Project Documentation

Introduction

Project Title: Rhythmic Tunes – Your Melodic Companion

Project leader:Thrisha V

Project member:Vaijanthimala P

Project member;Vinothini K

---

Project Overview

Purpose:

The project Rhythmic Tunes is a music-based companion web app designed to provide users with a smooth and interactive experience to explore, play, and enjoy music. Its goal is to act as a digital “melodic companion” that adapts to user preferences and provides a seamless listening journey.

Features:

Search and play songs using an API integration.

User-friendly music player with play, pause, next, and previous functionality.

Playlist creation and management.

Responsive design for desktop and mobile users.

Dark and light theme toggle for better user experience.

---

Architecture

Component Structure:

App.js – Main entry point handling routing and global state.

Navbar – Navigation bar with links to Home, Playlists, and Settings.

MusicPlayer – Core music player with controls and progress bar.

SongList – Displays songs fetched from the API.

PlaylistManager – Manages user-created playlists.

ThemeToggle – Switches between dark/light themes.

State Management:

Used Context API to handle global state such as current song, theme preference, and playlists.

Local states for component-specific interactions (e.g., song progress, form inputs).

Routing:

Implemented using react-router-dom:

/ → Home Page

/playlists → Playlist Manager

/settings → Preferences

---

Setup Instructions

Prerequisites:

Node.js (v16 or above)

npm or yarn

Installation:

1. Clone the repository:

git clone <https://github.com/yourusername/rhythmic-tunes.git>

2. Navigate to the client folder:

cd rhythmic-tunes/client

3. Install dependencies:

npm install

4. Start the development server:

npm start

---

Folder Structure

Client:

client/

  ├── public/

  ├── src/

  │ ├── components/

  │ │ ├── Navbar.js

  │ │ ├── MusicPlayer.js

  │ │ └── ThemeToggle.js

  │ ├── pages/

  │ │ ├── Home.js

  │ │ ├── Playlists.js

  │ │ └── Settings.js

  │ ├── assets/ (icons, images, music samples)

  │ ├── context/ (global state management)

  │ ├── App.js

  │ └── index.js

Utilities:

useLocalStorage.js → Custom hook to persist data in local storage.

api.js → Helper functions to fetch songs from external API.

---

Running the Application

To start the frontend server locally:

npm start

---

Component Documentation

Key Components:

MusicPlayer: Controls playback (play, pause, next, prev). Props: currentSong, onPlay, onPause.

SongList: Displays songs. Props: songs, onSelectSong.

PlaylistManager: Handles creation and storage of playlists.

Reusable Components:

Button (custom styled)

Card (for song and playlist display)

---

State Management

Global State:

Current playing song

Playlists

Theme preference

Local State:

Volume control inside MusicPlayer

Input fields in PlaylistManager

---

User Interface

Modern, clean design with responsive layouts.

Simple navigation with a persistent music player.

---

Styling

Framework: Tailwind CSS for utility-first styling.

Theming: Dark/light mode toggle with Context API.

---

Testing

Testing Strategy:

Unit testing for components using Jest + React Testing Library.

Integration testing for routing and global state updates.

Code Coverage:

Jest coverage reports ensure all major components are tested.

---

Known Issues

Some APIs may have rate limits causing search delays.

Playlist persistence depends on local storage (cleared if cache reset).

---

Future Enhancements

Add lyrics display feature.

Integration with Spotify API.

User authentication for personalized playlists.

Offline playback support.

--