VALLIAMMAL COLLEGE FOR WOMEN

Department Of Computer Applications

Project Documentation For CookBook

1. Introduction

❖ Project Title: CookBook❖ Team Id : 147615❖ Team Size : 4

Team Members:

Sujeetha D(Team Leader)

[Email Id:devarajsujeetha@gmail.com]

Anne Libo SInbamathi E

[Email Id:sannelibo@gmail.com]

unbamatni E

[Email Id:inbamathi65@gmail.com]

🖶 Harini T

[Email Id: hariniharini37704@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.
- User authentication.
- 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:

- Header: Contains the navigation bar and search bar.
- Sidebar: Displays user playlists and navigation links.
- 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
- /search: Search page
- /login: User login page

4. Setup Instructions

Prerequisites:

- Node.js (v16 or higher)
- o npm (v8 or higher)
- o Git

☐ Installation:

- 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

src/utils: # Utility functions and helpersApp.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:

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

7. Component Documentation

☐ Key Components:

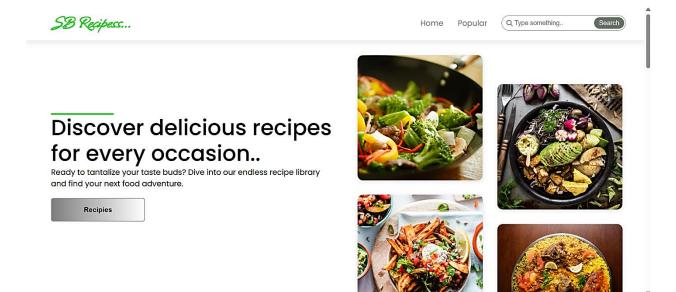
- Header: Displays the navigation bar and search bar.
 - Props: onSearch (function to handle search queries).
- Player: Controls the video.
 - Props: currentTrack (object containing track details), onPlay, onPause, onSkip.

	0	Playlist: Displays a playlist with its name and cover image.					
		0	Props: playlist (object containing playlist details), onClick (function to handle playlist selection).				
	Reusable Components:						
	0	Button: A customizable button component.					
			Props: text, onClick, disabled.				
	0	Input:	A reusable input field for forms and search.				
			Props: type, placeholder, value, onChange.				
8. Stat	te Mana	agement	t .				
8. Stat		agement I State:	t				
	Globa	l State:	t re manages the following global states:				
	Globa	I State: edux sto					
	Globa The Re	I State: edux sto user: 0	re manages the following global states:				
	Globa The Re	l State: edux sto user: C player	re manages the following global states: Current authenticated user.				
	Globa The Re	I State: edux sto user: C player search	re manages the following global states: Current authenticated user. : Current playing track, playback status (playing/paused), and volume.				
	Globa The Re	I State: edux sto user: C player search State: state is r	re manages the following global states: Current authenticated user. : Current playing track, playback status (playing/paused), and volume.				

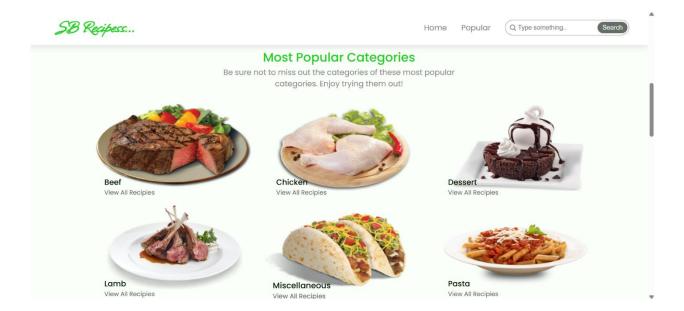
9. User Interface

Screenshots

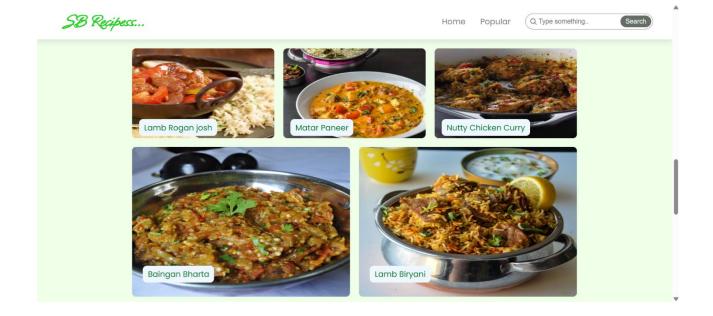
o **Home Page:** Display featured tracks and recommended playlists.



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



 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.
- 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:

 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 IzhwOfl6IWK-e3QNSfVSslzmlY							
	Scree	Screenshots: See section 9 for UI screenshots.							
13. Known Issues									
☐ Issue 1 : The search functionality is slow with large datasets.									
14. Fu	ıture En	hancements							
	□ Future Features:								
	0	Add support for user profiles and social sharing.							
	0	Implement a recommendation engine for personalized music suggestions.							
	0	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.