Frontend Development with React.js

Project Documentation format

1. Introduction
    - Project Title:CookBook – Virtual Kitchen Assistant
    - Team Members:
    - Kruthi.S-code execution
    - Divya.S-documentation
    - Divya dharshini-demo video
    - Hemalatha-demo video
2. Project Overview
    - Purpose:
    - CookBook is a web application that helps users discover, organize, and cook recipes easily. It acts as a virtual kitchen assistant by providing personalized recommendations, step-by-step instructions, and ingredient management.
    - Features:
    - Search and filter recipes by ingredients, cuisine, or dietary preferences.
    - Step-by-step cooking instructions with timers.
    - Save favorite recipes and create custom collections.
    - Suggest meals based on ingredients you already have.
    - Responsive design optimized for desktop, tablet, and mobile.
3. Architecture
    - Component Structure:
    - Header – App logo, search bar, navigation links.
    - RecipeList – Displays recipe cards fetched from the API.
    - RecipeDetails – Shows ingredients, steps, and nutrition info.
    - PantryManager – Manage available ingredients.
    - Favorites – User's saved recipes.
    - Timer – Integrated cooking timer component.
    - Footer – Links and copyright.
    - State Management:
    - Used Context API with useReducer for global state.
    - Maintains authentication status, favorites, pantry items, and theme mode.
    - Routing:
    - Implemented with React Router v6.
    - Routes: / (Home), /recipe/:id, /favorites, /pantry, /about.
4. Setup Instructions
    - Prerequisites:
    - Node.js ≥ 18
    - npm or yarn
    - Git
    - Installation:
    - git clone https://github.com/yourusername/cookbook.git
    - cd cookbook

- npm install
- npm start
- Create a .env file with API keys (e.g., for a recipe API).
- Configure REACT_APP_API_KEY and REACT_APP_API_URL.

5. Folder Structure
   - Client:
   - /src
   - /assets → Images, icons
   - /components → Reusable UI components (Button, Timer, Card)
   - /pages → Home, RecipeDetails, Favorites, Pantry
   - /context → Global state providers
   - /utils → API helpers, custom hooks
   - index.js
   - App.js
   - Utilities:
   - useFetch.js – Custom hook for API calls.
   - formatTime.js – Helper to format timers.
6. Running the Application
   - Frontend:
   - npm start
   - Runs the app locally on http://localhost:3000.
7. Component Documentation
   - Key Components:
   - RecipeCard: Displays recipe image, title, and cooking time.
   - Reusable Components:
   - Button, Modal, Spinner, Timer.
8. State Management
   - Global State:
   - Managed with Context API for favorites, pantry, and user settings.
   - Reducers handle actions like ADD_FAVORITE, REMOVE_FAVORITE, ADD_PANTRY_ITEM.
   - Local State:
   - Components use useState for form inputs, search queries, and timers.
9. User Interface

10. Styling

- CSS Frameworks/Libraries:
- Tailwind CSS for utility-first styling.
- Styled-Components for theme-based styling.
- Theming:
- Supports light/dark mode with theme context.

11. Testing

- Testing Strategy:
- Unit tests for components using React Testing Library and Jest.
- Integration tests for routing and state flow.
- End-to-end tests with Cypress for recipe search and saving.
- Code Coverage:
- Jest coverage reports ensure ≥80% test coverage.

12. Screenshots or Demo

13. Known Issues

- Slow loading for very large recipe images on low-end devices.
- Timer notification sound may not play on Safari due to autoplay restrictions.

14. Future Enhancements

- Add voice guidance for hands-free cooking.
- Allow users to upload their own recipes.
- Integration with smart kitchen devices.
- Offline mode for saved recipes.