CookBook: Your Virtual Kitchen Assistant

TEAM ID: SWTID1741155911

TEAM LEADER : DIVYA M
EMAIL ID: divyamppk@gmail.com
ROLE: Project Manager

TEAM MEMBER : LAKSHAYA M
EMAIL ID: lakshayamr2910@gmail.com
ROLE: Documentation

TEAM MEMBER : ASWINI M
EMAIL ID: kannankalpana346@gmail.com
ROLE: Demo Link Creator

TEAM MEMBER : SUBALAKSHMI K
EMAIL ID: queenblack87655@gmail.com
ROLE: API Integration

CLASS: THIRD- B.Sc.COMPUTER SCIENCE

CookBook :Your Virtual Kitchen Assistant

(React Application)

Introduction:

Cook Book is a revolutionary web application designed to change the way you discover, organize, and createrecipes.Itcaterstobothnoviceandprofessionalchefs,offeringauser-friendlyinterface,robust features, and a vast collection of inspiring recipes.

Description:

Welcome to the fore front of culinary exploration with Cook Book!

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, CookBookispoisedtorevolutionizetheentirerecipediscovery,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 onlysparksinspirationbutalsofosterscollaborationandsharingwithinthe vibrant culinary community.

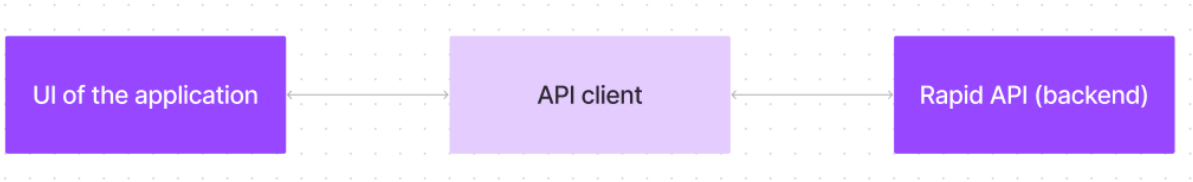
Embark on this gastronomic journey with us, where innovation seamlessly intertwines with tradition. Every click within CookBook propels you closer to are a lm of deliciouss possibilities. Join us and experience the evolution of recipe management, where each feature is meticulously crafted to offer a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

Scenario based introduction:

Sarah rummaged through the fridge, the fluorescent light casting an unappetizing glow on the wilting lettuce and forgotten container of yogurt. Dinnertime with her 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.Intriguedbythepromiseof"elevatingculinaryendeavors"and"arealmof

deliciouspossibilities,"Sarahgrabbedherlaptop."Holdthatthought,Ethan,"shedeclared,a flicker of hope igniting in her eyes."We might just be about to embark on a delicious adventure."

Technical Architecture:



The user experience starts with the CookBooks web application's UI, likely built with a frame work 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 feed so functionalities through Rapid API, enriching the user experience without building everything from scratch.

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 inter face 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** Lever age modern web development technologies ,includingReact.js, to ensure an efficient, and enjoyable user experience.

Feature so f Cook Books:

✓ **Recipes from the Meals DB API** :Access a vast library of inter national 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.

PRE-REQUISITES:

Herearethekeyprerequisitesfordevelopingafrontendapplicationusing 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 npmony our development machine ,as they are required to run JavaScript on the server-side.

- Download:https://nodejs.org/en/download/
- Installationinstructions: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 Java Script 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:3000in your web browser.

✓ **HTML,CSS,andJavaScript**:BasicknowledgeofHTMLforcreatingthestructureof 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 Web Storm.

- Visual Studio Code: Download from https://code.visualstudio.com/download•SublimeText:Downloadfrom https://www.sublimetext.com/download
- Web Storm :Download fromhttps://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:

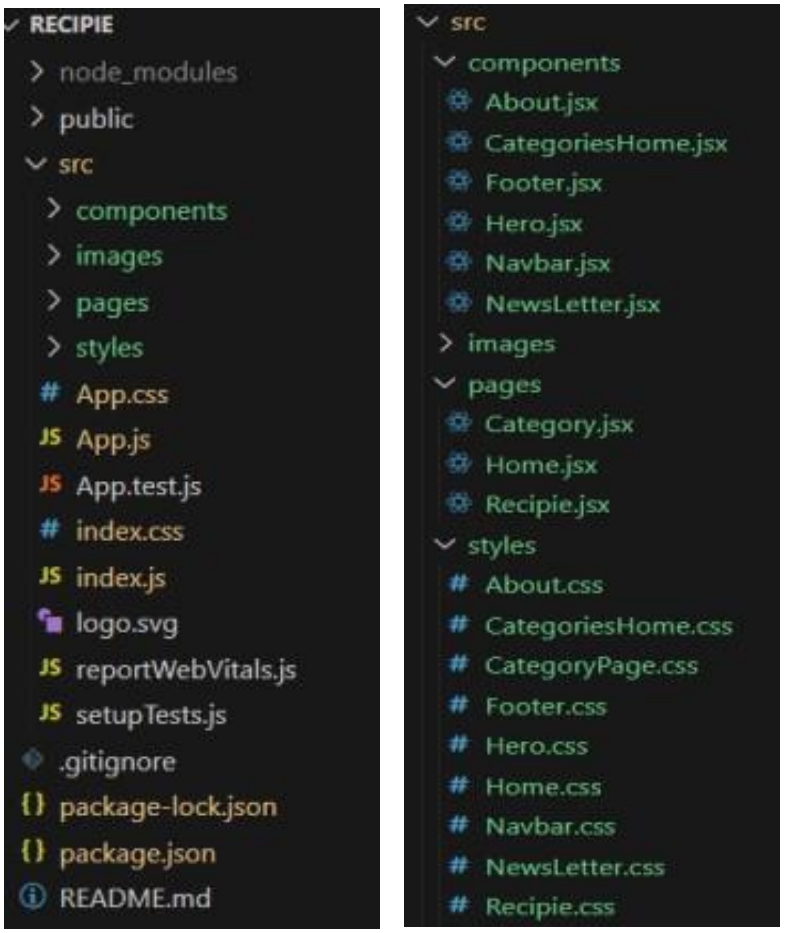
```
npm start
```

Access the App:

- Openyourwebbrowserandnavigatetohttp://localhost:3000.
- Youshouldseetherecipeapp'shomepage,indicatingthattheinstallation and setup were successful.

Youhavesuccessfullyinstalledandsetuptheapplicationonyourlocalmachine.Youcan now proceed with further customization, development, and testing as needed.

Project structure:



Inthisproject,we'vesplitthefilesinto3majorfolders,*Components,PagesandStyles.*Inthe

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

Milestone1:Projectsetupandconfiguration.

- **Installation of required tools**:

  To buildCookBook,we'llneedadeveloper'stoolkit.We'lluseReact.jsfortheinteractive interface, React Router Dom for seamless navigation, and Axios to fetch news data.Forvisual 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
  - ReactRouterDom
  - ReactIcons
  - Bootstrap/tailwindcss
  - Axios

- For further reference ,use the following resources
  - https://react.dev/learn/installation
  - https://react-bootstrap-v4.netlify.app/getting-started/introduction/
  - https://axios-http.com/docs/intro
  - https://reactrouter.com/en/main/start/tutorial

Milestone2:ProjectDevelopment

❖ 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 Nav bar and Hero components

❖ Codethepopularcategoriescomponentsandfetchthecategoriesfrom***themealsdb*** Api.

❖ Also ,add the trending dishes in the homepage.

❖ Now,developthecategorypagetodisplayvariousdishesunderthecategory.

❖ Finally,codetherecipepage,wheretheingredients,instructionsandademovideo 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 manageit 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 caseisalistofmealcategories.Initially,thecategoriesstatevariableisset 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.

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

const [items, setItems] = React.useState([])

useEffect(() => {
  fetchItems(id)
}, [window.location.href])

const fetchItems = async (idd) => {
  await axios.get(`https://www.themealdb.com/api/json/v1/1/filter.php?c=${idd}`)
    .then(response => {
      setItems(response.data.meals)
      console.log(response.data.meals)
    })
    .catch(error => console.error(error));

}
```

This React code snippet manages data fetching fro man API.

- ItleveragestheuseStatehooktoestablishastatevariablenamedcategories.Thisvariableacts as a container to hold the fetched data, which is initially set to an empty array [].
- TheuseEffecthookcomesintoplaytoexecuteasideeffect,inthisinstance,fetchingdatafrom 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 predeterminedAPIendpoint(https://www.themealdb.com/api/json/vi/1/categories.phpinthis 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 successfulresponsefromtheAPI.Insidethe.thenblock,thecoderetrievesthecategoriesdata 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 designedtomanageanyerrorsthatmightariseduringtheAPIrequest.Ifanerroroccurs,the .catch block logs the error details to the console using the console.error method. This rudimentary error handling mechanism provides away to identify and add respotential issues during the data fetching process.

➢ **Fetching Recipe details**

With the recipe id, we fetch the detail so 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));
}
```

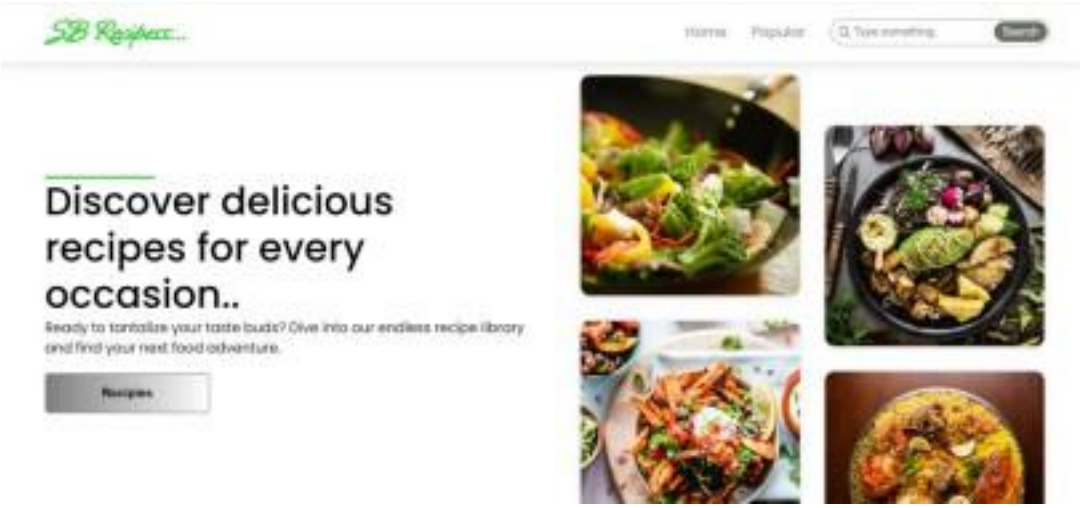
ThisReactcodemanagesfetchingrecipedatafromanAPIandstoringitwithinastatevariable.

- It leverages the use State 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 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 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 successfulresponsefromtheAPI.Insidethe.thenblock,thecoderetrievesthefirstrecipefrom the data. meals array in the response and updates the React component's state using the setRecipiefunction.Thisfunction,associatedwiththeuseStatehook,allowsformodificationof the recipie state variable. By calling set Recipie (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.
-

User Interface snips:

➢ **Hero components**

Theherocomponentoftheapplicationprovidesabriefdescriptionaboutourapplicationand a button to view more recipes.

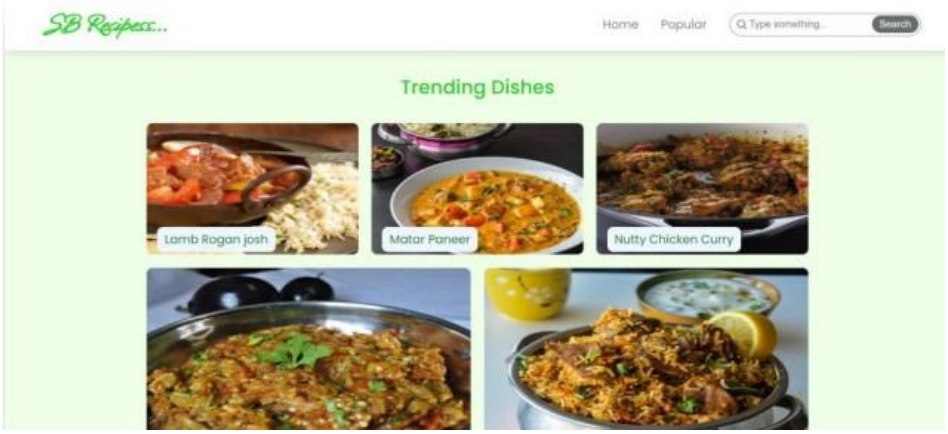


➢ **Popular categories**

This component contains all the popular categories of recipes..

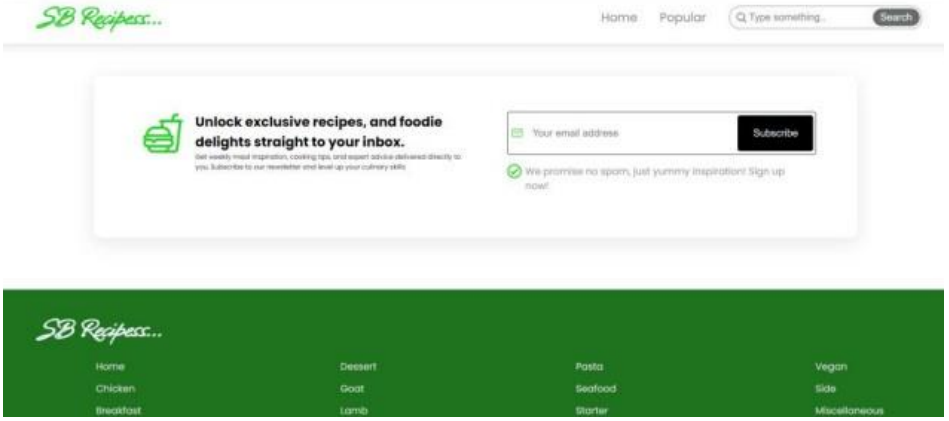
➢ **Trending Dishes**

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



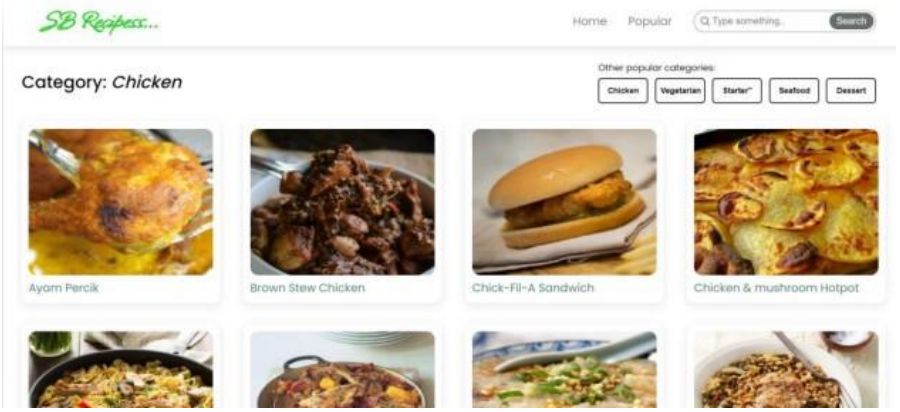
➢ **News Letter**

Thenewslettercomponentprovidesanemailinputtosubscribefortherecipe newsletters.



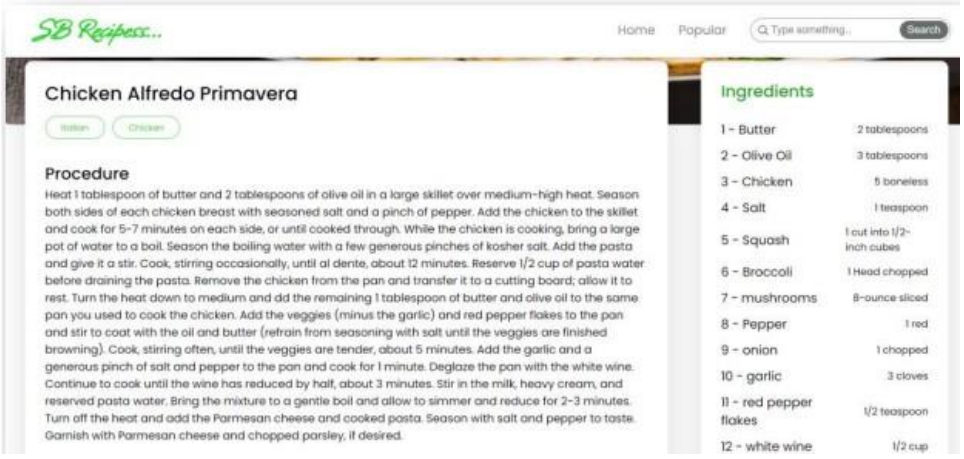
➢ **Category dishes page**

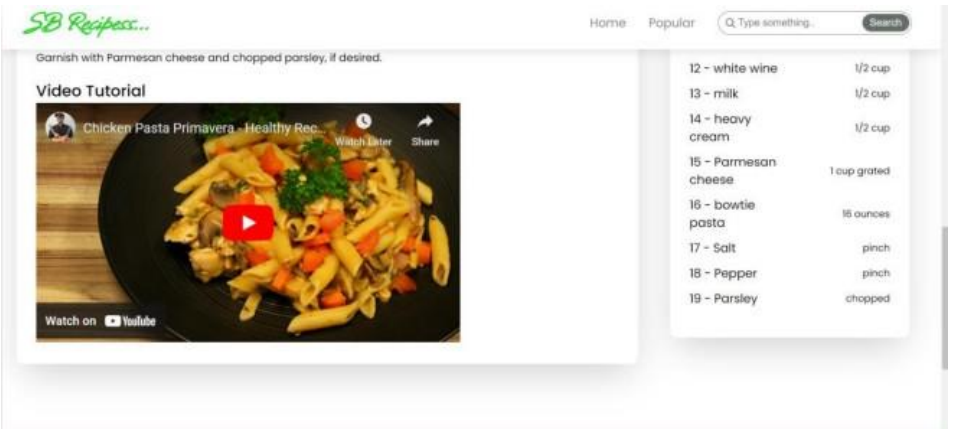
The category page contains the list of dishes under a certain category.



➢ **Recipe page**

Theimagesprovidedbelowshowstherecipepage,thatincludesimages,recipe instructions, ingredients and even a tutorial video.





Project demo link:
https://drive.google.com/file/d/1Ba-L7g6CsjMHzIpSFPz0AmDvYIcJDaF_/view?usp=drive_link

*** Happy coding!! ***