Project Documentation

**Project Title:** “COOKBOOK”

**1.Introduction**

• Project Title: CookBook: Your Virtual Kitchen Assistant.

• Team ID: NM2025TMID29906

• Team Leader: SIDDHESWARI K & [202400754@sigc.edu](mailto:202400754@sigc.edu)

• Team Members:

- Sharulatha R & [202400455@sigc.edu](mailto:202400455@sigc.edu)

- Yugabharathi K & [202400799@sigc.edu](mailto:202400799@sigc.edu)

- Priyadharshini V & [202400809@sigc.edu](mailto:202400809@sigc.edu)

**2. Project Overview**

• Purpose: **Your Virtual Kitchen Assistant** is a digital platform designed to help users manage their kitchens efficiently. It provides personalized recipe recommendations, shopping lists, and cooking guidance based on available ingredients. The goal is to reduce food waste, save time, and make cooking enjoyable.

• Features:

1. **Recipe Suggestions**: Recommends recipes based on the ingredients you already have in your kitchen.
2. **Smart Shopping List**: Automatically generates a shopping list based on the recipes chosen or your pantry inventory.
3. **Step-by-Step Cooking Instructions**: Provides detailed, easy-to-follow instructions for each recipe.
4. **Voice Assistance**: Allows hands-free interaction with the assistant to guide users through recipes while they cook.
5. **Pantry Management**: Tracks your kitchen inventory and alerts you when items are running low or about to expire.

**3. Architecture**

• Frontend: **Technology**: **React.js** for building dynamic, interactive UI.

* Backend: **Technology**: **Node.js** with **Express.js**.
* Database: **Technology**: **MongoDB**.

**4. Setup Instructions**

• Prerequisites:

# Node.js

# MongoDB

# Git

# Text Editor / IDE

# npm / yarn

• Installation Steps:

1. **Clone the Repository**:

git clone <https://github.com/your-repo/virtual-kitchen-assistant.git>

cd virtual-kitchen-assistant

1. **Install Frontend Dependencies**:

cd client

npm install

1. **Install Backend Dependencies**:

cd ../server

npm install

1. **Set Up MongoDB**:
   1. Create a MongoDB database (local or cloud, e.g., MongoDB Atlas).
   2. Update the **.env** file in the server folder with your MongoDB connection string.
2. **Start the Application**:
   1. Start the backend server:
   2. npm start
   3. Start the frontend client:
   4. cd ../client
   5. npm start
3. **Access the App**:
   1. Open your browser and go to http://localhost:3000 to use the app.
4. **Folder Structure:**

/virtual-kitchen-assistant

├── /client # Frontend React app

│ ├── /public # Static files (e.g., index.html)

│ ├── /src # React components, pages, and assets

│ │ ├── /components # Reusable UI components

│ │ ├── /pages # Page-level components (e.g., Home, Recipes)

│ │ └── /styles # CSS/SCSS styles

│ ├── package.json # Frontend dependencies

│ └── .env # Frontend environment variables

│

├── /server # Backend Node.js app

│ ├── /controllers # Logic for handling requests (recipes, pantry)

│ ├── /models # Database models (e.g., User, Recipe)

│ ├── /routes # API routes (e.g., /recipes, /pantry)

│ ├── /config # Server configuration (e.g., DB connection)

│ ├── /middleware # Custom middleware (e.g., authentication)

│ ├── server.js # Main server file

│ ├── .env # Backend environment variables

│ └── package.json # Backend dependencies

│

├── /node\_modules # Installed dependencies (generated after npm install)

├── .gitignore # Git ignore file

└── README.md # Project documentation

1. **Running the Application:**

**Start the Backend**:

* Navigate to the **server** directory:
* cd server
* Run the backend server:
* npm start

**Start the Frontend**:

* Navigate to the **client** directory:
* cd client
* Run the frontend client:

npm start

• Access: Visit <http://localhost:3000>

1. **API Documentation:**

**Authentication**

* **POST** /api/auth/login – Login with email and password.
* **POST** /api/auth/register – Register a new user.

**Recipes**

* **GET** /api/recipes – Get all recipes.
* **GET** /api/recipes/:id – Get a recipe by ID.

**Pantry**

* **GET** /api/pantry – Get pantry inventory.
* **POST** /api/pantry – Add an item to the pantry.

**Shopping List**

* **GET** /api/shopping-list – Get shopping list.
* **POST** /api/shopping-list – Add item to shopping list.

1. **User Interface:**

 **Home Screen**: Displays personalized recipe recommendations based on available ingredients.

 **Pantry Management**: Allows users to add, view, and update their kitchen inventory.

 **Recipe Details**: Shows ingredients, step-by-step cooking instructions, and estimated cooking time.

 **Shopping List**: Automatically generated and editable shopping list based on chosen recipes.

 **Search Bar**: Enables quick search for recipes by ingredient or name.

 **Responsive Design**: Works smoothly on desktop, tablet, and mobile devices.

1. **Testing:**

**Unit Testing**: Test individual components and functions using **Jest**.

**API Testing**: Verify backend endpoints with **Postman** or **Supertest**.

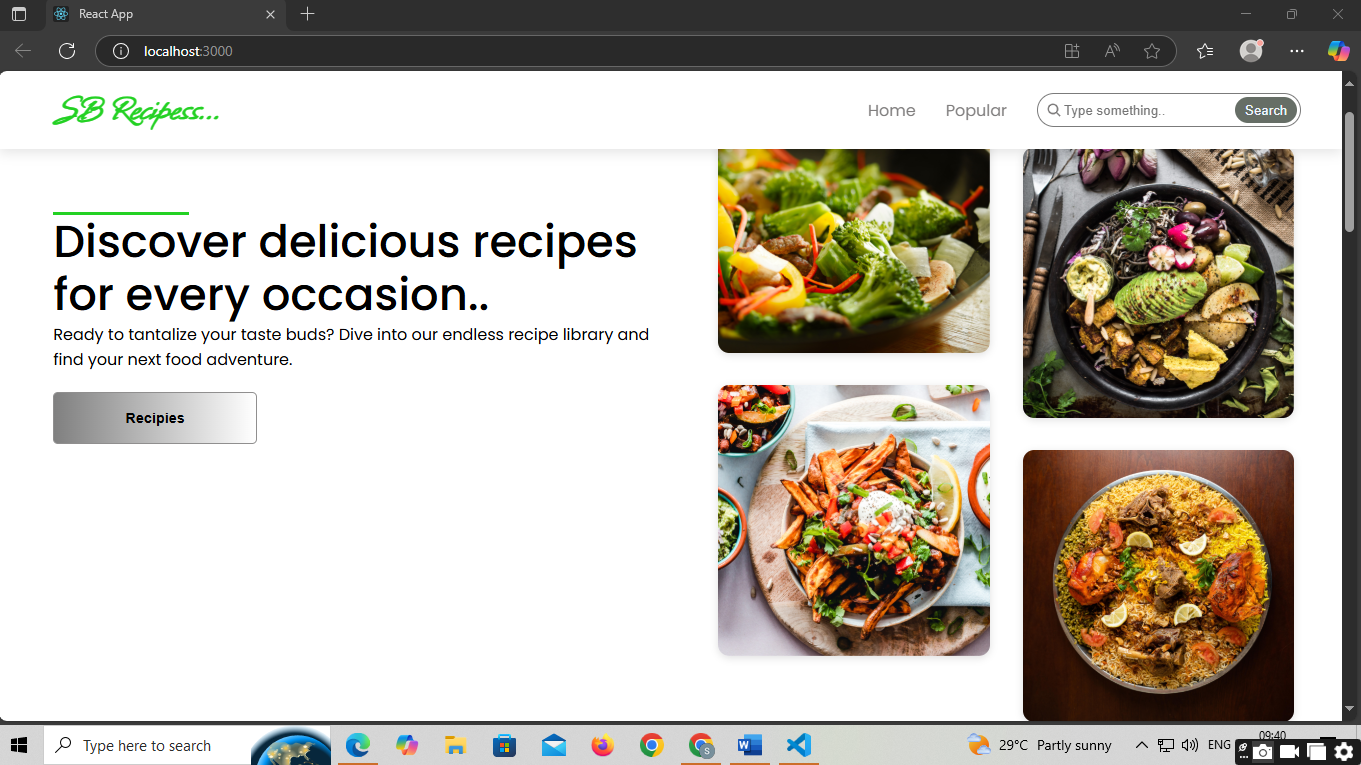
**UI Testing**: Use **React Testing Library** for frontend component tests.

**Manual Testing**: Check user flows like recipe search, pantry updates, and shopping list management.

**Bug Tracking**: Report and fix issues during development to ensure smooth performance.

1. **Screenshots or Demo:**

**Live Demo:** [**https://drive.google.com/file/d/1DJQoOsc0siyHnjZYFIZiPkddHKdQL43F/view?usp=drivesdk**](https://drive.google.com/file/d/1DJQoOsc0siyHnjZYFIZiPkddHKdQL43F/view?usp=drivesdk)



1. **Known Issues:**

* Occasional delays in real-time recipe updates.
* Limited recipe database — fewer options for specialized diets.
* Some UI elements may not be fully responsive on older mobile devices.
* Minor bugs in pantry item deletion process.
* Voice assistant may not recognize all accents accurately.

1. **Future Enhancements:**

Add AI-based personalized recipe recommendations.

Integrate voice-controlled cooking assistant.

Expand recipe database with diverse cuisines and dietary options.

Implement meal planning and calorie tracking features.

Develop a mobile app for on-the-go access.