**GOVERNMENT ARTS AND SCIENCE COLLEGE**



POONTHOTAM STREET, THERADI, THIRUVOTTIYUR

CHENNAI – 600 019

Cook Book : Your Virtual Kitchen Assistant

A NAAN MUDHALVAN PROJECT

***Submitted in Partial Fulfillment for the Award of***

**BATCHELOR OF COMPUTER APPLICATION**

BY

**GNANAPRIYA V (212405030)**

**BHUVANSURYA D (212405026)**

**ABARNA S (212405023)**

**ASWINI T (212405024)**

**BAKKIYALAKSHMI B (212405025)**

**2024 - 2025**

Cook Book :

Your Virtual Kitchen Assistant



**Introduction**

Welcome to **Vitruval Kitchen**, where the **art of cooking** meets the **science of balance**. Inspired by the timeless principles of the Roman architect **Vitruvius** — who believed that great design should embody **firmitas (strength), utilitas (function), and venustas (beauty)** — this cookbook applies the same ideals to the modern kitchen. Because great food, like great architecture, must not only be built well — it must **nourish, delight, and endure**.

**In this book, you’ll find:**

* **Deliberately designed recipes** that balance flavour, nutrition, and visual appeal.
* **Techniques** that empower beginners and challenge seasoned cooks.
* **Stories, history, and kitchen philosophy** that link culinary creativity to timeless design.

Whether you're preparing a humble weeknight meal or composing a grand dinner for guests, Vitruval Kitchen invites you to see cooking as an **intentional act of creation** — one that blends structure with soul.

So sharpen your knives, warm your pans, and step into a kitchen where every dish is **measured, meaningful, and made to last**.

**Description:**

Your Virtual Kitchen Assistant is more than just a cookbook—it’s your smart companion in the kitchen. Whether you’re a beginner learning the basics or a food enthusiast exploring new cuisines, this book guides you step by step with easy-to-follow instructions, cooking tips, and practical shortcuts. Blending traditional flavours with modern techniques, it helps you create delicious, healthy, and time-saving meals without the stress. From quick weeknight dinners to festive feasts, this assistant is always ready to inspire you with recipes, organization tips, and culinary creativity—all at your fingertips.

✨ **Key Features:**

* 👩🍳 **Interactive Step-by-Step Recipes**

Follow clear, voice-guided instructions with built-in timers and tips at every stage.

* 🛒 **Smart Shopping Lists**

Auto-generate ingredient lists based on selected recipes, with optional store integrations.

* 📱 **Hands-Free Mode**

Navigate recipes using voice commands — perfect when your hands are messy!

* 🧑🍳 **Personalized Meal Planning**

Create weekly meal plans tailored to your dietary needs, preferences, and time constraints.

* 🧠 **AI Cooking Assistant**

Get real-time answers to your kitchen questions: substitutions, cooking times, portion sizes, and more.

* 🥗 **Healthy Eating Made Easy**

Track calories, macros, and nutrition for every meal you prepare.

# PRE-REQUISITES

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.
* **Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like Git Hub or Bit bucket can host your repository.
* Git: Download and installation instructions can be found at: [https://gitscm.com/downloads](https://git-scm.com/downloads)

* **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>
* Sublime Text: Download from <https://www.sublimetext.com/download> • Web Storm: Download from <https://www.jetbrains.com/webstorm/download>

To get the Application project from drive:

Follow below steps:

**Install Dependencies:**

* Navigate into the cloned repository directory and install libraries: cd fitness-app-react npm install

 **Start the Development Server**:

* To start the development server, execute the following command:

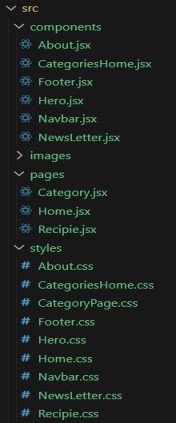
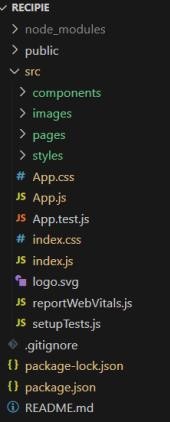
npm start

**Access the App:**

* Open your web browser and navigate to [http://localhost:3000.](http://localhost:3000/)
* You should see the application'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 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 setup and configuration

 **Project setup and configuration.**

* **Installation of required tools**:

To build Cook Book, 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

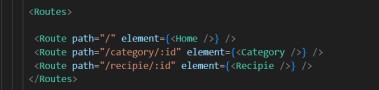
* For further reference, use the following resources o https://react.dev/learn/installation

o https://react-bootstrap-v4.netlify.app/getting-started/introduction/ o https://axioshttp.com/docs/intro o https://reactrouter.com/en/main/start/tutorial

**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 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 call back function (fetch Categories in this case) and an optional dependency array. The call back 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:**

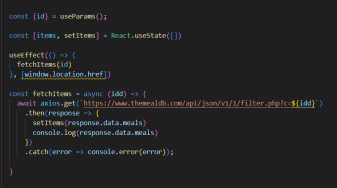
An asynchronous function named fetch Categories is defined to handle the API interaction. This function utilizes the axios.get method to make a GET request to a specified API endpoint (https://www.themealdb.com/api/json/vi/1/categories.php in this example). This particular endpoint presumably returns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

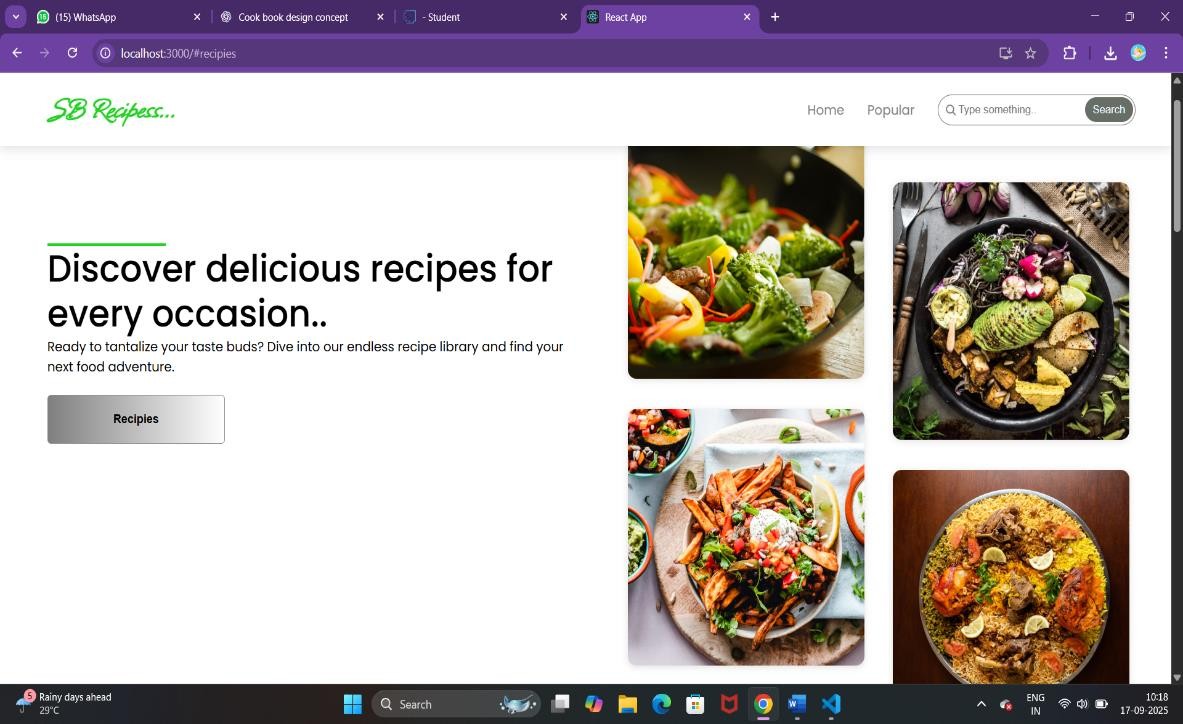
* The code snippet employs the .then method, which is chained to the axios.get call, to handle a successful response from the API. Inside the .then block, the code retrieves the first recipe from the data. Meals array in the response and updates the React component's state using the set Recipe function. 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.

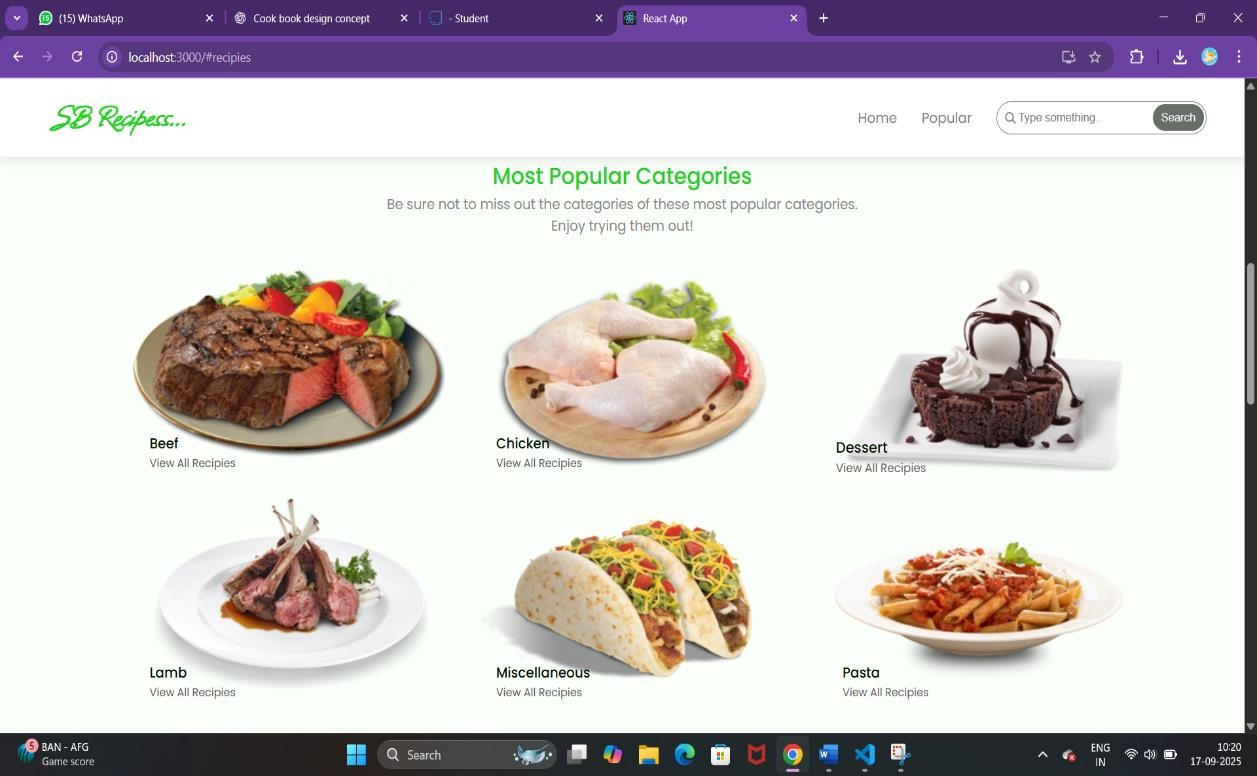
* An optional error handling mechanism is incorporated using the .catch block. This block is designed to manage any errors that might arise during the API request. If an error occurs, the .catch block logs the error details to the console using the console. Error method. This rudimentary error handling mechanism provides a way to identify and address potential issues during the data fetching process.

Error:

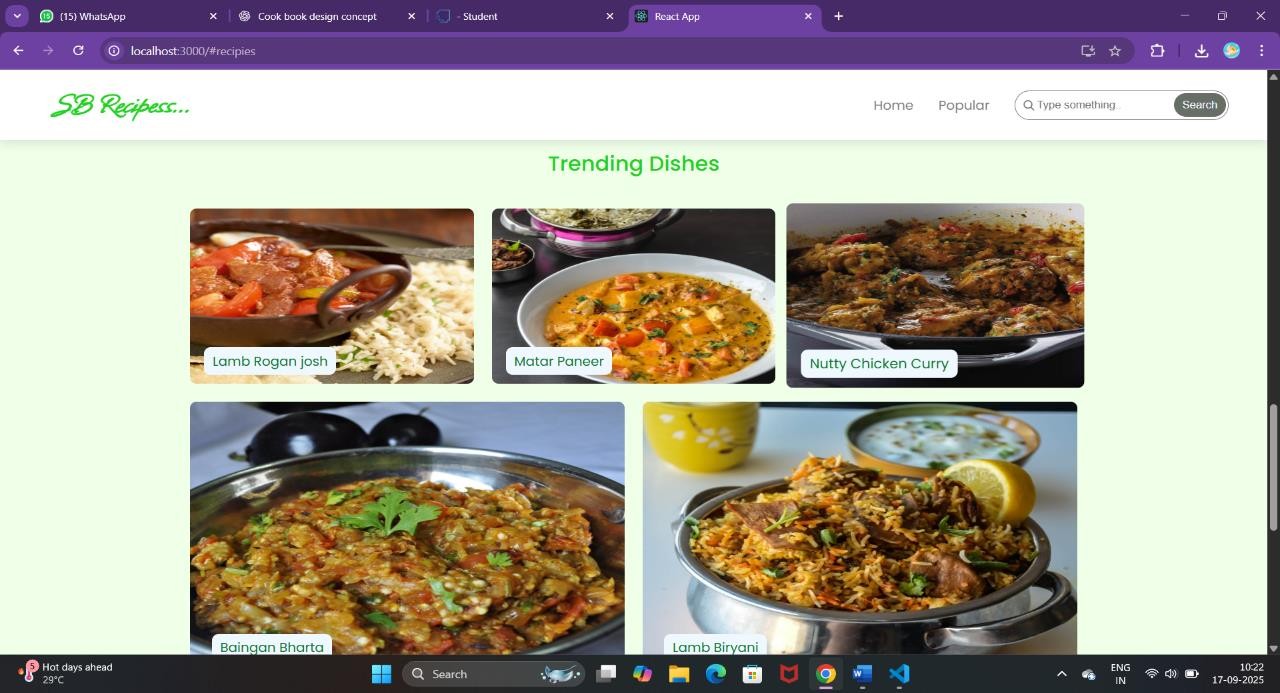


**Project Implementation & Execution :**

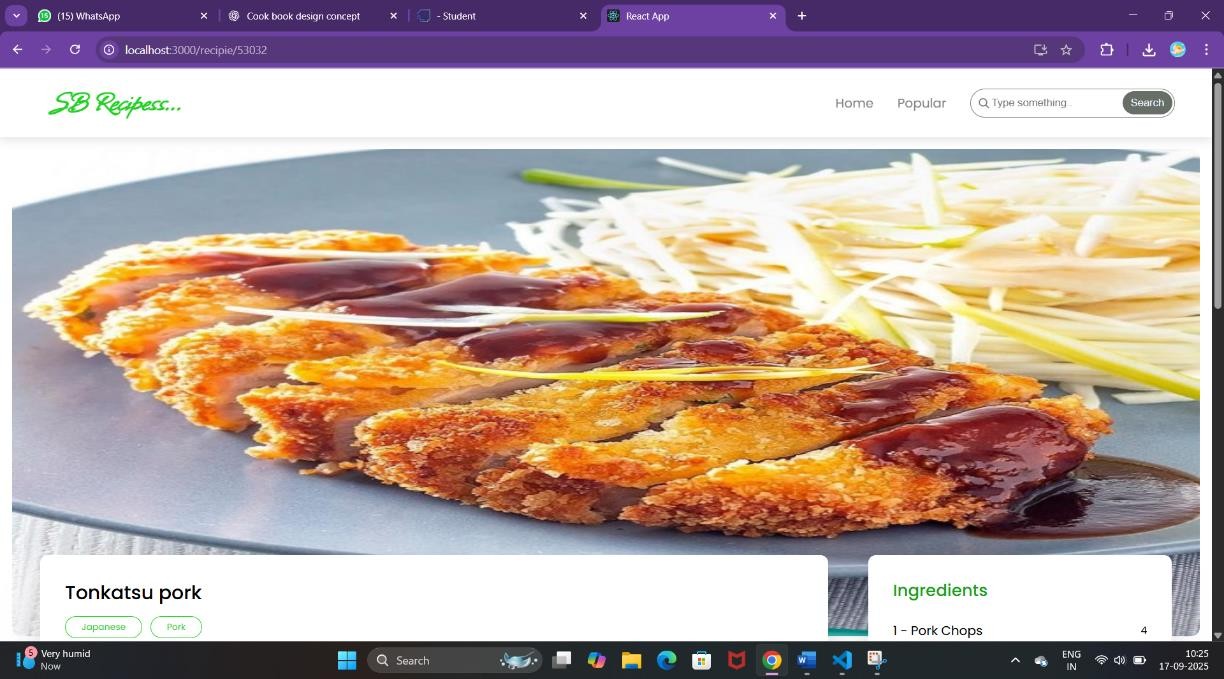


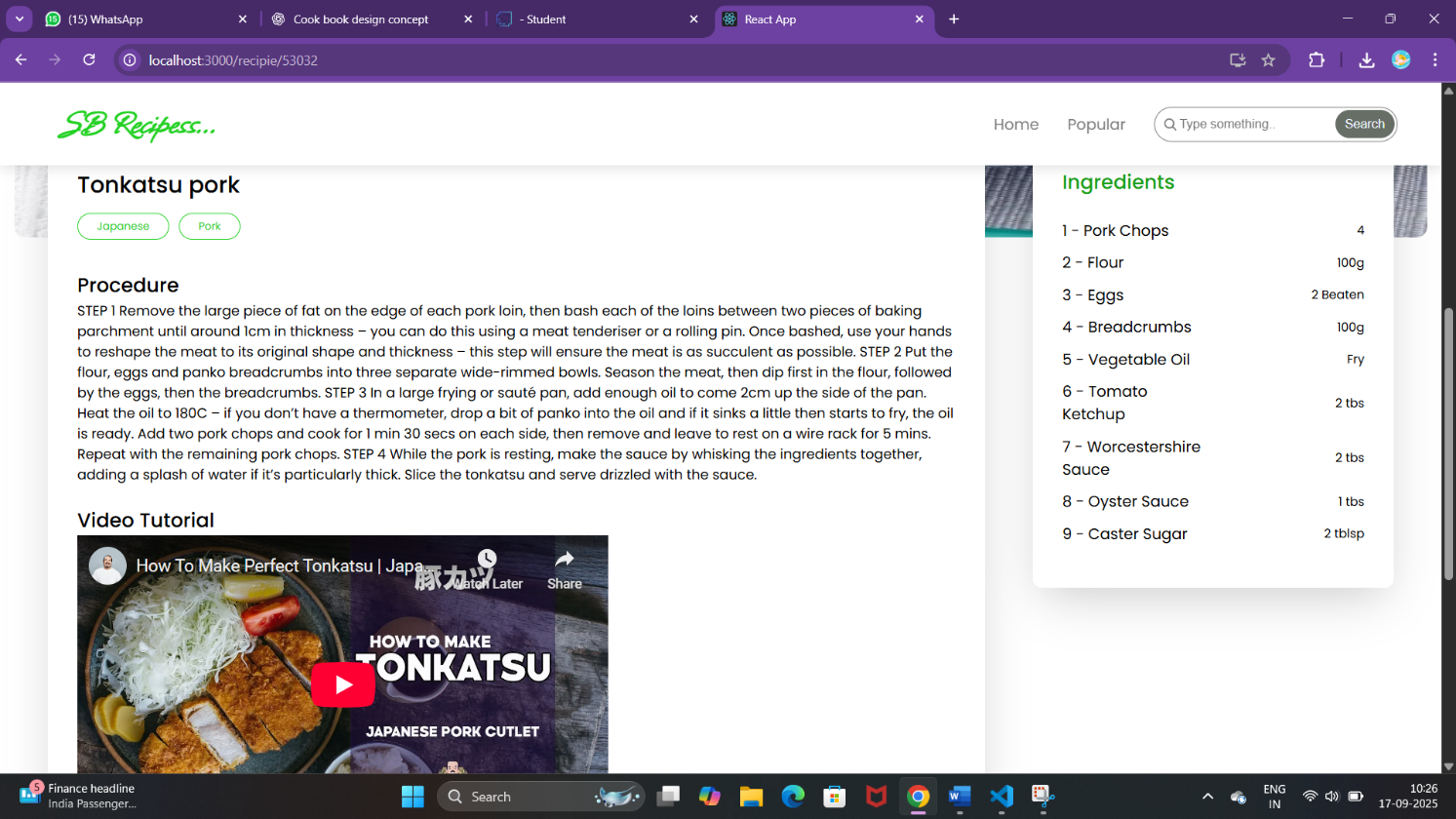


**Trending Dishes:**



**Recipe Method With Ingredients:**





THANKING YOU