Rhythmic Tunes (React) Documentation

---

1. Introduction

Welcome to Rhythmic Tunes, a cutting-edge music streaming application built using React.js. This application is designed to provide a seamless and immersive music experience for users by offering dynamic features such as playlist creation, favorite song marking, and an interactive UI.

Key Highlights:

- **Modern UI**: A visually stunning and intuitive interface.
- **Smart Features**: Playlist creation, song searching, and seamless playback.
- **Cross-Device Compatibility**: Works smoothly on desktop, tablet, and mobile.

2. Scenario-Based Introduction

Imagine walking down a busy street and needing the perfect music to match your mood. With Rhythmic Tunes, you can instantly access a curated playlist that enhances your journey. Whether you are working, commuting, or relaxing, our app adapts to your musical needs.

3. Target Audience

- **Music Enthusiasts**: People passionate about music streaming.
- **Casual Listeners**: Users looking for an easy-to-use platform.
- **Developers & Tech Enthusiasts**: Those interested in learning React-based app development.

4. Project Goals and Objectives

- **User-Friendly Interface**: An easy-to-navigate UI for smooth user experience.
- **Advanced Music Management**: Organize, search, and explore songs effortlessly.
- **Modern Tech Stack**: Built with React.js, ensuring performance and scalability.

5. Key Features

- **Song Listings**: Display songs with details like artist, genre, and release date.
- **Playlist Management**: Create, add, and organize songs.
- **Playback Control**: Play, pause, skip, and adjust volume.
- **Offline Listening**: Download songs for offline playback.
- **Search Functionality**: Quickly find songs, artists, or albums.

- **User Authentication**: Secure login and registration for personalized experiences.
- **Recommendations**: AI-powered song recommendations based on user preferences.

6. Prerequisites

Required Tools & Libraries:

- **Node.js & npm** (Download)
- **React.js** (Setup via `npm create vite@latest`)
- **Version Control (GitHub/Git)** (Download)
- **Development Environment** (VS Code, WebStorm, etc.)
- **Firebase or Backend API** (For user authentication and data storage)

7. Project Structure

```
RhythmicTunes/
|── src/
|    ├── components/
|    |    ├── Hero.js
|    |    ├── Playlist.js
|    |    ├── Favorites.js
|    |    ├── Songs.js
|    ├── App.js
|    ├── index.js
|── public/
|── db/db.json (Mock Database)
|── package.json
|── README.md
```

8. Project Flow

1. **User launches the app** → Navigates to the home page.
2. **User registers/logs in** → Access personalized playlists.
3. **Search for a song** → Uses the search bar to find tracks.
4. **Play a song** → Click on a track to start playback.
5. **Add to playlist** → Users can create and manage playlists.
6. **Favorite a song** → Save songs for quick access later.
7. **Receive recommendations** → AI suggests songs based on listening history.

9. Milestone 1: Project Setup & Configuration

Installation Steps:

Create a React App:

```
npm create vite@latest
```

```
cd project-name
npm install
```

1.

Run Development Server:

```
npm run dev
```

2.

Set up JSON Server (Mock API):

```
json-server --watch ./db/db.json
```

3.

Install Required Dependencies:

```
npm install react-router-dom axios firebase bootstrap
```

4.

10. Milestone 2: Project Development

Setting Up Routing:

```
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Songs from './components/Songs';
import Playlist from './components/Playlist';
import Favorites from './components/Favorites';
import Login from './components/Login';
import Register from './components/Register';

function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<Songs />} />
        <Route path="/playlist" element={<Playlist />} />
        <Route path="/favorites" element={<Favorites />} />
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Register />} />
      </Routes>
    </Router>
  );
}
export default App;
```

11. Fetching Songs from JSON Server

```
import { useState, useEffect } from 'react';
import axios from 'axios';

const [songs, setSongs] = useState([]);
useEffect(() => {
  axios.get('http://localhost:3000/items')
    .then(response => setSongs(response.data))
    .catch(error => console.error("Error fetching songs: ", error));
}, []);
```

12. Adding & Removing Songs from Playlist

```
const addToPlaylist = (itemId) => {
  axios.post('http://localhost:3000/playlist', { id: itemId })
    .then(() => setPlaylist([...playlist, itemId]));
};

const removeFromPlaylist = (itemId) => {
  axios.delete(`http://localhost:3000/playlist/${itemId}`)
    .then(() => setPlaylist(playlist.filter(id => id !== itemId)));
};
```

13. User Authentication with Firebase

```
import { getAuth, signInWithEmailAndPassword } from "firebase/auth";

const auth = getAuth();
const handleLogin = async (email, password) => {
  try {
    await signInWithEmailAndPassword(auth, email, password);
    console.log("User logged in");
  } catch (error) {
    console.error("Login failed: ", error);
  }
};
```

14. Running the Application

1. **Start React Application:** `npm start` or `npm run dev`
2. **Run JSON Server:** `json-server --watch ./db/db.json`
3. **Launch Rhythmic Tunes in Browser.**

15. Project Structured

    - **Home Page**
    - **Playlist View**
    - **Favorites View**
    - **Login Page**
    - **Song Recommendations**

Thank you !