recipes. It provides a platform to browse a wide variety of dishes, search for recipes by ingredients or categories, and create personalized recipe collections.

- \*\*Features:\*\*

- Recipe browsing with filters (e.g., cuisine, dietary preferences).

- Search functionality for recipes, ingredients, or cooking time.

- User authentication (login/signup).

- Personal recipe collection management (add, edit, delete).

- Responsive design for desktop and mobile use.

### 3. Architecture

- \*\*Component Structure:\*\*

The application is built using React.js with a modular, component-based architecture. Key components include:

</CONTENT>

</PAGE1>

<PAGE2>

<CONTENT>

- \*\*Header:\*\* Navigation bar and search input.

- \*\*RecipeCard:\*\* Displays individual recipe details (title, image, prep time).

- \*\*Sidebar:\*\* Filters and navigation links (e.g., categories, favorites).

- \*\*HomePage:\*\* Showcases featured recipes, trending dishes, and categories.

- \*\*SearchPage:\*\* Interface for searching recipes.

- \*\*CollectionPage:\*\* Displays and manages user recipe collections.

- \*\*State Management:\*\*

The app uses [Redux/Context API] for global state management, handling user data, recipe collections, and search results.

- \*\*Routing:\*\*

Navigation is managed with React Router. Defined routes include:

- `/`: Home page

- `/search`: Search page

- `/collection/:id`: User collection details page

- `/login`: Login page

### 4. Setup Instructions

- \*\*Prerequisites:\*\*

- Node.js (v16 or higher)

- npm (v8 or higher)

- Git

- \*\*Installation:\*\*

1. Clone the repository: `git clone [your-repo-link]`

2. Navigate to the project directory: `cd cookbook/client`

3. Install dependencies: `npm install`

4. Configure environment variables: Create a `.env` file in the client directory and add required variables (e.g., API keys, backend URL).

5. Start the development server: `npm start`

</CONTENT>

</PAGE2>

<PAGE3>

<CONTENT>

### 5. Folder Structure

- \*\*Client:\*\*

- `src/components`: Reusable components (e.g., Header, RecipeCard).

- `src/pages`: Page-specific components (e.g., HomePage, SearchPage).

- `src/assets`: Static files like images and icons.

- `src/[redux/context]`: State management setup (store, actions, reducers).

- `src/utils`: Utility functions and helpers.

- `App.js`: Main app component.

- `index.js`: Entry point.

- \*\*Utilities:\*\*

- `api.js`: Manages API calls to fetch recipe data.

- `auth.js`: Handles authentication logic and token management.

- `hooks/useRecipe.js`: Custom hook for recipe-related functionality.

### 6. Running the Application

- \*\*Frontend:\*\*

- Start the frontend server: `npm start` (in the `client` directory).

- Additional setup (if applicable):

- `npm install`

- `npx json-server [your-db-file]` (if using a mock backend).

- `npm run dev`

- Access the app at: `http://localhost:3000`

### 7. Component Documentation

- \*\*Key Components:\*\*

- \*\*Header:\*\* Renders navigation and search bar.

- \*\*Props:\*\* `onSearch` (function for search input handling).

- \*\*RecipeCard:\*\* Displays recipe summary (title, image, etc.).

- \*\*Props:\*\* `recipe` (recipe object), `onClick` (selection handler).

</CONTENT>

</PAGE3>

<PAGE4>

<CONTENT>

- \*\*CollectionCard:\*\* Displays a user’s recipe collection.

- \*\*Props:\*\* `collection` (collection object), `onEdit` (edit handler).

- \*\*Reusable Components:\*\*

- \*\*Button:\*\* Customizable button component.

- \*\*Props:\*\* `text`, `onClick`, `disabled`.

- \*\*Input:\*\* Reusable input field for forms/search.

- \*\*Props:\*\* `type`, `placeholder`, `value`, `onChange`.

### 8. State Management

- \*\*Global State:\*\*

The [Redux/Context] store manages:

- `user`: Authenticated user data.

- `recipes`: List of available recipes and filters.

- `collections`: User recipe collections.

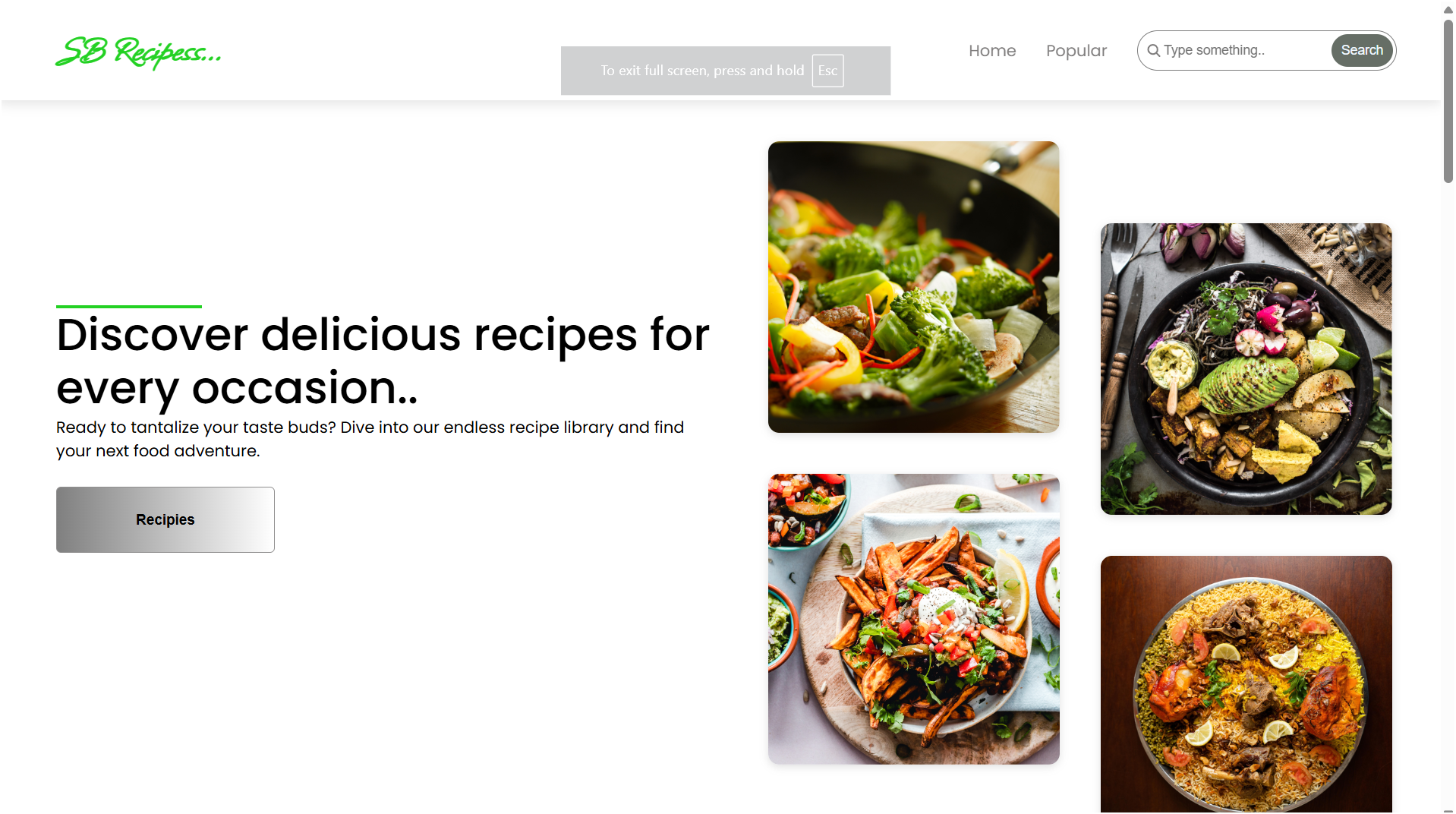
- `searchResults`: Results from search queries.

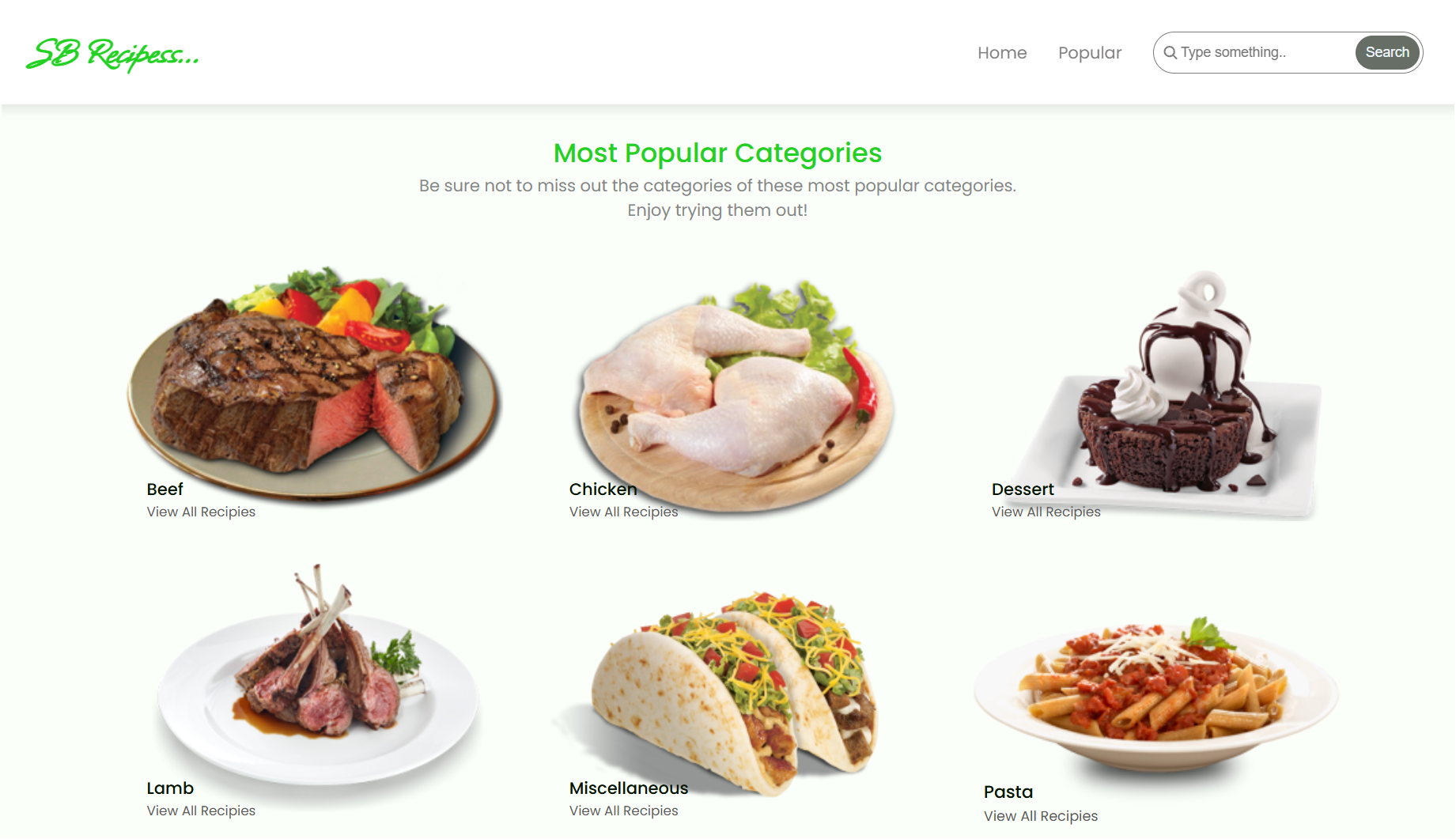
- \*\*Local State:\*\*

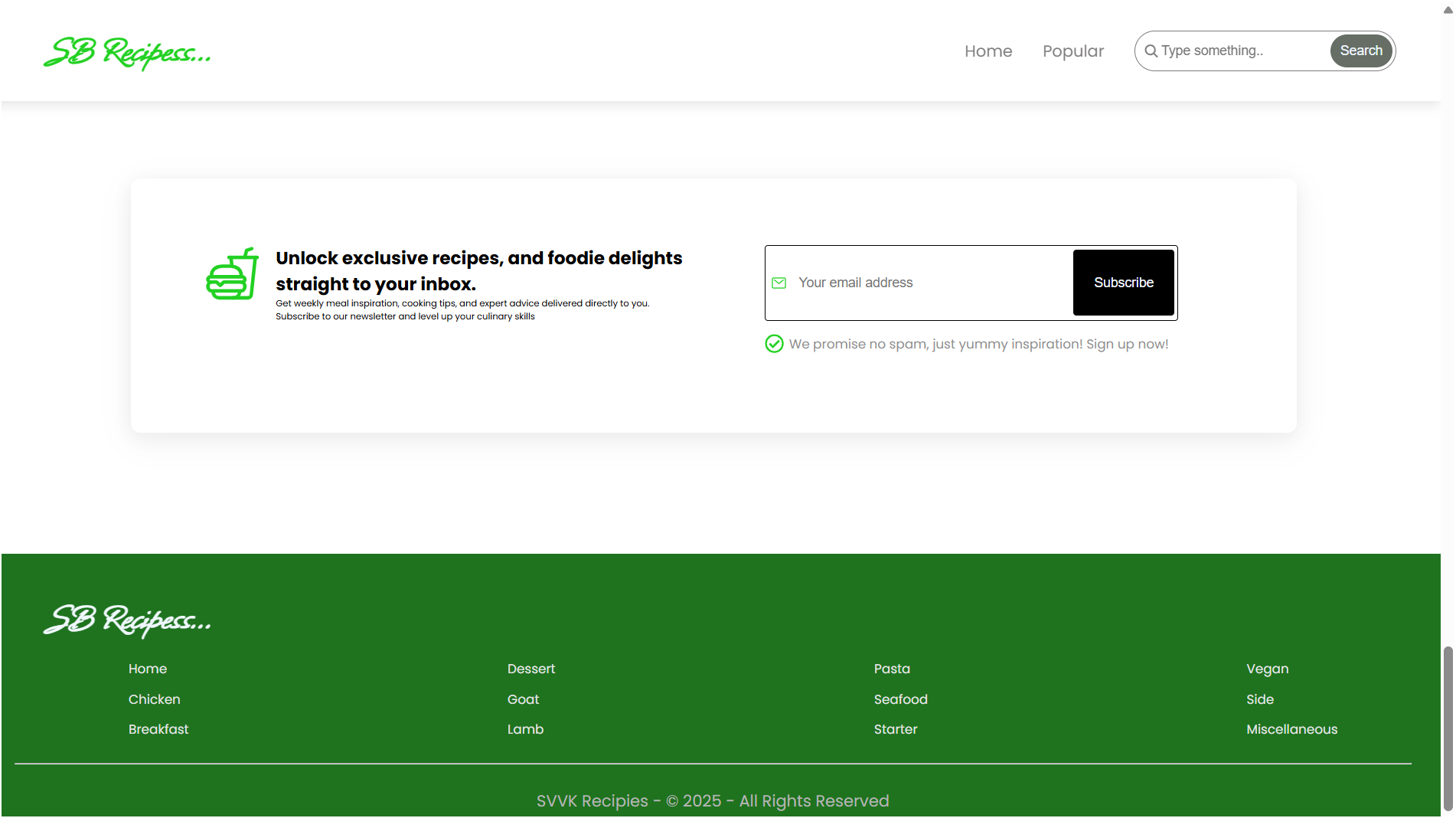
Local state is handled with React’s `useState` hook within components, e.g., search input state in `SearchPage`.

### 9. User Interface

- \*\*Screenshots:\*\*







### 10. Styling

- \*\*CSS Frameworks/Libraries:\*\*

The app uses [Styled-Components/Tailwind CSS/Material-UI] for styling, ensuring modular and visually appealing designs.

- \*\*Theming:\*\*

A custom theme is implemented with support for [light/dark modes or specific styles].

### 11. Testing

- \*\*Testing Strategy:\*\*

- \*\*Unit Testing:\*\* Conducted with Jest and React Testing Library.

- \*\*Integration Testing:\*\* Ensures component interoperability.

- \*\*End-to-End Testing:\*\* Performed with [Cypress/another tool] for user workflows.

- \*\*Code Coverage:\*\*

Monitored using [Jest/tool name], targeting [your coverage percentage, e.g., 80%].

### 13. Known Issues

- \*\*Issue 1:\*\* [e.g., Recipe filters reset unexpectedly on page refresh.]

- \*\*Issue 2:\*\* [e.g., Slow loading of large recipe datasets.]

### 14. Future Enhancements

- \*\*Future Features:\*\*

- Add a meal planner integrating saved recipes.

- Implement a shopping list generator based on recipe ingredients.

- Enhance UI with recipe preparation step animations.