Frontend Development with React.js

Project Documentation format

1. **Introduction**
   - **Project Title**:Rythmic Tunes: Your melodic Companion

   - **Team Leader:**
     Varsha.S- varshaini1409@gmail.com
   - **Team Members**:
     Preethi.M- preethimanivannan79@gmail.com
     RajaRajeshwari.R- anishramya72001@gmail.com
     Yuvalakshmi.S- yuvalakshmi1113a@gmail.com

2. **Project Overview**
   - **Purpose**: Rhythmic Tunes is a music streaming platform designed to provide a seamless and interactive listening experience. It allows users to play, pause, search, filter, and add tracks to their favorites, offering a personalized and user-friendly interface.

   - **Features**:
     - **Play/Pause Functionality:** Allows users to control music playback with intuitive play and pause controls.

     - **Search and Filter:** Enables users to search for tracks and filter by genre, artist, and other criteria.

     - **Add to Favorites:** Users can add or remove tracks from their favorites for quick access.

3. **Architecture**
   - **Component Structure**:
   - **MusicPlayer:** Controls play, pause, and track progress.
   - **SearchBar:** Allows users to search and filter music tracks.
   - **TrackList:** Displays a list of tracks and interacts with individual track items.
   - **TrackItem:** Handles track-specific actions like play, pause, and add to favorites.
   - **Favorites:** Manages favorite tracks and updates the UI in real-time.

   - **State Management**:
     We use the Context API for global state management:
       - **Playback Context**: Manages play, pause, and current track data.
       - **Search Context:** Stores search queries and filter parameters.
       - **Favorites Context:** Handles the addition and removal of favorite tracks.

- **Routing**:
  Implemented using React Router:
  - /: Home page with track listing and player.
  - /favorites: Displays user's favorite tracks.
  - /search: Search and filter page.

4. **Setup Instructions**
   - **Prerequisites**:
     - Node.js (v18 or later)
     - npm or yarn

5. **Installation**:
   - Clone the repository:

```
git clone [repository_url]

cd rhythmic-tunes
```

   - Install dependencies:

```
npm install
```

   - Start the development server:

```
npm start
```

6. **Folder Structure**

```
.
├── public/          # Static assets
├── src/
│   ├── components/  # React components
│   ├── context/     # Context API state management
│   ├── pages/       # Page-level components (Home, Favorites, Search)
│   ├── assets/      # Images, icons, etc.
│   └── App.js       # Root component
└── package.
```

7. **Running the Application**
   - To start the frontend application:
     npm start

8. **Component Documentation**
   - **Key Components**:
   - **MusicPlayer**:
     - **Props:** track, isPlaying, onPlayPause
     - **Purpose:** Controls music playback.
   - **SearchBar:**

        - **Props:** onSearch, onFilter
        - **Purpose:** Handles search and filter inputs.

    - **TrackItem:**
        - **Props:** track, onPlayPause, onFavorite
        - **Purpose:** Displays individual track information and actions.

    - **Reusable Components**:
        - **Button**: Generic button component.
        - **Modal**: Used for pop-up interactions.

9. **State Management**
    - **Global State**:
        - **Playback State**: Controls current track and playback status.
        - **Favorites State:** Stores the list of user's favorite tracks.
        - **Search State**: Manages search terms and filter options.

    - **Local State**: Handled within individual components for UI interactions (e.g., modals, input fields).

10. **User Interface**

The interface provides a clean and intuitive experience with sections for:

- Music playback
- Search and filter
- Favorites management

11. **Styling**

    - **CSS Frameworks/Libraries**: Tailwind CSS
    - **Theming**: Custom CSS variables for consistent styling across the app.

12. **Testing**

- Unit tests for components (using Jest and React Testing Library)
- Integration tests for user flows (e.g., play/pause, add to favorites)

13. **Screenshots or Demo**

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

- https://drive.google.com/file/d/1F-F_JF5GaoQgN7Cz-D1vz-2Jis7HSATr/view?usp=drivesdk

14. **Known Issues**

- Some browsers may not support advanced media controls.
- Minor latency during rapid search queries.

15. **Future Enhancements**

- Offline Mode: Enable track downloads for offline playback.
- User Authentication: Implement user login for personalized playlists.
- Enhanced Search: Add advanced filters (mood, tempo, etc.).