Frontend Development with Node.js

Project Documentation format

---

1. Introduction

- **Project Title:** Cook Book •
  Team Members:
  - [Devadharshini R] - [Project Manager] - dharshiniraja344@gmail.com
  - [Sudharsan K] - [Frontend Developer] - sudharsank2009@gmail.com
  - [Kumaraguru S] - [UI/UX Designer] - guruguru5310@gmail.com
  - [Dilliganesh V] – [Backend Developer] - dilliganesh054@gmail.com
  - [Aravindh R] – [Quality Assurance] - aravindhgpm07@gmail.com

---

2. Project Overview

- **Purpose:**
  The purpose of the Cook Book project is to provide users with an interactive platform to explore and share recipes, manage cooking instructions, and save favorite dishes. It allows users to add new recipes, rate dishes, and create a personalized cookbook.
- **Features:**
  ○ Recipe search and filter based on ingredients or categories. ○ Ability to create, update, and delete recipes. ○ User authentication and profile management.
  ○ Interactive rating and comment system for each recipe. ○ Save favorite recipes to a personal list.

---

3. Architecture

- **Component Structure:**
  The React application is organized into several components, such as: ○ **App:** The main container for routing and the global

state. o **Header:** Displays navigation links and user profile info.
- **RecipeList:** Displays a list of recipes. o **RecipeDetail:** Displays detailed information for a selected recipe.
- **Auth:** Manages user authentication and profile.

- **State Management:**
  We use the **Context API** to manage global state for authentication and user data. Local state within components is managed using React's `useState` hook.
- **Routing:**
  We use **React Router** for navigation between pages like the home page, recipe list, and recipe details page. The routing structure is as follows:
  - `/`: Home page. o `/recipes`: Displays the list of recipes. o `/recipes/:id`: Displays the details of a specific recipe. o `/login`: User login page.

---

4. Setup Instructions

- **Prerequisites:**
  - Node.js (version 14.x or higher)
  - npm (version 6.x or higher)
- **Installation:**
  1. Clone the repository: `git clone https://github.com/username/cookbook.git`
  2. Navigate into the project directory: `cd cookbook`
  3. Install dependencies: `npm install`
  4. Set up environment variables:
     - ✦ Create a `.env` file in the root directory and add any necessary API keys or configurations.

---

5. Folder Structure

- **Client:**
  The React app is structured as follows:
  - **/src/components**: Contains reusable components (e.g., Button, RecipeCard). o **/src/pages**: Contains page components (e.g., HomePage, RecipeDetailPage). o **/src/assets**: Stores static assets such as images and icons.
  - **/src/utils**: Includes utility functions, custom hooks (e.g., useAuth), and API requests.
- **Utilities:**

Helper functions include functions for API calls (`api.js`) and form validation utilities (`validate.js`).

---

6. Running the Application

- **Frontend:**
  To run the frontend server locally, navigate to the `client` directory and run:
- `npm start`

---

7. Component Documentation

- **Key Components:** o **RecipeCard:** Displays basic information about each recipe.
  - ✦ **Props:** `title`, `image`, `id`, `onClick` o **RecipeDetail:** Displays detailed information about a selected recipe.
  - ✦ **Props:** `recipeId`, `details`
- **Reusable Components:**
  - **Button:** A reusable button component.
    - ✦ **Props:** `label`, `onClick`, `type` o **InputField:** A reusable input component for forms.
    - ✦ **Props:** `label`, `type`, `value`, `onChange`

---

8. State Management

- **Global State:**
  The global state is managed using the Context API, which handles authentication and user-related data (e.g., user info, token). The global state is accessed via the `useAuth` hook.
- **Local State:**
  Local states for individual components are managed with the `useState` hook, such as the state for form inputs and toggleable UI elements.

---

9. User Interface

- **Screenshots/Demo:**
  Include images or GIFs to showcase different UI features:
  - Home page showing recipe search.

    - Recipe detail page with ingredients and instructions. ○ User profile page with saved recipes.

---

10. Styling

- **CSS Frameworks/Libraries:**
  We use **Styled Components** for styling. It allows for writing CSS-in-JS and provides component-level styling.
- **Theming:**
  A custom theme is implemented to handle consistent colors, fonts, and spacing. A `theme.js` file is used to define these properties.

---

11. Testing

- **Testing Strategy:**
  We use **Jest** and **React Testing Library** for unit and integration tests. For example, we test that components render correctly and user interactions work as expected.
- **Code Coverage:**
  We ensure code coverage through Jest's built-in coverage tool, which provides insights into how much of the code is covered by tests.

---

12. Screenshots or Demo

- Provide a link to a live demo or include images to demonstrate key features and the overall look of the application.

---

13. Known Issues

- **Bug 1:** Sometimes, recipe details are not displayed correctly if there's a delay in API response.
- **Bug 2:** User login sometimes fails if the backend server is slow or down.

---

14. Future Enhancements

- **Features to Add:**
    - Implement a rating system for recipes (1-5 stars). ○ Allow users to upload images for their recipes.

    - Implement search filters based on cuisine or difficulty level.
- **Improvements:** ○ Enhance styling for mobile responsiveness. ○ Add more user-friendly error handling and loading indicators.
