1. **INTRODUCTION**

* **PROJECT TITLE:**

**COOK BOOK: Your Virtual Kitchen Assistant**

**(React Application)**

* **TEAM MEMBERS:**

|  |  |  |
| --- | --- | --- |
| **TEAM MEMBERS** | **E-MAIL ID** | **ROLE** |
| VINODHA K  [TEAM LEADER] | kvinodha636@gmail.com | Tracking the progress of the project. |
| POOJA V | poojavelu2004@gmail.com | Planning and organizing. |
| ROGINI K | roginikubendran@gmail.com | Test the product for bugs and report defects. |
| NISHA V | nishatharun57@gmail.com | created the  demo video for this project. |
| MUTHULAKSHMI V | devivijay023@gmail.com | Managed the documentation of this project. |

**2.PROJECT OVERVIEW**

* **PURPOSE:**

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. 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.

**Key goals 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:**

**Core features**

**✓ 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.

**React – Specific Features**

* **Components:** The app is built using React components, which are reusable and modular.
* **JSX:** The app uses JSX to render components and manage state.
* **Hooks:** The app uses React Hooks (e.g. use State, use Effect) to manage state and side

effects.

**3.ARCHITECTURE:**

* **COMPONENTS STRUCTURE:**

**1. About.js Component**

* **Purpose**: This component provides information about the application, its mission, and how it helps users. It may include details about the developers, the inspiration behind the app, and any unique features it offers.
* **User Value**: Helps users understand the context of the app and builds trust by sharing the story behind it.

**2. Navbar.js Component**

* **Purpose**: The navigation bar (navbar) allows users to navigate between different sections of the application, such as Home, Recipes, Categories, About, and Contact.
* **User Value**: Enhances usability by providing quick access to different parts of the app, improving overall user experience and engagement.

**3. Newsletter.js Component**

* **Purpose**: This component is typically a subscription form that allows users to sign up for newsletters, updates, or special offers related to recipes and cooking.
* **User Value**: Engages users and encourages them to return by keeping them informed about new recipes, tips, or blog posts. It can also help build a community around the app.

**4. Hero.js Component**

* **Purpose**: The hero component usually features a large, eye-catching image or banner with a welcoming message or call to action (e.g., “Discover Delicious Recipes”).
* **User Value**: Captures users’ attention immediately when they land on the site, creating an inviting first impression and encouraging them to explore further.

**5. Footer.js Component**

* **Purpose**: The footer typically contains links to important information such as privacy policy, terms of service, social media links, and additional navigation options.
* **User Value**: Provides a space for users to find essential information and links without cluttering the main content area. It also adds professionalism to the app.

**6. Categories.js Component (or Categories Home)**

* **Purpose**: This component displays different recipe categories (e.g., Appetizers, Main Courses, Desserts) and allows users to filter recipes based on their interests.
* **User Value**: Enhances user experience by making it easier for users to find specific types of recipes, improving navigation and discovery within the app.
* **STATE MANAGEMENT:**

The state management approach used in this React application is a combination of

React Context API and Redux.

**1.React Context API:**

The React Context API is used to manage state that needs to be shared between

components, but doesn't require a full-fledged state management solution like Redux.

**- Contexts:** Multiple contexts are created to manage different types of state, such as

user authentication, news articles, and categories.

**- Providers:** Context providers are used to wrap the entire application, providing the

state to all components that need it.

**- Consumers:** Context consumers are used to access the state in individual

components.

**2.Redux:**

Redux is used to manage global state that requires more complex management, such as

news article fetching and caching.

**- Store:** A single Redux store is created to manage the global state.

**- Actions:** Actions are dispatched to trigger state changes, such as fetching news

articles or saving a user's preferences.

**- Reducers:** Reducers are used to handle actions and update the state accordingly.

**- Selectors:** Selectors are used to access the state in individual components.

* **ROUTING:**

**Routing Structure:**

The routing structure is designed to provide a logical and intuitive

navigation experience for users.

**Route Configuration:**

The route configuration is defined in a separate file (routes.js) and is imported into the

main app component (App.js).

**4.SETUP INSTRUCTION**

* **PRE-REQUISITES:**

Here are the key prerequisites for developing a frontend application:

**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.

**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.

**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 GitUp for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

**Development Environment:**

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

* **INSTALLATION:**

**Clone the repository:**

**Step 1:** Create a GitHub Account

**Step 2:** Create a New Repository

**Step 3:** Initialize the Repository

**Step 4:** Add Files to the Repository

**Step 5:** Push Changes to GitHub

**Step 6:** Configure Repository Settings

**Step 7:** Create a README File

**Step 8:** Create a License File

**Step 9:** Create a .gitignore File

**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 applications 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.

**Configure Environment Variables:**

Environment variables are values that are set outside of a program and are used to configure

the program's behavior. In the context of a Cook Book, environment variables can be used to store

sensitive information such as API keys, database credentials, and other configuration settings.

**Configuring Environment Variables:**

1. Create a .env File

2. Add Environment Variables

**5.FOLDER STRUCTURE**

* **CLIENT:**

The React application is organized into several folders, each with its own specific purpose. The

main folders are:

- **components/:** Reusable UI components

**- pages/:** Top-level routes for the application

**- styles/:** Global CSS styles

**-context/:** General Context

* **UTILITIES:**

**Helper Functions**

**- api.js:** A file containing helper functions for making API requests, such as fetchData, postData,

and putData.

**- storage.js:** A file containing helper functions for storing and retrieving data in local storage,

such as storeData and retrieveData.

**- string.js:** A file containing helper functions for string manipulation, such as capitalize and

truncate.

**Utilities**

**- constants.js:** A file containing constants used throughout the application, such as API endpoints

and color schemes.

**- enums.js:** A file containing enumerations used throughout the application, such as status codes

and error messages.

**Classes**

**- NewsArticle.js:** A class representing a news article, with properties such as title, author, and

content.

**- User.js:** A class representing a user, with properties such as name, email, and password.

**Custom Hooks**

**- useFetchData.js:** A custom hook for fetching data from an API, with options for caching and

error handling.

**- useStorage.js:** A custom hook for storing and retrieving data in local storage, with options for

encryption and expiration.

**- useForm.js:** A custom hook for managing form state and validation, with options for custom

validation rules and error messages.

**6.RUNNING THE APPLICATION**

The commands to start the frontend using npm start in the client directory:

**Navigate to the Client Directory**

1. Open a terminal or command prompt.

2. Navigate to the client directory using the cd command:

cd recipe-app-react

**Install Dependencies (Optional)**

* 1. If you haven't installed dependencies before, run the following command:

npm install

**Start the Frontend**

1. Start the frontend using the following command:

npm start

This will start the development server, and you can access your application at

http://localhost:3000 (or the port number specified in your package.json file).

**Verify the Application**

1. Open a web browser and navigate to http://localhost:3000.

2. Verify that your application is running correctly.

**7.COMPONENTS DOCUMENTATION**

* **KEY COMPONENTS:**

**1. About Component**

* **Purpose**: This component provides information about the application, its mission, and how it helps users. It may include details about the developers, the inspiration behind the app, and any unique features it offers.
* **User Value**: Helps users understand the context of the app and builds trust by sharing the story behind it.

**2. Navbar Component**

* **Purpose**: The navigation bar (navbar) allows users to navigate between different sections of the application, such as Home, Recipes, Categories, About, and Contact.
* **User Value**: Enhances usability by providing quick access to different parts of the app, improving overall user experience and engagement.

**3. Newsletter Component**

* **Purpose**: This component is typically a subscription form that allows users to sign up for newsletters, updates, or special offers related to recipes and cooking.
* **User Value**: Engages users and encourages them to return by keeping them informed about new recipes, tips, or blog posts. It can also help build a community around the app.

**4. Hero Component**

* **Purpose**: The hero component usually features a large, eye-catching image or banner with a welcoming message or call to action (e.g., “Discover Delicious Recipes”).
* **User Value**: Captures users’ attention immediately when they land on the site, creating an inviting first impression and encouraging them to explore further.

**5. Footer Component**

* **Purpose**: The footer typically contains links to important information such as privacy policy, terms of service, social media links, and additional navigation options.
* **User Value**: Provides a space for users to find essential information and links without cluttering the main content area. It also adds professionalism to the app.

**6. Categories Component (or Categories Home)**

* **Purpose**: This component displays different recipe categories (e.g., Appetizers, Main Courses, Desserts) and allows users to filter recipes based on their interests.
* **User Value**: Enhances user experience by making it easier for users to find specific types of recipes, improving navigation and discovery within the app.
* **REUSABLE COMPONENTS:**

some reusable components and their configurations:

**1. Button Component**

**- Purpose:** A reusable button component.

**- Configurations:**

**- variant:** Can be one of primary, secondary, success, danger, or warning.

**- size:** Can be one of small, medium, or large.

**- onClick:** A callback function to handle button clicks.

**- disabled:** A boolean indicating whether the button is disabled.

**2. Input Component**

**- Purpose:** A reusable input component.

**- Configurations:**

**- type:** Can be one of text, email, password, or number.

**- placeholder:** A string to display as a placeholder.

**- onChange:** A callback function to handle input changes.

**- value:** The initial value of the input.

**3. Select Component**

**- Purpose:** A reusable select component.

**- Configurations:**

**- options:** An array of options to display.

**- onChange:** A callback function to handle select changes.

**- value:** The initial value of the select.

**4. Table Component**

**- Purpose:** A reusable table component.

**- Configurations:**

**- columns:** An array of column definitions.

**- data:** An array of data to display.

**- onRowClick:** A callback function to handle row clicks.

**5. Modal Component**

**- Purpose:** A reusable modal component.

**- Configurations:**

**- isOpen:** A boolean indicating whether the modal is open.

**- onClose:** A callback function to handle modal closure.

**- title:** The title of the modal.

**- children:** The content of the modal.

**8.STATE MANAGEMENT OVERVIEW**

The cookbook application uses a combination of global and local state management to store and manage data.

* **GLOBAL STATE:**

The global state is used to store data that is accessible across the entire application.

**Global State Components**

**- User:** stores user information, such as name, email, and password

**- Recipes:** stores a list of all recipes in the application

**- Categories:** stores a list of all categories in the application

**- SearchQuery:** stores the current search query

**Global State Actions**

**- login:** updates the user global state with the logged-in user's information

**- logout:** clears the user global state

**- addRecipe:** adds a new recipe to the recipes global state

**- removeRecipe:** removes a recipe from the recipes global state

**- updateSearchQuery:** updates the search query global state

* **LOCAL STATE:**

The local state is used to store data that is specific to a particular component or feature.

**Local State Components**

**- RecipeDetail:** stores the current recipe being viewed

**- RecipeForm:** stores the current recipe being edited

**- SearchResults:** stores the current search results

**Local State Actions**

**- loadRecipe:** loads a recipe into the Recipe Detail local state

**- saveRecipe:** saves a recipe to the Recipe Form local state

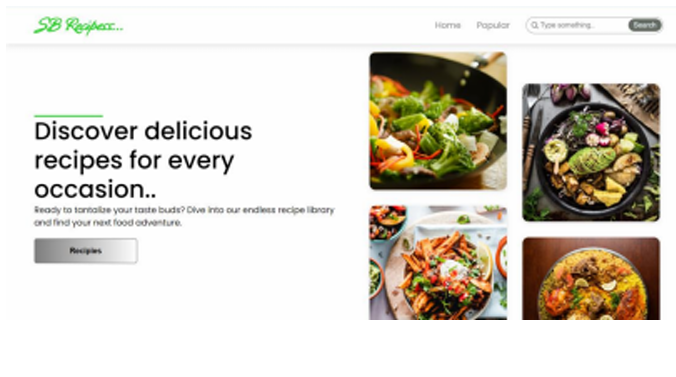
**- searchRecipes:** searches for recipes and updates the Search Results local state

**9.USER INTERFACE**

A user interface (UI) is the visual representation of a software application, website, or device that

allows users to interact with it. The UI provides a way for users to input data, navigate through the

application, and view the output.



**Screens**

**1.Home Screen:** Displays a list of top news stories, with images, headlines, and summaries.

**2.Popular Screen:** Displays a popular Dishes, with images, content and videos.

**Buttons and Icons**

**1. Home Button:** Returns the user to the Home Screen.

**2. Search Button:** Activates the search function.

**Animations and Transitions**

**1. Page Transitions:** Smooth transitions between pages.

**2. Button Animations:** Buttons animate on hover and click.

**3. Loading Animations:** Loading animations display while content is loading.

**10.STYLING**

* **CSS FRAMEWORKS/LIBRARIES:**

**CSS Frameworks**

**1. Bootstrap:** A popular front-end framework that provides a comprehensive set of CSS classes and

components for building responsive and mobile-first UI components.

**2. Tailwind CSS:** A utility-first CSS framework that provides a set of pre-defined classes for styling

HTML elements.

**3. Material-UI:** A popular CSS framework developed by Google that provides a set of pre-built UI

components based on the Material Design specification.

**4. Bulma:** A modern CSS framework that provides a set of pre-built UI components and a simple

grid system.

**5. Foundation:** A popular front-end framework that provides a comprehensive set of CSS classes

and components for building responsive and mobile-first UI components.

**CSS Libraries**

**1. Normalize.css:** A small CSS library that provides a set of CSS rules to normalize the styling of

HTML elements across different browsers.

**2. Reset CSS:** A small CSS library that provides a set of CSS rules to reset the styling of HTML

elements to a consistent baseline.

**3. Animate.css:** A CSS library that provides a set of pre-defined animation classes for adding

animations to HTML elements.

**CSS Preprocessors**

**1. Sass:** A popular CSS preprocessor that provides a set of features such as variables, nesting, and

mixing for writing more efficient and modular CSS code.

**2. Less:** A CSS preprocessor that provides a set of features such as variables, nesting, and mixing

for writing more efficient and modular CSS code.

**3. Stylus:** A CSS preprocessor that provides a set of features such as variables, nesting, and mixing

for writing more efficient and modular CSS code.

**CSS-in-JS Libraries**

**1. Styled Components:** A popular CSS-in-JS library that provides a set of features such as dynamic

styling and theming for writing more efficient and modular CSS code.

**2. Emotion:** A CSS-in-JS library that provides a set of features such as dynamic styling and theming

for writing more efficient and modular CSS code.

* **THEMING:**

Theming allows you to change the visual appearance of your application, such as colors,

typography, and layout, without modifying the underlying code. In CookBook, theming can be

implemented using various techniques:

**1. CSS Variables:** Define CSS variables for colors, typography, and other visual elements. Then,

use these variables in your CSS code. This way, you can easily switch between different themes by

changing the values of these variables.

**2. Theme Provider:** Create a theme provider component that wraps your entire application. This

component can provide a theme object that contains the visual styles for your application. Then, use

this theme object in your components to apply the styles.

**3. Styled Components:** Use a library like Styled Components, which allows you to define styles as

JavaScript functions. This way, you can easily switch between different themes by changing the

styles.

**Custom Design System**

A custom design system is a collection of reusable UI components, guidelines, and assets that

define the visual design of your application. In CookBook, a custom design system can be

implemented using various techniques:

**1. Create a Design Language System (DLS):** Define a set of principles, guidelines, and assets that

describe the visual design of your application. This includes typography, colors, spacing, and other

visual elements.

**2. Build a UI Component Library:** Create a library of reusable UI components that adhere to the

design language system. This includes components such as buttons, inputs, cards, and other UI

elements.

**3. Use a Design System Tool:** Use a tool like Recipebook, Bit, or Lona to create, manage, and

document your design system.

**11.TESTING**

* **TESTING STRATEGY:**

**1.Unit Testing**

Unit testing involves testing individual components or functions in isolation. For React components,

unit testing typically involves rendering the component and verifying that it renders correctly.

**Tools:**

**- Jest:** A popular testing framework for React applications.

**- React Testing Library:** A testing library that provides a set of APIs for testing React components.

**2.Integration Testing**

Integration testing involves testing how multiple components interact with each other. For React

applications, integration testing typically involves rendering a component tree and verifying that the

components interact correctly.

**Tools:**

**- Jest:** A popular testing framework for React applications.

**- React Testing Library:** A testing library that provides a set of APIs for testing React components.

**3.End-to-End Testing**

End-to-end testing involves testing the entire application from start to finish, simulating user

interactions and verifying that the application behaves correctly.

**Tools:**

**- Cypress:** A popular end-to-end testing framework for web applications.

**- Playwright:** A browser automation framework that can be used for end-to-end testing.

* **CODE COVERAGE:**

**Testing Tools:**

**-Jest:** A popular testing framework for React applications.

**-React Testing Library:** A testing library that provides a set of APIs for testing React

components.

**-Cypress:** A popular end-to-end testing framework for web applications.

**-Enzyme:** A testing library that provides a set of APIs for testing React components.

**Testing Techniques:**

**-Unit Testing:** Test individual React components in isolation.

**-Integration Testing:** Test how multiple React components interact with each other.

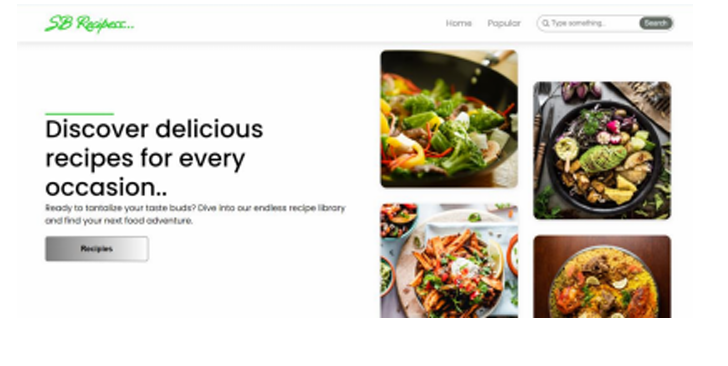
**-End-to-End Testing:** Test the entire application from start to finish.

**-Snapshot Testing:** Test that the component's UI matches a previously saved snapshot.

**12.SCREENSHOTS OR DEMO**

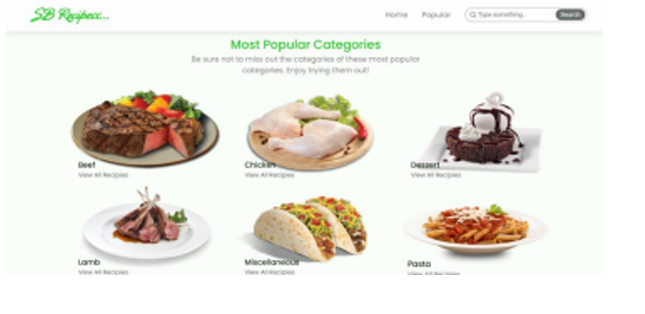
* **HERO COMPONENTS:**

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



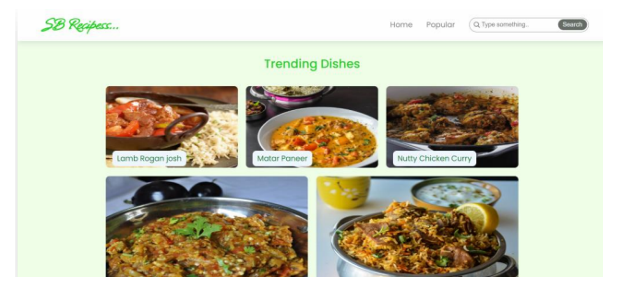
* **POPULAR CATEGORIES:**

This component contains all the popular categories of recipes..



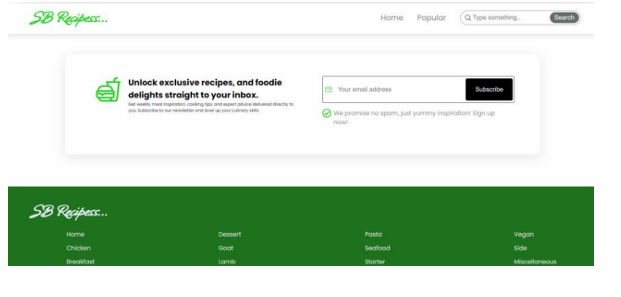
* **TRENDING DISHES:**

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



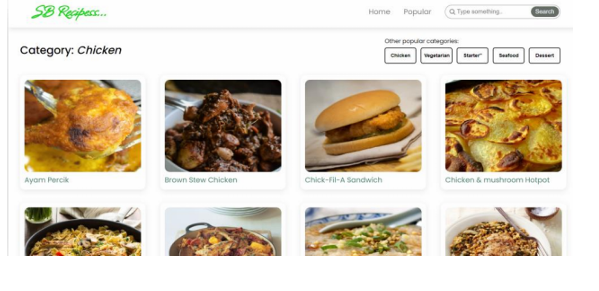
* **NEWS LETTER:**

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



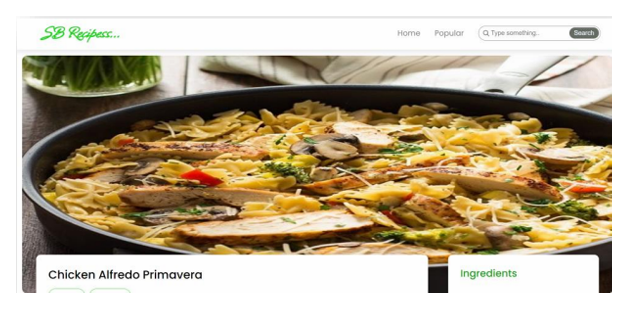
* **CATEGORY DISHES PAGE:**

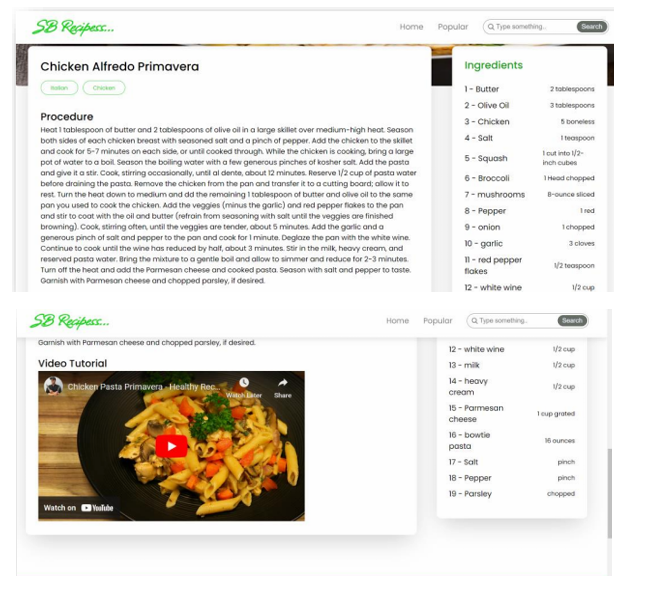
The category page contains the list of dishes under a certain category.



* **RECIPE PAGE:**

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





* **PROJECT DEMO LINK:**

<https://drive.google.com/file/d/1Cj12YtH3NTunfVfycRwQbTKHcb3WVEbl/view?usp=sharing>

**13.KNOWN ISSUES**

* **SOME KNOWN BUGS AND ISSUES ARE:**

▪ Article image loading issue

▪ Performance issue

▪ Accessibility issue

**14.FUTURE ENHANCEMENT**

Some potential future features or improvements for CookBook:

* **NEW COMPONENTS:**

1. Personalized Recipe Recommendations

2. Meal Planning

3. Recipe Import and Export

4. Smart Kitchen Appliance Integrations

* **ANIMATIONS:**

1. Article Loading Animation

2. Component Transition Animations

3. Scrolling Animations

* **ENHANCED STYLING:**

1. Customizable Themes

2. Improved Typography

3. Consistent Spacing

4. Enhanced Image Display