

CookBook: Your Virtual Kitchen Assistant

INTRODUCTION:

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.

- ❖ **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 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.
- ❖ **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
 - ❖ Sublime Text
 - ❖ WebStorm

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:

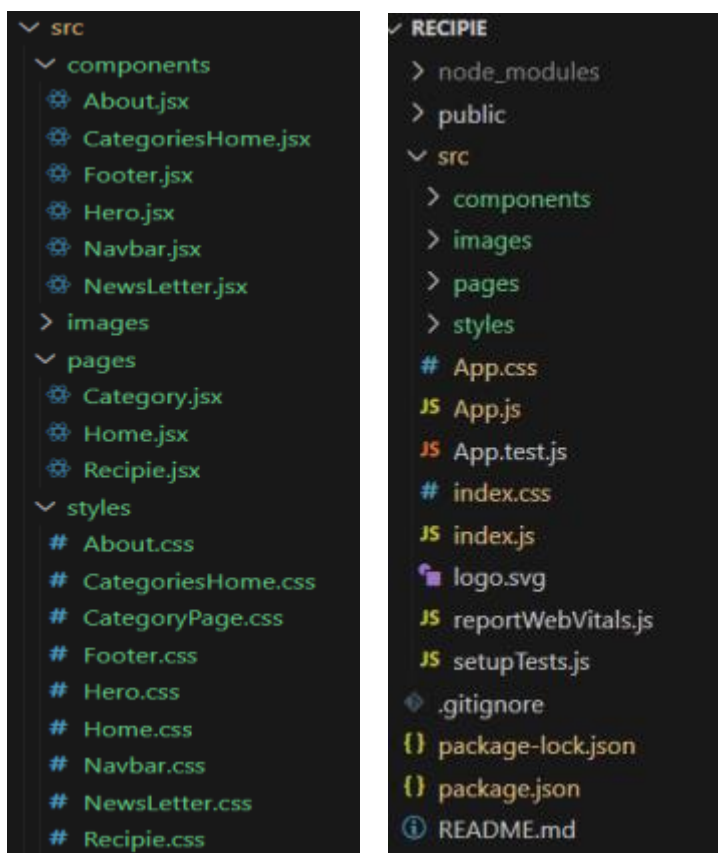
```
npm start
```

Access the App:

- ❖ Open your web browser and navigate..
- ❖ 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:

- ❖ React Js
- ❖ React Router Dom
- ❖ React Icons
- ❖ Bootstrap/tailwind css
- ❖ Axios

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="/recipe/:id" element={<Recipe />} />
</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/v1/1/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 (id) => {
  await axios.get(`https://www.themealdb.com/api/json/v1/1/filter.php?c=${id}`)
  .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/v1/1/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 [recipe, setRecipe] = React.useState()

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

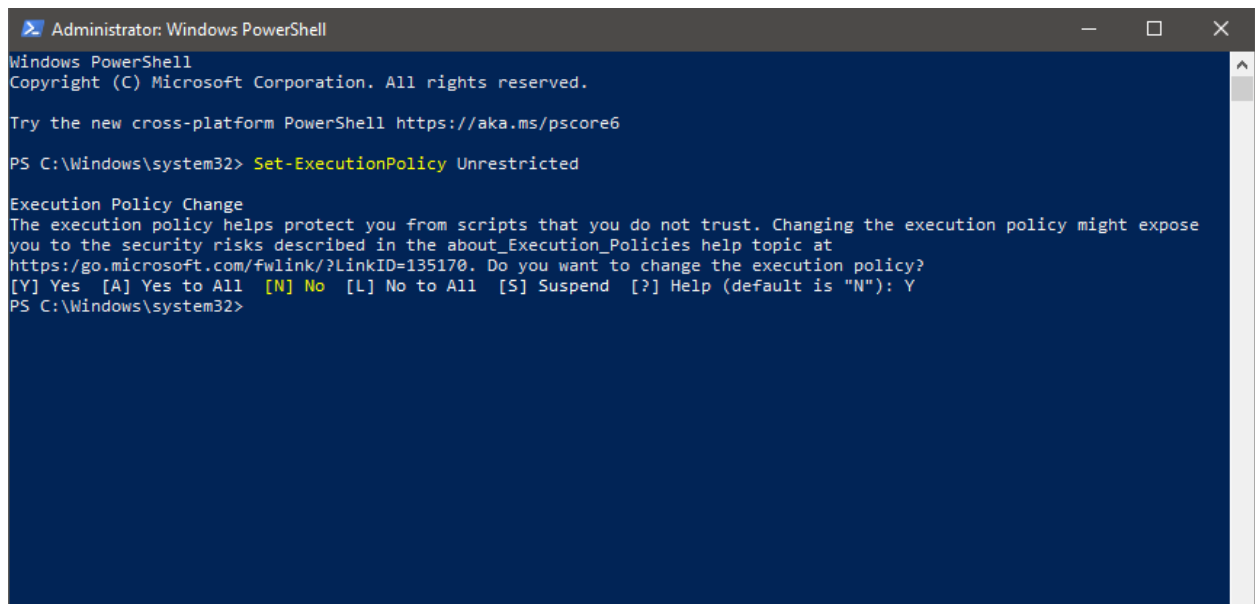
const fetchRecipe = async () => {
  await axios.get(`https://www.themeadb.com/api/json/v1/1/lookup.php?i=${id}`)
    .then(response => {
      setRecipe(response.data.meals[0])
      console.log(response.data.meals[0])
    })
    .catch(error => console.error(error));
}
```

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 recipe (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 (fetchRecipe 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 fetchRecipe 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 `setRecipe` function. This function, associated with the `useState` hook, allows for modification of the recipe state variable. By calling `setRecipe(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.

PROJECT IMPLEMENTATION & EXECUTION:

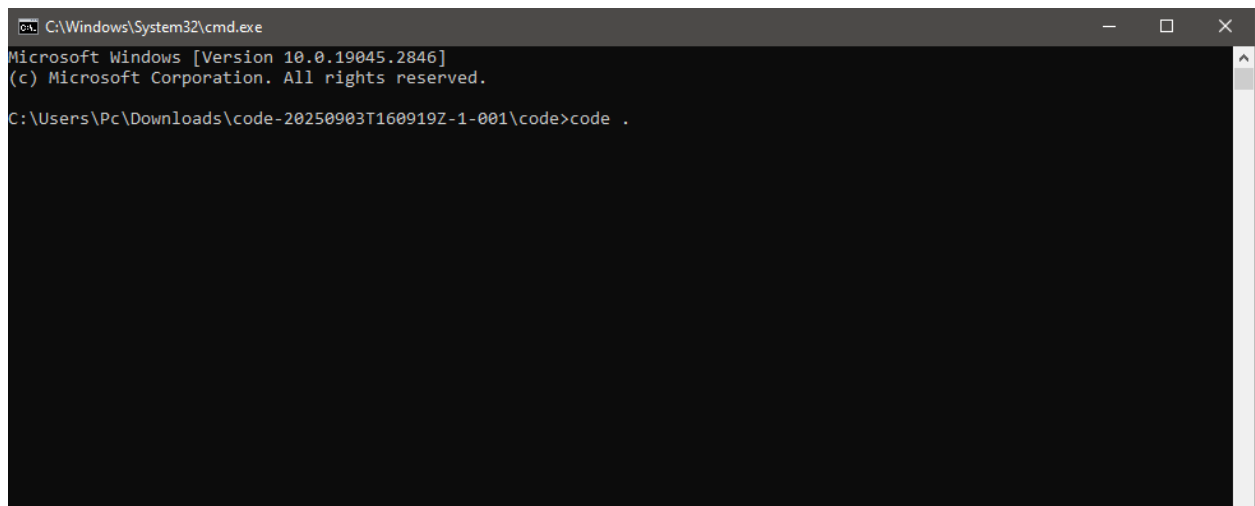
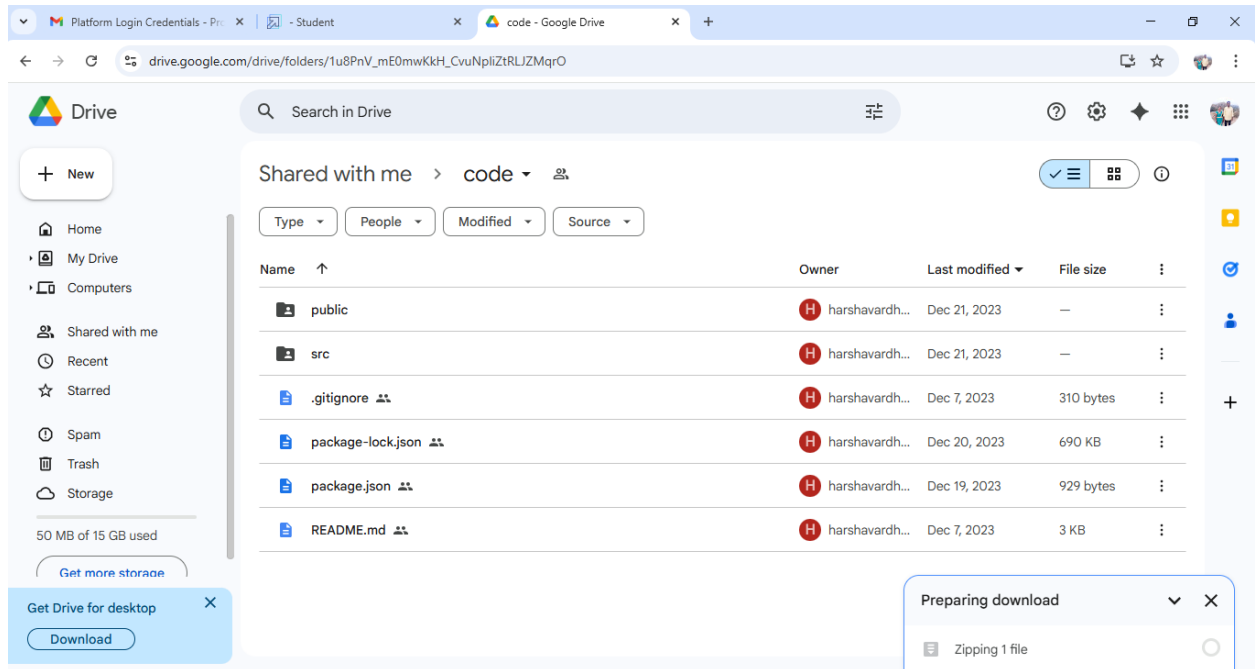


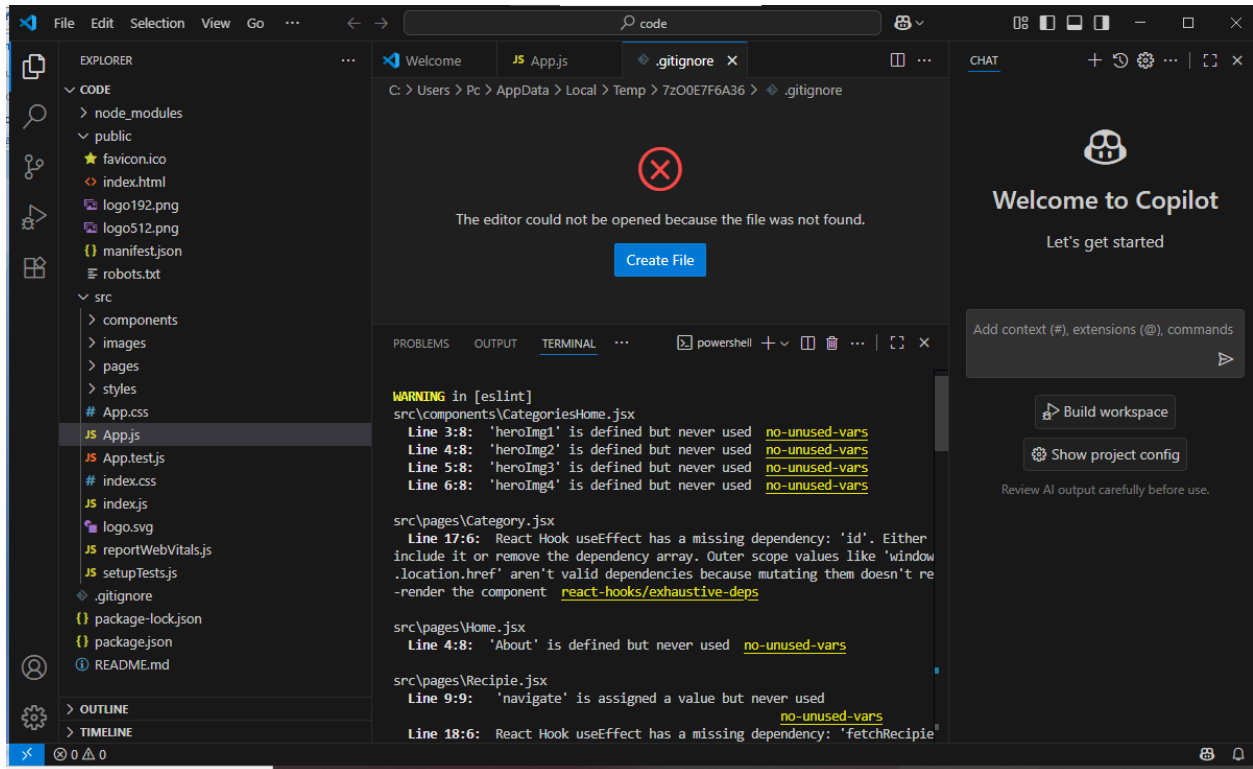
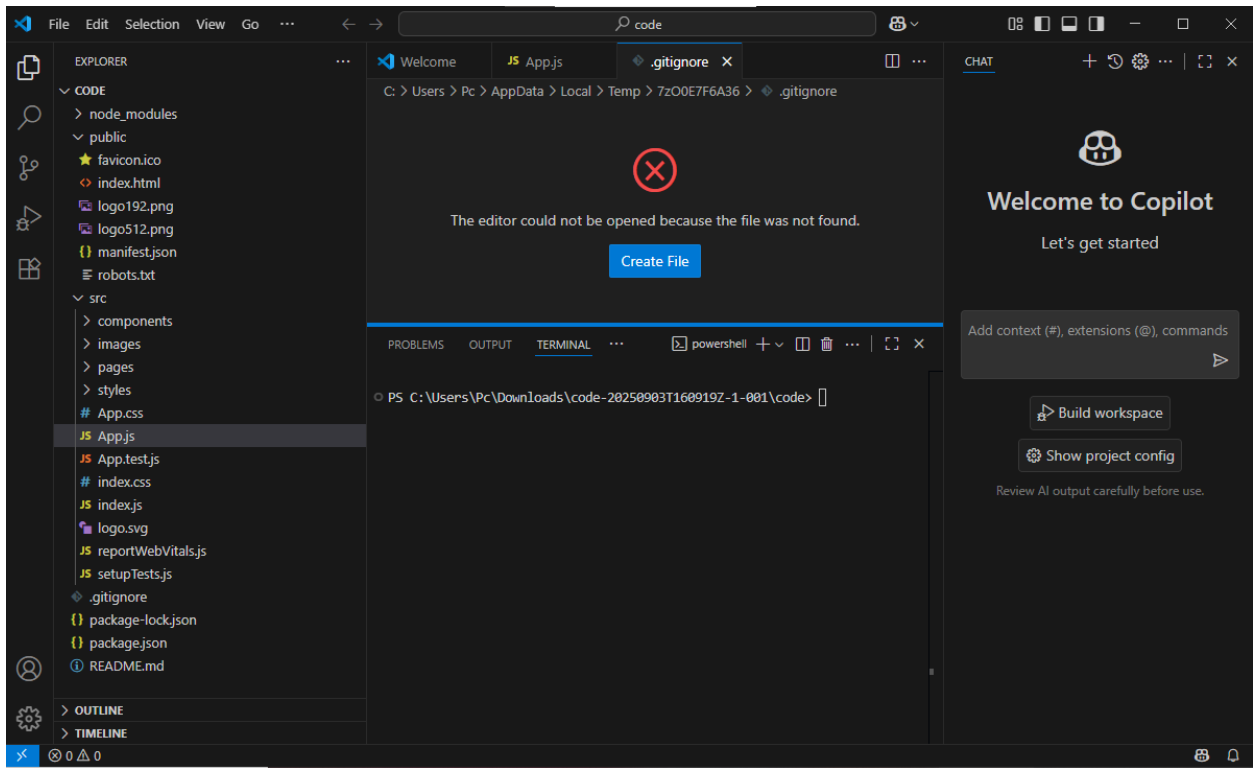
```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

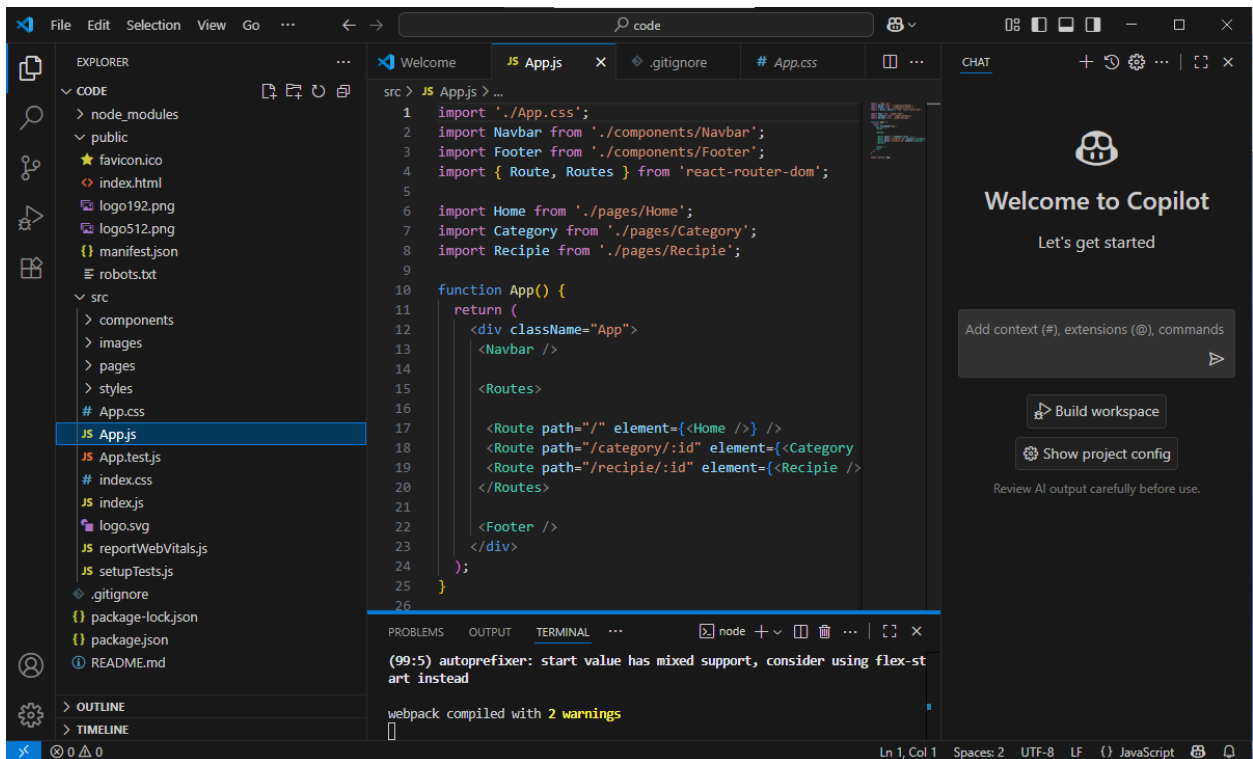
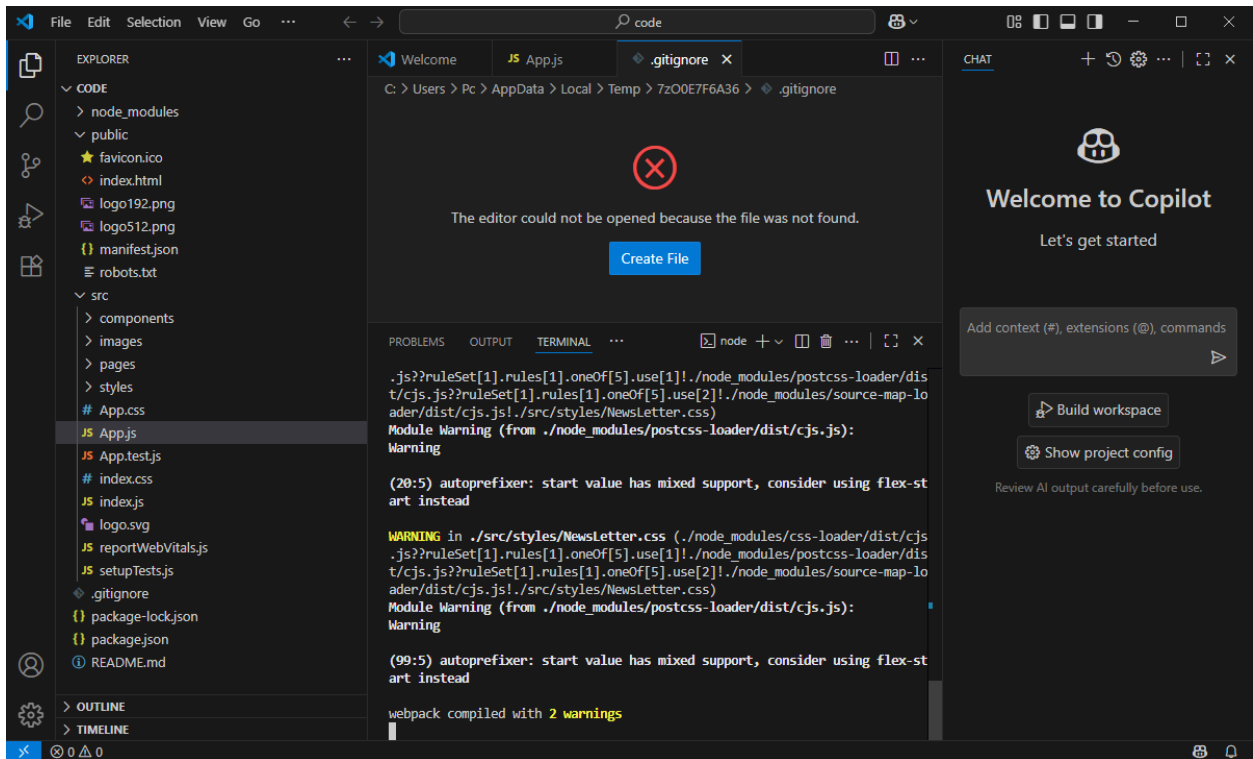
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> Set-ExecutionPolicy Unrestricted

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y
PS C:\Windows\system32>
```

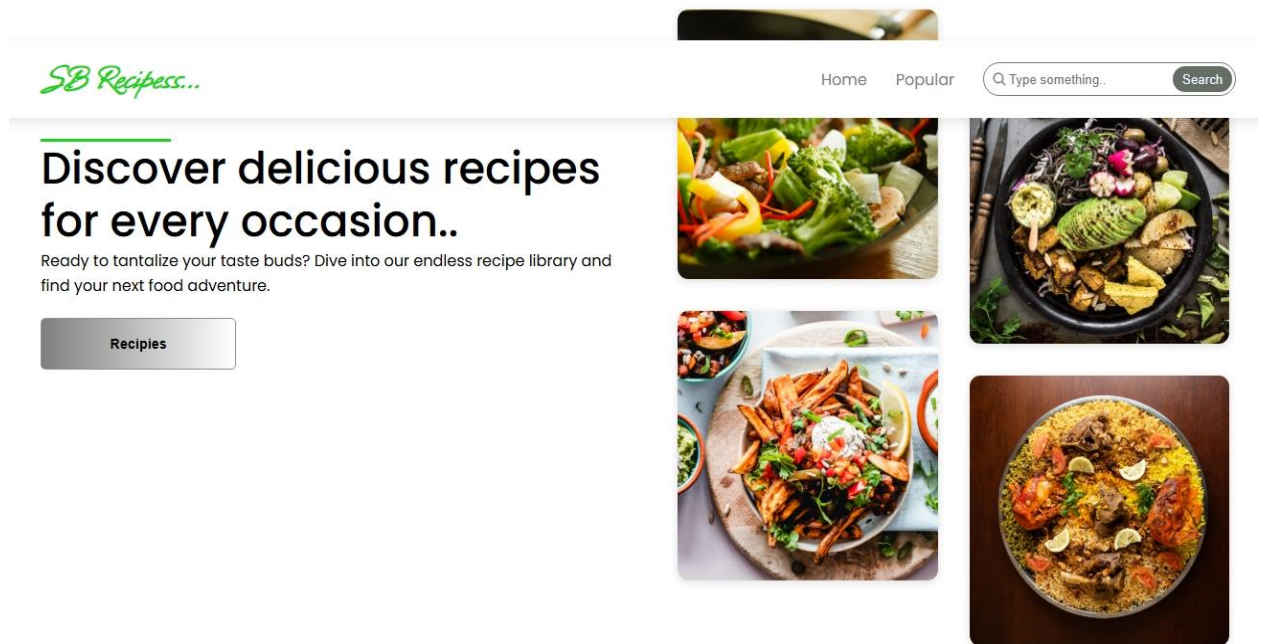






? Hero components

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



? 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.

Your email address

Subscribe

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

SB Recipes...

Home
Chicken
Seafood

Dessert
Diet
Lunch

Breakfast
Seafood
Starter

Vegetarian
Side
Miscellaneous

SB Recipes - © 2023 - All Rights Reserved

? Category dishes page

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

SB Recipes...

Home Popular

Type something...

Search

Category: Chicken

Other popular categories:

Chicken Vegetarian Starter Seafood Dessert



15-minute chicken & halloumi burgers



Ayam Perik



Brown Stew Chicken



Chicken-Fit-A Sandwich



? Recipe page

The images provided below shows the recipe page

that includes images

recipe instructions

ingredients and even a tutorial video.



Lamb Biryani

Indian Lamb

Ingredients

1 - Cashew nuts

12

Lamb Biryani

Indian Lamb

Procedure

Grind the cashew, poppy seeds and cumin seeds into a smooth paste, using as little water as possible. Set aside. Deep fry the sliced onions when it is hot. Don't overcrowd the oil. When the onions turn light brown, remove from oil and drain on paper towel. The fried onion will crisp up as it drains. Also fry the cashewnuts till golden brown. Set aside. Wash the rice and soak in water for twenty minutes. Meanwhile, take a big wide pan, add oil in medium heat, add the sliced onions, add the blended paste, to it add the green chillies, ginger garlic paste and garlic and fry for a minute. Then add the tomatoes and sauté them well till they are cooked and not mushy. Then to it add the red chilli powder, biryani powder, mint, coriander leaves and sauté them well. Add the yogurt and mix well. I always move the skillet away from the heat when adding yogurt which prevents it from curdling. Now after returning the skillet back to the stove, add the washed lamb and salt and $\frac{1}{2}$ cup water and mix well. Cook for 1 hour and cook it covered in medium low heat or put it in a pressure cooker for 6 whistles. If the water is not drained totally, heat it by keeping it open. Take another big pan, add thrice the cup of rice you use, and boil it. When it is boiling high, add the rice, salt and jeera and mix well. After 7 minutes exact or when the rice is 80% done. Switch off and drain the rice. Now, the layering starts. To the lamb, pat and level it. Add the drained hot rice on the top of it. Garnish with fried onions, ghee, mint, coriander leaves and saffron dissolved in milk. Cover the dish and bake in a 350f oven for 15 minutes or till the cooked but not mushy. Or cook in the stove medium heat for 12 minutes and lowest heat for 5 minutes. And switch off. Mix and serve hot! Notes 1. If you are cooking in oven, do make sure to cook in a big oven safe pan and cover it tight and then keep in oven for the final step. 2. You can skip biryani masala if you don't have and add just garam masala (1 tsp and red chilli powder - 3 tsp instead of 1 tsp) 3. If it is spicy in the end, squeeze some lemon, it will reduce the heat and enhance the flavors also.

Video Tutorial



Ingredients

1 - Cashew nuts	12
2 - Khus khus	$\frac{1}{2}$ tbsp
3 - Cumin seeds	$\frac{1}{2}$ tbsp
4 - Onions	3 sliced thinly
5 - Ginger garlic paste	2 tsp
6 - Garlic	4 whole
7 - Mint	Leaves
8 - Cilantro	Leaves
9 - Saffron	$\frac{1}{2}$ tsp dissolved in $\frac{1}{2}$ cup warm milk
10 - Ghee	2 tbsp
11 - Basmati rice	2 Cups
12 - Full fat yogurt	$\frac{1}{2}$ cup
13 - Cumin Seeds	1 tbsp
14 - Bay leaf	$\frac{1}{2}$
15 - Cinnamon	1 thin piece
16 - Cloves	3
17 - Cardamom	2
18 - Lamb	1 lb
19 - Red Chilli powder	1 tsp
20 - Biryani masala	1 tbsp