CookBook: Your Virtual Kitchen Assistant.

Welcome to the forefront of culinary exploration with CookBook!

Our cutting-edge web application is meticulously crafted to transcend the boundaries of culinary experiences, catering to the tastes of both passionate cooking enthusiasts, and seasoned professional chefs. With an emphasis on an intuitive user interface and a robust feature set, CookBook is poised to revolutionize the entire recipe discovery, organization, and creation process.

Designed with a commitment to user-friendly aesthetics, CookBook immerses users in an unparalleled culinary adventure. Navigate seamlessly through a vast expanse of culinary inspiration with features such as dynamic search effortlessly.

From those taking their first steps in the kitchen to seasoned professionals, CookBook embraces a diverse audience, nurturing a dynamic community united by a shared passion for the art of cooking. Our vision is to reshape how users interact with recipes, presenting a platform that not only sparks inspiration but also fosters collaboration and sharing within the vibrant culinary community.

Embark on this gastronomic journey with us, where innovation seamlessly intertwines with tradition. Every click within CookBook propels you closer to a realm of delicious possibilities. Join us and experience the evolution of recipe management, where each feature is meticulously crafted to offer a glimpse into the future of culinary exploration. Elevate your culinary endeavours with CookBook, where every recipe becomes an adventure waiting to be discovered and savoured.

Project Objective

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.

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

This 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

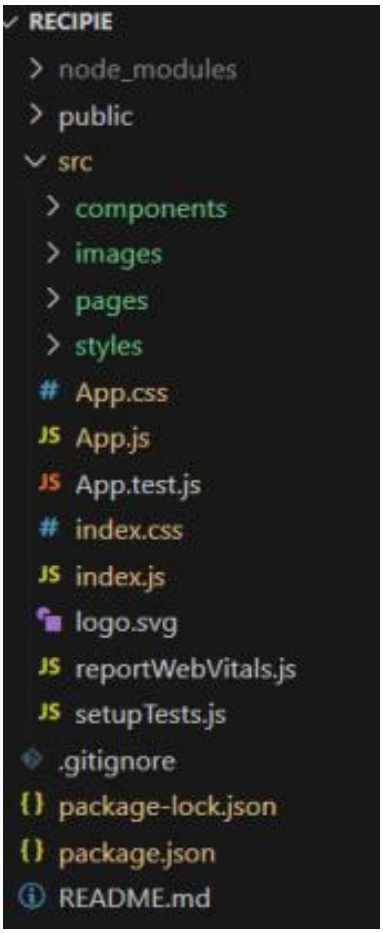
  Access the App:

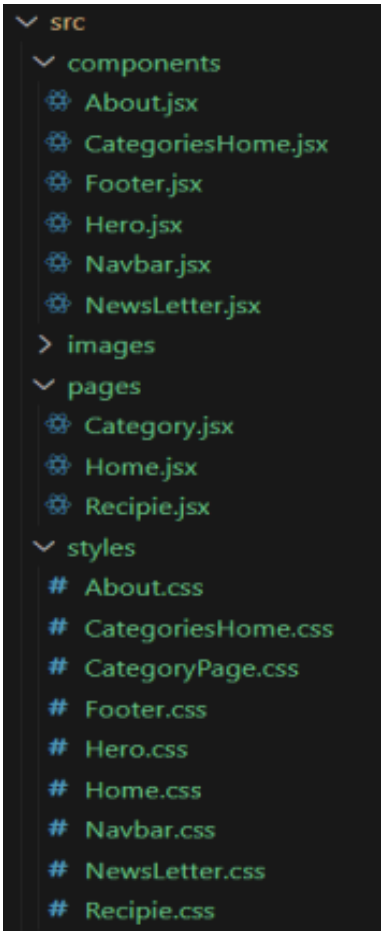
  • Open your web browser and navigate to http://localhost:3000.

• 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.

Project 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

Project setup and configuration

- **Project setup and configuration.**

• **Installation of required tools**:

To build CookBook, we'll need a developer's toolkit. We'll use React.js for the interactive interface, React Router Dom for seamless navigation, and Axios to fetch news data. For visual design, we'll choose either Bootstrap or Tailwind CSS for pre-built styles and icons.

Open the project folder to install necessary tools, In this project, we use:

o React Js

o React Router Dom

o React Icons

o Bootstrap/tailwind css

o Axios

• For further reference, use the following resources

o https://react.dev/learn/installation

o https://react-bootstrap-v4.netlify.app/getting-started/introduction/ o https://axios-http.com/docs/intro

o https://reactrouter.com/en/main/start/tutorial

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:

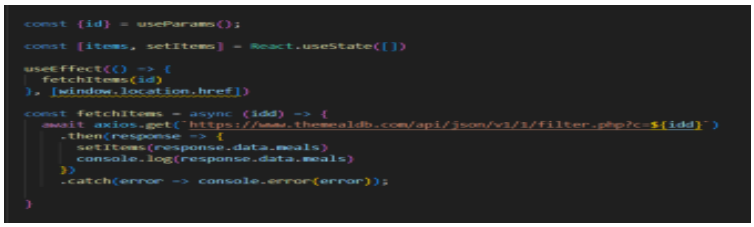
An asynchronous function named fetchCategories is defined to handle the API interaction. This function utilizes the axios.get method to make a GET request to a specified API endpoint (https://www.themealdb.com/api/json/vi/1/categories.php in this example). This particular endpoint presumably returns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



This React code snippet manages data fetching from an API.

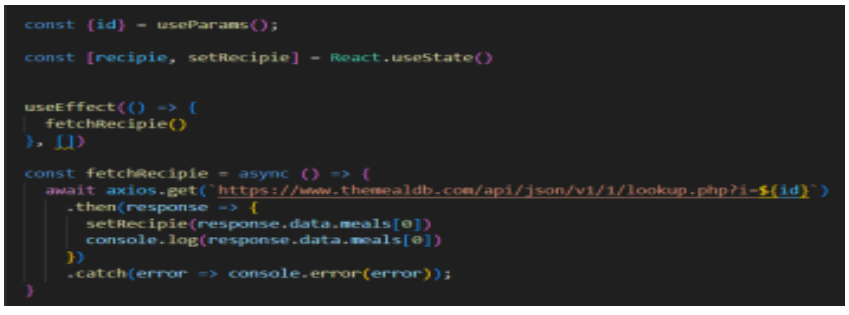
- It leverages the useState hook to establish a state variable named categories. This variable acts as a container to hold the fetched data, which is initially set to an empty array [].
- The useEffect hook comes into play to execute a side effect, in this instance, fetching data from an API endpoint. 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 dependenc 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**

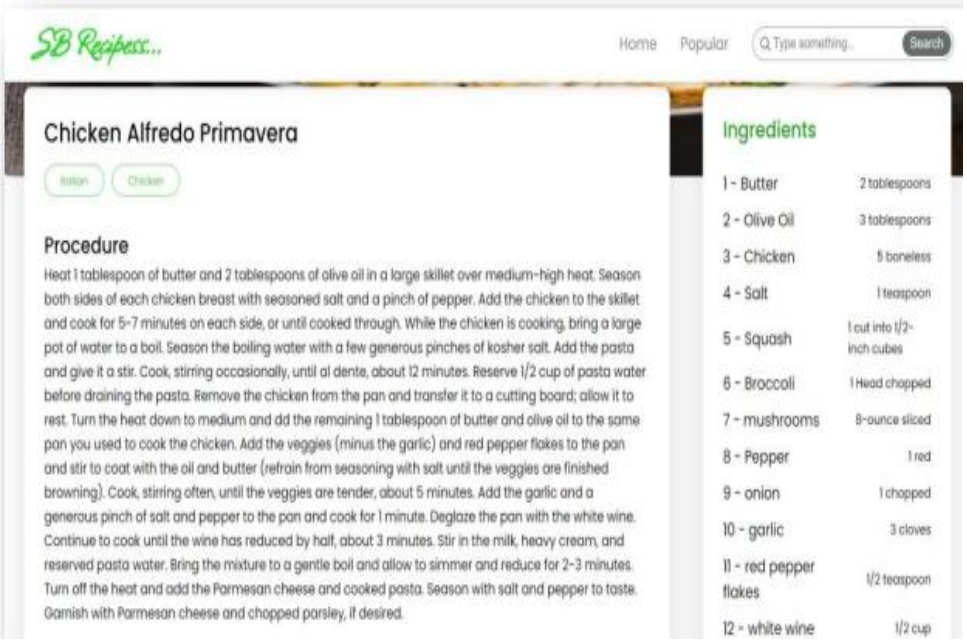
With the recipe id, we fetch the details of a certain recipe.

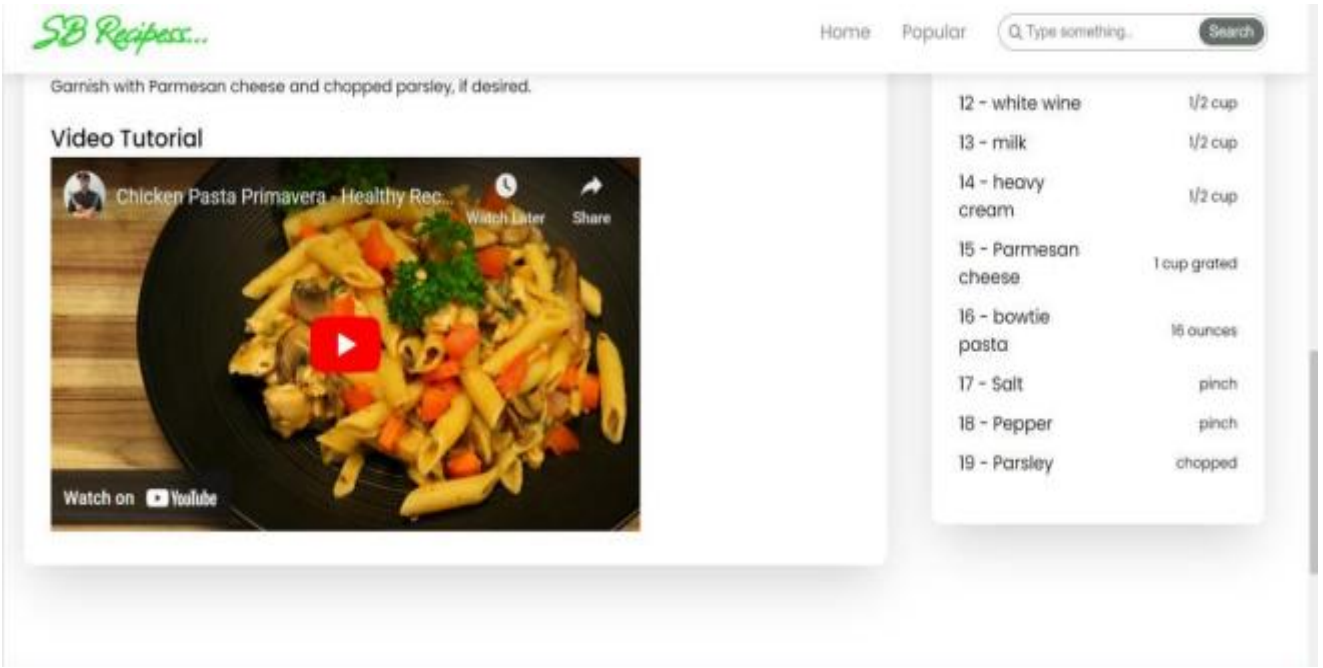


```
const {id} = useParams();

const [recipie, setRecipie] = React.useState()


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

const fetchRecipie = async () => {
  await axios.get(`https://www.themealdb.com/api/json/v1/1/lookup.php?i=${id}`)
    .then(response => {
      setRecipie(response.data.meals[0])
      console.log(response.data.meals[0])
    })
    .catch(error => console.error(error));
}
```

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.


SB Recipess...
Home
Popular
Type something...
Search
Chicken Alfredo Primavera
Italian
Chicken
Procedure
Heat 1 tablespoon of butter and 2 tablespoons of olive oil in a large skillet over medium-high heat. Season both sides of each chicken breast with seasoned salt and a pinch of pepper. Add the chicken to the skillet and cook for 5-7 minutes on each side, or until cooked through. While the chicken is cooking, bring a large pot of water to a boil. Season the boiling water with a few generous pinches of kosher salt. Add the pasta and give it a stir. Cook, stirring occasionally, until al dente, about 12 minutes. Reserve 1/2 cup of pasta water before draining the pasta. Remove the chicken from the pan and transfer it to a cutting board; allow it to rest. Turn the heat down to medium and dd the remaining 1 tablespoon of butter and olive oil to the same pan you used to cook the chicken. Add the veggies (minus the garlic) and red pepper flakes to the pan and stir to coat with the oil and butter (refrain from seasoning with salt until the veggies are finished browning). Cook, stirring often, until the veggies are tender, about 5 minutes. Add the garlic and a generous pinch of salt and pepper to the pan and cook for 1 minute. Deglaze the pan with the white wine. Continue to cook until the wine has reduced by half, about 3 minutes. Stir in the milk, heavy cream, and reserved pasta water. Bring the mixture to a gentle boil and allow to simmer and reduce for 2-3 minutes. Turn off the heat and add the Parmesan cheese and cooked pasta. Season with salt and pepper to taste. Garnish with Parmesan cheese and chopped parsley, if desired.
Ingredients
1 - Butter 2 tablespoons
2 - Olive Oil 3 tablespoons
3 - Chicken 5 boneless
4 - Salt 1 teaspoon
5 - Squash 1 cut into 1/2-inch cubes
6 - Broccoli 1 Head chopped
7 - mushrooms 8-ounce sliced
8 - Pepper 1 red
9 - onion 1 chopped
10 - garlic 3 cloves
11 - red pepper flakes 1/2 teaspoon
12 - white wine 1/2 cup


SB Recipess...
Home
Popular
Type something...
Search
Garnish with Parmesan cheese and chopped parsley, if desired.
Video Tutorial
Chicken Pasta Primavera - Healthy Rec...
Watch Later
Share
Watch on YouTube
12 - white wine 1/2 cup
13 - milk 1/2 cup
14 - heavy cream 1/2 cup
15 - Parmesan cheese 1 cup grated
16 - bowtie pasta 16 ounces
17 - Salt pinch
18 - Pepper pinch
19 - Parsley chopped