# Music Player Frontend Documentation

## 1. Introduction

### Project Title: RYTHEMIC TUNES – An music streaming platform

### Team Leader:

* Dhanraj S

### Team Members:

* Boopathy.G
* Manoj A
* Surendar V
* Madhumitha R

## 2. Project Overview

### Purpose

The **Music Player Frontend** is a React-based web application built using Vite. It provides an interactive UI for users to browse, play, and manage music tracks with a smooth experience.

### Features

* Music playback with play, pause, and next/previous controls.
* Dynamic UI built using React and Tailwind CSS.
* User-friendly navigation and interactive design.
* JSON-based music database (powered by json-server).
* Dark and light mode support.

## 3. Architecture

### Component Structure

* **App Component:** The root component managing global state and layout.
* **Navbar Component:** Handles navigation between sections.
* **Player Component:** Manages audio playback.
* **TrackList Component:** Displays the list of available music tracks.
* **Favorites Component:** Allows users to save favorite songs.

### State Management

* Uses React's useState and useEffect for local state management.

### Routing

Utilizes **React Router** for page navigation:

* / - Home page
* /tracks - Track listing
* /favorites - Saved songs
* /settings - User settings

## 4. Setup Instructions

### Prerequisites

* **Node.js** (v16+)
* **npm** or **yarn**

### Installation

1. Clone the repository:

git clone https://github.com/your-repo/music-player-frontend.git

1. Navigate to the project directory:

cd music-player-frontend

1. Install dependencies:

npm install

## 5. Folder Structure

music-player-frontend/

│── public/

│── src/

│ ├── components/

│ ├── pages/

│ ├── assets/

│ ├── store/

│ ├── utils/

│ ├── App.jsx

│ ├── main.jsx

│── .env

│── package.json

│── vite.config.js

## 6. Running the Application

To start the development server:

npm run dev

## 7. Component Documentation

### Key Components

* **TrackList:** Displays a list of available music tracks.
* **Player:** Controls audio playback.
* **SearchBar:** Allows users to search for songs.

### Reusable Components

* **Button:** Customizable button component.
* **Card:** Styled container for displaying data.

## 8. State Management

### Global State

* Uses React Context API for theme switching and user preferences.
* JSON-based database (json-server) for storing tracks and user data.

### Local State

* useState for UI-based interactions (e.g., volume control, play/pause button).

## 9. User Interface

* Responsive design supporting mobile and desktop views.
* Built with Tailwind CSS for styling.

## 10. Styling

### CSS Frameworks/Libraries

* **Tailwind CSS** for component styling.
* **Styled Components** for dynamic theming.

### Theming

* Supports **light and dark modes** using context-based theming.

## 11. Testing

### Testing Strategy

* **Unit Testing:** Jest & React Testing Library for component tests.
* **Integration Testing:** Ensuring API data rendering correctly.
* **End-to-End Testing:** Cypress for user interaction testing.

### Code Coverage

* Jest is configured for test coverage reports.

## 12. Screenshots or Demo

* [Insert screenshots or a link to the live demo]

## 13. Known Issues

* API rate limits might affect real-time updates.
* Mobile responsiveness improvements in progress.

## 14. Future Enhancements

* Integration with third-party music APIs (Spotify, SoundCloud, etc.).
* AI-based song recommendations.
* Offline mode for playing downloaded tracks.