***Frontend Development with React.js***

***Project Documentation for CookBook***

**1.** **Introduction**

• **Project** **Title: CookBook**

• **Team** **Members**:

**Syed Ali Aasin J** (**Team** **Leader**)

**Swetha S**

**Swetha** **S**

**Swetha R**

**Sundarakamatchi I**

[Email Id: [syedaliaasinj22cs088@gmail.com]](mailto:syedaliaasinj22cs088@gmail.com])

[Email Id: [swethas22cs086@gmail.com]](mailto:swethas22cs086@gmail.com])

[Email Id: [swetha22cs087@gmail.com]](mailto:swetha22cs087@gmail.com])

[Email Id: [swetha.r22cs085@gmail.com]](mailto:swetha.r22cs085@gmail.com])

[Email Id:[sundarakamatchi.i22cs083@gmail.com]](mailto:sundarakamatchi.i22cs083@gmail.com])

**2.** **Project** **Overview**

• **Purpose**:

A Cookbook Recipe Application is a digital platform that allows users to explore, save, and manage various recipes. Here are a few key points about such an application:

• **Features**:

o Recipe Library.

o Ingredient List.

o User authentication.

o Step by Step Cooking Instruction.

o Detailed Procedure.

**3.** **Architecture**

• **Component** **Structure**:

The application is built using React.js with a component-based architecture. Major components include:

o **Header**: Contains the navigation bar and search bar.

o **Sidebar**: Displays user playlists and navigation links.

o **HomePage**: Most Popular Categories, and Trending dishes.

o **SearchPage**: Allows users to search for recipe, ingredient, and video.

• **State** **Management**:

The application uses **Redux** for global state management. The Redux store manages user authentication, recipe, ingredients, and search results.

• **Routing**:

The application uses **React** **Router** for navigation. Routes include:

o /: Home page

o /search: Search page

o /login: User login page

**4.** **Setup** **Instructions**

• **Prerequisites**:

o Node.js (v16 or higher)

o npm (v8 or higher)

o Git

• **Installation**:

1. Clone the repository: git clone : <https://github.com/Syed1284/CookBook_SyedAliAasin>

2. Install dependencies: npm install

3. Configure environment variables: Create a .env file in the client directory and add the necessary variables (e.g., API keys).

4. Start the development server: npm start

**5.** **Folder** **Structure**

• **Client**:

o **src/components:** # Reusable components (Header, Player, etc.) o **src/pages:** # Page components (HomePage, SearchPage, etc.) o **src/assets:** # Images, icons, and other static files

o **src/redux:** # Redux store, actions, and reducers o **src/utils:** # Utility functions and helpers

o **App.js:** # Main application component o **index.js:** # Entry point

• **Utilities**:

o **api.js**: Handles API requests to the backend.

o **auth.js**: Manages user authentication and token storage.

o **hooks/usePlayer.js**: Custom hook for managing the music player state.

**6.** **Running** **the** **Application**

**Frontend**:

o To start the frontend server, run the following command in the client directory: npm start

o npm install

o npx json-server ./db/db.json o npm run dev

o The application will be available at http://localhost:3000

**7.** **Component** **Documentation**

• **Key** **Components**:

o **Header**: Displays the navigation bar and search bar.

▪ Props: onSearch (function to handle search queries).

o **Player**: Controls the video.

▪ Props: currentTrack (object containing track details), onPlay, onPause, onSkip.

o **Playlist**: Displays a playlist with its name and cover image.

▪ Props: playlist (object containing playlist details), onClick (function to handle playlist selection).

• **Reusable** **Components**:

o **Button**: A customizable button component.

▪ Props: text, onClick, disabled.

o **Input**: A reusable input field for forms and search.

▪ Props: type, placeholder, value, onChange.

**8.** **State** **Management**

• **Global** **State**:

The Redux store manages the following global states:

o **user:** Current authenticated user.

o **player:** Current playing track, playback status (playing/paused), and volume.

o **searchResults:** Results from the search functionality.

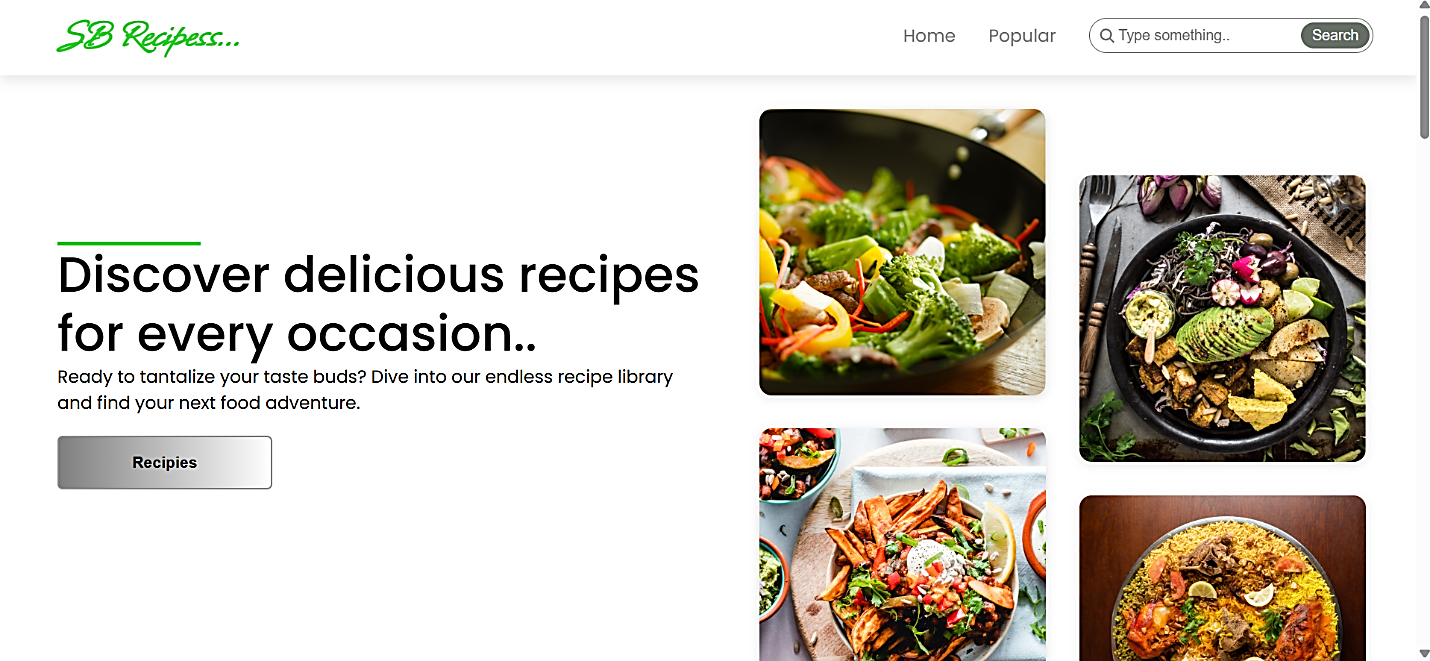
• **Local** **State**:

Local state is managed using React's useState hook within components. For example, the SearchPage component manages the search query input locally.

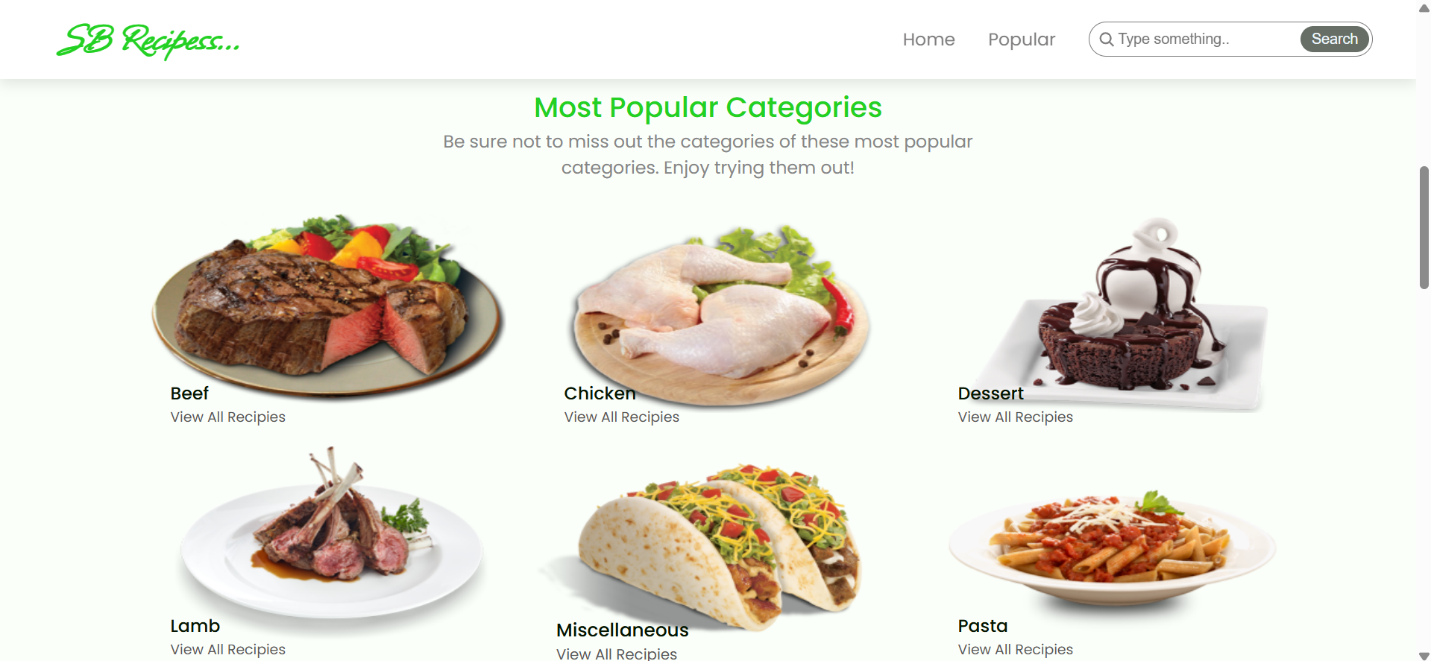
**9.** **User** **Interface**

• **Screenshots**

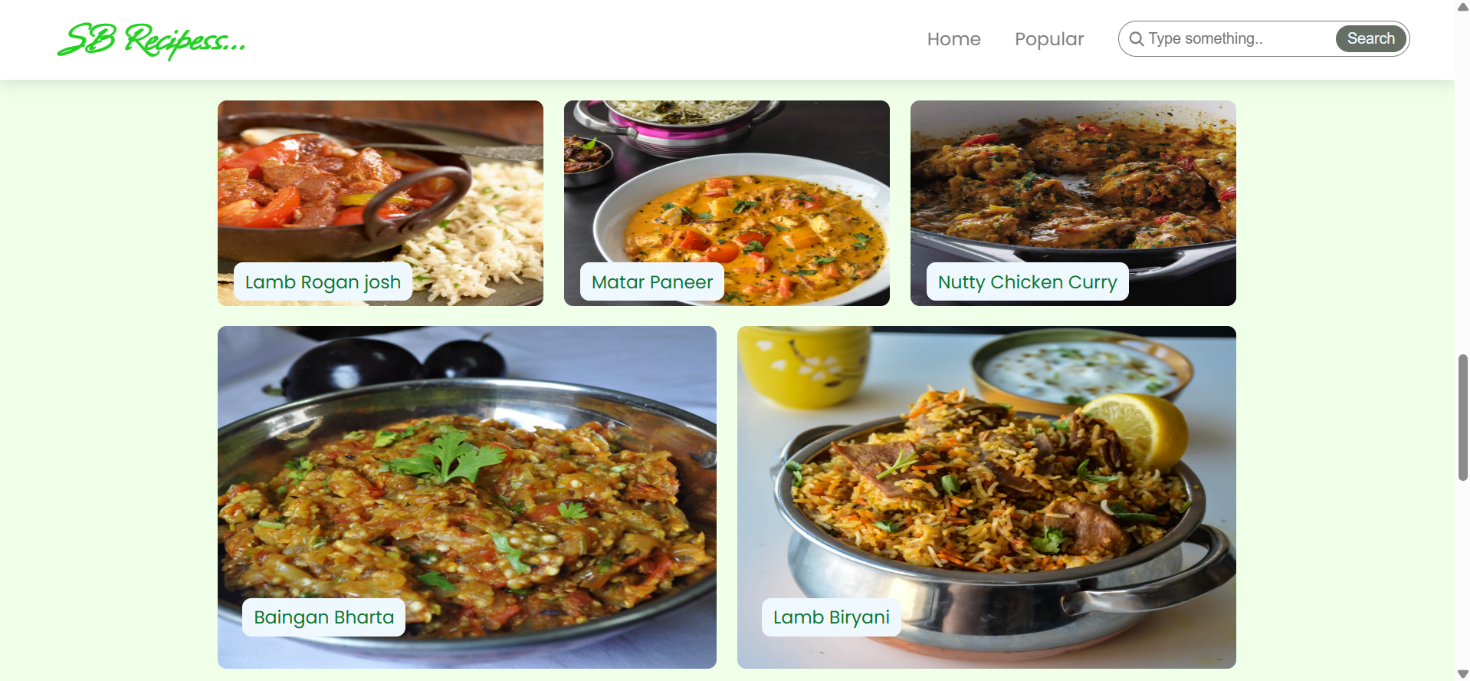
o **Home** **Page:** Display featured tracks and recommended playlists.



o **Most popular Categories:** This component contains all the popular categories of recipes.



o **Trending Dishes:** this components contains some of the trending dishes in the application.



**10.** **Styling**

• **CSS** **Frameworks/Libraries**:

The application uses **Styled-Components** for styling. This allows for modular and scoped CSS within components.

• **Theming**:

A custom theme is implemented using Styled-Components, with support for light and dark modes.

**11.** **Testing**

• **Testing** **Strategy**:

o **Unit** **Testing:** Using **Jest** and **React** **Testing** **Library**.

o **Integration** **Testing**: Is performed to ensure that components work together as expected.

o **End-to-End** **Testing:** **Cypress** is used for end-to-end testing of user flows.

• **Code** **Coverage**:

o Code coverage is monitored using Jest’s built in coverage tool. The current coverage is 85%.

**12.** **Screenshots** **or** **Demo**

• **Demo** **Link:**

<https://drive.google.com/drive/folders/1vGGiA_lzhwOfl6IWK-e3QNSfVSslzmlY>

• **Screenshots:** See section 9 for UI screenshots.

**13.** **Known** **Issues**

• **Issue** **1**: The search functionality is slow with large datasets.

**14.** **Future** **Enhancements**

• **Future** **Features**:

o Add support for user profiles and social sharing.

o Implement a recommendation engine for personalized music suggestions.

o Add animations and transitions for a smoother user experience.

This documentation provides a comprehensive overview of the **CookBook** project, including its architecture, setup instructions, and future plans.