RHYTHMIC TUNES – YOUR MELODIC COMPANION

Frontend Development with React.js

PROJECT DOCUMENTATION

INTRODUCTION:

Project Title

"Rhythmic Tunes – Your Melodic Companion"

Team Members:

- **PRIYADARSHINI.M- Team Leader**
- **SHARMILA.C**
- **SRUTHI.T**
- **NASRIN BEGUM.I**

PROJECT OVERVIEW

Purpose:

The purpose of this project is to design and develop a melodic companion web application where users can play songs, manage playlists, add/remove favourites, and search music by name, singer, or genre.

Features:

- **Play and pause songs**
- **Increase or decrease volume.**
- **Manage playback speed**
- **Add or remove favourites.**

- **Search songs by song name, singer name, or genre.**
- **Manage playlists using a JSON SERVER**

Architecture:

Component Structure:

-App.js – Root component, handles routing
Home.js – Displays songs with controls (play, pause, speed, volume, -favourites).
-search.js – Allows searching by keywords.
-Favourites.js – Displays user's saved songs.
-Playlist aylist.js – Manages playlists.
-SongCard.js – Reusable component for displaying song details with buttons.
-Navbar.js – Navigation bar for accessing pages.

State Management:

Local state handled via React Hooks (useState, useEffect).

Routing:

Implemented using React Router v6 for smooth navigation between Home, Favourites, Search, and Playlist.

SETUP INSTRUCTIONS

Prerequisites:

- Node.js installed

- Visual Studio Code installed

Installation:

1. Clone or download the repository.
2. **Open the project in Visual Studio Code.**
3. **In the terminal, run**
   npm install
   npm run dev

This will start the Vite development server and provide a link to open the app.

4. **Split the terminal and move to the db folder:**

   cd db
   npm install -g json-server

5. **Run the JSON server:**

   json-server --watch db.json --port 3000

Folder Structure :

Client:

src/components/ – Reusable components (SongCard, Navbar).

src/pages/ – Page components (Home, Search, Favourites, Playlist).

src/assets/ – Media files and styles.

Utilities:

Axios used for fetching songs and handling backend communication.

Helper functions for search and filtering.

RUNNING THE APPLICATION

Start the frontend (Vite dev server):
npm run dev

Start the backend JSON server:

json-server --watch db.json --port 3000

COMPONENT DOCUMENTATION

Key Components:

Home.js: Displays list of songs, play/pause controls, favourites button.

Search.js: Filters and displays search results.

Playlist.js: Handles playlists from db.json.

Favourites.js: Manages favourite songs.

Reusable Components:

SongCard.js: Displays single song with details and actions.

Navbar.js: Provides navigation between pages.

STATE MANAGEMENT

Global State:

Managed through context for favourites and playlists.

Local State:

Controlled using hooks within individual components.

USER INTERFACE

Home Page: List of songs, play controls, volume, favourites.

Search Page: Search by song, singer, or genre.

Playlist Page: View and manage playlists.

Favourites Page: Add/remove songs to favourites.

STYLING

CSS Frameworks/Libraries:

- Tailwind CSS for utility-first responsive design.
- Bootstrap & React-Bootstrap for layout and components.
- React Icons for consistent icons

Theming:

Custom color palette applied for a modern music UI feel.

TESTING

Testing Strategy:

Manual testing of UI interactions (play, pause, volume, search).

JSON server tested for proper CRUD operations.

Code Coverage:

Linting enforced via ESLint for code quality.

SCREENSHOTS OR DEMO

Demo link with explanation:

https://drive.google.com/file/d/1txjqOqG-2_slTwwE7fabLz8K4HnEc-x7/view?usp=drivesdk

KNOWN ISSUES

- JSON server must be restarted if manually closed.
- No user authentication system.

FUTURE ENHANCEMENTS

- Integration with a real music API.
- User login and profile-specific playlists.
- Advanced filtering options (album, year, mood).
- Mobile app version using React Native.