Frontend Development with React.js

Project Documentation format

1. **Introduction**
    - **Project Title**: Rhythmic tunes: Your melodic companion
    - **Team leader:**
    - Saraswathi.A
    - **Team Members**:
    - Sujitha.S
    - Swetha.J
    - Sangeetha.J
2. **Project Overview**
    - **Purpose:**
    - Rhythmic Tunes is a music-based web application designed to help users discover, play, and organize their favorite tunes efficiently. It aims to deliver an intuitive and immersive experience for music lovers.
    - **Goals:**
    - Provide a seamless UI for browsing and playing music.
    - Implement an intuitive playlist and favourites management system.
    - Deliver a fast and responsive interface using React.js.
    - – Ensure scalability for future enhancements and integrations.
    - **Features**:
    - Music Player: Play, pause, seek, and adjust volume.
    - Search & Filter: Easily find songs by artist, genre, or title.
    - Playlist Management: Create and manage custom playlists.
    - User Authentication: Secure login and user-specific preferences.
    - Responsive Design: Works across different screen sizes.
3. **Architecture**

    - **Component Structure:**
    - App.js – Main entry point, routes setup.
    - Player Component- Handles audio playback.
    - Playlist Component – Displays user-created playlists.
    - Search Component- Enables searching for songs.
    - Auth Component – Manages user authentication.

    - **State Management**:
    - Global State: Redux Toolkit is used to manage app-wide state (e.g., user preferences, playlists).
    - Local State: React's use State & use Reducer manage UI interactions.

    - **Routing** :
    - React Router is used for navigation between pages (e.g., Home, Library, Now Playing).

4. **Setup Instructions**

- **Prerequisites**:
- Ensure the following dependencies are installed:
- Node.js (v16+)
- NPM or Yarn
- Git
- **Installation**: Provide a step-by-step guide to clone the repository, install dependencies, and configure environment variables.

5. **Folder Structure**
   - **Client**: Describe the organization of the React application, including folders like components, pages, assets, etc.
   - **Utilities**: Explain any helper functions, utility classes, or custom hooks used in the project.
6. **Running the Application**
   - Provide commands to start the frontend server locally.
     - **Frontend**: npm start in the client directory.
7. **Component Documentation**
   - **Key Components**: Document major components, their purpose, and any props they receive.
   - **Reusable Components**: Detail any reusable components and their configurations.
8. **State Management**
   - **Global State**: Describe global state management and how state flows across the application.
   - **Local State**: Explain the handling of local states within components.
9. **User Interface**
   - Provide screenshots or GIFs showcasing different UI features, such as pages, forms, or interactions.
10. **Styling**

- **CSS Frameworks/Libraries**: Describe any CSS frameworks, libraries, or pre-processors (e.g., Sass, Styled-Components) used.
- **Theming**: Explain if theming or custom design systems are implemented.

11. **Testing**

- **Testing Strategy**: Describe the testing approach for components, including unit, integration, and end-to-end testing (e.g., using Jest, React Testing Library).
- **Code Coverage**: Explain any tools or techniques used for ensuring adequate test coverage.

12. **Screenshots or Demo**

- Provide screenshots or a link to a demo showcasing the application's features and design.

- https://drive.google.com/drive/folders/1bU5xgQG2pKdVSug42Gp4yEQEvIyALlJB?usp=sharing

13. **Known Issues**

- Slow API response-on large song searches.
- Limited mobile optimization-for smaller screens.
- Playlist sorting issue- when adding multiple songs.

14. **Future Enhancements**

- Dark Mode Toggle
- Offline music playback
- Integration with Spotify API
- AI-based music recommendations