COOKBOOK- YOUR VIRTUAL KITCHEN ASSISTANT

A PROJECT REPORTED FOR FRONTEND DEVELOPMENT

DEPARTMENT OF COMPUTER SCIENCE

SHRI KRISHNASWAMY COLLEGE FOR WOMENS 2025-2026

**SUBMITTED BY:**

AFRIN SHIFANA M

KAVYA K

DELPHIYA P

DIVYA DHARSHINI D

**1.INTRODUCTION:**

The Cookbook Project is a web-based application designed to make cooking easier, more interactive, and more enjoyable. It provides users with a collection of recipes, categorized by cuisine, meal type, and ingredients. The project allows users to search for recipes, explore detailed cooking steps, and learn about ingredients and nutrition values in a simple and user-friendly interface.

With the help of modern frontend technologies like React.js, the Cookbook offers a responsive design, smooth navigation, and dynamic features such as a search bar, interactive dashboard, and personalized recipe recommendations.

The goal of this project is not only to store recipes but also to serve as a digital assistant for cooking enthusiasts, students, and anyone who wants to try new dishes. It combines functionality with an attractive design, ensuring that users can quickly find and cook recipes with ease.

**2.PROJECT OVERVIEW:**

This Cookbook Project is a web-based application designed to provide users with an easy and interactive way to explore,organize,and share recipies. In today’s digital age, but many platforms lack personalization and user-friendly design, Our project aims to build a modern, responsive, and dynamic cookbook application.

.

.

**2.1 CORE FEATURES:**

* **Home Page Features:**
* **Hero Section:** Attractive banner with app name/logo + tagline.
* **Navigation Bar:**Links to Home, Categories, Contact, About.
* **Search Bar:** Search recipes by name, ingredient, or cuisine.
* **Dish Categories Panel:**
* Categories like Starters, Main Course, Desserts, Drinks.
* Filter or sort dishes by cuisine (Indian, Italian, Chinese) or type (Veg/Non-Veg).
* Interactive cards with dish images + quick preview.

**3.ARCHITECTURE:**

**3.1. COMPONENT STRUCTURE:**

****

**3.2 STATE MANAGEMENT:**

State Management in the frontend of the cooking recipe project plays a key role in handling data across different components, such as popular dishes, trending dishes, and dish categories. Since multiple parts of the application need to access and update shared data (for example, when a user searches for a recipe, adds it to favorites, or switches categories), state management ensures consistency and smooth interaction between components. By using techniques like React’s built-in useState and useContext, or more advanced tools like Redux, the application can maintain a centralized state, avoid unnecessary re-renders, and provide a seamless user experience. This helps the cookbook project stay scalable and organized as more features are added.

**3.3 ROUTING:**

Routing Architecture in the cookbook frontend project defines how users navigate between different pages and sections, such as the Home page, Popular Dishes, Trending Dishes, Dish Categories, and Contact page. Using React Router, the application follows a client-side routing approach, which allows smooth transitions without reloading the entire page. Each route corresponds to a specific component, for example, /home for the Home page, /categories for the dish category panel, and /contact for the contact details. Nested routes can also be used to display individual recipe details under categories, such as /categories/desserts/chocolate-cake. This routing structure keeps the application organized, improves user experience, and ensures scalability as more features or pages are added in the future.

**4. SETUP INSTRUCTION:**

**4.1 PRE-REQUISITES:**

This PRE-REQUISITES section is telling the tools, software, and basic knowledge must have before developing this CookBook frontend project using React.js.

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.

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

**Download:** https://code.visualstudio.com/download

**• Sublime Text:** Download from

https://www.sublimetext.com/download

To clone and run the Application project from Google drive:

**Follow below steps:**

✓ Get the code:

\* Download the code from the drive link given below: https://drive.google.com/drive/folders/1u8PnV\_mE0mwKkH\_CvuNpliZtRLJZMqrO?usp=sharing

**Install Dependencies:**

\* Navigate into the cloned repository directory and install libraries:

cd recipe-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 recipe app'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.

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.

**Demo link:**

<https://drive.google.com/file/d/14SyY8TBk7so62asyd5_lB4QprKzx_hyF/view?usp=sharing>

**folder link:**

<https://drive.google.com/drive/folders/10kjjTo9tL4twiXa7cil4T1DB7QaL6fqW?usp=sharing>

**Milestone 1:** 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

ohttps://react-bootstrap-v4.netlify.app/getting-started/introduction/

o https://axios-http.com/docs/intro

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

**Milestone 2:** **Project Development**

Setup the Routing paths

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

❖ 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.



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:**

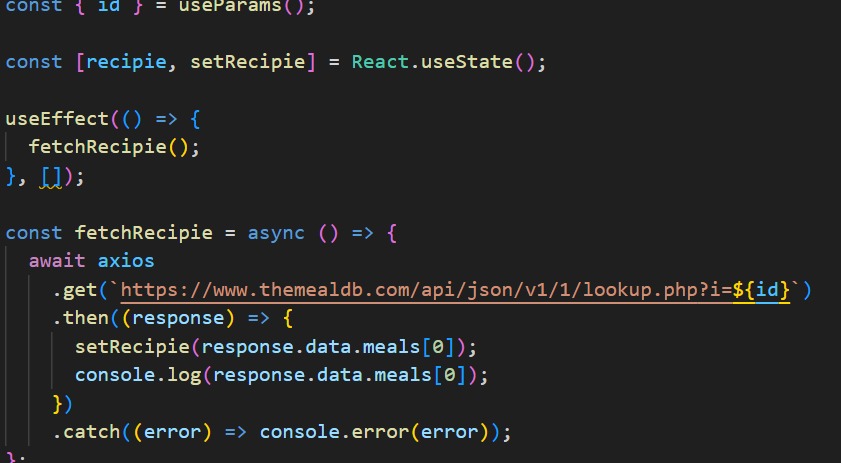
The useEffect hook is employed to execute a side effect, in this instance, fetching data from an API. The hook takes a callback function (fetchCategories in this case) and an optional dependency array. The callback function is invoked after the component renders and whenever the dependencies in the array 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(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.

● 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.

****

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.

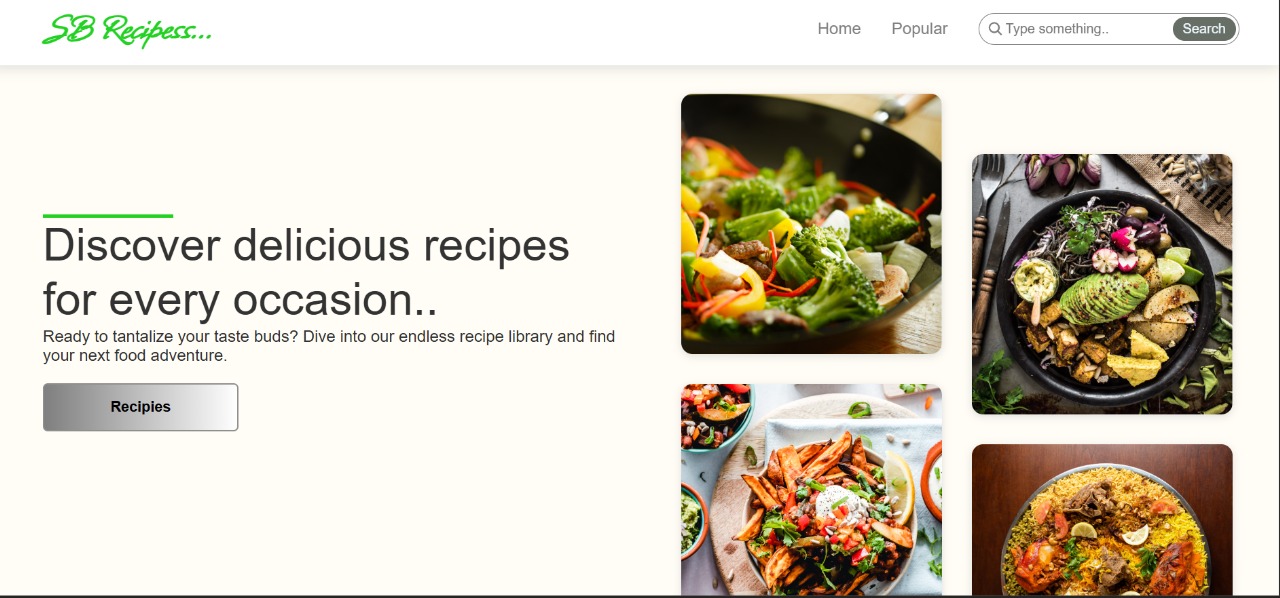
● The code snippet employs the .then method, which is chained to the axios.get call, to handle a successful response from the API. Inside the .then block, the code retrieves the categories data from the response and updates the React component's state using the setCategories function. This function, 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.

**User Interface snips**:

**➢ Hero components**

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

****

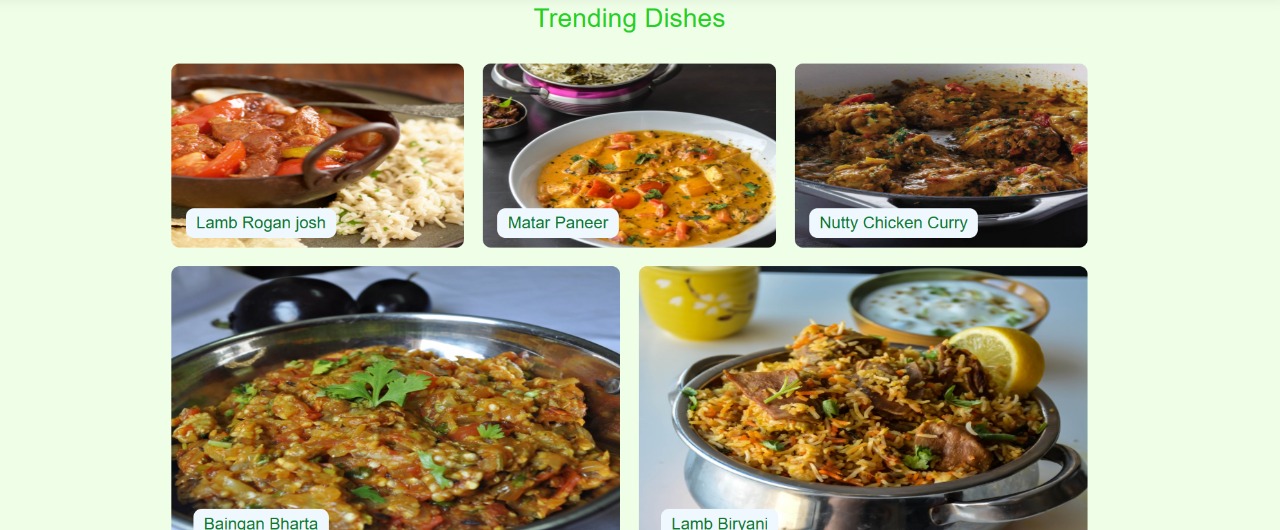
**➢ Popular categories**

This component contains all the popular categories of recipes..



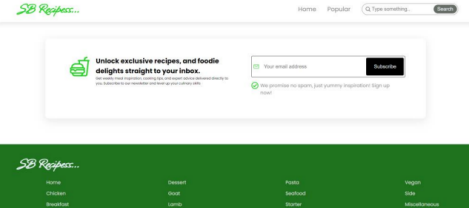
**➢ Trending Dishes**

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



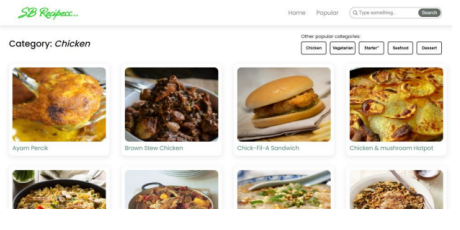
**➢ News Letter**

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

****

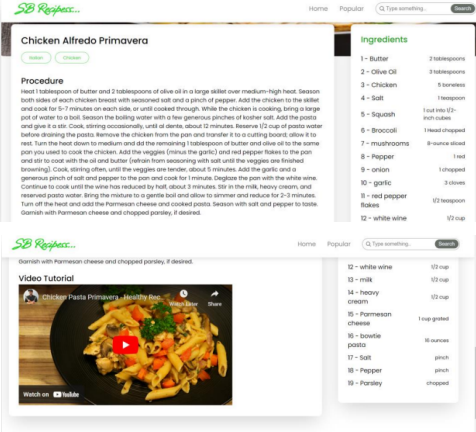
**➢ Category dishes page**

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

****

**➢ Recipe page**

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



**…..THANK YOU.....**