NAAN MUDHALVAN PROJECT

_____

COOKBOOK-YOUR VITRUAL KITCHEN ASSISTANT

1. Introduction

Project Title:Cook book

Team ID:NM2025TMID47120

Team Leader:K.Namitha

Email Id:namitha179@gmail.com

Team members:

1.K.Namitha-namitha179@gmail.com -Document making

2.T.Srividhya–srisrividhya2007@gmail.com –Document making

3.S.Jayamalini –jaimalini53@gmail.com -Demo vedio maker

4.R.Harini-harinihari7319@gmail.com-Coding developer

5.S.Mageshwari-mageshwari07042007@gmail.com-Editing and Digital Setup.

2. Project Overview

Purpose:
- SmartCook is a React.js-based online cookbook application that allows users to browse,
- save, and share recipes. It aims to make cooking easier by providing detailed recipes,
- step-by-step instructions, and personalized recommendations.

Features:
- Browse recipes by category, cuisine, or ingredients
- Save favourite recipes to personal profile
- Add and edit user-submitted recipes
- Responsive UI for mobile and desktop

3. Architecture

Component Structure:
- Header (navigation bar)
- RecipeList (list of recipes)
- RecipeDetail (individual recipe page)
- Favourites (user's saved recipes)
- AddRecipeForm (form to add new recipes)
- Footer

State Management:
- Using React Context API to manage global state for recipes and user favourites.
- Routing:
- React Router v6 handles navigation between Home, Recipe Detail, favourites, and Add Recipe
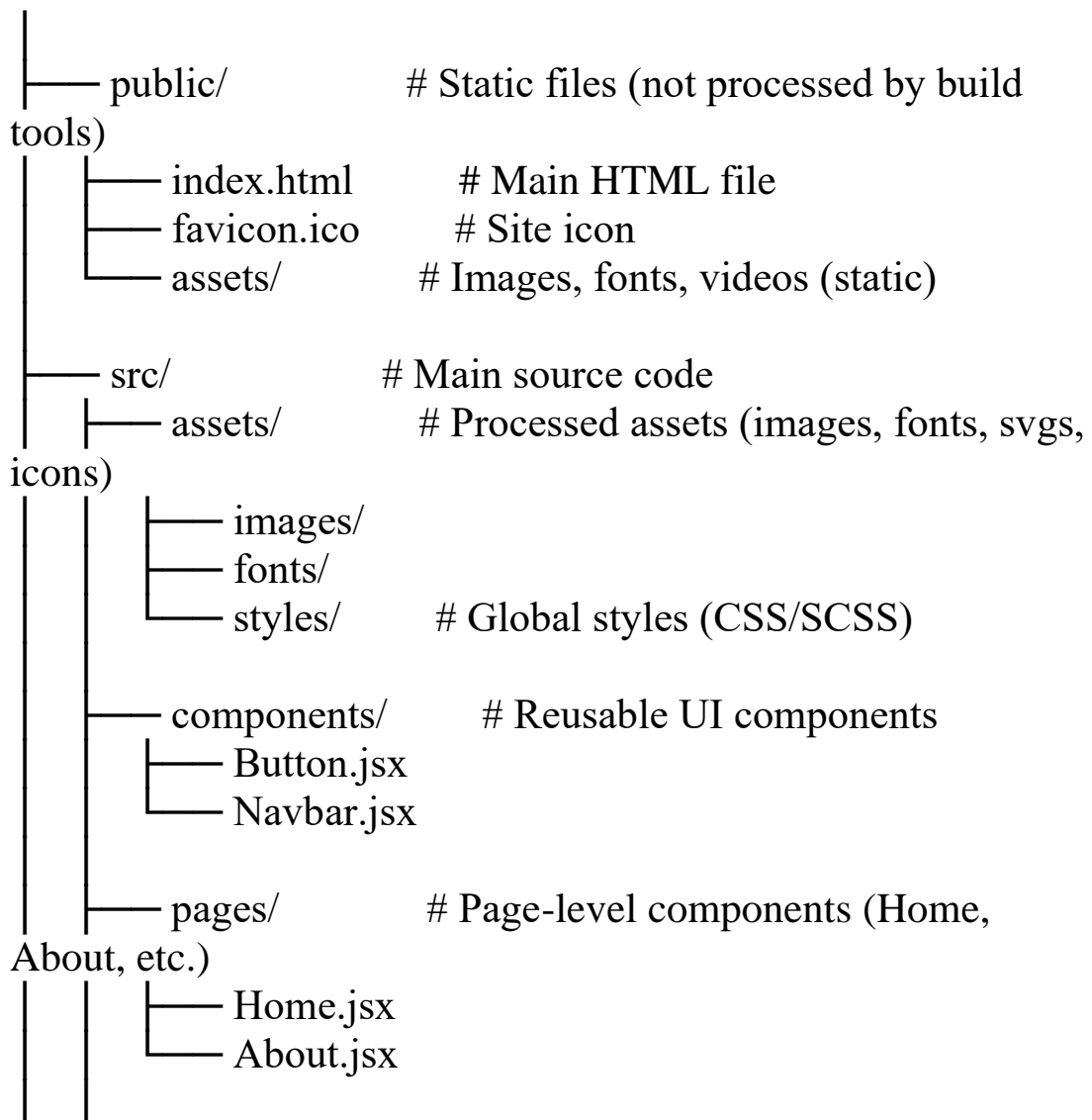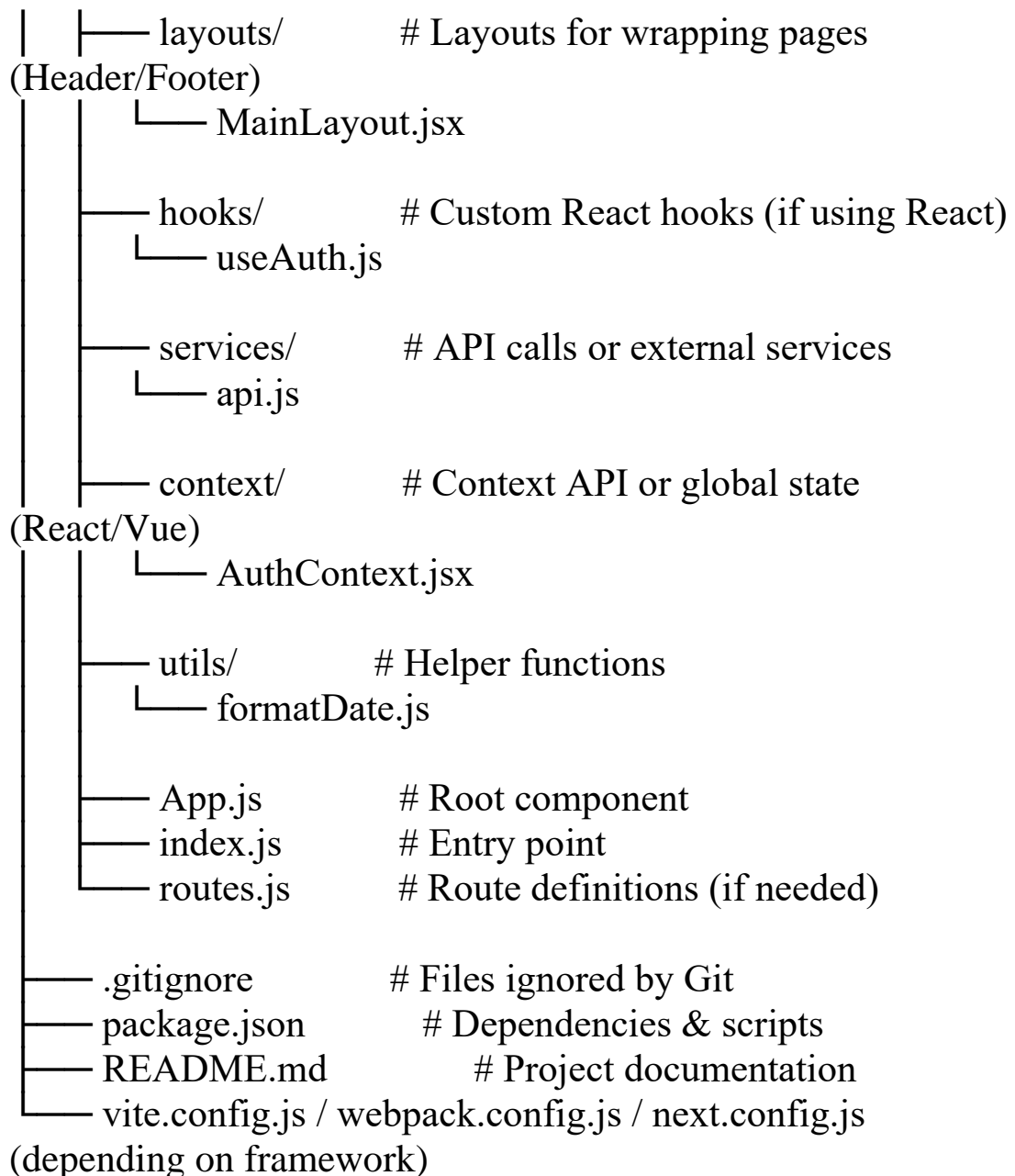- Pages

4. Setup Instructions

Prerequisites:

- Node.js
- MongoDB
- Git
- React.js
- Express.js – Mongoose – Visual Studio Code
- Installation Steps
- Clone the Repository

5. Folder Structure

Cookbook:virtual kitchen Assistant

```
Cookbook:virtual kitchen Assistant
│
├── 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. Running the Application

- Frontend:cd client
- npm start
- Backend:cd server npm start
- Access: 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:
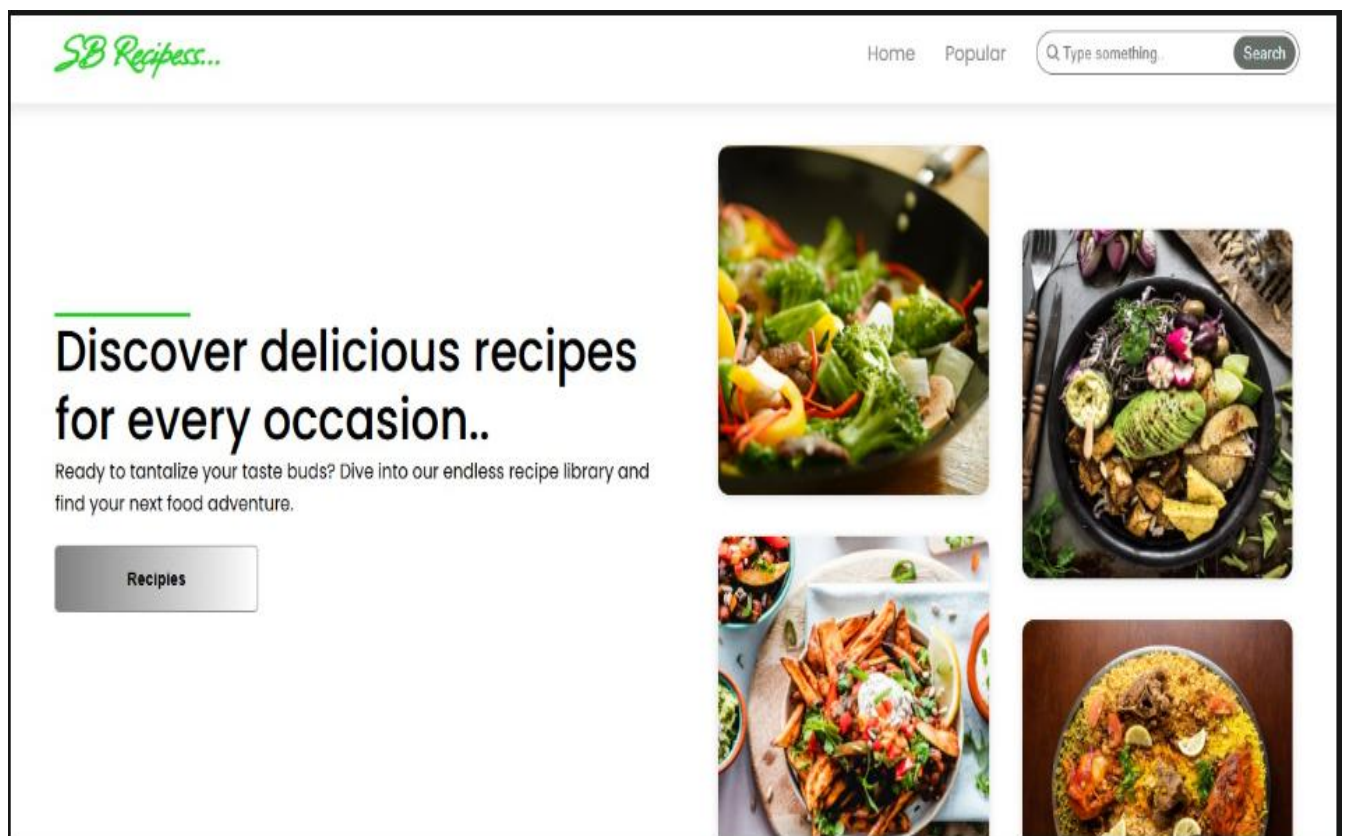- Dark and light mode toggle implemented via context.
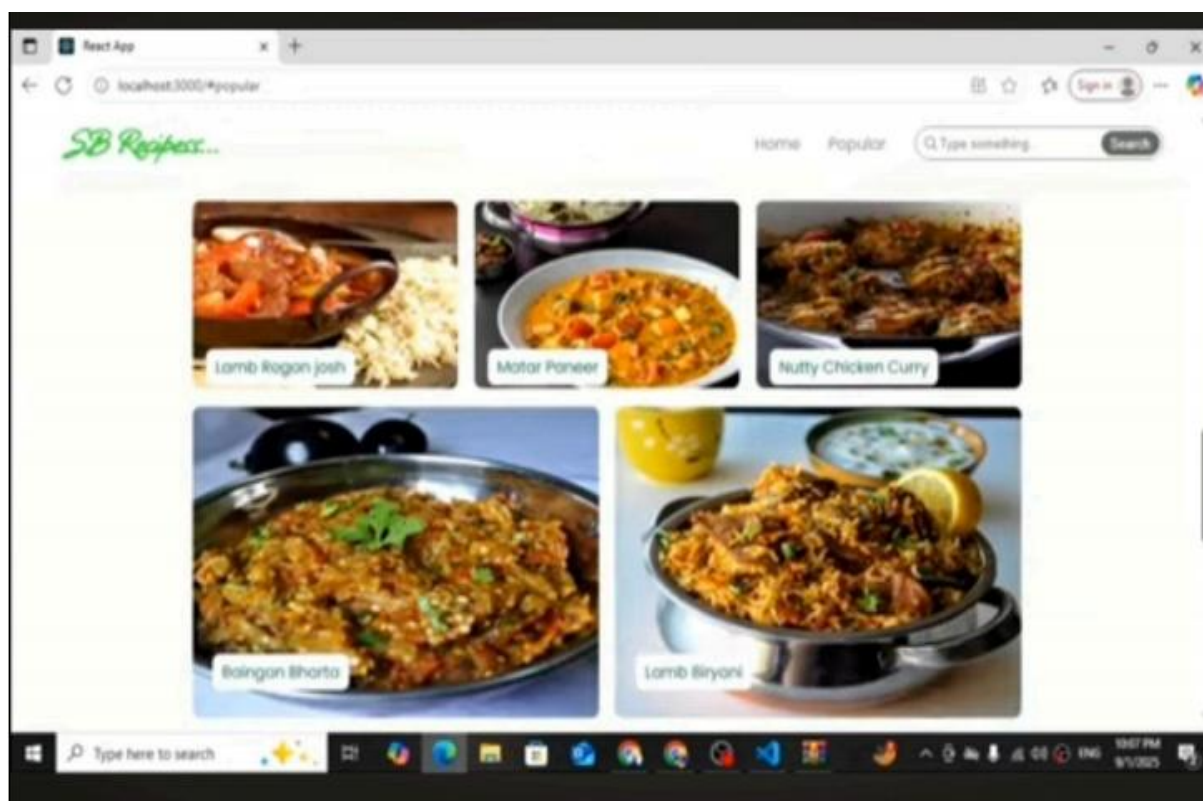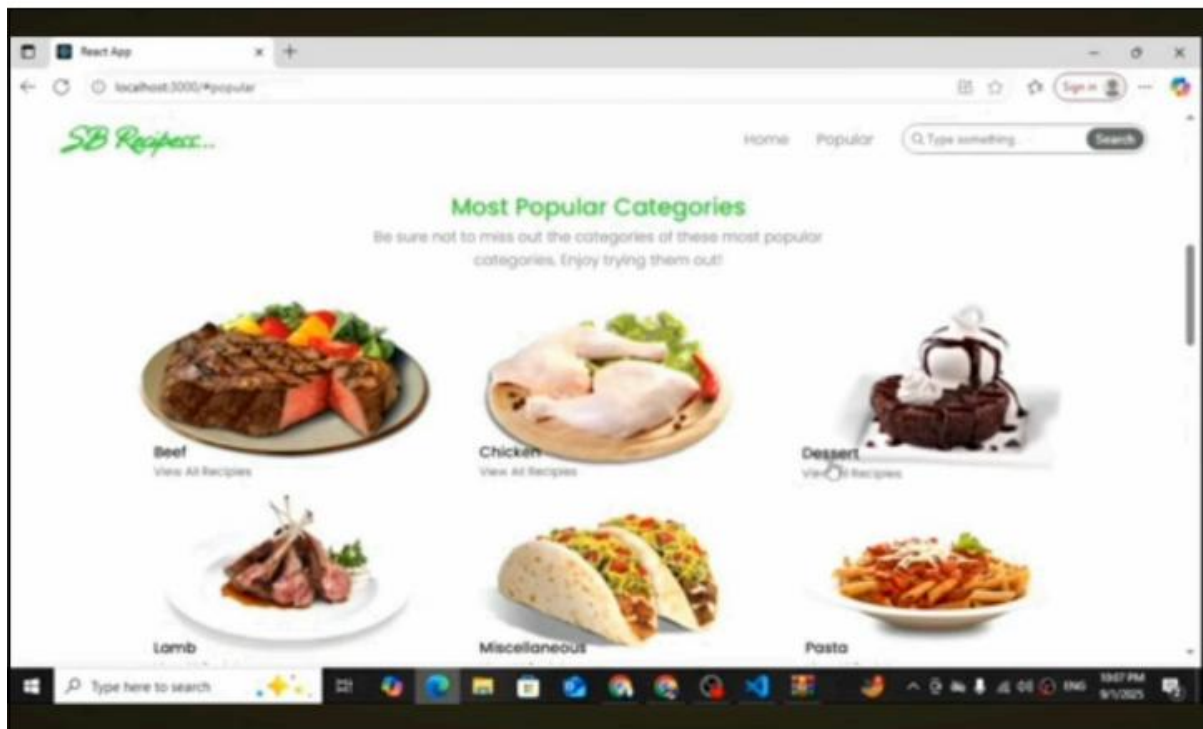
11. Testing

Testing Strategy:

- Jest + React Testing Library for unit and integration tests.

Code Coverage:

- Using jest --coverage to ensure 80%+ coverage

12. Screenshots or Demo

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