**Cookbook** : **Your Virtual Kitchen Assistant**

**(React Application)**

**1.Introduction:**

* **PROJECT NAME :** Cookbook : Your Virtual Kitchen Assistant (React Application)
* **TEAM MEMBERS:**
* **Team Leader :** NIVETHA K
* **Team member :** NAIEEMARILA S
* **Team member :** NIKITHA C
* **Team member :**  PRIYADHARSHINI R
* **Team member :**  PRIYADHARSHINI R

**2.Project Overview:**

Cookbook is a revolutionary web application designed to change the way you discover, organize, and create recipes. It caters to both novice and professional chefs, offering a user-friendly interface, robust features, and a vast collection of inspiring recipes.

**Purpose:**

* 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.

**Scenario based introduction:**

teenage son, Ethan, was fast approaching, and her usual creative spark was missing. "What are we even going to eat?" Ethan groaned from the doorway; his phone glued to his ear. Suddenly, a memory surfaced. Her friend, Maya, had been raving about a new

recipe platform called Cookbook. Intrigued by the promise of "elevating culinary endeavours" and "a realm of delicious possibilities," Sarah grabbed her laptop. "Hold that thought, Ethan," she declared, a flicker of hope igniting in her eyes. "We might just be about to embark on a delicious adventure."

**Project Goals and Objectives:**

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.

**Features of Cookbooks:**

✓ Recipes from the Meals DB API: Access a vast library of international recipes spanning diverse cuisines and dietary needs.

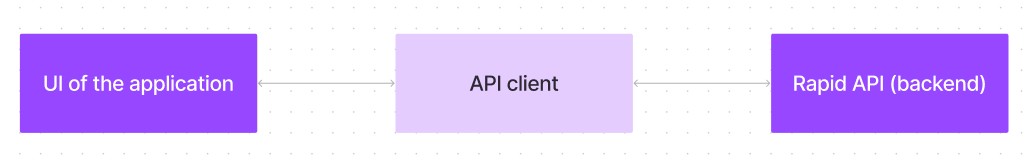
✓ Visual recipe browsing: Explore recipe categories and discover new dishes through curated image galleries.

✓ Intuitive and user-friendly design: Navigate the app effortlessly with a clean, modern interface and clear navigation.

✓ Search feature: various dishes can be accessed easily through the search feature.

**3.Architecture:**

Technical Architecture:



The user experience starts with the Cookbooks web application's UI, likely built with a framework like React or Vue.js for a smooth, single-page experience. This UI interacts with an API client specifically designed for Cookbooks. This client handles communication with the backend, but with a twist: it leverages Rapid API, a platform providing access to various external APIs. This suggests Cookbooks might integrate external data feeds or functionalities through Rapid API, enriching the user experience without building everything from scratch.

**4.Setup Instructions:**

**PRE-REQUISITES**:

Here are the key prerequisites for developing a frontend application using

React.js:

✓ Node.js and n p m:

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 n pm 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.

**5.Running the Application**

✓ **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.

• 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 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**:

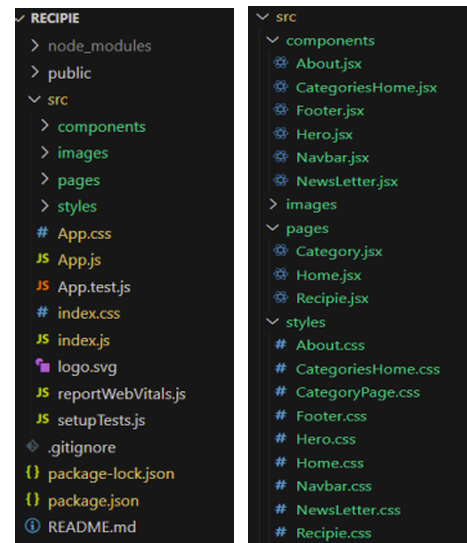
• Open your web browser and navigate to <http://localhost:3000>.

• 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.

**6. Folder Structure:**

**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 Flow:**

Project demo:

Before starting to work on this project, let’s see the demo.

Demo link:

<https://drive.google.com/file/d/1khMJkccySgKyqRaEZgCpgDACHi572Llj/view?usp=sharing>

Use the code in:

<https://drive.google.com/drive/folders/1u8PnV_mE0mwKkH_CvuNpliZtRLJZMqrO?usp=sharing>

**7. Component Documentation:**

**Key Components**

* **Recipes**: A list or collection of recipes.
* **Recipe Title**: The name of the recipe, serving as the first impression.
* **Ingredients:** A list of all items needed for the recipe.
* **Instructions:** Step-by-step directions on how to prepare the dish.
* **Preparation Time**: How long it takes to get the dish ready to cook.
* **Cook Time:** The duration needed for cooking the dish.
* **Serving Size:** How many people the recipe serves.

**Reusable Components**

* **Image:** Visual representation of the finished dish or steps.
* **Notes:** Personal notes, memories, or recommendations related to the recipe.
* **Tips:** Serving suggestions or ingredient substitutions.
* **Extras:** Information on if the recipe is gluten-free, vegetarian.

**8. State Management:**

* **Global State**
* Managed using: Redux Toolkit (or Context API if applicable).
* Stores: Recipe data, user authentication, shopping list, and

theme preferences.

**Example:**

cons initial State = {

recipes: [],

user: null,

shopping List: [],

theme: "light"

    };

**Local State**

* Managed using: Reacts use State for UI-specific states.
* Stores: Form input values, modal visibility, and temporary UI states.

**Example:**

cons [isModalOpen, set IsModalOpen] = use State(false);

cons [search Query, set Search Query] = use State("");

**9. User Interface:**

* Login/Registration Page: For user authentication.
* Home/Dashboard:
  + - Displays featured recipes, new recipes, or popular recipes.
    - Search bar for finding specific recipes.
    - Categories or tags for browsing recipes (e.g., by cuisine, ingredient, dietary restriction).
* Recipe Detail Page:
* Recipe name and image.
* Ingredients list.
* Step-by-step instructions.
* Nutritional information (optional).
* User rating and reviews section (optional).
* Save to favourites or add to meal plan
* Profile Page: User settings and preferences.

**10. Styling:**

**CSS Frameworks/Libraries**

* **Tailwind CSS** for utility-based styling.
* **Styled Components** for component-specific styles.

**Theming**

* Supports light and dark mode using context.
* Example:
* cons { theme } = use Context(Theme Context);

**11. Testing:**

* **Testing Strategy**
* **Unit Testing**: Using Jest and React Testing Library.
* **Integration Testing**: Ensuring components work together correctly.
* **End-to-End Testing**: Cypress is used for full user flow testing.
  + **Code Coverage**
* Jest configured to check test coverage.

Example command:

npm test -- --coverage

* Generates reports to ensure key functionality is well-tested.

**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:

* React JS
* React Router Dom
* React Icons
* Bootstrap/tailwind CSS
* Axios

For further reference, use the following resources

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

o <https://react-bootstrap-v4.netlify.app/getting-started/introduction/>

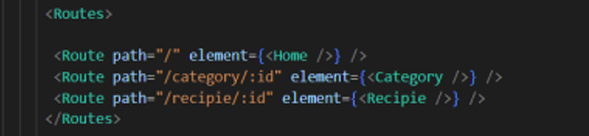
o <https://axios-http.com/docs/intro>

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

**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 themes 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 use State Hook:**

The code utilizes the use State 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 use Effect Hook:**

The use Effect hook is employed to execute a side effect, in this instance, fetching data from an API. The hook takes a callback function (fetch Categories 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 fetch Categories Function:**

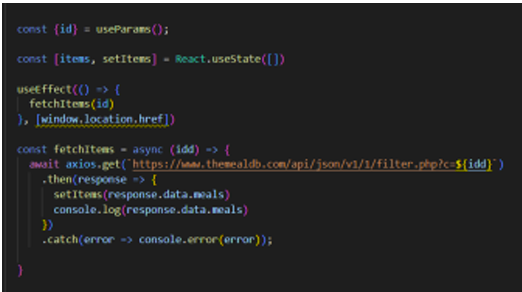
An asynchronous function named fetch Categories is defined to handle the API interaction. This function utilizes the axios. get method to make a GET request to a specified API endpoint (https://www.themealdb.com/api/json/vi/1/categories.php in this example). This particular endpoint presumably returns 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 set Categories function. This function, associated with the use State hook, allows for modification of the categories state variable. By calling set Categories(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 use State 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 use Effect hook comes into play to execute a side effect, in this instance, fetching data from an API endpoint. The hook takes a callback function (fetch Categories 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 fetch Categories 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 set Categories function. This function, associated with the use State hook, allows for modification of the categories state variable. By calling set Categories(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 use State hook to establish a state variable named recipes (which is initially empty). This variable acts as a container to hold the fetched recipe data.

● The use Effect hook comes into play to execute a side effect, in this instance, fetching data from an API endpoint. The hook takes a callback function (fetch Recipie 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 fetch Recipie 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 recipe retrieved from somewhere else in the code (not shown in the snippet).

● The code snippet employs the. then method, which is chained to the axis. 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 set Recipe function. This function, associated with the use State hook, allows for modification of the recipes state variable. By calling set Recipe(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.

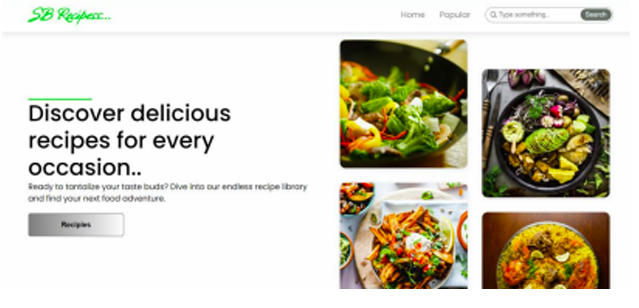
**12.Screenshorts:**

**Project Execution:**

After completing the code, run the react application by using the command “npm start” or “npm run dev” if you are using vite.js

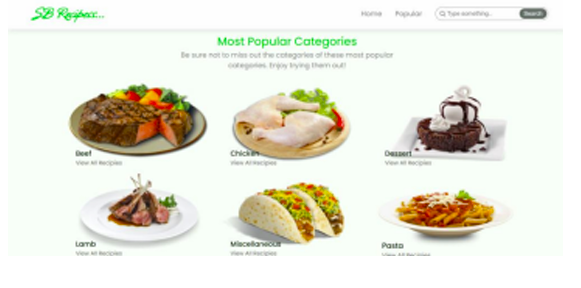
Here are some of the screenshots of the application

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



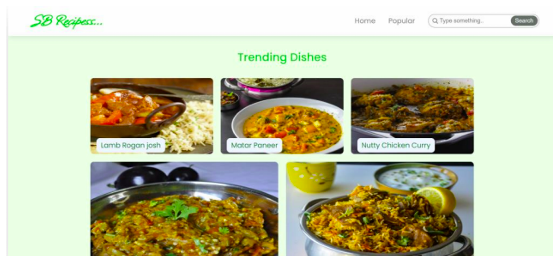
**Popular categories**

This component contains all the popular categories of recipes..



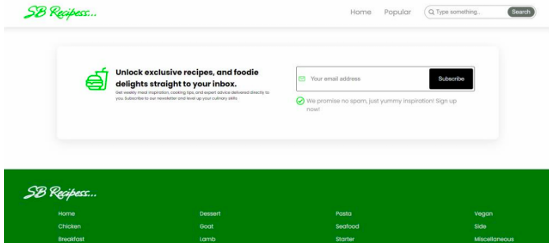
**Trending Dishes**

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



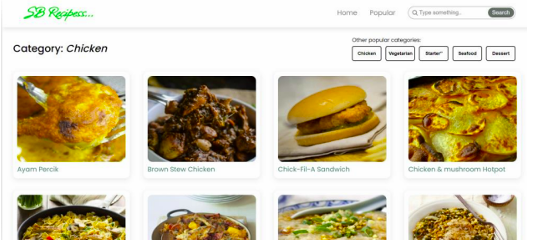
**News Letter**

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

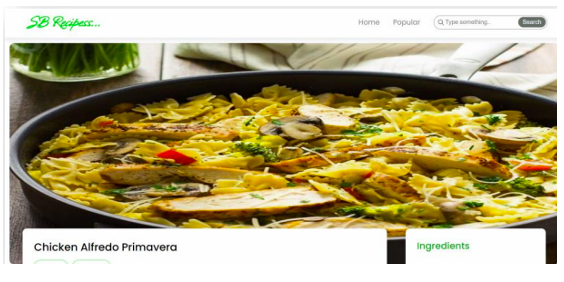


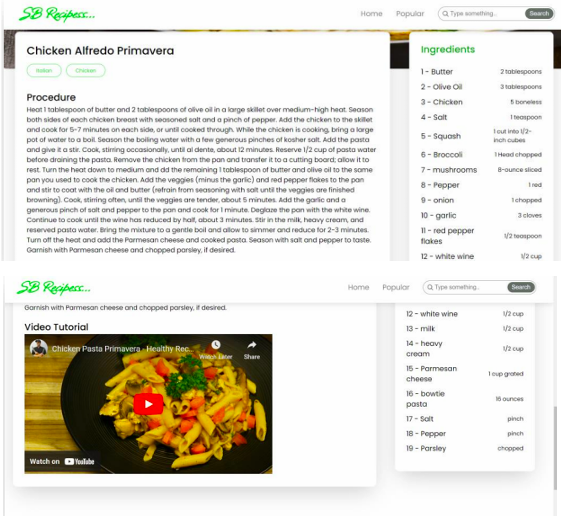
**Category dishes page**

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.





Project demo link:

<https://drive.google.com/file/d/1khMJkccySgKyqRaEZgCpgDACHi572Llj/view?usp=sharing>

**13** **. Known Issues:**

* **Inaccurate Measurements:**

Some recipes may have incorrect measurements, leading to inconsistent results.

* **Unclear Instructions:**

Certain recipes might contain vague or ambiguous instructions, making them difficult to follow.

* **Missing Information:**

Some recipes may lack essential details such as cooking times, oven temperatures, or ingredient specifics.

* **Ingredient Availability:**

A few recipes could call for ingredients that are hard to find or are not readily available in all locations.

* **Formatting Errors:**

The layout and formatting of some recipes might be inconsistent, making them hard to read or understand.

* **Nutritional Information Inaccuracies:**

The provided nutritional information for some recipes could be incorrect or outdated.

**14. Future Enhancements:**

* **New Recipe Categories**:

Add more recipe types (e.g., vegan, gluten-free, international) and expand filtering options (e.g., by cuisine, dietary restriction, ingredient).

* **Visual Improvements:**

Enhance the user experience with smoother transitions between steps and high-quality images.

* **Offline Access:**

Allow users to access saved recipes and shopping lists without an internet connection.

* **Smart Recipe Suggestions**:

Use user preferences and past cooking history to suggest recipes.

* **Improved Mobile Experience:**

Enhance navigation and layout for mobile users for easier access on the go.

**Conclusion**

This cookbook is designed to provide users with an intuitive and engaging cooking experience. With its well-structured organization, clear instructions, and accessible format, it aims to make meal preparation more effective and enjoyable. While there may be areas for future improvement, continuous enhancements are planned to refine recipes and user experience. Future updates will focus on adding new recipes, addressing user feedback, and optimizing the overall cooking journey, further establishing this cookbook as a comprehensive culinary companion.

**THANK YOU!!!**