Frontend Development with React.js

Project Documentation: RhythmicTunes

Introduction

Project Title: RhythmicTunes – Your Melodic Companion

Team Leader: P. Sivakumar

Team Members: E. Rekha, D. Dharshini, K. Jeevitha

Project Overview

Purpose: RhythmicTunes is a music streaming application designed to provide a seamless and enjoyable listening experience. It allows users to discover, organize, and play music while offering advanced features like personalized recommendations, playlists, and offline access.

Features:

- Song Listings: Displays a list of songs with details (title, artist, genre, release date).
- Playlist Creation: Allows users to create and manage playlists.
- Playback Control: Provides play, pause, skip, and volume adjustment options.
- Offline Listening: Users can download songs for offline playback.
- Search Functionality: Enables quick searches by song, artist, or album.

Architecture

Component Structure:

- App.js – Root component
- Songs.js – Displays a list of songs
- Playlist.js – Manages user-created playlists
- Favorites.js – Shows favorite songs
- Sidebar.js – Navigation menu

State Management: Uses useState and useEffect for local state management.

Routing: Uses React Router for navigation between pages.

Setup Instructions

Prerequisites:

- Node.js & npm: Required for package management and running the app.

- React.js: Core framework.

- Development Tools: Git, VS Code, Sublime, WebStorm.

Installation:

1. Clone the repository.

2. Install dependencies using 'npm install'.

3. Start the development server using 'npm run dev'.

4. Start JSON server with 'json-server --watch ./db/db.json'.

5. Open the app at http://localhost:5173.

Folder Structure

```
rhythmic-tunes/
|── src/
|   |── components/  # Reusable UI components
|   |── pages/       # Main pages (Songs, Playlist, Favorites)
|   |── assets/      # Images, icons, and styles
|   |── utils/       # Helper functions and API calls
|── public/
|── db/              # JSON server database
|── package.json
|── README.md
```

Running the Application

Start the frontend: 'npm run dev'

Start the backend (JSON server): 'json-server --watch ./db/db.json'

Component Documentation

Key Components:

- Songs.js – Displays available songs.

- Playlist.js – Manages user-created playlists.

- Favorites.js – Allows users to add/remove favorite songs.

- AudioPlayer.js – Handles music playback.

State Management

Global State:

- Future integration with Redux or Context API for state persistence.

Local State:

- useState and useEffect manage song data, playlist updates, and search filters.

User Interface

Fully responsive for desktop, tablet, and mobile.

Uses Bootstrap and Tailwind CSS for styling.

Dark mode support (future enhancement).

Testing

Testing Strategy:

- Uses Jest and React Testing Library for unit testing.
- Cypress for end-to-end testing (planned).

Code Coverage:

- Enforced via Jest's '--coverage' flag.

Screenshots or Demo

Demo Link: https://drive.google.com/file/d/1-jSQg705jDpn5mmzuufQmkFTHpLtRgOl/view?usp=drivesdk

Known Issues

- Some UI components need optimization for performance.
- JSON server requires manual setup for persistent data.

Future Enhancements

- AI-Powered Music Recommendations
- Lyrics Integration
- User Authentication & Profile Management
- Social Sharing Features