**1. Introduction**

* **Project Title**: COOK BOOK
* **Team Members**:

TEAM LEADER: NALINI K

TEAM MEMBER: SHANMATHI R

TEAM MEMBER: SASIKALA M

TEAM MEMBER: YUVARANI V

TEAM MEMBER: ANUSHIYA M

**Overview**:  
The Multi-Cuisine Cookbook is a web-based platform designed to offer a curated collection of recipes from around the world. Each recipe is equipped with ingredients, step-by-step instructions, and YouTube video tutorials. The front-end uses React.js to provide a dynamic, user-friendly experience for users of all skill levels. This document outlines the development process, architecture, and instructions for setting up and using the app.

**2. Project Overview**

* **Purpose**:  
  The *Multi-Cuisine Cookbook* aims to provide users with a comprehensive platform to discover global recipes. Users can search for recipes by cuisine, dish type, or ingredient. Each recipe is accompanied by detailed cooking instructions, ingredients, and YouTube tutorials for better understanding. The frontend was developed using React.js to ensure a rich user experience with quick data updates and real-time interactions.
* **Features**:
  + **Recipe Search**: Allows users to search by ingredients, cuisines, or recipe type.
  + **Recipe Details**: View detailed recipe instructions, ingredients, and a YouTube tutorial.
  + **Mobile Optimization**: Fully responsive, allowing users to view and interact with the content on various devices.
  + **Light/Dark Mode**: Theme toggle functionality that allows the user to switch between light and dark modes.
  + **Filter by Dietary Preferences**: Users can filter recipes based on vegetarian, vegan, gluten-free, or other dietary needs.
* **Technologies**:
  + React.js
  + react-router-dom
  + Styled-Components for CSS
  + Axios for API calls

**3. Architecture**

* **Component Structure**:
  + **App.js**: Root component that includes routing and the global layout structure.
  + **Header.js**: Displays the main navigation bar, logo, and the search bar component.
  + **RecipeList.js**: A container component that fetches and displays recipes.
  + **RecipeCard.js**: A reusable component that displays a short summary of a recipe (name, image, brief description).
  + **RecipeDetail.js**: Displays the full details of a selected recipe including ingredients, instructions, and YouTube video tutorial.
  + **SearchBar.js**: Contains the search functionality, accepting user input and filtering the recipes.
  + **Footer.js**: Displays copyright, social media links, and additional information.
* **State Management**:
  + **Global State**:
    - **SearchContext**: Manages search queries and updates the recipe list when a query is submitted.
    - **ThemeContext**: Manages the application’s theme (light/dark mode) using Context API.
  + **Local State**:
    - **useState** is used to manage component-specific states, such as loading states or the selected recipe.
* **Routing**:
  + The application uses **react-router-dom** to manage routing. Key routes include:
    - /: Home page where users see a list of recipes.
    - /recipe/:id: Detailed page showing information about a specific recipe.
    - /search: Search results page displaying filtered recipes based on search criteria.

**4. Setup Instructions**

* **Prerequisites**:
  + **Node.js** (version 14 or higher)
  + **npm** (Node Package Manager) or **yarn** for installing dependencies
  + A **GitHub** account to clone the repository
* **Installation Steps**:
  + Clone the repository to your local machine:

bash

Copy

git clone https://github.com/your-username/multi-cuisine-cookbook-frontend.git

* + Navigate to the project folder:

bash

Copy

cd multi-cuisine-cookbook-frontend

* + Install the project dependencies:

bash

Copy

npm install

* + Set up any required environment variables (e.g., API keys, base URLs):
    - Create a .env file in the root directory and add necessary environment configurations like REACT\_APP\_API\_URL.
  + Start the development server:

bash

Copy

npm start

Visit http://localhost:3000 in your browser to view the app.

**5. Folder Structure**

The project follows a clean, modular folder structure to ensure scalability and maintainability:

* **public/**: Contains static files such as the index.html, favicons, and other assets.
* **src/**: Contains all source code for the React app.
  + **components/**: Contains reusable components like Header, RecipeCard, and Footer.
  + **pages/**: Includes page-level components like HomePage.js and RecipeDetailPage.js.
  + **assets/**: Stores images, icons, and other media assets.
  + **utils/**: Contains utility functions and custom hooks, such as useFetch.js for data fetching and useThemeToggle.js for theme switching.
  + **App.js**: Root component that incorporates the global layout, routing, and state management.
  + **styles/**: Contains global CSS files or Styled-Components for styling the app.

**6. Running the Application**

* To run the frontend server locally, use the following command:

bash

Copy

npm start

The application will be available at http://localhost:3000.

* **Testing**:
  + For development builds, ensure to use npm run build to create a production-ready build of the application.
  + The build folder can be deployed on any static file hosting platform like Netlify, Vercel, or Firebase Hosting.

**7. Component Documentation**

* **Key Components**:
  + **RecipeList**:
    - **Purpose**: Fetches and renders a list of recipes based on the search query or filter criteria.
    - **Props**: recipes (an array of recipe objects).
    - **State**: Uses local state for loading status and error handling.
  + **RecipeDetail**:
    - **Purpose**: Displays detailed information for a specific recipe.
    - **Props**: recipe (an object containing recipe details).
    - **State**: Uses useEffect to fetch data when the component is mounted.
* **Reusable Components**:
  + **SearchBar**:
    - **Purpose**: Provides a search bar for users to input a query and filter recipes.
    - **Props**: onSearch (function that handles search query).

**8. State Management**

* **Global State with Context API**:
  + **SearchContext**:
    - Stores the search query and provides it to all components requiring search functionality.
  + **ThemeContext**:
    - Manages the theme state, allowing users to toggle between light and dark modes.
* **Local State**:
  + Local states like loading states or error handling are managed using useState.
  + useEffect is used for side-effects, such as fetching data when the component loads.

**9. User Interface**

* **Key UI Elements**:
  + **Homepage**: Displays a dynamic list of recipes based on a user’s search or category selection. The search bar is always accessible at the top.
  + **Recipe Details**: Detailed page showcasing the recipe instructions, ingredients, and a YouTube tutorial.
  + **Search and Filter**: Real-time filtering of recipes based on search input.
* **Screenshots/GIFs**: *(Insert screenshots of the app here to demonstrate its functionality and design)*

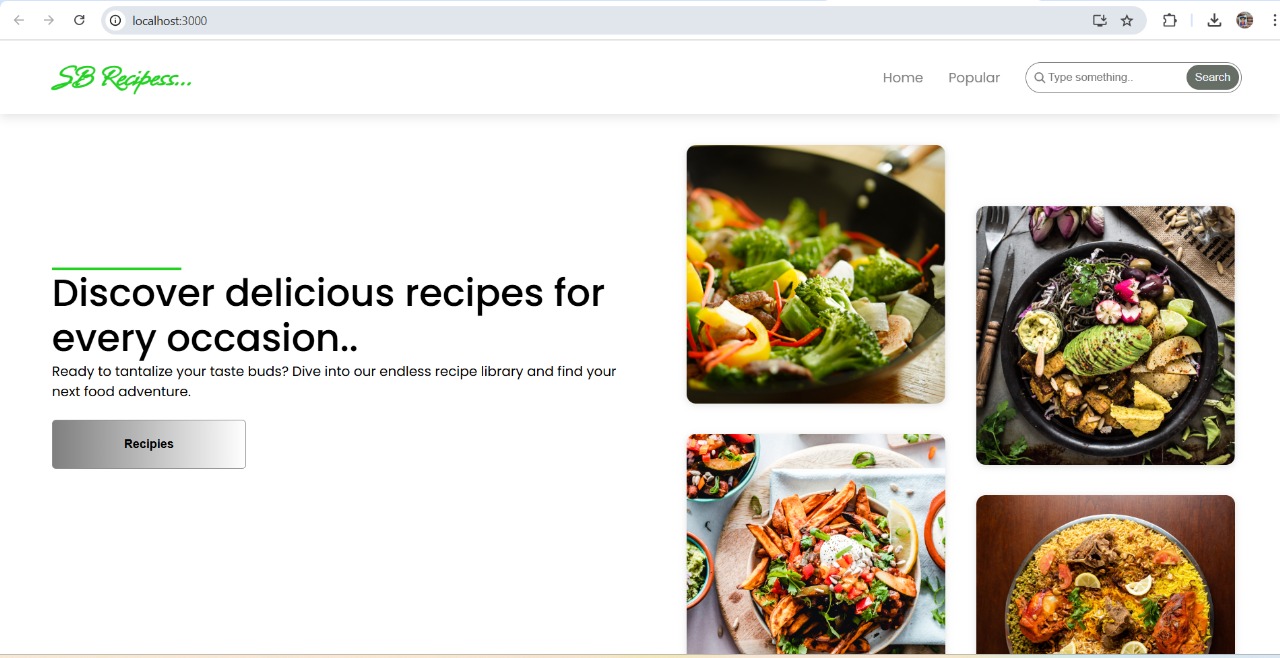
**10. Styling**

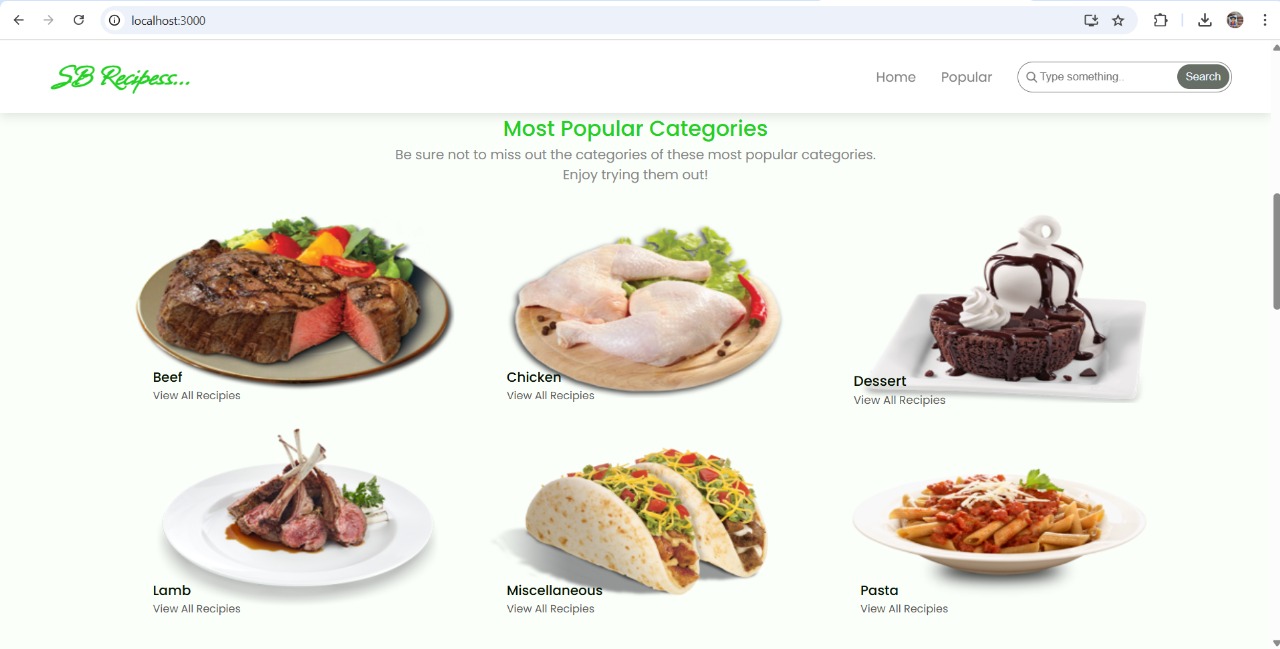
* **CSS Frameworks/Libraries**:
  + **Styled-Components**: Enables scoped styling and theming within components.
  + **SASS**: For more complex styles with reusable variables and mixins.
* **Theme**:
  + Implemented using **React Context API**, the theme can be toggled between light and dark modes. The selected theme is stored in local storage to persist across sessions.

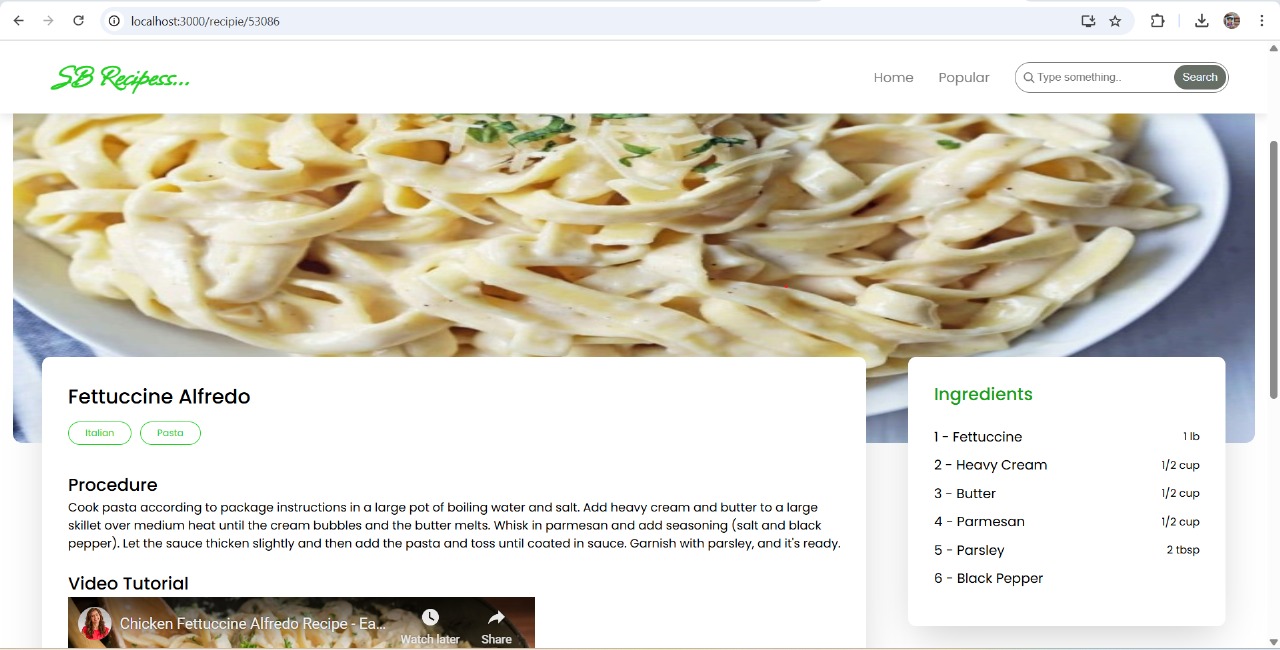
**11. Testing**

* **Testing Strategy**:
  + **Unit Testing**: Components are unit tested with Jest.
  + **Integration Testing**: Using React Testing Library to simulate user interactions and test component integration.
  + **End-to-End Testing**: Future plans include using Cypress for complete user interaction testing.
* **Code Coverage**:
  + Jest’s built-in coverage tool is used to ensure that all critical parts of the app are covered by tests.

**12. Screenshots or Demo**







**13. Known Issues**

* **YouTube Embeds**: Some YouTube videos fail to load if the external API has restrictions.
* **Search Delays**: Search functionality might be slow when the recipe database grows large.

**14. Future Enhancements**

* **User Authentication**: Implement login features to allow users to save their favorite recipes and track their cooking history.
* **Recipe Ratings and Reviews**: Allow users to rate and leave comments on recipes.
* **Advanced Filtering**: Additional filters like preparation time and difficulty level could be added.