🥗 SB Recipiess - Frontend Development with React.js

1. Introduction

- **Project Title**: SB Recipiess
- **Team Members**:

  Dhivakaran L, Aadithya D, Chandhru T, Hariashwinth P

---

2. Project Overview

- **Purpose**:
  SB Recipiess is a responsive and visually engaging recipe application designed to help users explore a variety of recipes categorized by type (e.g., chicken, vegetarian, seafood, etc.). It allows users to browse, search, and view dishes for every occasion.
- **Features**:
  - Homepage with highlighted images and intro text.
  - Category browsing (e.g., Chicken, Vegetarian, Dessert).
  - Search functionality.
  - Interactive navigation bar.
  - Recipe cards with images and names.

---

3. Architecture

- **Component Structure**:
    - `Navbar`: Handles navigation and search.
    - `Home`: Displays the landing section with featured images and a call-to-action.
    - `Category`: Displays recipes by category (e.g., Chicken).
    - `RecipeCard`: A reusable component for individual recipe previews.
    - `Footer`: (Optional, based on actual implementation)
- **State Management**:
    - Primarily uses **React `useState` and `useEffect` hooks** for local component state.
    - May use **Context API** for shared data like selected categories (if implemented).
- **Routing**:
    - Implemented using **React Router DOM**.
    - Example routes:
        - `/`: Home
        - `/category/:type`: Category-based recipe listing
        - `/#recipes`: Anchor link navigation to recipe section

---

4. Setup Instructions

- **Prerequisites**:
    - Node.js (v14 or higher recommended)
    - npm or yarn

Installation:

```
 git clone https://github.com/your-username/sb-recipiess.git
cd sb-recipiess
npm install
```

-

---

5. Folder Structure

```
client/
│
├── public/
│   └── index.html
│
├── src/
│   ├── assets/          # Images and static assets
│   ├── components/      # Reusable components (Navbar, RecipeCard)
│   ├── pages/           # Home, Category
│   ├── App.js           # Main component with routing
│   ├── index.js         # Entry point
│   └── styles/          # CSS or styled-components
```

- **Utilities**:
    - Custom hooks or utility functions for filtering or API calls can be placed in `src/utils/` (if applicable).

---

6. Running the Application

npm start

Runs the app in development mode at http://localhost:3000

---

7. Component Documentation

Key Components

- **Navbar**
    - Props: None
    - Features: Navigation links (`Home`, `Popular`), search bar
- **Home**
    - Landing section with introductory message, call-to-action button, and recipe grid.
- **CategoryPage**
    - Props: `category` (e.g., Chicken) from route params
    - Fetches and displays relevant recipes.
- **RecipeCard**
    - Props: `title`, `image`, etc.
    - Reusable card layout for each recipe.

---

8. State Management

- **Global State**:
    - Minimal (if Context API is not used). Could be added to manage selected categories or user preferences.
- **Local State**:
    - Managed using `useState` within components (e.g., search input, fetched recipe data).

---

9. User Interface

UI Highlights

- Clean and modern UI
- Large banner with title and CTA
- Recipe image cards with hover effects
- Category filter buttons
- Responsive layout

📸 **Screenshots**:

- Homepage: Hero section with intro and food grid.
- Category Page: Filtered recipe cards by selected category.

---

10. Styling

- **CSS Frameworks/Libraries**:
    - Basic **CSS** or **CSS Modules**
    - Optionally styled-components or Tailwind (if used, mention it)
- **Theming**:
    - Consistent green color for branding (`SB Recipiess`)
    - Light theme with clean whitespace and padding

---

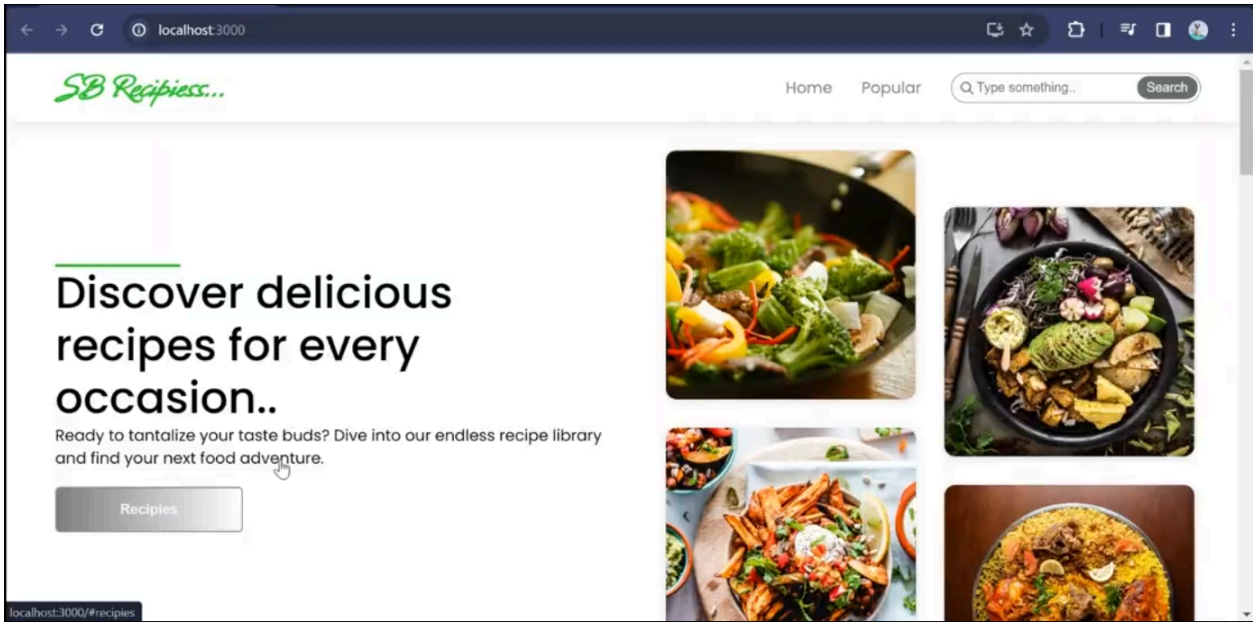
11. Testing

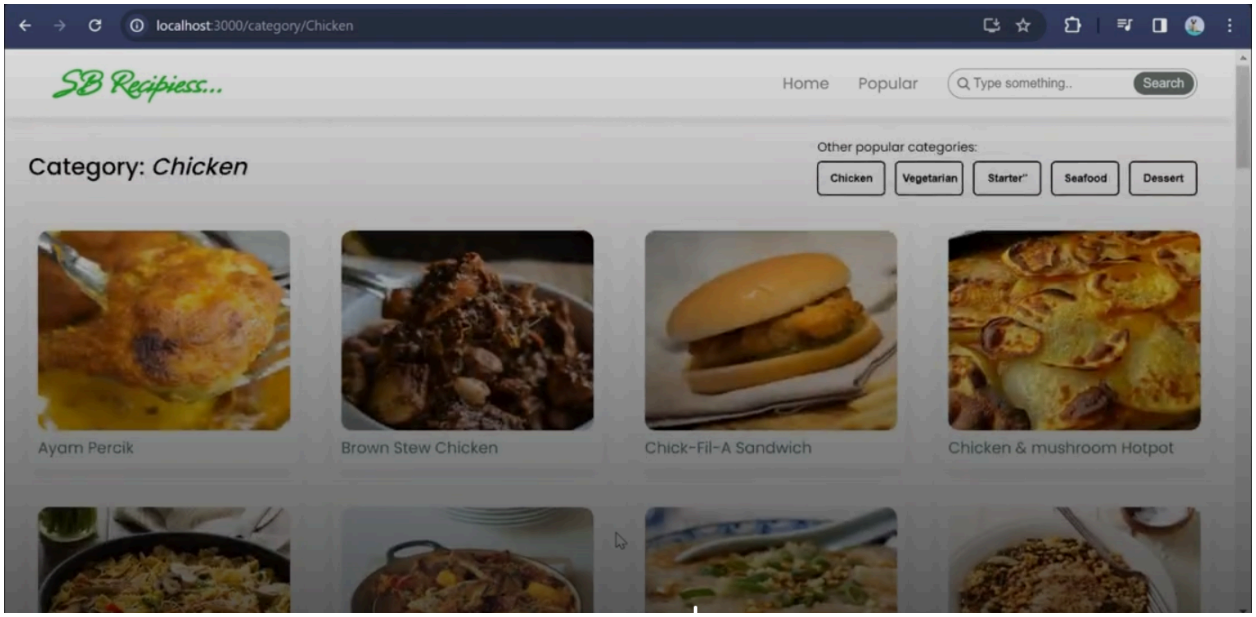
- **Testing Strategy**:

    - [Planned or basic testing using React Testing Library]
    - Component-level tests (e.g., for `RecipeCard`, `Navbar`)
- **Code Coverage**:
    - Not implemented yet or minimal (add tools like Jest if used)

---

12. Screenshots / Demo





---

13. Known Issues

- No actual backend/API integration (if applicable).
- Search functionality may not be fully dynamic.
- No pagination or lazy loading for recipe cards.

---

14. Future Enhancements

- Integrate a real recipe API (e.g., Spoonacular).
- Add individual recipe detail pages.
- Add user login and favorite recipe features.
- Improve search filtering and sorting options.
- Enhance UI with animations and mobile responsiveness.
- Implement dark mode and accessibility features.
