

Project Documentation

RhythmicTunes: Your Melodic Companion

1. Introduction

- RhythmicTunes: Your Melodic Companion
- Team ID : NM2025TMID33654
- Team Size : 5
- Team Leader : MOHAMED ISTHI.A (mohamedisthi19@gmail.com)
- Team member : MURALI.P (pakkirisami9047@gmail.com)
- Team member : PREMKUMAR.N (nrpremkumar67@gmail.com)
- Team member : PRIZEGOD.V (prizegod2616@gmail.com)
- Team member : VEERRAMANI.G (vv1238486@gmail.com)

2. Project Overview

Name: Rhythmic Tunes

Type: Frontend Web Application

Framework: React.js

Purpose: Deliver an immersive music experience with interactive playback, playlists, and personalization.

Key Features: Music playback, playlist management, favorites, responsive UI, theme toggle.

—

3. Architecture

Frontend: React.js (Functional Components, Hooks)

Routing: React Router

State Management: Redux / Context API

Styling: Tailwind CSS / Material UI

Music API: Integration with third-party APIs or local JSON data

Deployment: Netlify /

Vercel Diagram (optional):

User → React UI → State Management → Audio API → External Music API (optional)

—

4. Setup Instructions

Prerequisites

Node.js (v16+)

npm or yarn

Steps

Clone repository

git clone https://github.com/username/rhythmic-

tunes.git # Navigate into project

cd rhythmic-tunes

```
# Install dependencies
npm install

# Start development server
npm start

# Build production
files npm run build
```

—

5. Folder Structure

Rhythmic-Tunes/

```
| — public/      # Static assets (favicon, index.html)
| — src/
|   └— assets/   # Images, icons
|   └— components/ # Reusable UI components (Player, Navbar, Sidebar)
|   └— pages/    # Application pages (Home, Library, Playlist)
|   └— context/  # Context API providers (if used)
|   └— store/    # Redux setup (if used)
|   └— hooks/    # Custom React hooks
|   └— utils/    # Helper functions
|   └— App.js    # Main app entry
|   └— index.js  # React entry point
| — package.json
| — README.md
```

6. Running the Application

Development Mode: `npm start`

Production Build: `npm run build`

Preview Production Build: `npm run preview`

7. Component Documentation

Navbar: Provides navigation between pages.

Sidebar: Contains playlists and quick links.

MusicPlayer: Handles play, pause, next, previous, volume.

SongCard: Displays individual song details.

SearchBar: Allows users to search for music.

ThemeToggle: Switch between light and dark mode.

8. State Management

Global state handled via Redux or Context API.

Stores

Current song

Playlist data

Playback state (playing/paused)

User preferences (theme, favorites)

—

9. User Interface

Home Page: Displays featured music.

Library: Shows playlists and saved songs.

Player Page: Full-screen music player with controls.

Responsive Design: Works on desktop, tablet, and mobile.

—

10. Styling

Tailwind CSS for utility-first styling.

Theme Support: Dark/Light mode with CSS variables.

Consistent typography and color scheme

—

11. Testing

Unit Tests: Jest + React Testing Library.

Integration Tests: Component rendering & state changes.

End-to-End Tests: Cypress (optional).

—

12. Screenshots or Demo

(Insert screenshots or provide a live demo link here)

—

13. Known Issues

Limited offline playback support.


API rate limiting when using third-party services.


Some UI components may need optimization on smaller screens.

—


14. Future Enhancements

🔊 Voice command support.

 AI-based song recommendations.

 Multi-language support.

 User authentication & profiles.

 Mobile app version.