COOKBOOK-RECEIPE APPLICATION

ABSTRACT

The Cookbook Application is a dynamic and interactive platform designed to help users explore, manage, and share recipes efficiently. It serves as a digital repository where users can browse diverse recipes, search based on categories, and retrieve detailed ingredient lists and preparation steps. This application leverages React.js for a seamless user experience, utilizing state management through use State and efficient data fetching with use Effect. Recipes are categorized for easy navigation, and users can filter meal options based on preferences. API integration ensures real-time access to recipe details, images, and ingredient information.

DESCRIPTION

The Cookbook Application is a dynamic and user-friendly digital platform designed to help users explore, save, and manage recipes effortlessly. It serves as a virtual recipe collection that caters to various cooking preferences, dietary needs, and culinary interests. With an extensive database of recipes, the application allows users to search, filter, and categorize meals based on ingredients, cuisine types, cooking time, and difficulty level. Whether users are looking for quick snacks, elaborate gourmet dishes, or dietary-specific recipes (such as vegan, keto, or gluten-free), the Cookbook Application provides a well-organized and easily accessible repository.

INTRODUCTION

Cooking is an essential part of daily life, and with the rise of digital technology, accessing recipes has become easier than ever. The Cookbook Application is designed to be a comprehensive digital recipe manager that enables users to explore, organize, and share their favourite recipes effortlessly. This application simplifies the process of finding and storing recipes by categorizing meals, fetching data from an API, and displaying step-by-step cooking instructions. Built using React.js, it utilizes state management through use State and data fetching with use Effect, ensuring a smooth and interactive user experience.

GOALS AND OBJECTIVE OF COOKBOOK

The Cookbook Application aims to provide an intuitive and feature-rich platform for users to explore, organize, and manage their favourite recipes efficiently. Below are its key goals and objectives:

Goals

1. Enhance Accessibility to Recipes – Provide users with an extensive collection of recipes that are easy to browse, filter, and access.

2. Improve User Engagement – Ensure an interactive and user-friendly interface that encourages users to explore and try new dishes.

3. Simplify Recipe Management – Allow users to save, organize, and retrieve their favourite recipes effortlessly.
The Cookbook Application aims to provide an intuitive and feature-rich platform for users to explore, organize, and manage their favorite recipes efficiently.

FEATURES OF COOKBOOK

The Cookbook Application is designed to provide a seamless and engaging cooking experience. It offers various features that make recipe discovery, meal planning, and cooking more efficient and enjoyable.

1. Recipe Exploration & Search
Users can search for recipes by name, ingredients, or cuisine.

Filtering options allow users to refine their search based on dietary preferences (e.g., vegan, keto, gluten-free).

2. Recipe Categorization
Recipes are organized into categories such as Breakfast, Lunch, Dinner, Snacks, Desserts, and Beverages.
Users can browse by meal type, cooking time, or difficulty level.

PRE-REQUISITES

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.

• 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

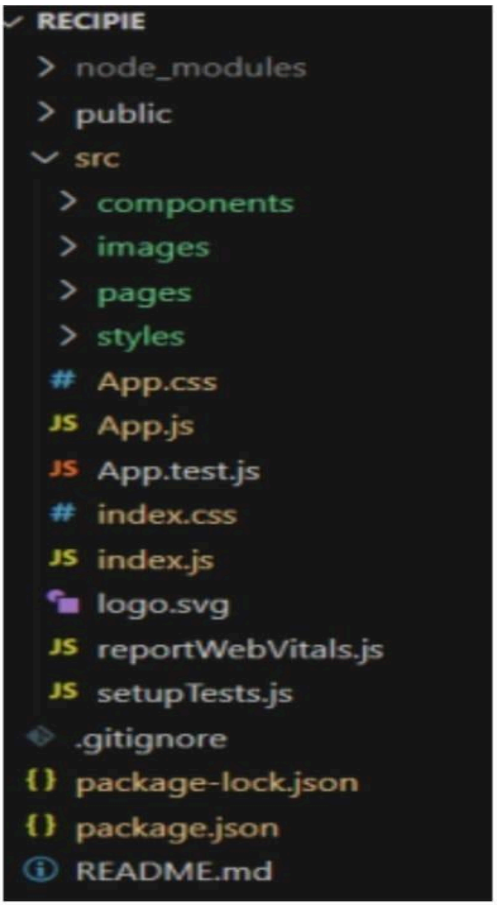
Access the App:

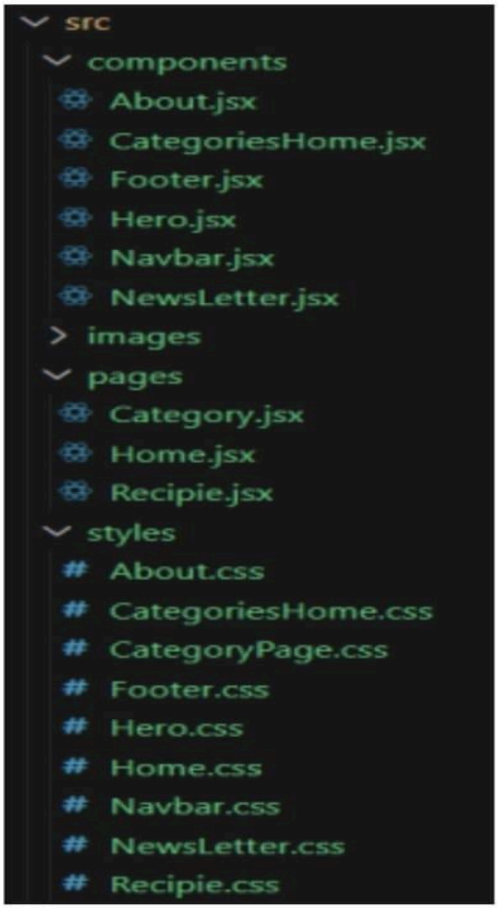
• *Open your web browser and navigate to* *http://localhost:3000*.

• *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.

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/19W8L4bdDwTYppQA_SKFqig0Ps-AjDMj1/view?usp=drivesdk

Use the source code in:
https://drive.google.com/drive/folders/1rrGzfNH4vdPYgpmUrLfACuEeTkDDYj3a

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

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.

```
<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 the meals db.

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 () => {
  try {
    const response = await axios.get(

"https://www.themealdb.com/api/json/v1/1/categorie
s.php"
    );
    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 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:

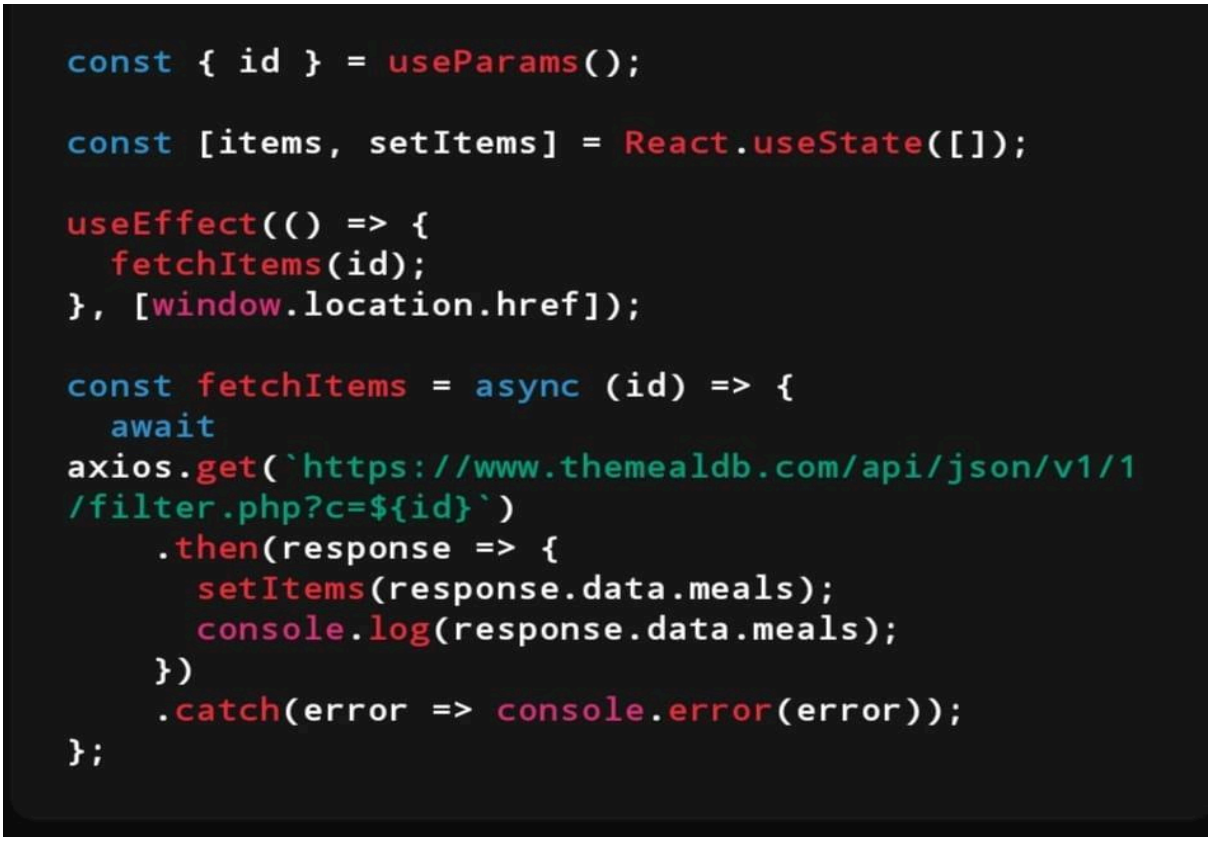
An asynchronous function named fetch Categories is defined to handle the API interaction. This function utilizes the axios.get method to make a GET request to a specified API endpoint (https://www.themealdb.com/api/json/vi/1/categories.php in this example). This particular endpoint presumably returns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

```
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 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 recipe (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 Recipe 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

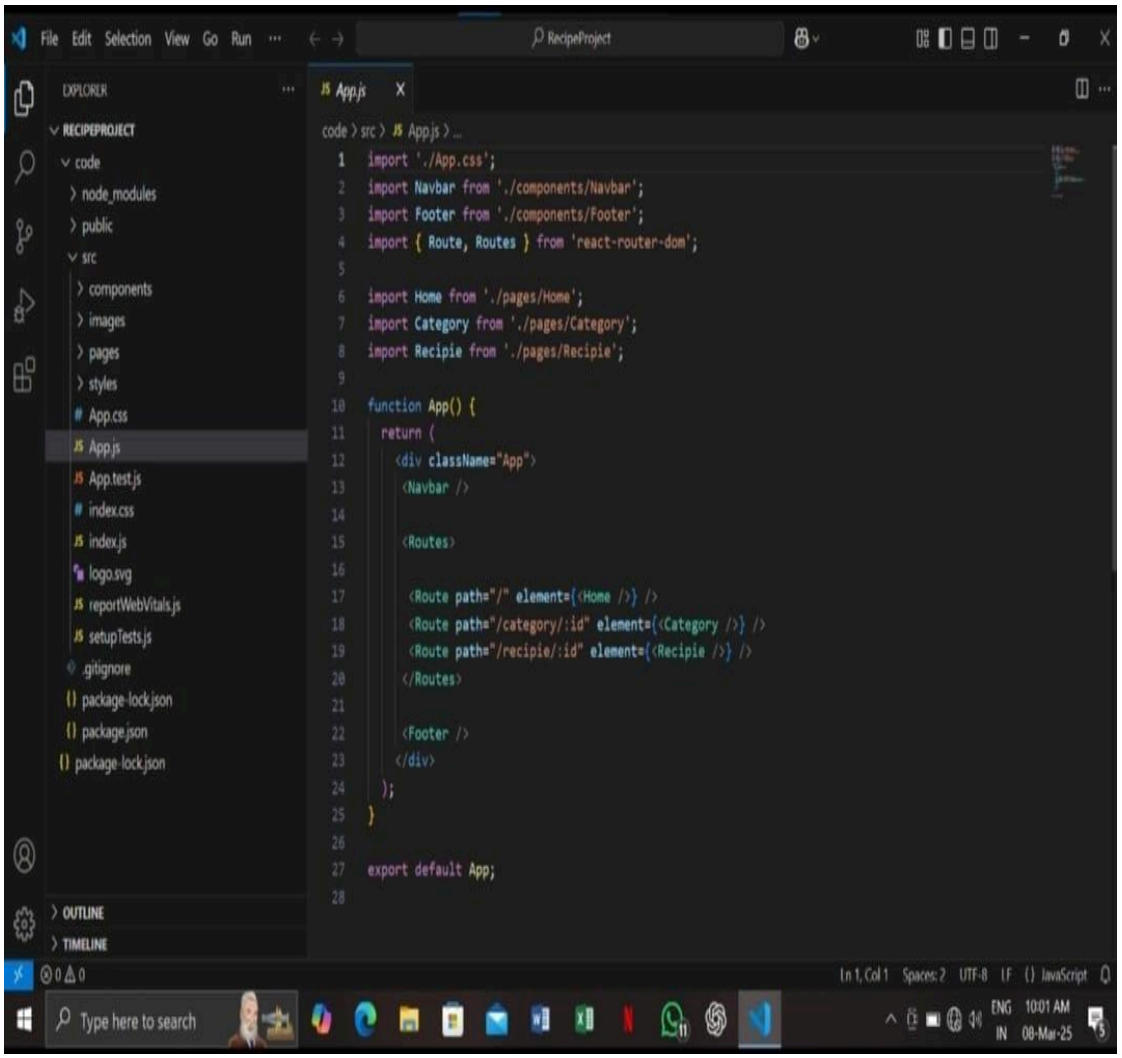
.catch block logs the error details to the console using the console.error method. This

rudimentary error handling mechanism provides a way to identify and address potential issues

- **IMPLEMENT DATA BINDING**

  REFERENCE VIDEO LINK:
  https://drive.google.com/file/d/19W8L4bdDwTYppQA_SKFqig0Ps-AjDMj1/view?usp=drivesdk

  REFERENCE IMAGE:

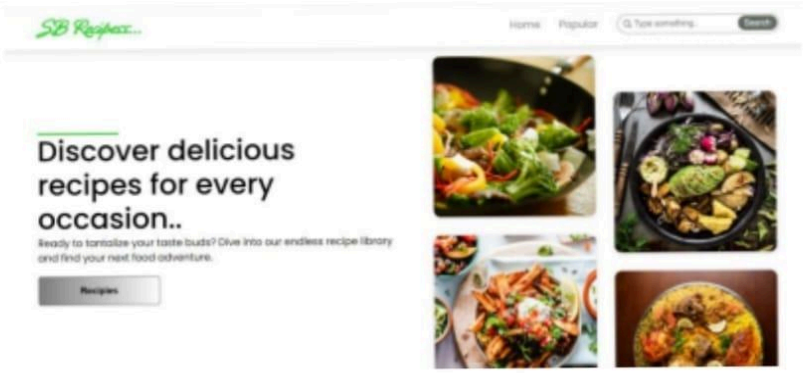


User Interface snips:

➢ **Hero components**

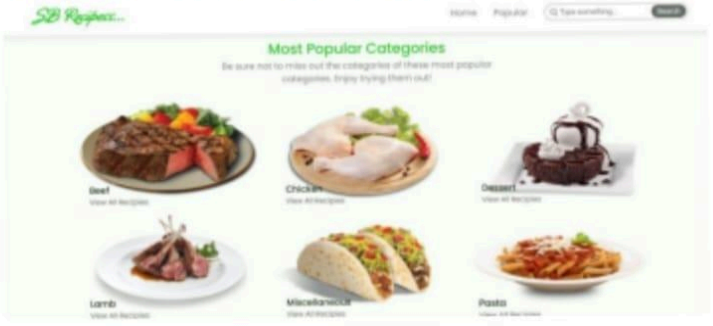
The hero component of the application provides a brief description about our application and a button to view more recipes

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..

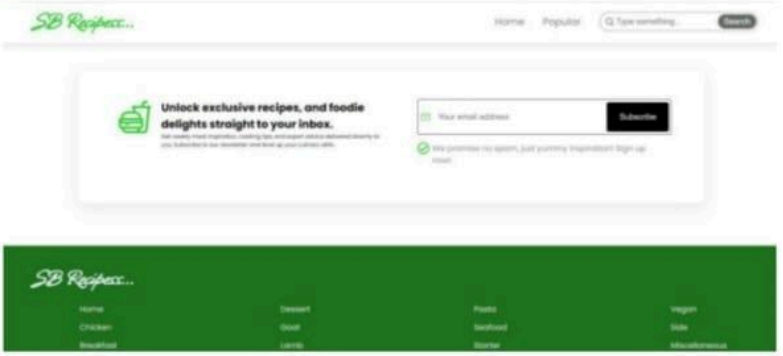


➢ **Trending Dishes**

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

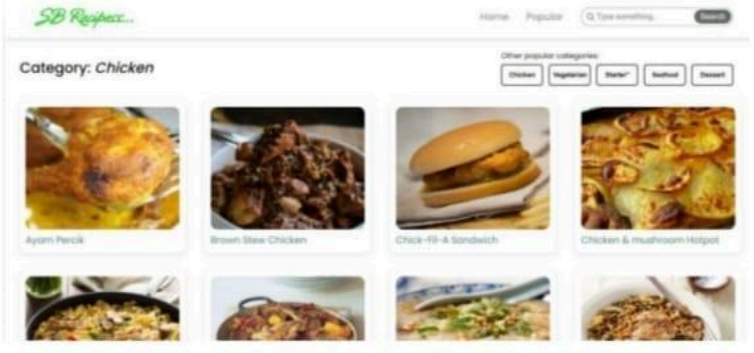
➢ **News Letter**

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



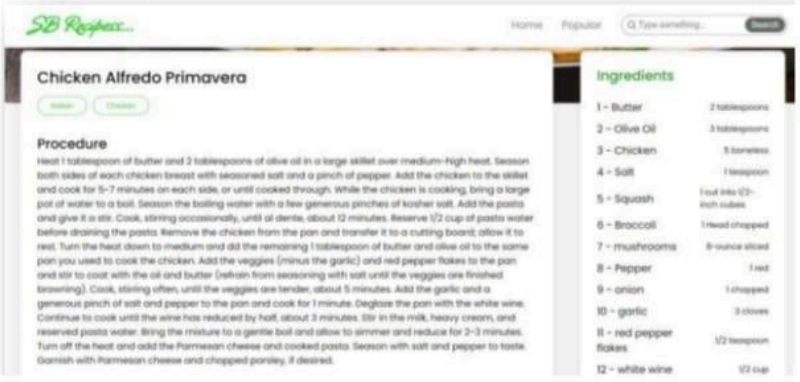
➢ **Category dishes page**

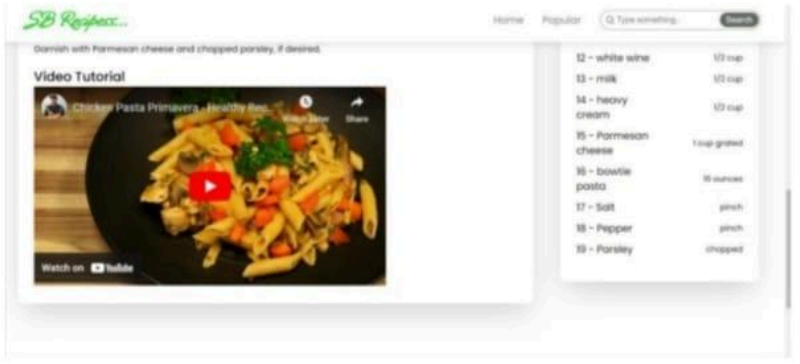
The category page contains the list of dishes under a certain category.



➢ **Recipe page**

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





PROJECT DEMO:

https://drive.google.com/file/d/153wIy7qcn6d3BEloJthL89J-sfKBuZL2/view?usp=sharing

SOURCE CODE LINK:

https://drive.google.com/drive/folders/1rrGzfNH4vdPYgpmUrLfACuEeTkDDYj3a?usp=sharing