**Project Documentation**

**Cook book : your virtual kitchen assistant**

# 1. Introduction

* **Project Title :** Cook Book your virtual assistant
* **Team ID**  : NM2025TMID29984
* **Team Leader :** SWETHA S
* **LEADER MAIL ID :**  [202400044@sigc.edu](mailto:202400044@sigc.edu)
* **Team Members:**

ROOBA P - [202400519@sigc.edu](mailto:202400519@sigc.edu)

RITHIKA SHREE V - [202400673@sigc.edu](mailto:202400673@sigc.edu)

SWATHY P - [202400822@sigc.edu](mailto:202400822@sigc.edu)

# 2. Project Overview

Purpose: The primary purpose of the Virtual Assistant Cookbook project is to develop an easily accessible, dynamic, and practical guide to creating, managing, and optimizing virtual assistants. This project aims to serve both technical developers and non-technical users who are interested in building or using AI-powered assistants for personal, business, or educational purposes.

* **Features:**

\* **Project Posting and Bidding:**

Users can post projects and place bids, enabling easy collaboration between clients and service providers.

\* **Secure Chat System:**

Built-in encrypted messaging for real-time communication between users, ensuring privacy and security.

\* **Feedback and Review System:**

Clients and service providers can leave ratings and reviews, fostering transparency and trust.

\* **Admin Control Panel:**

Admins can manage users, monitor activities, and control platform settings from a centralized dashboard.

# 3. Architecture

* Frontend:
* **User Interface (UI):**

\* A responsive web/mobile interface where users can post projects, bid, chat, and leave feedback.

\* Built with frameworks like React.js, Angular, or Vue.js for dynamic rendering.

* **Backend:**
* **API Layer:**

\* RESTful APIs to handle requests between the frontend and the server (Node.js, Django, or Laravel).

\* Manages authentication, project posting, bidding, and feedback submission.

* **Real-Time Communication:**

\* WebSocket-based service for the secure chat system (Socket.io, Firebase).

\* Ensures instant communication between users.

* **Database:**
* **Relational Database (SQL):**

\* Stores user data, project information, bids, reviews, and feedback (MySQL, PostgreSQL).

\* Structured tables for projects, bids, user profiles, etc.

# 4. Setup Instructions

* **Prerequisites:**

Before you begin, ensure that the following tools and services are installed and set up:

-Node.js : It is required to run the backend server and manage dependencies.

* **Installation:**
* Download and install Node.js from here.
* **Verify installation**:

- node -v

- npm -v

* **MongoDB**
* It is used to store recipe data, user profiles, and other application data.
* Install MongoDB by following the official MongoDB setup guide.
* Alternatively, you can use a cloud service like [MongoDB Atlas](https://www.mongodb.com/cloud/atlas).
* After installation, start the MongoDB service:

sudo systemctl start mongodb

sudo systemctl enable mongodb

* **Git**

It is necessary for version control and managing your source code.

* **Installation:**
* Download Git from [here](https://git-scm.com/).
* Verify installation:

- git --version

- React.js

 React.js is used to build the frontend of your cookbook application.

* If you don't have create-react-app installed, use the following:
  + npm install -g create-react-app
  + Express.js **–** Mongoose **–** Visual Studio Code

npm install -g create-react-app

-Create a new React project (or clone your existing repo):

create-react-app cookbook-app

Install necessary dependencies by navigating to the project directory:

cd cookbook-app

npm install

# 5. Folder Structure

# SB-Works/

# │

# ├── client/ # React frontend

# │ ├── components/ # Reusable UI components (e.g., buttons, forms)

# │ └── pages/ # React components representing different pages/views

# │

# └── server/ # Node.js backend

# ├── routes/ # API route handlers (e.g., /api/recipes, /api/users)

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

# └── controllers/ # Business logic for routes (e.g., createRecipe, getRecipes)

* **Folder Details:**
* **client/**: Contains all the frontend code.
  + **components/**: Holds reusable components (buttons, form fields, etc.).
  + **pages/**: Contains React components for different pages like Home, Recipe List, Recipe Detail, etc.
* **server/**: Contains the backend code (API and business logic).
  + **routes/**: Defines API endpoints and links them to controller functions.
  + **models/**: Defines the database schemas for MongoDB (e.g., Recipe model, User model).
  + **controllers/**: Contains the logic to handle requests (e.g., adding a new recipe, fetching recipes).

# 6. Running the Application

**Frontend:**

1. Navigate to the client directory:

cd client

2. Start the React frontend:

npm start

**Backend:**

1. Navigate to the server directory:

cd server

1. Start the Node.js backend:

npm start

**Access the Application:**

1.Once both the frontend and backend are running, open your browser and go to:

http://localhost:3000

# 7. API Documentation

**User**

* /api/user/register

**Projects**

* /api/projects/create
* /api/projects/:id

**Applications**

* /api/apply

**Chats**

* /api/chat/send
* /api/chat/:userId

# 8.AUTHENTICATION:

# JWT-based Authentication

* Description: The application uses JSON Web Tokens (JWT) to secure login and protect private routes.
* User Login: When a user logs in, the server validates their credentials.
* JWT Generation: Upon successful login, a JWT token is generated and sent to the client.

# 9. User Interface

**Landing Page**

* **Key Components**:
  1. Navigation Bar
  2. Hero Section
  3. Features Section

**Freelancer Dashboard**

* **Purpose**: The freelancer dashboard is the personal workspace for freelancers where they can manage their profile, applications, projects, and communication with clients.
* **Key Components**:
  1. Profile Overview
  2. My Projects
  3. Messages/Chat
  4. Notifications
  5. Earnings and Stats

Top of Form

Bottom of Form

# 10. Testing

**1. Unit Testing**

**Purpose**: Test individual components and backend functions.

**Tools**: Jest, React Testing Library (Frontend), Mocha/Chai (Backend)

**2. Integration Testing**

**Purpose**: Test how frontend and backend work together.

**Tools**: Cypress (Frontend), Super test (Backend)

**3. API Testing**

**Purpose**: Test API endpoints for CRUD operations.

**Tools**: Super test, Postman

**4. UI/UX Testing**

**Purpose**: Ensure a responsive, user-friendly UI.

**Tools**: Cypress, Lighthouse

**5. Performance Testing**

**Purpose**: Test app performance under load.

**Tools**: Lighthouse, JMeter

# Run Tests:

**Frontend**: npm test

**Backend**: npm test

**E2E (Cypress)**: npx cypress open

1. **Known Issues**
2. **Search Functionality**:
   * **Issue**: Search results may not always be accurate due to partial matches in the recipe database.
   * **Solution**: We are improving the search algorithm for better accuracy.
3. **User Authentication**:
   * **Issue**: Users may experience occasional delays when logging in or logging out due to session management.
   * **Solution**: Session handling improvements are being worked on.
4. **Performance**:
   * **Issue**: Loading times for recipe lists can be slower when there are a large number of entries.
   * **Solution**: Pagination and lazy-loading features are being implemented to improve performance.

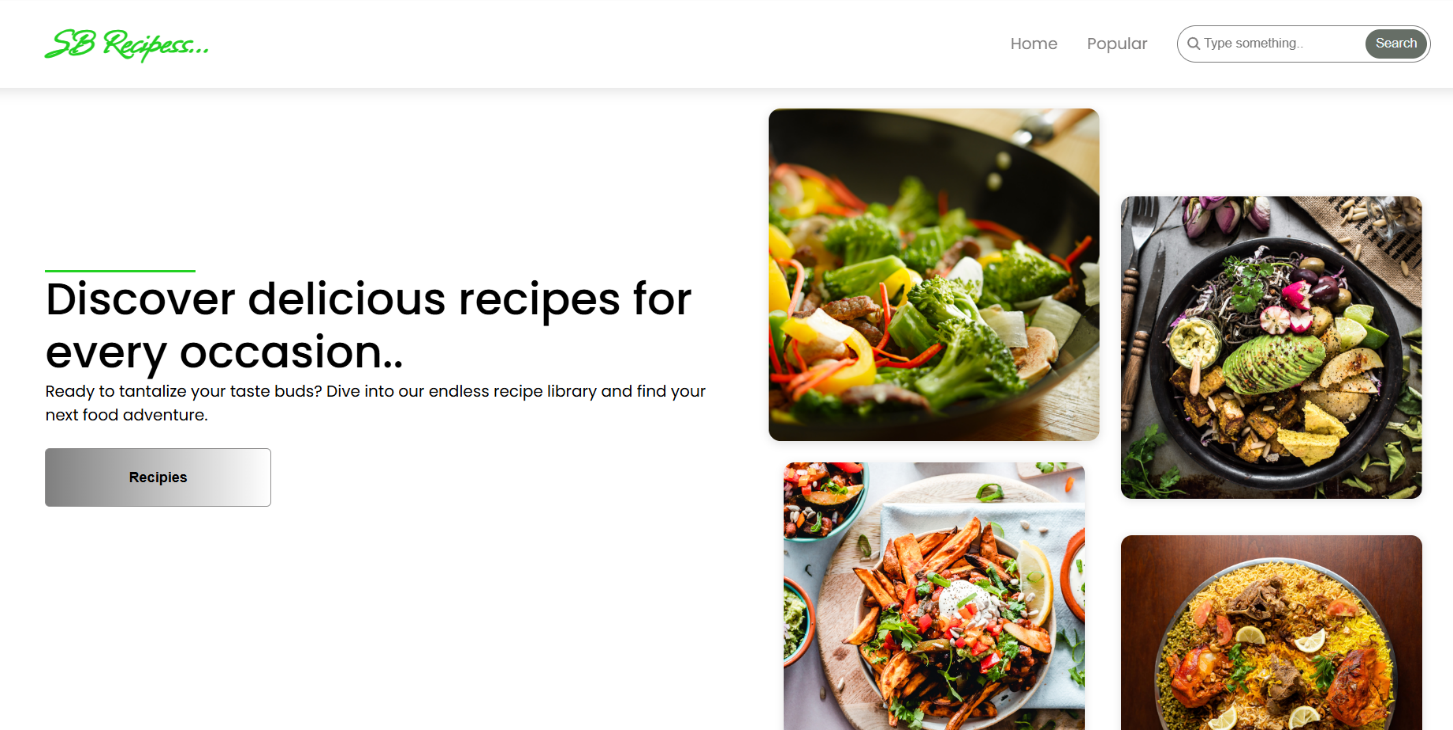
# 11. Future Enhancements

* **Advanced Search Filters**: Add filters for cuisine, diet, and cooking time.
* **User Ratings & Reviews**: Allow users to rate and review recipes.
* **Recipe Categories**: Organize recipes by categories (e.g., Desserts, Quick Meals).

# 13.Demo video link

<https://drive.google.com/file/d/1HKjp0DhScL49NtqvE9UPy1u4gV_Tj9TF/view?usp=drive_link>

# 12.ScreenShot



# 