■ **Music Player Web App - Full Documentation**

**Team ID:** NM2025TMID39227

**TEAM SIZE**: 4

**TEAM LEADER**: THUKKARAM S

**TEAM MEMBER**: RAHAMAN SHERIFF B

**TEAM MEMBER**: YOGESHWAR A

**TEAM MEMBER**: DHANUSH L

# ■ Introduction

The Music Player Web App is designed to provide users with an easy-to-use platform for playing and managing music directly in the browser. It supports uploading songs from the local system, customizing album art, and controlling playback seamlessly.

# ■ Project Overview

This project was built using React + Vite for modern development speed and efficiency. It demonstrates core web development concepts including component-based design, state management, and responsive UI. The app is lightweight, fast, and extendable.

# ■■ Architecture

The application follows a modular React component architecture: - **App.js**: Root component that manages layout. - **MusicPlayer.js**: Core player component with controls. - **SongList.js**: Handles playlist rendering. - **Upload.js**: Manages adding songs and album art. State is passed down via props, ensuring a unidirectional data flow.

# ■■ Setup Instructions

1. Download and extract the project zip file.
2. Navigate to the project folder in terminal: cd music-player
3. Install dependencies: npm install
4. Start development server: npm run dev
5. Access the app via: http://localhost:5173

# ■ Folder Structure

music-player/ ■ public/ ■ ■ covers/ (default and uploaded covers) ■ src/ ■ ■ components/ ■ ■ ■

MusicPlayer.jsx ■ ■ ■ SongList.jsx ■ ■ ■ Upload.jsx ■ ■ App.jsx ■ ■ main.jsx ■ package.json ■ vite.config.js

# ■■ Running the Application

Once the setup is complete, run `npm run dev`. The application will start on http://localhost:5173. Users can upload songs, add covers, and control playback through the UI.

# ■ Component Documentation

* App.jsx – Main layout and entry point.
* MusicPlayer.jsx – Handles audio context, controls, and playback logic.
* SongList.jsx – Renders playlist with song details and album art.
* Upload.jsx – Provides input for adding songs and covers.

# ■ State Management

The app uses React's useState and useEffect hooks for state management. Uploaded songs are stored in a playlist array. Playback state (current song, progress, isPlaying) is managed at the MusicPlayer level.

# ■ User Interface

The UI is clean and minimal, focusing on usability. It provides clear controls for Play, Pause, Next, Previous, and progress tracking. Album art is displayed dynamically based on user uploads.

# ■ Styling

The app uses custom CSS for styling. Flexbox ensures responsive layouts, and hover effects are applied for interactivity. Colors are chosen for readability and accessibility.

# ■ Testing

Manual testing was conducted to ensure: - Songs upload and play correctly. - Custom album art is displayed properly. - Playback controls (Play, Pause, Next, Prev) function as expected. Responsive behavior across devices.

■■ **Demo Video**

■ Demo Link: https://drive.google.com/file/d/1Iv1ZYLLjfGz2PWaFxU0Yjrc2axAJxb6r/view?usp=sharing

# ■ Known Issues

* No backend integration – songs reset on page refresh.
* Large files may take time to load depending on browser.
* Limited browser testing – optimized for Chrome and Edge.

# ■ Future Enhancements

* Persistent playlist storage using localStorage or database.
* Support for multiple audio formats (e.g., WAV, FLAC).
* Improved mobile-first design.
* Online streaming support.
* Playlist sharing and collaboration features.