Frotend Development with React.Js

1. Introduction

Project Title : CookBook: Your Virtual Kitchen Assistant.

Team Members :

T.KANIHA (HEAD) –ROLE-(Project Coding and Demo Video)

P.GOWSALYA- ROLE-(Project Coding)

S.HARIPRIYA- ROLE-(Project Documentation)

J.HARSHINI- ROLE-(Demo Video)

2. Project Overview

CookBook is a modern web application designed to transform recipe discovery, organization, and sharing. Built with an intuitive interface and powerful features, it caters to both beginners and professional chefs.

With dynamic search, easy navigation, and personalized recommendations, users can seamlessly explore and manage recipes. CookBook also fosters a collaborative community where cooking enthusiasts connect, share, and inspire one another.

Blending innovation with tradition, CookBook redefines culinary exploration—turning every recipe into an exciting adventure.

CookBook is designed to transform the way people discover, organize, and share recipes. It aims to bridge the gap between home cooks and professional chefs by offering a user-friendly platform that makes cooking inspiration, recipe management, and community interaction seamless and enjoyable.

Purpose :

The primary goal of CookBook is to provide a user-friendly platform that caters to individuals passionate about cooking, baking, and exploring new culinary horizons. Our objectives include:

- **User-Friendly Experience:** Create an interface that is easy to navigate, ensuring users can effortlessly discover, save, and share their favourite recipes.
- **Comprehensive Recipe Management:** Offer robust features for organizing and managing recipes, including advanced search options.
- **Technology Stack:** Leverage modern web development technologies, including React.js, to ensure an efficient, and enjoyable user experience.

3. Setup Instructions

PRE-REQUISITES:

Here are the key prerequisites for developing a frontend application using React.js:

- **Node.js and npm**:

  Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the local environment. It provides a scalable and efficient platform for building network applications.

  Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

  - Download: https://nodejs.org/en/download/
  - Installation instructions: https://nodejs.org/en/download/package-manager/

- **React.js**:

  React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

  Install React.js, a JavaScript library for building user interfaces.

  - Create a new React app:

    ```
    npx create-react-app my-react-app
    ```

    Replace my-react-app with your preferred project name.

  - Navigate to the project directory:

    ```
    cd my-react-app
    ```

  - Running the React App:

    With the React app created, you can now start the development server and see your React application in action.

  - Start the development server:

    ```
    npm start
    ```

  command launches the development server, and you can access

  your React app at http://localhost:3000 in your web browser.

- **HTML, CSS, and JavaScript**: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

- **Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.
    - Git: Download and installation instructions can be found at: https://git-scm.com/downloads

- **Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

- Visual Studio Code: Download from https://code.visualstudio.com/download
- Sublime Text: Download from https://www.sublimetext.com/download
- WebStorm: Download from https://www.jetbrains.com/webstorm/download

To get the Application project from drive:

Follow below steps:

Install Dependencies:

- Navigate into the cloned repository directory and install libraries:

```
cd fitness-app-react
npm install
```

- **Start the Development Server**:

- To start the development server, execute the following command:

```
npm start
```

Access the App:

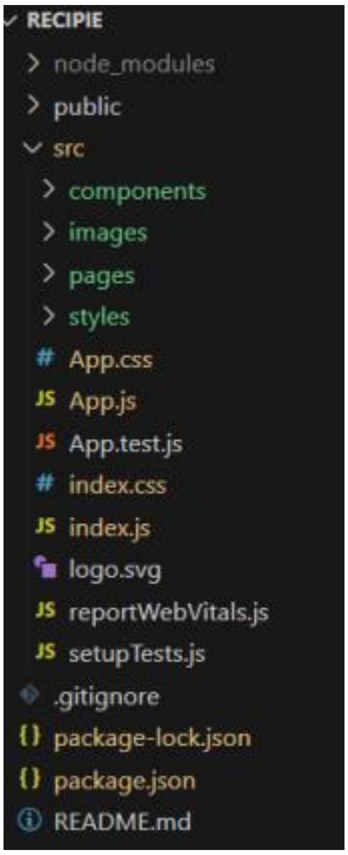
- Open your web browser and navigate to http://localhost:3000.

- You should see the application's homepage, indicating that the installation and setup were successful.

You have successfully installed and set up the application on your local machine. You can now proceed with further customization, development, and testing as needed.

4.Folder structure

In this project, we've split the files into 3 major folders, *Components, Pages and Styles.* In the pages folder, we store the files that acts as pages at different url's in the application. The components folder stores all the files, that returns the small components in the application. All the styling css files will be stored in the styles folder.



```
RECIPIE
  > node_modules
  > public
  v src
    > components
    > images
    > pages
    > styles
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

Running the Application

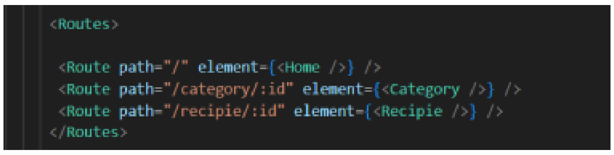
- **Start the Development Server**:
- To start the development server, execute the following command:

```
npm start
```

5.Project Development

- ? Setup the Routing paths

  Setup the clear routing paths to access various files in the application.



```
<Routes>

  <Route path="/" element={<Home />} />
  <Route path="/category/:id" element={<Category />} />
  <Route path="/recipie/:id" element={<Recipie />} />
</Routes>
```

? Develop the Navbar and Hero components
? Code the popular categories components and fetch the categories from ***themealsdb*** Api.
? Also, add the trending dishes in the home page.

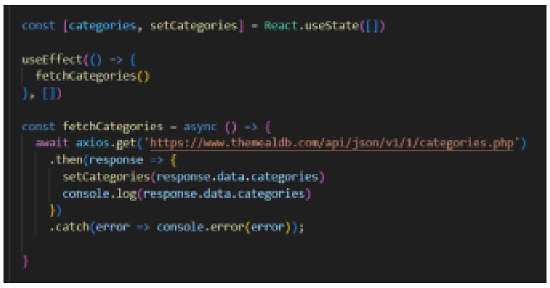
? Now, develop the category page to display various dishes under the category.

? Finally, code the recipe page, where the ingredients, instructions and a demo video will be integrated to make cooking much easier.

Important Code snips:

? **Fetching all the available categories**

Here, with the API request to Rapid API, we fetch all the available categories.



```
const [categories, setCategories] = React.useState([])

useEffect(() => {
  fetchCategories()
}, [])

const fetchCategories = async () => {
  await axios.get('https://www.themealdb.com/api/json/v1/1/categories.php')
    .then(response => {
      setCategories(response.data.categories)
      console.log(response.data.categories)
    })
    .catch(error => console.error(error));

}
```

This code snippet demonstrates how to fetch data from an API and manage it within a React component. It leverages two key functionalities: state management and side effects.

State Management with useState Hook:

The code utilizes the useState hook to create a state variable named categories. This variable acts as a container to hold the fetched data, which in this case is a list of meal categories. Initially, the categories state variable is set to an empty array [].

Fetching Data with useEffect Hook:

The useEffect hook is employed to execute a side effect, in this instance, fetching data from an API. The hook takes a callback function (fetchCategories in this case) and an optional dependency array. The callback function is invoked after the component renders and whenever the dependencies in the array change. Here, the dependency array is left empty [], signifying that the data fetching should occur only once after the component mounts.

Fetching Data with fetchCategories Function:

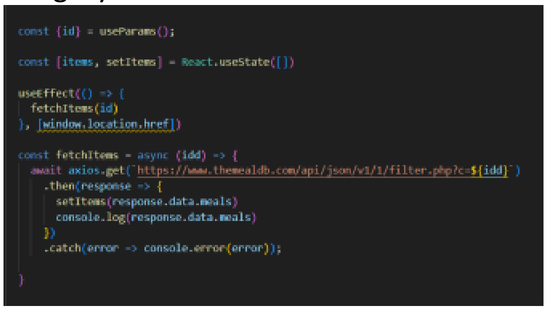
An asynchronous function named fetchCategories is defined to handle the API interaction. This function utilizes the axios.get method to make a GET request to a specified API endpoint (https://www.themealdb.com/api/json/vi/1/categories.php in this example). This particular endpoint presumably returns a JSON response containing a list of meal categories.

Processing API Response:

The .then method is chained to the axios.get call to handle a successful response from the API. Inside the .then block, the code retrieves the categories data from the response and updates the React component's state using the setCategories function. This function, associated with the useState hook, allows for modification of the categories state variable. By calling setCategories(response.data.categories), the component's state is updated with the fetched list of meal categories.

? **Fetching the food items under a particular category**

Now, with the API request, we fetch all the available food items under the certain category.



```
const {id} = useParams();

const [items, setItems] = React.useState([])

useEffect(() => {
  fetchItems(id)
}, [window.location.href])

const fetchItems = async (idd) => {
  await axios.get(`https://www.themealdb.com/api/json/v1/1/filter.php?c=${idd}`)
    .then(response => {
      setItems(response.data.meals)
      console.log(response.data.meals)
    })
    .catch(error => console.error(error));

}
```

This React code snippet manages data fetching from an API.

- It leverages the useState hook to establish a state variable named categories. This variable acts as a container to hold the fetched data, which is initially set to an empty array [].
- The useEffect hook comes into play to execute a side effect, in this instance, fetching data from an API endpoint. The hook takes a callback function (fetchCategories in this case) and an optional dependency array. The callback function is invoked after the component renders and whenever the dependencies in the array change. Here, the dependency array is left empty [], signifying that the data fetching should occur only once after the component mounts.
- The fetchCategories function is an asynchronous function responsible for handling the API interaction. This function utilizes the axios.get method to make a GET request to a predetermined API endpoint (https://www.themealdb.com/api/json/vi/1/categories.php in this example). This particular endpoint presumably returns a JSON response containing a list of meal categories.
- The code snippet employs the .then method, which is chained to the axios.get call, to handle a successful response from the API. Inside the .then block, the code retrieves the categories data from the response and updates the React component's state using the setCategories function. This function, associated with the useState hook, allows for modification of the categories state variable. By

calling setCategories(response.data.categories), the component's state is updated with the fetched list of meal categories.

- An optional error handling mechanism is incorporated using the .catch block. This block is designed to manage any errors that might arise during the API request. If an error occurs, the .catch block logs the error details to the console using the console.error method. This rudimentary error handling mechanism provides a way to identify and address potential issues during the data fetching process.

? **Fetching Recipe details**
With the recipe id, we fetch the details of a certain recipe.

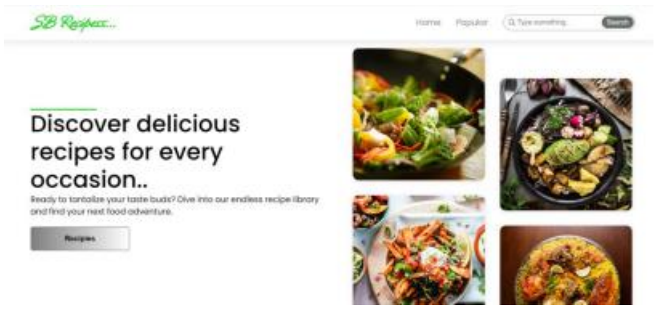
This React code manages fetching recipe data from an API and storing it within a state variable.

- It leverages the useState hook to establish a state variable named recipie (which is initially empty). This variable acts as a container to hold the fetched recipe data.
- The useEffect hook comes into play to execute a side effect, in this instance, fetching data from an API endpoint. The hook takes a callback function (fetchRecipie in this case) and an optional dependency array. The callback function is invoked after the component renders and whenever the dependencies in the array change. Here, the dependency array is left empty [], signifying that the data fetching should occur only once after the component mounts.
- The fetchRecipie function is an asynchronous function responsible for handling the API interaction. This function likely utilizes the axios.get method to make a GET request to a predetermined API endpoint, the exact URL construction of which depends on a recipeId retrieved from somewhere else in the code (not shown in the snippet).
- The code snippet employs the .then method, which is chained to the axios.get call, to handle a successful response from the API. Inside the .then block, the code retrieves the first recipe from the data.meals array in the response and updates the React component's state using the setRecipie function. This function, associated with the useState hook, allows for modification of the recipie state variable. By calling setRecipie(response.data.meals[0]), the component's state is updated with the fetched recipe data, effectively making it available for use throughout the component.
- An optional error handling mechanism is incorporated using the .catch block. This block is designed to manage any errors that might arise during the API request. If an error occurs, the .catch block logs the error details to the console using the console.error method. This rudimentary error handling mechanism provides a way to identify and address potential issues during the data fetching process.

6.Screenshots or Demo

? **Hero components**

The hero component of the application provides a brief description about our application and a button to view more recipes.

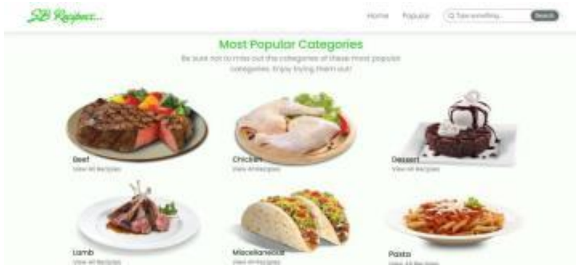


Discover delicious recipes for every occasion..

Ready to tantalize your taste buds? Dive into our endless recipe library and find your next food adventure.

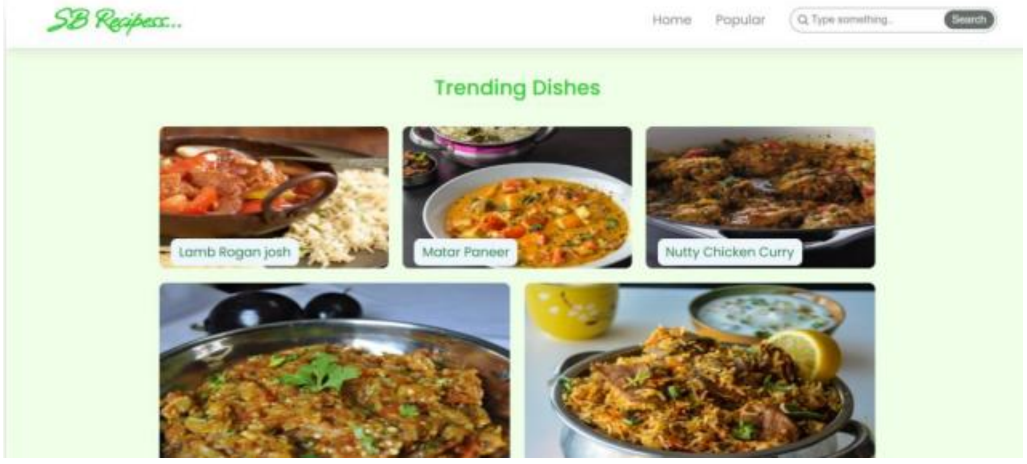
? **Popular categories**

This component contains all the popular categories of recipes..



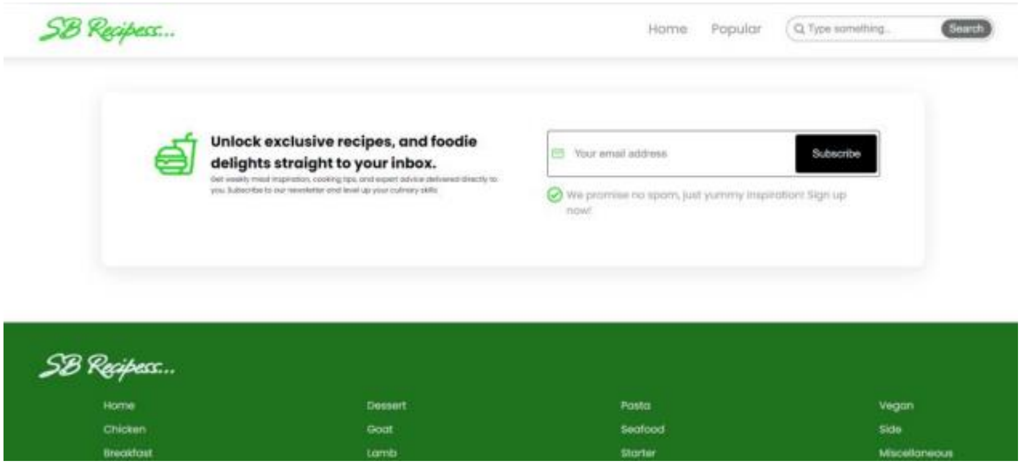
? **Trending Dishes**

This component contains some of the trending dishes in this application.



? **News Letter**

The news letter component provides an email input to subscribe for the recipe newsletters.



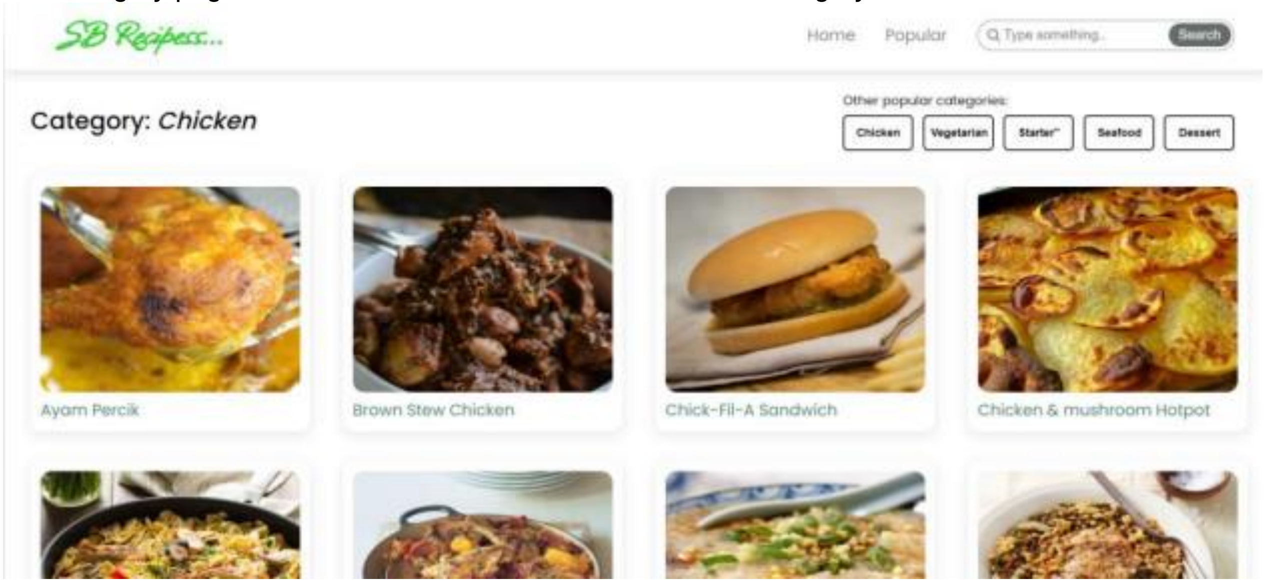
Unlock exclusive recipes, and foodie delights straight to your inbox.

Get weekly meal inspiration, cooking tips, and expert advice delivered directly to you. Subscribe to our newsletter and level up your culinary skills

We promise no spam, just yummy inspiration! Sign up now!

? Category dishes page

The category page contains the list of dishes under a certain category.



Category: *Chicken*

Ayam Percik

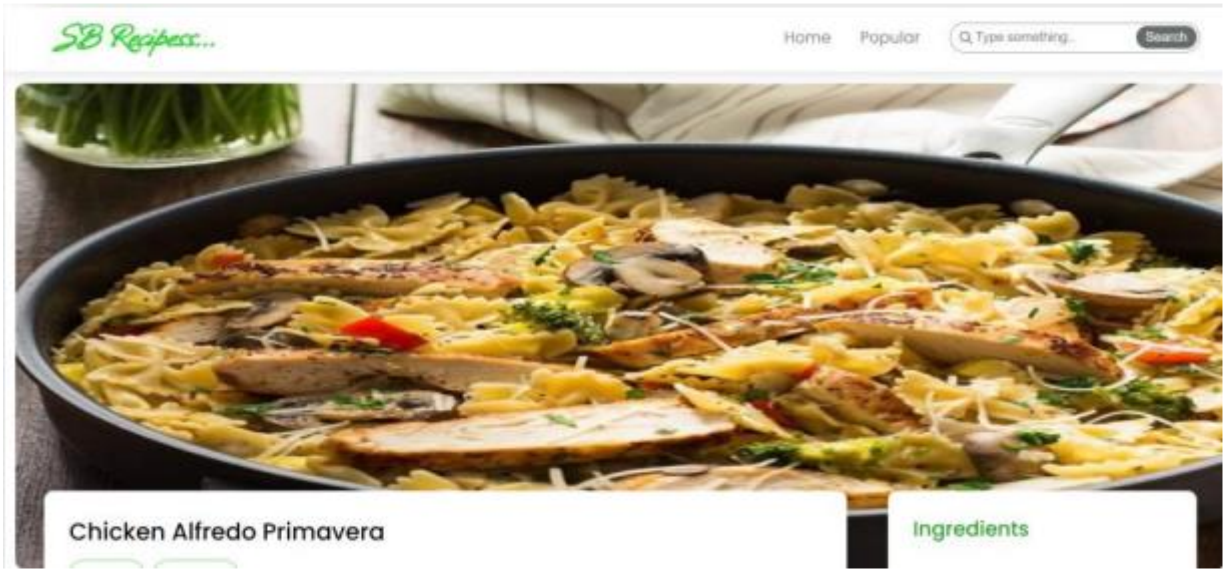
Brown Stew Chicken

Chick-Fil-A Sandwich

Chicken & mushroom Hotpot

? Recipe page

The images provided below shows the recipe page, that includes images, recipe instructions, ingredients and even a tutorial video.



Chicken Alfredo Primavera

Ingredients

7. Future Enhancements

1. **AI-Powered Recipe Suggestions**
   - Use machine learning to recommend recipes based on user preferences, past cooking history, and dietary needs.
2. **Voice Assistant Integration**
   - Enable voice-guided cooking so users can follow recipes hands-free.
3. **Smart Kitchen Device Connectivity**
   - Connect with IoT-enabled devices (smart ovens, refrigerators, etc.) for automated ingredient tracking and cooking assistance.
4. **Meal Planner & Grocery List Generator**
   - Provide weekly meal planning tools with auto-generated shopping lists.