**Frontend Development with React Application**

**Project Documentation for Rhythmic Tunes**

**1. Introduction**

. Project Title: **Rhythmic Tunes**

• Team Members:

MUTHULAKSHMI.M Team Leader[EmailId:dhana890lakshmi@gmail.com]

MONISHA.R [Email Id: m26155708@gmail.com]

MAHASUNDHARI.C [Email Id: mahaseena7338@gmail.com]

MARY.N [Email Id: sandhiyanagaiya2001@gmail.com]

**2. Project Overview**

Purpose:

Rhythmic Tunes is a web application designed to provide users with a seamless music listening experience. The application allows users to browse, search, and play music tracks, create playlists, and discover new music based on their preferences.

Features:

\*Music player with

\* play pause

\* skip andvolume control.

**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.
    - **Player**: Music player controls (play, pause, volume, etc.).
    - **Sidebar**: Displays user playlists and navigation links.
  + **State Management**:

The application uses **Redux** for global state management. The Redux store manages user authentication, current playing track, playlist data, and search results.

* + **Routing**:

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

/: Home page /search: Search page /playlist/:id: Playlist details page /login: User login page

**4. Setup Instructions:**

Prerequisites:

Node.js(v16 or higher)

npm

Git

* + **Installation**:
    1. Clone the repository: git clone
    2. Navigate to the client directory: cd rhythmic-tunes/client
    3. Install dependencies: npm install
    4. Configure environment variables: Create a .env file in the client directory and add the necessary variables (e.g., API keys).
    5. Start the development server: npm start

1. **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**:
    - **api.js**: Handles API requests to the backend.
    - **auth.js**: Manages user authentication and token storage.
    - **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

* + - 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**:

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

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

o **Player**: Controls the music playback.

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

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

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

**8. State Management**

* + **Global State**:

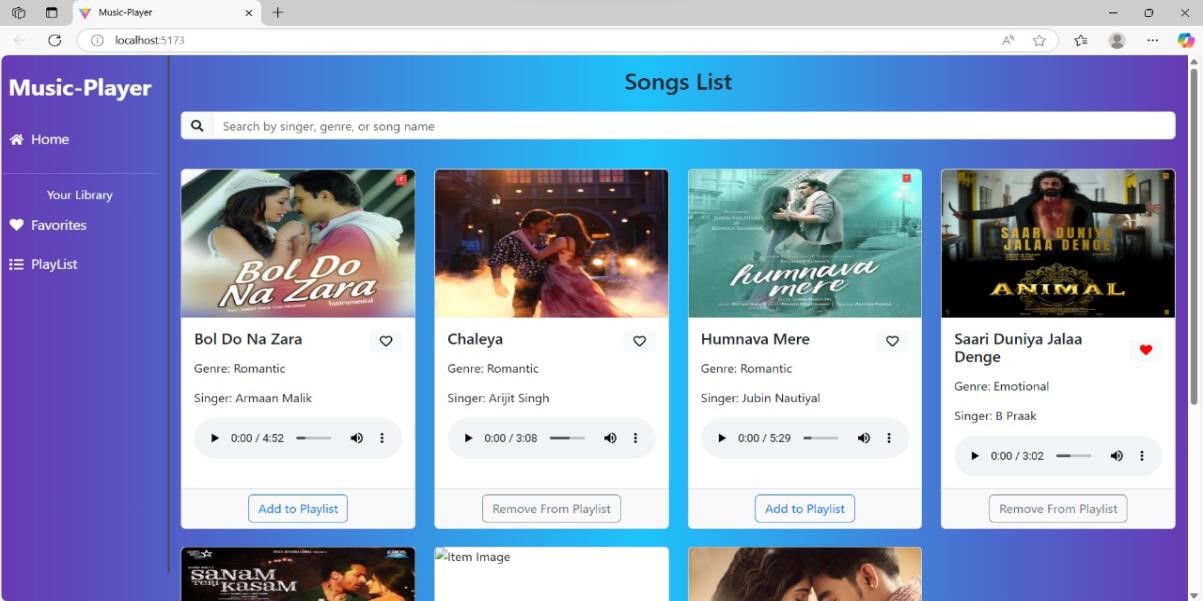
The Redux store manages the following global states:

* + - **user:** Current authenticated user.
    - **player:** Current playing track, playback status (playing/paused), and volume.
    - **playlists:** User-created playlists.
    - **searchResults:** Results from the search functionality.
  + **Local State**:

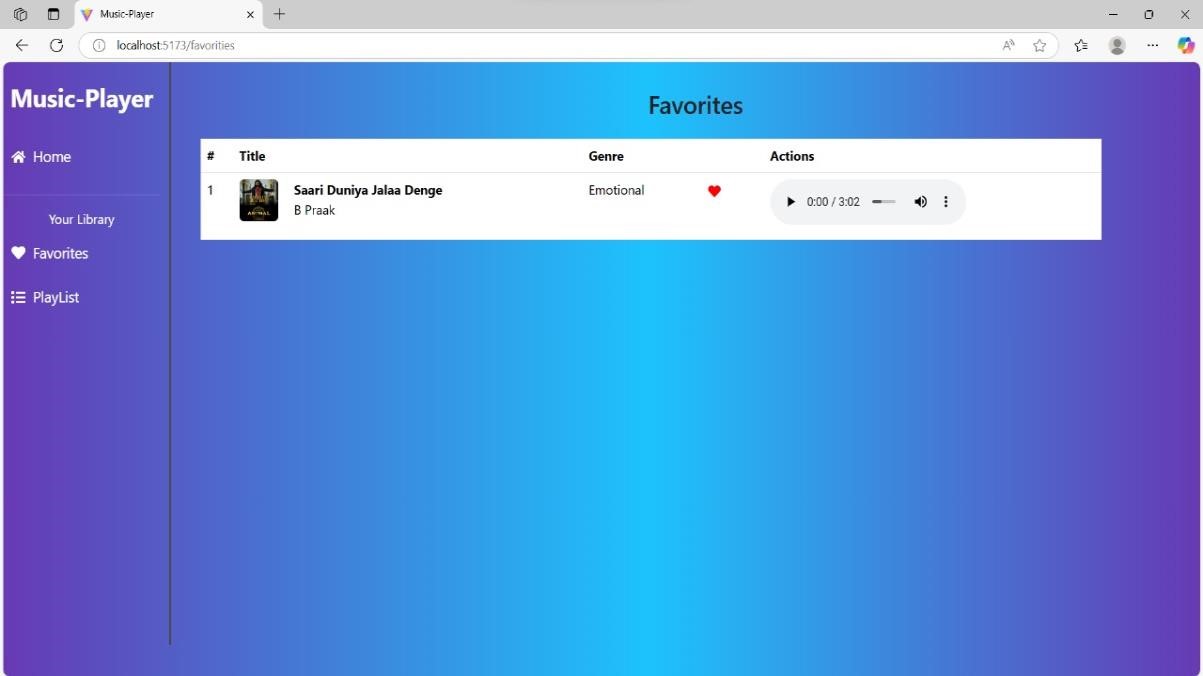
Local state is managed using React's useState hook within components. For example, the SearchPage component manages the search query input locally.

**9.User Interface**

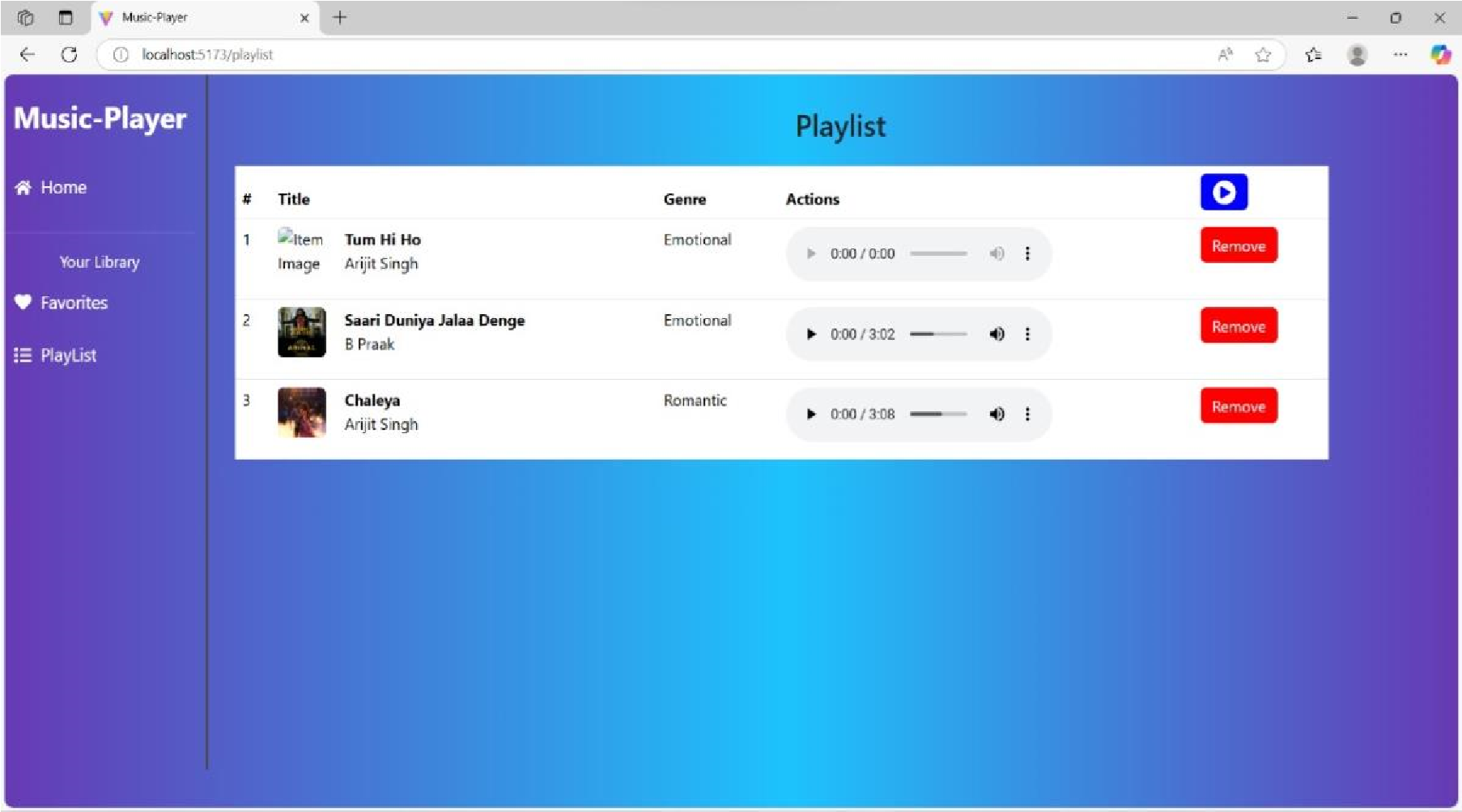
* + **Screenshots** o **Home Page:** Display featured tracks and recommended playlists.



* + - **Search Page:** Allows users to search for songs, albums, and artists.



* + - **Playlist Page:** Displays user-created playlists and allows playlist management.



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**:
    - **Unit Testing:** Using **Jest** and **React Testing Library**.
    - **Integration Testing**: Is performed to ensure that components work together as expected.
    - **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/file/d/1E50q8uXdA9Hhyu8eX3ts2nC9QWZriqbc/view?usp=drivesdk**](https://drive.google.com/file/d/1E50q8uXdA9Hhyu8eX3ts2nC9QWZriqbc/view?usp=drivesdk)
  + **Screenshots:** See section 9 for UI screenshots.

1. **Known Issues** 
   1. **Issue 1**: The music player sometimes skips tracks unexpectedly.
   2. **Issue 2**: The search functionality is slow with large datasets.

1. **Future Enhancements** 
   1. **Future Features**:
      * Add support for user profiles and social sharing. o Implement a recommendation engine for personalized music suggestions.
      * Add animations and transitions for a smoother user experience.

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