

# **NAAN MUDHALVAN PROJECT REPORT**

**PROJECT TITLE: COOK BOOK –YOUR VIRTUAL KITCHEN ASSISTANT**

**TEAM ID:** NM2025TMID47172

## **TEAM LEADER:**

**NAME** : P.Ilakkiya (code developer)

**EMAIL\_ID:** [ilakkiyaprakash7@gmail.com](mailto:ilakkiyaprakash7@gmail.com)

## **TEAM MEMBERS:**

<b>NAME</b>	<b>MAIL ID</b>
1) S.Sandhiya (code developer)	<a href="mailto:sandhiya0420@gmail.com">sandhiya0420@gmail.com</a>
2) V.Deepajothi (documentation)	<a href="mailto:vadiveldinesh824@gmail.com">vadiveldinesh824@gmail.com</a>
3)S.karthika (demo video linking)	<a href="mailto:karthika8126@gmail.com">karthika8126@gmail.com</a>
4)P.Priya (documentation)	<a href="mailto:priyapragaspathi831@gmail.com">priyapragaspathi831@gmail.com</a>

## **2.PROJECT OVERVIEW:**

A **Cookbook Virtual Kitchen Assistant** is a smart, interactive application designed to assist users in the kitchen. It acts like a digital sous-chef, helping with recipe management, meal planning, grocery shopping, and step-by-step cooking guidance. The assistant can be voice-enabled, app-based, or integrated with smart home devices like Alexa, Google Assistant, or kitchen appliances.

Features :

1. Smart Recipe Search & Suggestions
  - Search recipes by ingredient, cuisine, dietary preference, or cooking time.
  - Get personalized suggestions based on past cooking history or available ingredients.
2. Step-by-Step Cooking Instructions (Voice & Visual)
  - Interactive, hands-free instructions with voice guidance.
  - Includes images, videos, timers, and tips for each cooking step.
3. Automatic Grocery List Generator
  - Creates a shopping list based on selected recipes or weekly meal plans.
  - Organizes items by category for easier shopping.
4. Meal Planning & Scheduling
  - Plan daily or weekly meals.
  - Sync meal plans with grocery lists and calendar reminders.

### **3.ARCHITECTURE:**

This architecture supports a scalable, modern web application that provides interactive recipe guidance, meal planning, and pantry management through an intuitive interface.

Frontend: React.js + Bootstrap + Material UI

#### *Role:*

The user interface that delivers a smooth, responsive, and interactive experience.

#### *Technologies Used:*

- React.js: Component-based structure for dynamic UI.
  - Bootstrap: Layout grid system, responsiveness, and basic styling.
  - Material UI: Modern, sleek UI components (buttons, cards, modals, etc.).
- Backend:**  
**Node.js + Express.js**

#### *Role:*

Handles business logic, API routing, user authentication, and connection with the database.

#### *Technologies Used:*

- **Node.js:** Event-driven, non-blocking backend runtime for handling high concurrency.
- **Express.js:** Lightweight framework to build RESTful APIs and manage server-side logic.

Database: MongoDB

Stores structured and unstructured data in flexible JSON-like documents.

[ React.js (Frontend) ]

|

| REST API Calls

↓

[ Node.js + Express.js (Backend) ]

|

| Mongoose Queries

↓

[ MongoDB (Database) ]

## **4.SETUP INSTRUCTIONS:**

Prerequisites:

- Node.js
- MongoDB
- Git
- React.js
- Express.js – Mongoose – Visual Studio Code

### Installation Steps

#### **1. Clone the Repository**

```
git clone <your-repo-url>  
cd <repo-folder-name>
```

#### **2. Install Client Dependencies**

```
cd client  
npm install
```

### 3. Install Server Dependencies

```
cd ../server  
npm install
```

### Start the Application

#### Start Client (Frontend):

```
bash
```

```
npm start
```

#### Start Server (Backend):

```
cd server
```

```
npm start
```

### 5. FOLDER STRUCTURE:

#### COOK BOOK

```
|  
|  
| — public/          # Static files (not processed by build tools)  
| | — index.html     # Main HTML file  
| | — favicon.ico    # Site icon  
| | — assets/        # Images, fonts, videos (static)  
| |  
| — src/             # Main source code  
| | — assets/        # Processed assets (images, fonts, svgs, icons)  
| | | — images/  
| | | — fonts/  
| | | — styles/      # Global styles (CSS/SCSS)  
| |  
| | — components/    # Reusable UI components
```

```
| | | └─ Button.jsx
| | | └─ Navbar.jsx
| |
| | └─ pages/          # Page-level components (Home, About, etc.)
| | | └─ Home.jsx
| | | └─ About.jsx
| |
| | └─ layouts/        # Layouts for wrapping pages (Header/Footer)
| | | └─ MainLayout.jsx
| |
| | └─ hooks/          # Custom React hooks (if using React)
| | | └─ useAuth.js
| |
| | └─ services/       # API calls or external services
| | | └─ api.js
| |
| | └─ context/        # Context API or global state (React/Vue)
| | | └─ AuthContext.jsx
| |
| | └─ utils/          # Helper functions
| | | └─ formatDate.js
| |
| | └─ App.js          # Root component
| | └─ index.js        # Entry point
| | └─ routes.js       # Route definitions (if needed)
```

|

|— .gitignore           # Files ignored by Git

|— package.json        # Dependencies & scripts |— README.md        # Project documentation

|— vite.config.js / webpack.config.js / next.config.js (depending on framework)

## 6. RUN THE APPLICATION:

### frontend

cd client

npm start

### backend

cd server

npm start

Access: visit <http://localhost:3000>

## 7. COMPONENT DOCUMENTATION:

### Key Components:

- RecipeList: Displays recipe cards fetched from API. Props: recipes[].
- RecipeDetail: Shows full recipe details. Props: recipeId.

### Reusable Components:

- RecipeCard – used in RecipeList and Favorites.
- Button – styled button component used throughout the app

## **8. STATE MANAGEMENT:**

### **Global State:**

Managed by RecipesContext for recipes, favorites, and user login status.

### **Local State:**

Form input states managed inside AddRecipeForm.

## **9. USER INTERFACE:**

### **Include screenshots or GIFs of:**

- Home page showing recipes
- Recipe detail page
- Adding a recipe

## **10. STYLING:**

### **CSS Frameworks/Libraries:**

Tailwind CSS for styling; Styled Components for scoped styles.

### **Theming:**

Dark and light mode toggle implemented via context.



## **11. TESTING:**

### **Unit testing:**

Testing individual components or functions in isolation to ensure they work correctly.

### **Integration testing:**

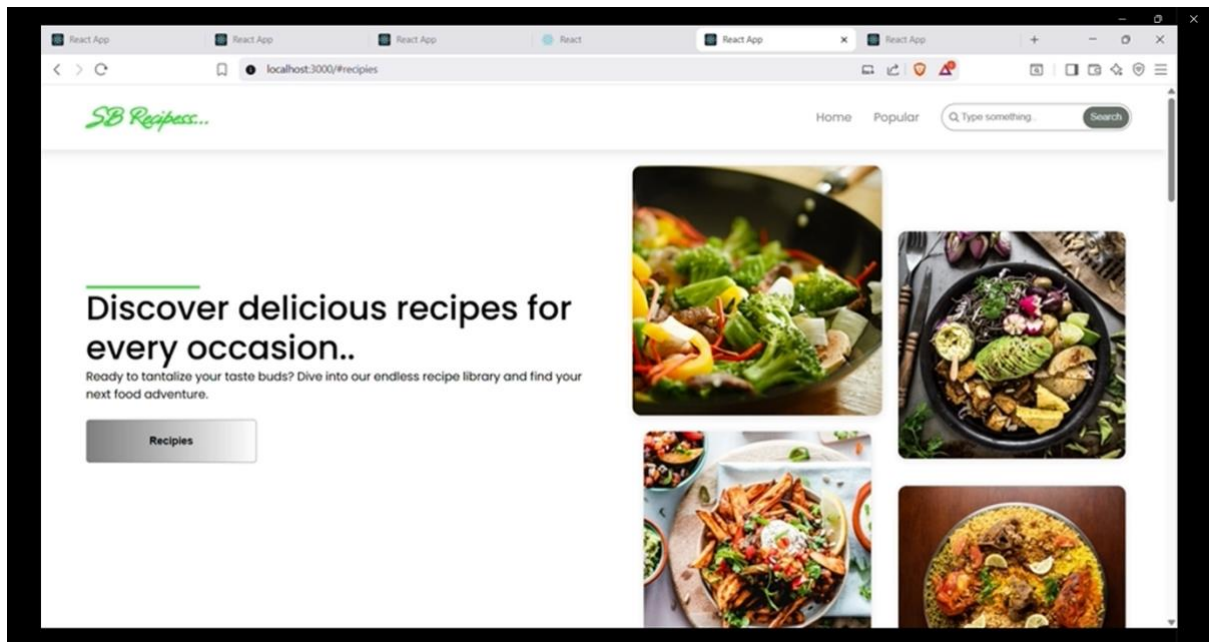
Testing how different components or modules work together as a whole

## **12. SCREENSHOTS OR DEMO:**

### **Add actual screenshots or a demo link:**

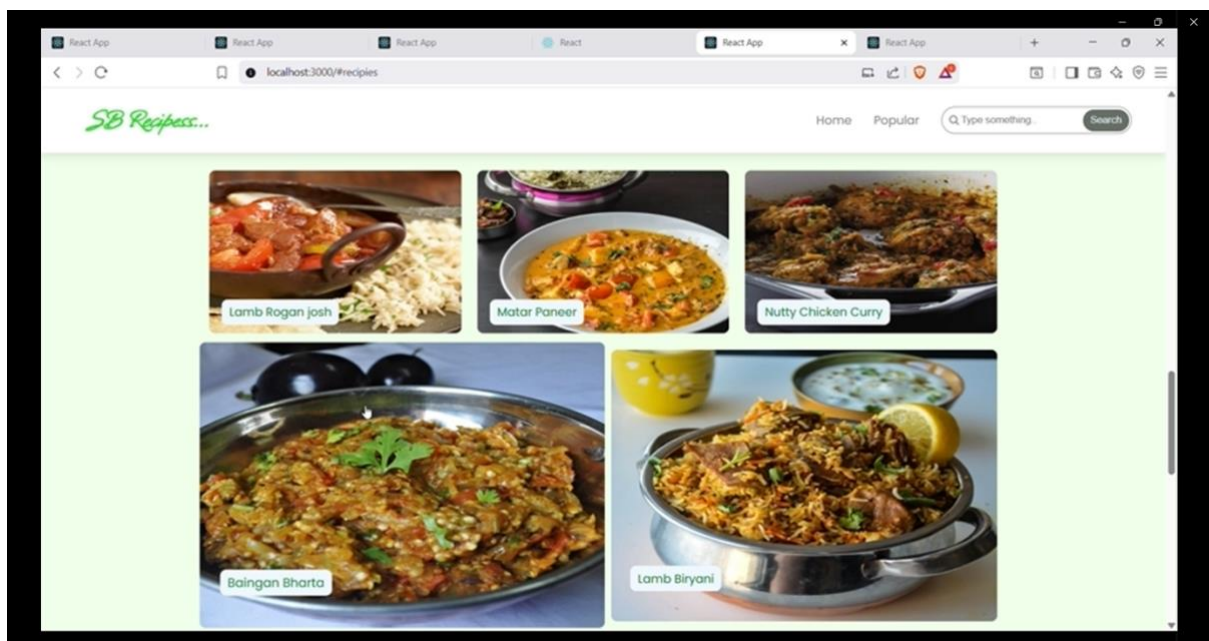
### **Hero components:**

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



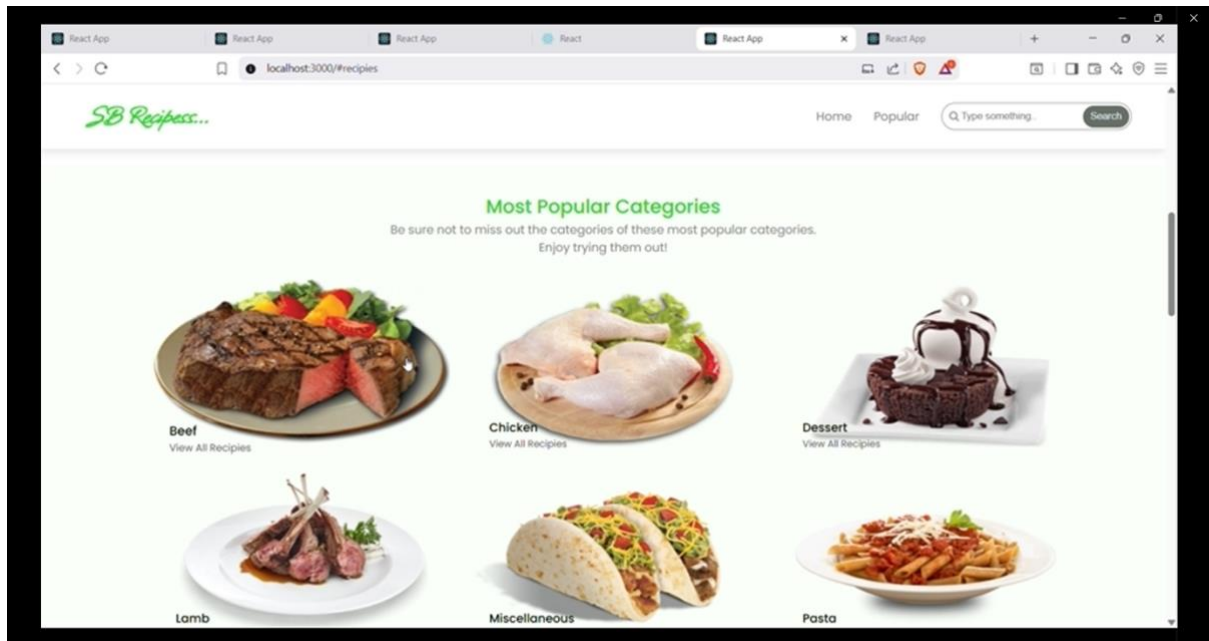
## Trending Dishes :

This component contains some of the trending dishes in this application



## Popular categories :

This component contains all the popular categories of recipes..



### - Demo Link:

<https://drive.google.com/file/d/13hEdzu6G7NDgwELG16bbygyrTbcXLIR/view?usp=drivesdk>

### **13. KNOWN ISSUES :**

Search filtering may be slow with very large recipe lists.

Image upload sometimes fails on slow networks.

### **14. FUTURE ENHANCEMENTS :**

Add shopping list generation from recipe ingredients

Introduce push notifications for new recipes

Offline mode for saved recipes