**Frontend Development with React.js**

**Project Documentation**

**Introduction**

**Project Title:** Rhythmic Tunes: Your Melodic Companion

**Team leader : POOJASRI.R –** [**poojasriramasamy4@gmail.com**](mailto:poojasriramasamy4@gmail.com)

**Team members :**

1. RAKSHANADHARANI.S – [rakshanadharani2007@gmail.com](mailto:rakshanadharani2007@gmail.com) – code developer
2. SRI ANNAPOORANI.S – [sriannapoorani24@gmail.com](mailto:sriannapoorani24@gmail.com) – demo video
3. SRINATHI.S – [srinathisuresh@gmail.com](mailto:srinathisuresh@gmail.com) – document creator

**Project Overview**

**Purpose:**  
Rhythmic Tunes is a music discovery and playback web application designed to be a melodic companion for users. It allows them to explore trending tracks, curated playlists, and personalized recommendations with a modern, responsive interface.

**Features:**

* Browse trending tracks and curated playlists.
* Personalized recommendations based on user preferences.
* Search functionality for songs, artists, and albums.
* Seamless music player with play, pause, skip, and volume controls.
* Responsive design for mobile, tablet, and desktop.
* User authentication for personalized playlists and favorites.

**Architecture**

**Component Structure:**

* App.js – Root component handling routes and layout.
* Navbar – Navigation bar for pages.
* MusicPlayer – Controls playback.
* TrackList – Displays list of songs/playlists.
* TrackCard – Reusable component for track details.
* SearchBar – Enables song/artist search.
* UserProfile – Displays user details and playlists.

**State Management:**

* Redux Toolkit for global state (authentication, user preferences, player state).
* Local component state for UI interactions (e.g., modals, dropdowns).

**Routing:**

* React Router v6:
  + / – Home (Trending tracks)
  + /search – Search results
  + /playlist/:id – Playlist details
  + /profile – User profile

**Setup Instructions**

**Prerequisites:**

* Node.js (v16+)
* npm or yarn

**Installation:**

1. Clone repository:
2. git clone https://github.com/yourusername/rhythmic-tunes.git
3. cd rhythmic-tunes
4. Install dependencies:
5. npm install
6. Setup environment variables (.env):
7. REACT\_APP\_API\_KEY=your\_api\_key

**Folder Structure**

**Client:**

src/

├── assets/ # Images, icons

├── components/ # Reusable UI components

├── pages/ # Page-level components

├── redux/ # State management (slices, store)

├── utils/ # Helper functions & hooks

├── App.js

└── index.js

**Utilities:**

* api.js – API fetch helpers.
* useFetch.js – Custom hook for fetching music data.
* formatTime.js – Utility for song duration formatting.

**Running the Application**

To start the app locally:

npm start

This will run the development server on <http://localhost:3000>.

**Component Documentation**

**Key Components:**

* MusicPlayer – Accepts props like currentTrack, onPlay, onPause.
* TrackList – Receives tracks array and renders TrackCard.
* SearchBar – Accepts onSearch callback.

**Reusable Components:**

* Button – Styled button with variants (primary, secondary).
* Modal – Reusable modal for popups.

**State Management**

**Global State:**

* Redux store holds:
  + auth → user authentication data.
  + player → current track, playing status.
  + playlist → user playlists and favorites.

**Local State:**

* Modal toggles
* Form input values

**User Interface**

* Home page with trending tracks carousel.
* Music player docked at the bottom.
* Search results displayed in a responsive grid.
* Profile page with personalized playlists.

**Styling**

* Tailwind CSS for styling.
* Styled-components for custom theming.
* Dark and light mode support.

**Testing**

**Testing Strategy:**

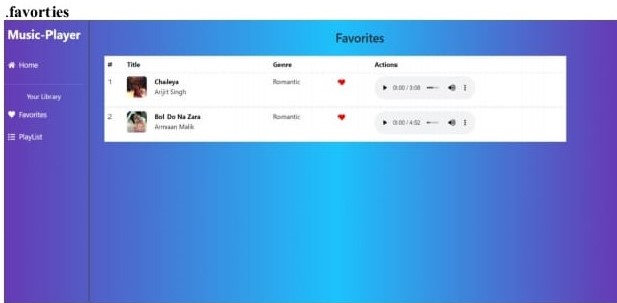
* Unit testing with Jest & React Testing Library.
* Integration tests for API fetches.
* End-to-end testing with Cypress.

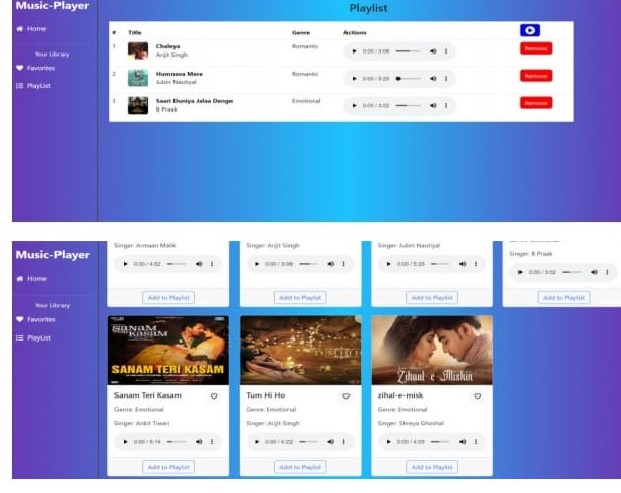
**Code Coverage:**

* Jest + Istanbul for coverage reports.

**Screenshots or Demo**

* [Insert screenshots of Home, Player, and Profile pages here]
* [Optional: Demo link to live application]





**Known Issues**

* Some APIs have rate-limiting restrictions.
* Mobile player controls need further optimization.

**Future Enhancements**

* Add offline playback.
* Integrate lyrics display.
* AI-based playlist recommendations.
* Social sharing for playlists.

Bottom of Form