**PROJECT DOCUMENTATION**

**Project Title: COOKBOOK: YOUR VIRTUAL KITCHEN ASSISTANT**

****

**Team ID: NM2025TMID36561**

**Team Leader:**

* **Afrin. Z- 24csafrin003@gmail.com**

**Team Members:**

* **Abinaya. A- 24csabinaya001@gmail.com**
* **Abirami. S- 24csabirami002@gmail.com**
* **Akshaya. M- 24csakshaya004@gmail.com**

**Project Overview**

**1. Purpose**

The purpose of the project is to create a visual and interactive cookbook that serves as a kitchen assistant for users. It provides step-by-step recipes with clear instructions and supporting visuals to make cooking easier, especially for beginners. The project aim to simplify the cooking process through the visual guidance.

**2.Introduction**

The Cookbook-Virtual Kitchen Assistant aims to provide a digital platform for food Enthusiasts, chefs, and home cook to discover new recipes, organize meal plans, and connect with others who share the same passion for cooking. Overall, it acts as a convenient Kitchen companion for both beginners and experienced cooks.

**Benefits Of Cook Book**

* Acts like a personal kitchen helper, making it easier for beginners to cook confidently.
* The users can quickly find a variety of recipes in one place without searching multiple sources.
* Encourages users to try new dishes and improve their cooking skill.
* Works as a digital cookbook that’s always accessible on phone or laptop.
* It can recommend recipes based on dietary preferences, allergies, or cuisine types.

**Features:**

* It give a clear instructions with photos or video for step.
* A well-organized collection of recipes (by cuisine, meal type, difficulty level, etc.)
* It Automatically creates a shopping lists from selected recipes.
* Display calories and nutrients for each recipes.
* User can search recipes by ingredients, time, or dietary needs.
* It save the favorite recipes for quick access.

### 

**3.Architecture:**

### **Cookbook- Your Virtual Kitchen Assistant has 3 main layers:**

### **1**.**Presentation Layer(Front-End):**

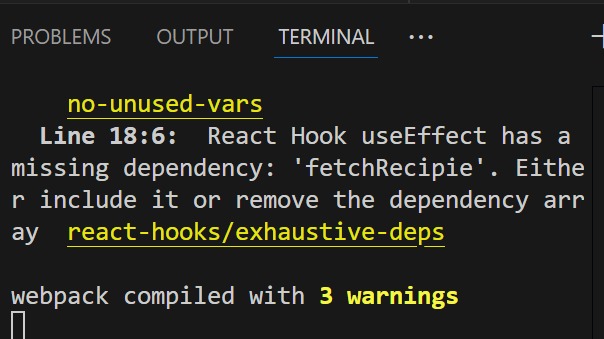
### Built with HTML/ CSS/ JAVA SCRIPT (or React/ Angular if advanced.

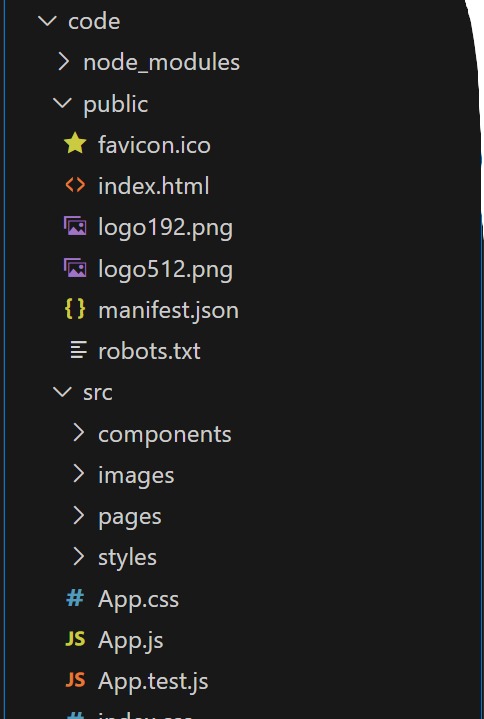
### **2.** **Application layer(Back- End):**

### Communicates with the data base using APIs (REST/Graph QL)

### **3.Data layer(Data Base):**

### Stores all recipes, ingredients, nutritional info, user accounts, saved recipes, shopping lists etc.

****

****

****

**4.Set-up Instruction**

**Prerequisities**

To begin a web development project that uses Node.js, MongoDB, Git, React.js, Express.js, Mongoose, and Visual Studio Code, you must have a solid understanding of several key technologies.

* **Node.js**
* A runtimeenvironment that let you run JavaScript code outside the browser.
* Essential for building backend services and running tools like NPM (Node Package Manager) .
* **Mongo DB**
* A NoSQL data base used to store application data in flexible,JSON-like documents.
* Ideal for dynamic and scalable web applications.
* **Git**
* A version control system that help you track changes in your code.
* Useful for collaboration and managing project history.
* **React.js**
* A java script library for building user interface, especially single -page applications.
* Helps to create dynamic and responsive frontends.
* **Express.js**
* A light weight web frame work for node.js.
* Used to build APIs handled server-side logic efficiently.
* **Visual Studio Code**
* A powerful and free code editor development by Microsoft.
* Supports extensions, debugging, and integrated terminal-perfect for full- stack development

**Installation Steps:**

* **Clone the repository**

**git clone <repo-link>**

* This command to copies the entire repository from a remote server (eg. Github, Gitlab) to your local machine.
* It allows to you to get the project code so that you can work with it locally.
* You need to replace <repo-link>with the actual URL of the repository you want to clone.
* **Install client dependencies**

**cd client**

**npm install**

* This command navigates into the client directory of the project. Typically, this folder contains the font-end code of your project (like react or angular).
* This command installs the necessary JavaScript dependencies for the front-end, as listed in the package.json file in the client directory.
* You need to install the dependencies that your front-end code requires to run. These dependencies could include libraries like React, Axios, etc.
* **Install server dependencies**

**cd ../server**

**npm install**

* This command navigates from the client folder to the server folder (which is typically one level above the client directory).
* This installs the dependencies for the server side of the project, as specified in the package.json file within the server folder. The server could be running with Node.js, Express, or other backend technologies.
* The server will need its own set of dependencies to run correctly, like libraries for handling requests, databases, authentication, etc

.



**Folder Structure:**

**Cookbook-App/**

**│── src/**

**│ ├── App.js → Main application logic (state + functions + layout)**

**│ ├── components/ → Folder for reusable components**

**│ │ ├── CookRecipesForm.js → Form to add new recipes (name + picture)**

**│ │ ├── FoodList.js → Shows list of all recipes**

**│ │ └── FoodStyleItem.js → Handles a single recipe card (picture + style)**

**│ ├── index.js → Entry point of React app (renders <App />)**

**│ └── App.css → Styling for the app**

**Running the Application:**

**FRONTEND**

* **Frontend (Client)**

The client folder contains all your HTML/CSS/JS or React code that displays the cookbook UI.

cd client # go to the frontend folder

npm install # install required packages (first time only)

npm start # start the frontend dev server

By default this runs on http://localhost:3000 (React’s default port).

This will open automatically in your browser.

You’ll see the Cook Book Home Page with vibrant colors, featured recipes, and category cards (Indian, Italian, Desserts, etc.)

* **Backend (Server)**

The server folder contains all the backend logic, like storing recipes, user profiles, and handling new submissions.

cd server # go to the backend folder

npm install # install backend dependencies (first time only)

npm start # start the backend server

By default this runs on http://localhost:5000 (or whatever port is configured in server.js).

The backend provides APIs like:

GET /Api/recipes – list all recipes

POST /Api/recipes – add a new recipe

GET /Api/recipes/:id – fetch a single recipe

* **Accessing the App**

The frontend will talk to the backend via API calls to http://localhost:5000 (or your configured URL).

and now browse the digital cookbook:

Search recipes by name or ingredient.

Filter by cuisine (Indian, Chinese, Italian…).

View high-resolution food pictures.

**7. API Documentation**

Users

GET /v1/users/:id — Get a public user profile (name, avatar, bio, project Count)  
GET/v1/users/me—Get current user profile (protected)PUT /v1/users/me — Update profile (protected)  
DELETE /v1/users/me — Delete account (protected) Projects.

**GET /v1/projects** — List all projects (public)  
Query params: page, limit, q (search by title/description), sort.

**GET /v1/projects/:id** — Get project details (includes owner, members, created At)  
**PUT /v1/projects/:id** — Update project (protected — only owner/admin)  
**DELETE /v1/projects/:id** — Delete project (protected)

**POST /v1/projects/:id/members** — Add member to project (protected)  
**DELETE /v1/projects/:id/members/:user Id** — Remove member from project

**8.Authentication**

**Authentication is a mechanism used to verify the identity of a user before granting access to personalized or restricted features of the Cook Book application. It ensures that each user’s saved recipes, preferences, and account details remain secure and private.**

**Key Objectives:**

**Allow users to register and log in securely**

**Protect user data and personal recipe collections**

**Enable personalized experiences (Favorites, saved recipes, shopping lists)**

**User -> Registration Form -> Backend (save hashed password) -> Database**

**User -> Login Form -> Backend (verify credentials) -> JWT issued -> Stored in browser (HTTP Only cookie)**

**User -> Access protected routes (like “My Recipes”) -> Backend verifies JWT -> Access granted**

### **9. User Interface**

### **Landing Page**

### Intro, features, and popular projects.

### Options: **Login / Sign Up** and **Post a Project**.

### **Freelancer Dashboard**

### Shows **bids, recommended projects, messages, and earnings**.

### Quick actions: **Apply, View Stats, Manage Profile**.

### **Admin Panel**

### Manage **users, projects, disputes, and settings**.

### Tools: **tables, filters, analytics, approve/suspend actions**.

### **Project Details Page**

### Displays **title, budget, skills, description, client info**.

### Actions: **Freelancers → Bid/Edit Bid** | **Clients → Manage Proposals & Milestones**.

### **12. Known Issues**

### **Authentication Error** • Token expiration may cause users to be logged out unexpectedly. • JWT refresh mechanism not fully implemented.

### **UI/UX Limitations** • Some pages are not fully responsive on smaller devices. • Limited accessibility support (e.g., screen reader compatibility).

### **Performance Constraints** • Slow load times for pages with large datasets (e.g., project listings). • Search and filter functions may lag under heavy load.

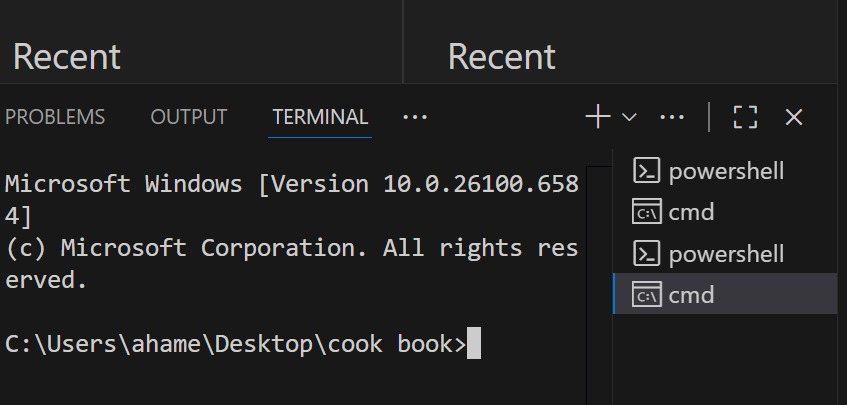
### **Chat System** • Occasional message delivery delays. • No offline message caching.

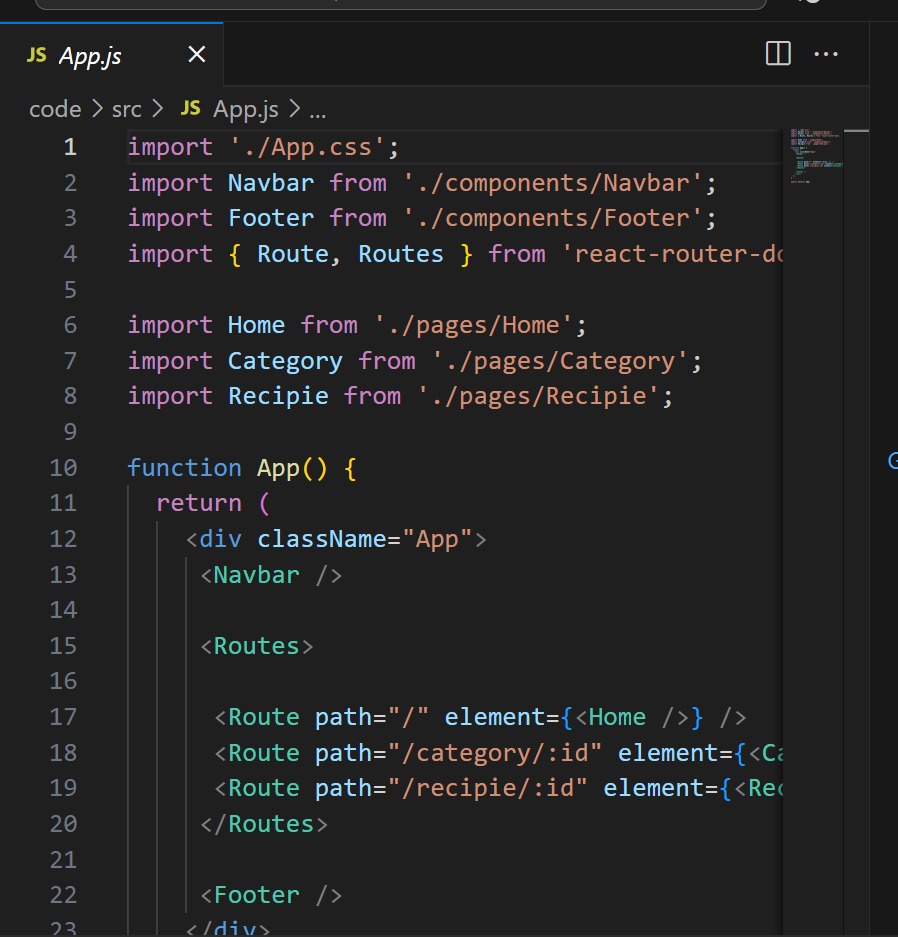
### **Error Handling** • Generic error messages displayed instead of descriptive ones. • Inconsistent validation feedback across forms.

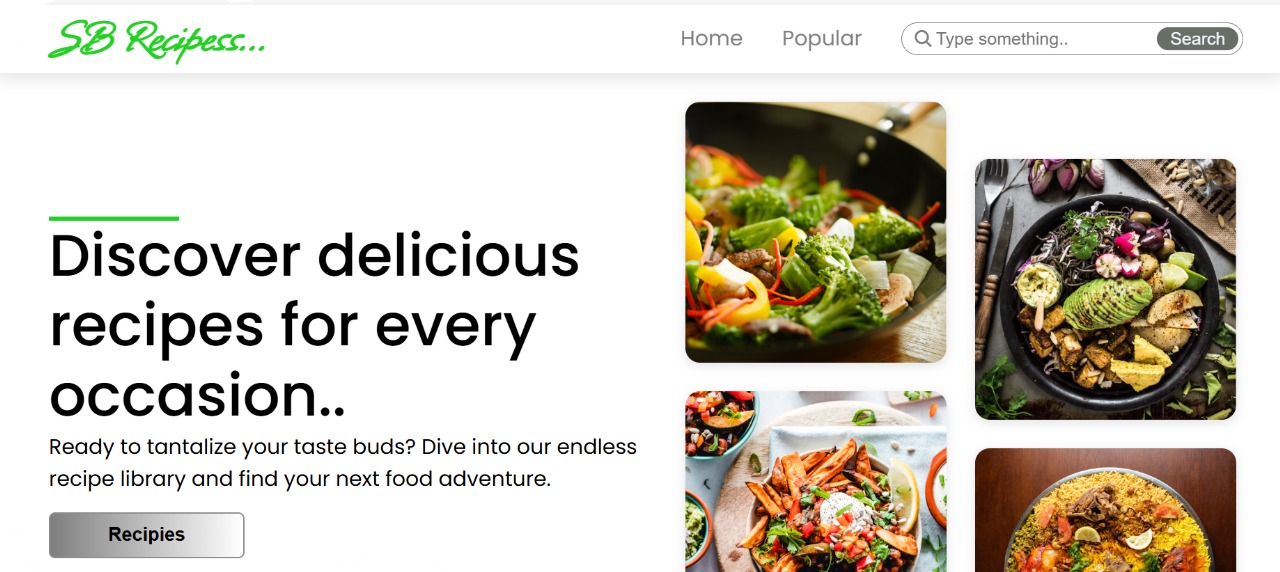
### **Deployment Issues** • Environment configuration varies between local and production. • Limited logging for debugging in production mode.

### **Testing Gaps** • Automated testing not fully integrated. • Manual testing needed for edge cases and high-traffic scenarios.

**SCREENSHOTS:**

****

****

****