FrontendDevelopmentwithReact.js

ProjectDocumentationforRhythmicTunes

---

1. Introduction

- **Project Title**: **Rhythmic Tunes**
- **Team Members**:

- **Sakthivelan T (Team Leader)** [Email Id: svelan004@gmail.com]
- **Kathirvel K** [Email Id: kathirkumar231@gmail.com]
- **Manoj S** [Email Id: manoj5647564@gmail.com]
- **Vinoth Kumar S** [Email Id: vasuvino36@gmail.com]

---

2. Project Overview

- **Purpose**:
  RhythmicTunesisawebapplicationdesignedtoprovideuserswithaseamlessmusic listening experience. The application allows users to browse, search, and play music tracks, create playlists, and discover new music based on their preferences.
- **Features**:
  - o 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.
  - o Responsive design for mobile and desktop.

---

3. Architecture

- **Component Structure**:
  TheapplicationisbuiltusingReact.jswithacomponent-basedarchitecture.Major components include:

- o **Header**: Contains the navigation bar and search bar.
- o **Player**: Music player controls (play, pause, volume, etc.).
- o **Sidebar**: Display suserplay lists and navigation links.
- o **Home Page**: Displays featured tracks, recommended playlists, and new releases.
- o **Search Page**: Allows users to search for songs, albums, and artists.
- o **Playlist Page**: Displays user-created playlists and allows playlist management.

- **State Management**:
  Theapplicationuses**Redux**forglobalstatemanagement.TheReduxstoremanages user authentication, current playing track, playlist data, and search results.
- **Routing**:
  The application uses **React Router** for navigation. Routes include:
  - /:Homepage
  - /search:Searchpage
  - /playlist/:id:Playlistdetailspage
  - /login:Userloginpage

---

4. Setup Instructions

- **Prerequisites**:

  o Node.js (v16 or higher)

  o npm (v8 or higher)

  o Git
- **Installation**:
  1. Clone the repository: git clonehttps://github.com/Rifairoshan/Music-Streaming.git
  2. Navigate to the client directory: cd rhythmic-tunes/client
  3. Install dependencies: npm install
  4. Configureenvironmentvariables:Createa.envfileintheclientdirectoryand add the necessary variables (e.g., API keys).
  5. Start the development server: npm start

---

5. Folder Structure

- **Client**:
  - **src/components:** # Reusable components (Header, Player, etc.)
  - **src/pages:** # Page components (HomePage, SearchPage, etc.)
  - **src/assets:** # Images, icons, and other static files
  - **src/redux:** # Redux store, actions, and reducers
  - **src/utils:** # Utility functions and helpers
  - **App.js:** # Main application component
  - **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. RunningtheApplication

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**:
  - **Header**: Displays the navigation bar and search bar.
    - Props: on Search (function to handle search queries).

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

  - Props: current Track(object containing track details), on Play, on Pause, on Skip.

- o **Playlist Card**: Displays a playlist with its name and cover image.
    - Props: play list(object containing play list details), on Click(function to handle playlist selection).

- **Reusable Components**:
  - o **Button**: A customizable button component.
    - Props: text, on Click, disabled.
  - o **Input**: A reusable input field for forms and search.
    - Props: type, placeholder, value, on Change.

---

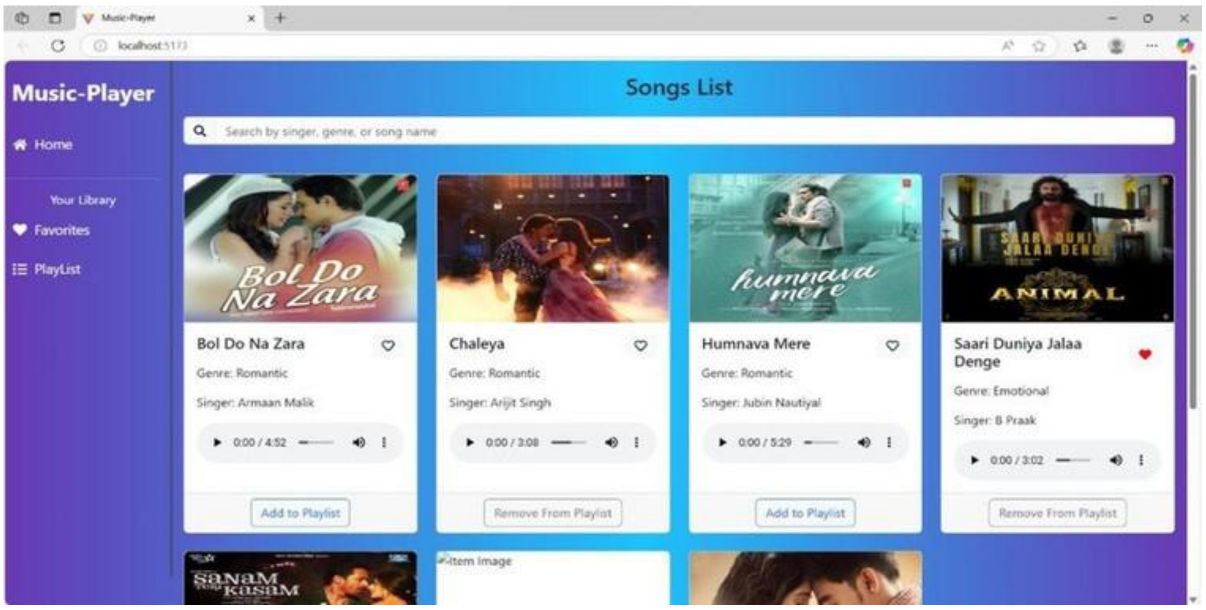
8. State Management

- **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 **playlists:** User-created playlists.
  - o **search Results:** Results from the search functionality.
- **Local State**:
  LocalstateismanagedusingReact'suseStatehookwithincomponents.Forexample, the Search Page component manages the search query input locally.

---

9. User Interface

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



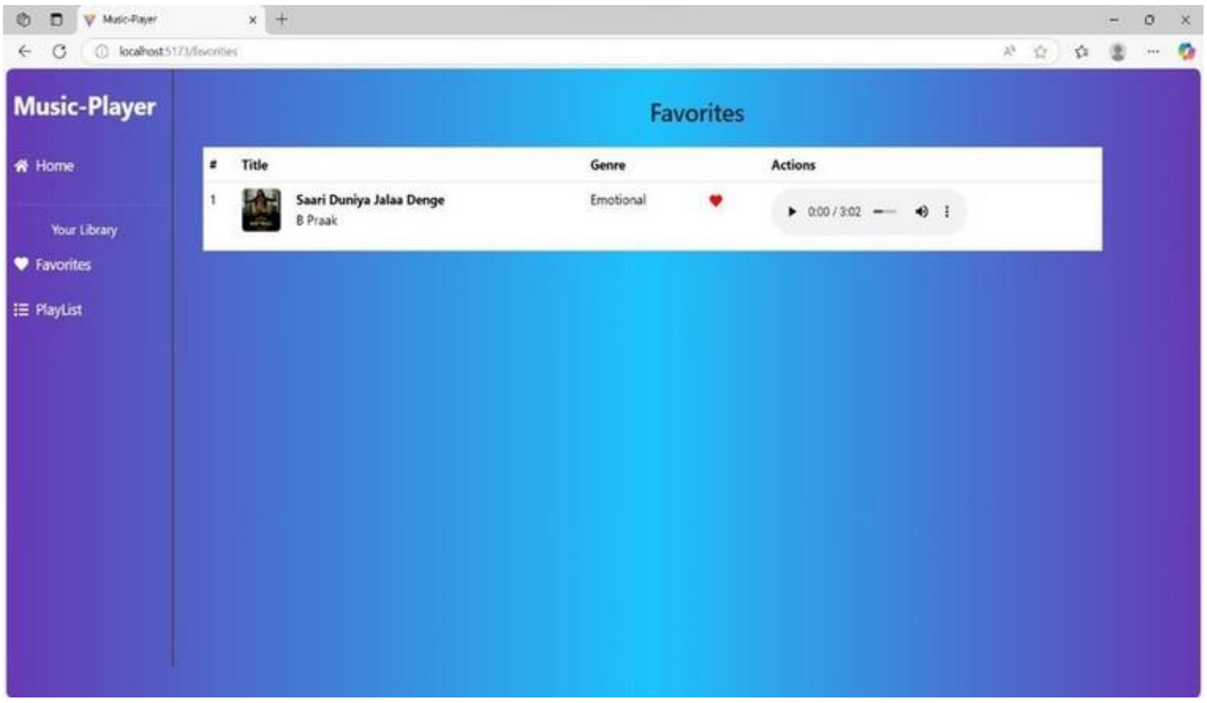
Bol Do Na Zara
Genre: Romantic
Singer: Armaan Malik
0:00 / 4:52

Chaleya
Genre: Romantic
Singer: Arijit Singh
0:00 / 3:08

Humnava Mere
Genre: Romantic
Singer: Jubin Nautiyal
0:00 / 5:29

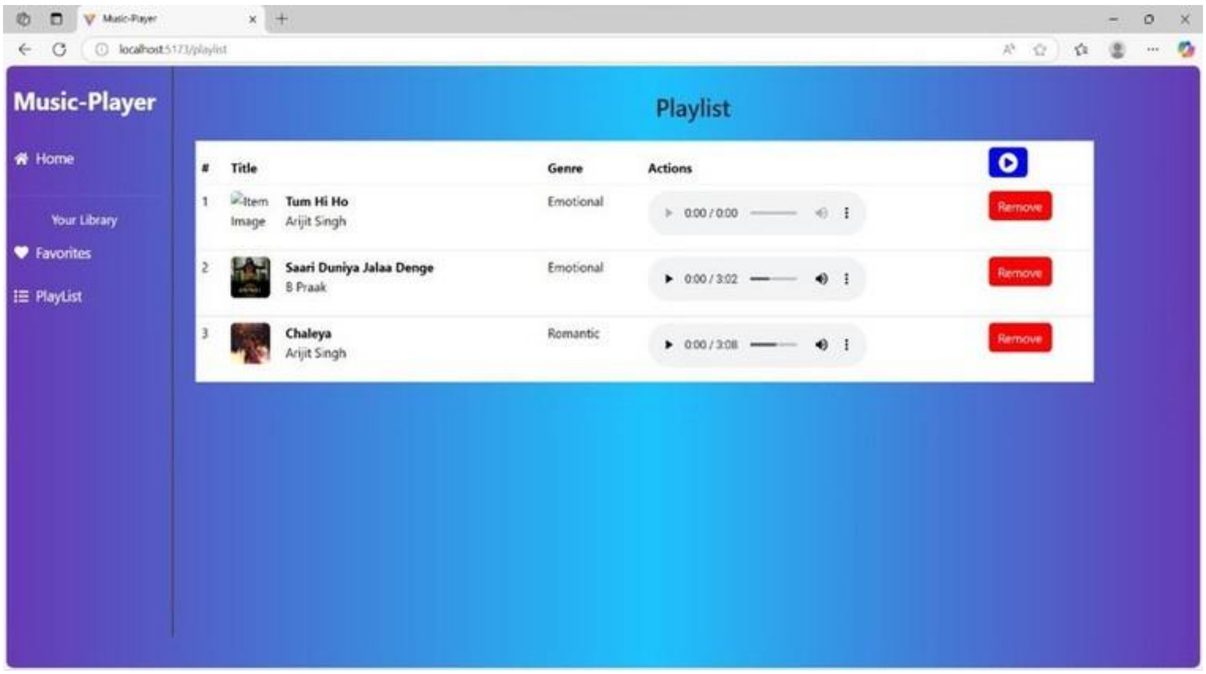
Saari Duniya Jalaa Denge
Genre: Emotional
Singer: B Praak
0:00 / 3:02

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



#	Title	Genre	Actions
1	Saari Duniya Jalaa Denge B Praak	Emotional	0:00 / 3:02

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



Playlist

#	Title	Genre	Actions
1	Item Image Tum Hi Ho Arijit Singh	Emotional	0:00 / 0:00
2	Saari Duniya Jalaa Denge B Praak	Emotional	0:00 / 3:02
3	Chaleya Arijit Singh	Romantic	0:00 / 3:08

---

10. Styling

- **CSS Frameworks/Libraries**:
  TheapplicationusesStyled-Componentsforstyling.Thisallowsformodularand 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 CodecoverageismonitoredusingJest'sbuiltincoveragetool.Thecurrent coverage is 85%.

---

12. Screenshots or Demo

- **https://drive.google.com/file/d/1IhcnjlYu9Es5ISPVb0QbQZzAENA3Prhq/view?usp=sharing**
- **Screenshots:** See section 9 for UI screenshots.

13. Known Issues

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

---

14. Future Enhancements

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

Thisdocumentationprovidesacomprehensiveoverviewofthe**RhythmicTunes**project, including its architecture, setup instructions, and future plans.