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

***Project Documentation for Rhythmic Tunes***

* + **INTRODUCTION**
  + **Project Title**: Rhythmic Tunes
  + **Team Members**:

**Saranya S** (**Team Leader**) Email Id [saranya20041013@gmail.com](mailto:saranya20041013@gmail.com) ]

**Priyadharshini V** [Email Id: [priyadharshinivisvanathan@gmail.com](mailto:priyadharshinivisvanathan@gmail.com) ]

**D Vanishka** [Email Id: [vanishkad2005@gmail.com](mailto:vanishkad2005@gmail.com) ]

**Jayasri S** [Email Id: [jayashreesudhakar2004@gmail.com](mailto:jayashreesudhakar2004@gmail.com) ]



1. **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, and volume control. o Search functionality to find songs, albums, and artists. o User authentication (login/signup). o Playlist creation and management.
    - Responsive design for mobile and desktop.

1. **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.). o **Sidebar**: Displays user playlists and navigation links.
    - **HomePage**: Displays featured tracks, recommended playlists, and new releases.
    - **SearchPage**: Allows users to search for songs, albums, and artists.
    - **PlaylistPage**: Displays user-created playlists and allows playlist management.
  + **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:

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

1. **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/unm12912137/rhythmictunes.git](https://github.com/unm12912137/rhythmic-tunes.git)
    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.

1. **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

1. **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).

* + **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.

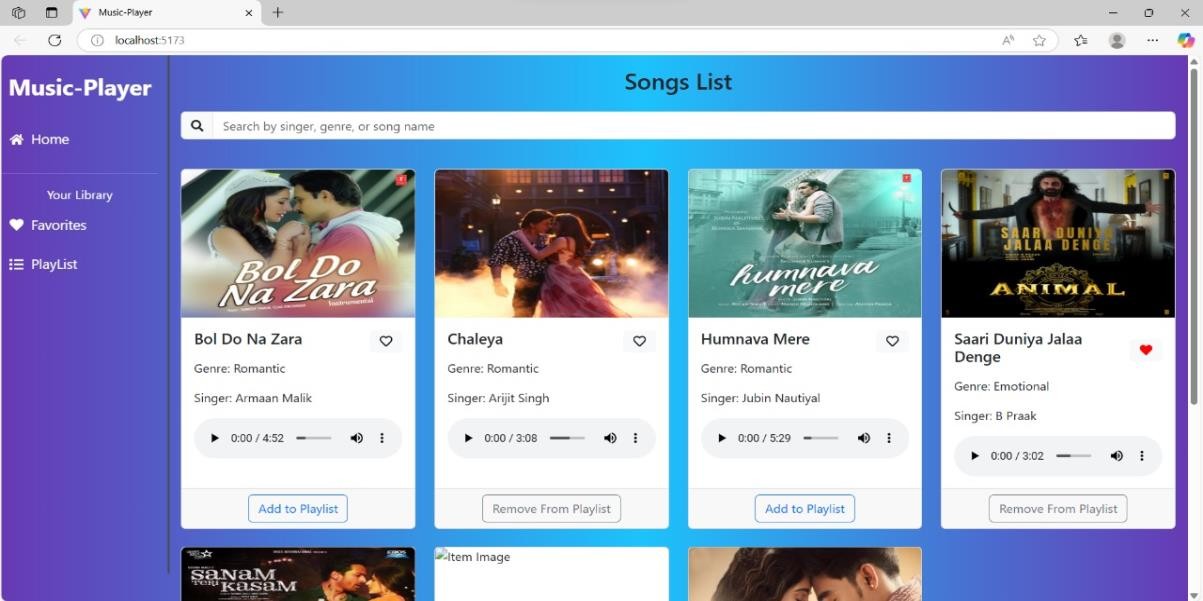
1. **STATE MANAGEMENT**
   * **Global State**:

The Redux store manages the following global states:

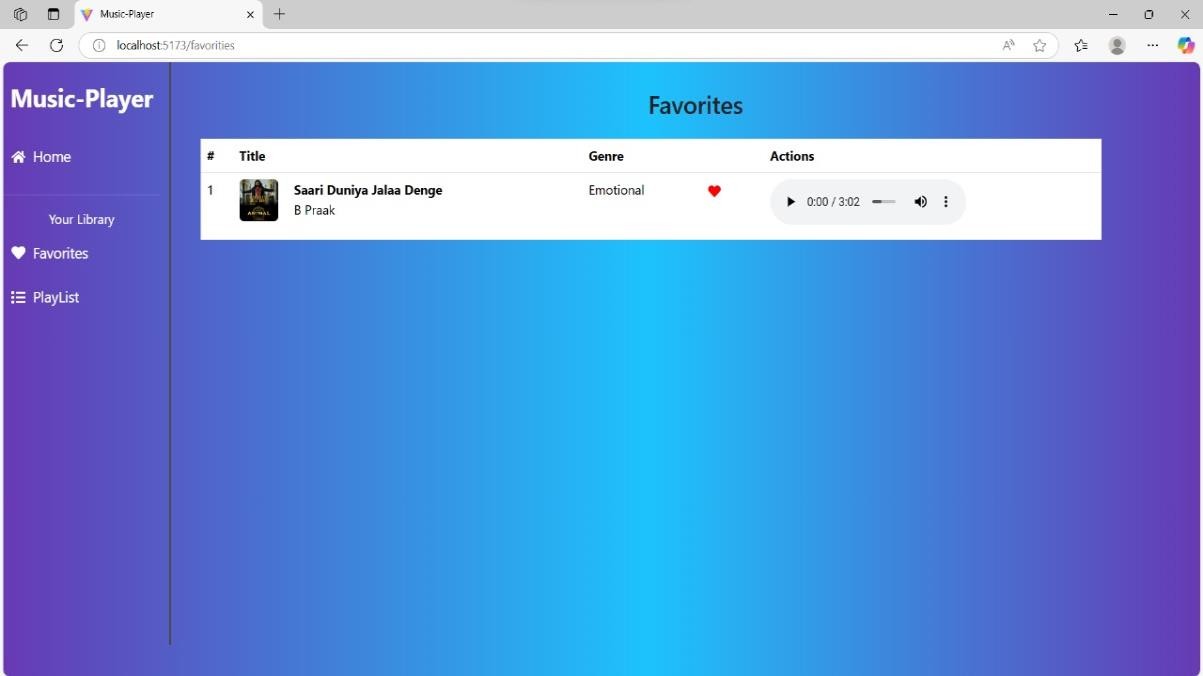
* + - **user:** Current authenticated user.
    - **player:** Current playing track, playback status (playing/paused), and volume. o **playlists:** User-created playlists.
    - **searchResults:** Results from the search functionality.
  + **Local State**:

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

1. **USER INTERFACE**
   * **Screenshots** o **Home Page:** Display featured tracks and recommended playlists.



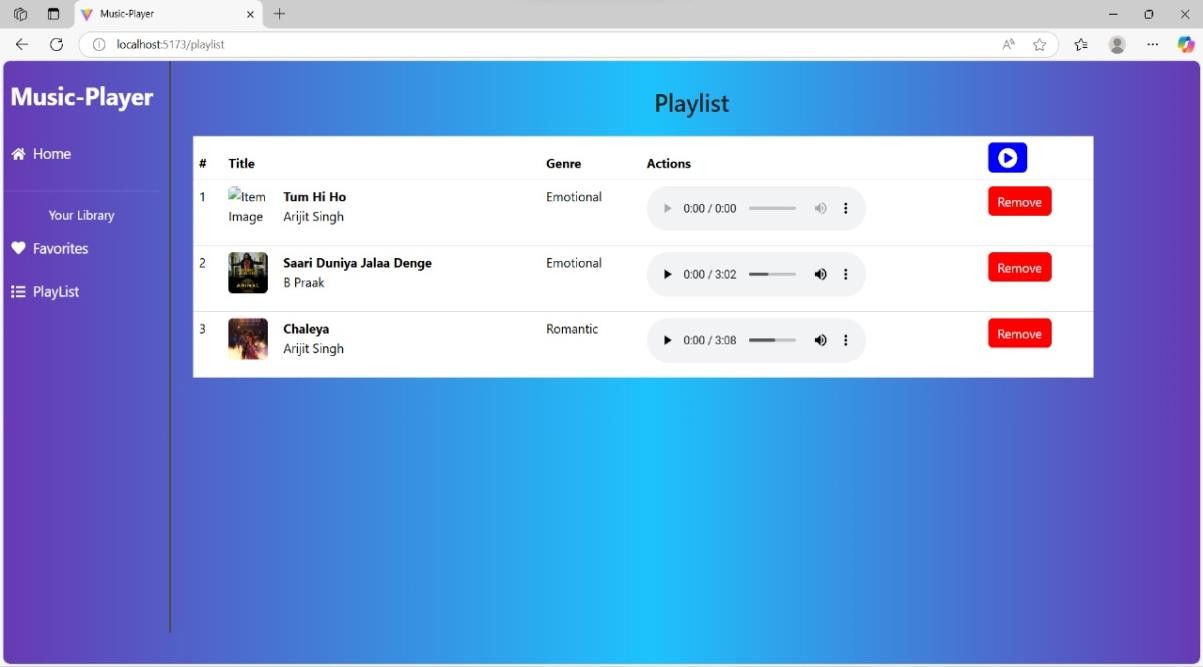
* + - **Search Page:** Allows users to search for songs, albums, and artists.



* + - **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.

1. **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**:
     + Code coverage is monitored using Jest’s built in coverage tool. The current coverage is 85%.

1. **SCREENSHOTS OR DEMOS** 
   * **Demo Link:** [https://drive.google.com/file/d/1ROVO0udGYwpFo\_rTD9KGNFiUPm34ZvNS/view?us p=drivesdk](https://drive.google.com/file/d/1ROVO0udGYwpFo_rTD9KGNFiUPm34ZvNS/view?usp=drivesdk)
   * **Screenshots:** See section 9 for UI screenshots.

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

1. **FUTURE ENHANCEMENTS**
   * **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.

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