**COOK BOOK**

**Introduction:**

Project Title: [cook book ]

Team Members: [Ayesha,Gudesh,Nasrin,Thasilma]

* M.Ayesha (documentation,video editing,collect recipe)
* P. Gudesh (Frontend coding and collect recipe)
* N. Nasrin (Frontend coding and collect recipe)
* S.Thaslima (documentation,video editing,collect recipe)

**1.Project Overview:**

* **Purpose:**

The purpose of this Cookbook project is to provide an interactive and user-friendly web platform where users can explore, share, and organize recipes. The goal is to make cooking easier by offering categorized recipes, search functionality, and a responsive design for all devices.

* **Features:** 
  + Responsive Design -- Works on all screen sizes.
  + Navigation Menu --Includes categories like Veg, Indian, Chinese, Korean, About Us.
  + Search Bar -- Quickly find recipes.
  + Recipe Cards -- Display dish image, name, and a short description.
  + Filter/Sort Options --Sort recipes by cuisine, popularity, or preparation time.
  + Interactive UI Elements--Buttons, hover effects, and animations for a modern feel.

**2. Architecture**:

This project is built using HTML, CSS, JavaScript, React, and Node.js. The structure is modular, with reusable components and a clean layout. Navigation between sections is handled manually through buttons, links.

**\*Component Structure:**

I divided the website into separate parts to keep everything clean and easy to manage. The top section has a navigation bar with a centered logo and a search button.

Below that, there's a panel with links for Home, Recipe, Veg, Non-Veg, About Us, and Sign In. The hero section includes a background image and a welcome note to introduce the site. Each section is built separately so I can update or style them without affecting the rest of the layout

**\*State Management:**

The project links pages manually using direct links and basic logic to control what content is displayed. Sections such as Home, Veg, Non-Veg, and Recipe are connected through buttons or links, so when a user clicks, the corresponding content is loaded.

**3.Setup Instructions:**

**\*Prerequisites:**

Before starting, make sure the following tools are installed on your system:

* Node.js
* npm (comes with Node.js)
* Git
* A code editor like VS Code
* A browser (Chrome or any modern browser)

**\*Installation:**

* Clone the Repository Open your terminal and run: git clone

(https://github.com/your-username/your-repo-name.git)

* Go to the Project Folder cd your-repo-name
* Install Dependencies npm install
* Start the Server Run: node server.js or npm start (depending on your setup) • Configure Environment Variables (if needed) Create a .env file and add:

Code

(PORT=3000

DB\_URL=your-database-url)

* View the Site Open your browser and go to: [(http://localhost:3000)](http://localhost:3000/)

**4.Folder Structure:**

**\*Client**:

The project is structured into clear folders to keep everything modular and easy to manage.

* components/ – Contains reusable UI parts like navbar, panel, hero section, recipe cards, and footer.
* pages/ – Includes full-page layouts such as Home, Recipe, Veg, Non-Veg, About Us, and Sign In.
* assets/ – Stores images, icons, and other media files.
* styles/ – Holds CSS files for consistent styling across the site.

**5. Running the Application:**

**Frontend:**

To run the frontend locally, open your terminal, navigate to the project’s client directory,

(cd client)

npm start

**6.Component Documentation:**

**\*key compent:**

Nav bar – Displays the logo, search button, and navigation links. It helps users move between pages.

panel– Contains quick-access buttons like Home, Recipe, Veg, Non-Veg, About Us, and Sign In.

Hero Section – Shows a background image and welcome note. It sets the tone for the homepage.

Recipe Card – Displays recipe details like title, image, and category. Props include title, image, type, and description.

\***Reusable Components:**

Button – A styled button used across panels and forms. Props include label, onClick, and type.

Input Field – Used in search and sign-in forms. Props include placeholder, value, and onChange.

**7.State Management:**

**\*Global State:**

The project uses React’s Context API to manage global state, storing shared data like selected categories or search queries. This allows components such as the NavBar, Panel, and RecipeList to access and update the same data without passing props through multiple levels.

**\*Local State:**

Local state is handled directly within each component using simple variables and basic logic to manage component-specific data, such as controlling which content is displayed in sections like Home, Veg, Non-Veg, or Recipe.

**8. Interface:**

The project features a clean and user-friendly interface. The NavBar allows easy navigation, the Panel organizes sections like Home, Veg, Non-Veg, Recipe, About Us, and Sign In, and the HeroSection displays a prominent banner image. The WelcomeNote greets users, and recipe previews are shown using RecipeCard components. You can include screenshots or GIFs of different pages, search functionality, filters, and interactions to showcase the design and usability of the application.

**9.Styling:**

**\*CSS Frameworks/Libraries:**

The project uses plain CSS to style all components, including layout, fonts, buttons, and responsive design. No external CSS frameworks are used, keeping the design lightweight and simple.

**\*Theming:**

The Cookbook project follows a red and white theme, with red used for highlights, buttons, and important elements, and white as the primary background color. Black is used for text to ensure readability. This consistent color scheme gives the application a clean and attractive look while reinforcing the cookbook branding.

**10.Testing:**

Manual testing is performed by navigating all pages, clicking buttons, using the search feature, and verifying that content displays correctly. This ensures the application functions smoothly and provides a good user experience.

**\*Unit testing**

focuses on testing the smallest parts of code, such as functions, methods, or classes, in complete isolation. The purpose is to ensure that each unit performs its logic correctly without depending on external systems like databases or APIs.

**\*integration testing**

These tests are usually slower and more complex because they involve multiple parts of the system, but they are crucial for catching issues that only appear when components interact. In short, unit tests validate correctness of individual pieces, while integration tests validate the cooperation between those pieces.

.

**11.Screenshots :**











