📑 **RhythmicTunes - Project Documentation**

Frontend Development with React.js

**Project Title:** RhythmicTunes: Your Melodic Companion

🔹 **Introduction:**

RhythmicTunes: Your Melodic Companion is a music streaming application built with React.js, designed to provide users with a seamless and engaging way to enjoy music. The platform allows users to explore songs, create playlists, and save favorites through a modern, responsive, and user-friendly interface. By combining powerful functionality with an intuitive design, RhythmicTunes aims to redefine the digital music experience for enthusiasts of all kinds.

🤝 **Team ID: NM2025TMID46597**

🤝 **Team Leader:**

**PRAVEEN KUMAR AM Email id: ampraveenkumar2007@gmail.com**

🤝 **Team Members:**

**SELVAM V Email id:** [**selvamselvam007@gmail.com**](mailto:selvamselvam007@gmail.com)

**PRAVEENSHREE B Email id: praveenshreeb@gmail.com**

**MUTHUVEL A Email id: muthuvel2612@gmail.com**

**VINETH KUMAR S Email id: vkvinith82@gmail.com**

🎯 **Project Overview:**

**Purpose:**

The purpose of RhythmicTunes is to provide a seamless, engaging, and user-friendly music streaming platform built with React.js. It allows users to explore, stream, and manage their favorite tracks with an intuitive interface.

**Features:**

**\*** Browse and play songs with integrated audio controls.

**\*** Add/remove songs from Wishlist (Favorites).

**\*** Create and manage Playlists.

**\*** Advanced search functionality (search by name, singer, or genre).

**\*** Responsive design for desktop, tablet, and mobile.

**\*** Seamless routing between pages (Songs, Favorites, Playlist).

🏗️ **Architecture:**

**Component Structure:**

**\*** App Component → Root component, manages routing and layout.

**\*** Sidebar Component → Navigation panel.

**\*** Songs Component → Displays list of songs, search bar, and actions.

**\*** Favorites Component → Shows songs added to wishlist.

**\*** Playlist Component → Displays and manages playlist items.

**\*** Reusable Components → Buttons, Search Input, Card UI, Audio Player.

**State Management:**

**\* Local State:** useState for managing search term, currently playing song, etc.

**\* Global State:** Managed through props drilling (can be extended later to Context API or Redux if needed).

**Routing:**

Implemented using react-router-dom:

/ → Songs Component

/favorites → Favorites Component

/playlist → Playlist Component

⚙️ **Setup Instructions:**

**Prerequisites:**

**\*** Node.js & npm → Download Here

**\*** Git for version control → Download Here

**\*** Code Editor (VS Code recommended)

**Installation:**

# Clone the repository

git clone <repository\_url>

# Navigate to project directory

cd tunes

# Install dependencies

npm install

Start the Application

# Start React app

npm start

# Start JSON server (in a separate terminal)

json-server --watch ./db/db.json

Access at: [http://localhost:3000](http://localhost:3000/)

📂 **Folder Structure:**

**tunes/**

│── public/

│── src/

│ ├── assets/ # Images, icons, etc.

│ ├── components/ # Reusable components (Sidebar, SearchBar, etc.)

│ ├── pages/ # Songs, Favorites, Playlist

│ ├── utils/ # Helper functions, axios instance, hooks

│ ├── App.js # Root component

│ ├── App.css # Global styles

│ └── index.js # Entry point

│── db/

│ └── db.json # Mock JSON server database

│── package.json

▶️ **Running the Application:**

**\*** **Frontend:** Run npm start in project root.

**\*** **Backend (Mock):** Run json-server --watch ./db/db.json.

🧩 **Component Documentation:**

**Key Components:**

**\*** Songs.js → Displays songs list, handles search and playback.

**\*** Favorites.js → Renders saved favorite songs.

**\*** Playlist.js → Shows user’s playlist.

**Reusable Components:**

**\*** Card Component → Displays song details with actions.

**\*** Button Component → Custom styled buttons for actions (wishlist/playlist).

**\*** Search Input → Search bar with icon.

🔄 **State Management:**

**Global State:**

**\*** Data fetched from JSON server (items, favorites, playlist).

**\*** Controlled by axios requests and component states.

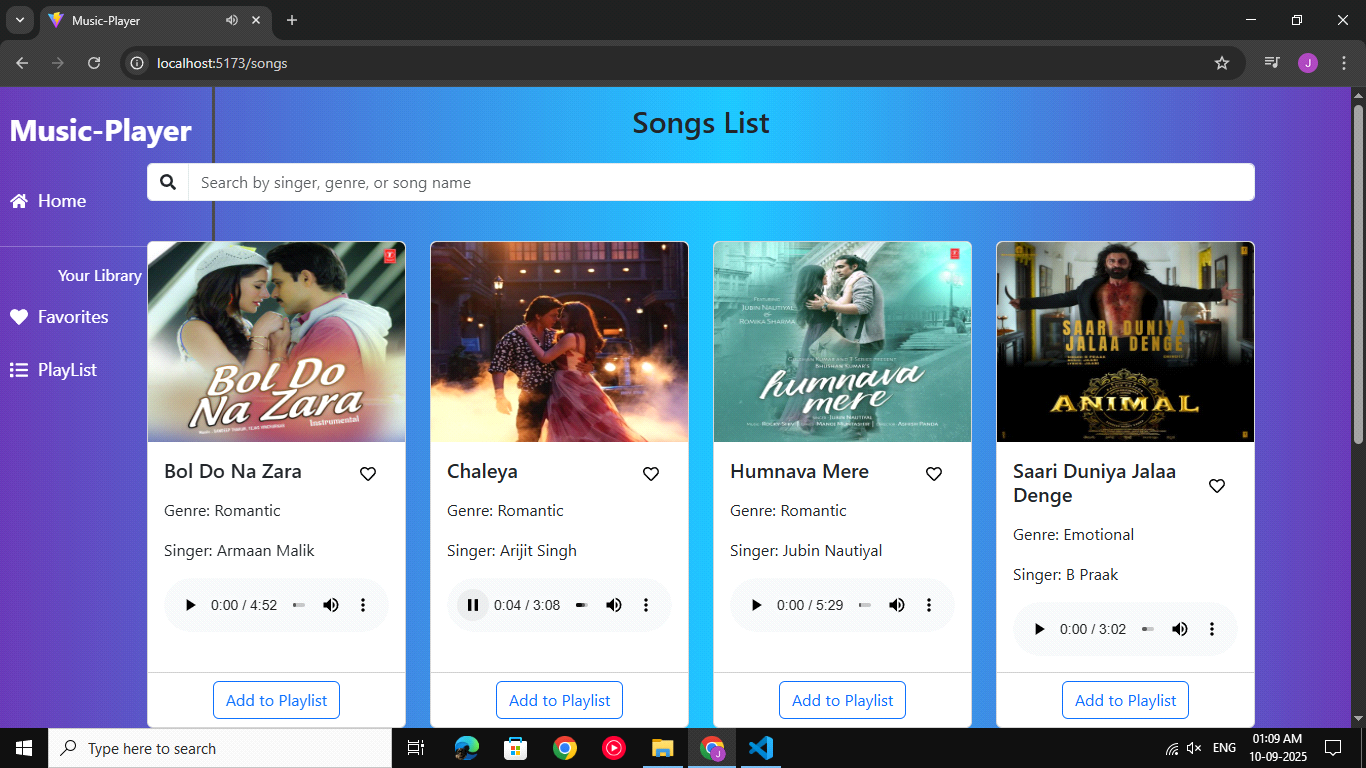
**Local State:**

**\*** searchTerm → Stores query for filtering songs.

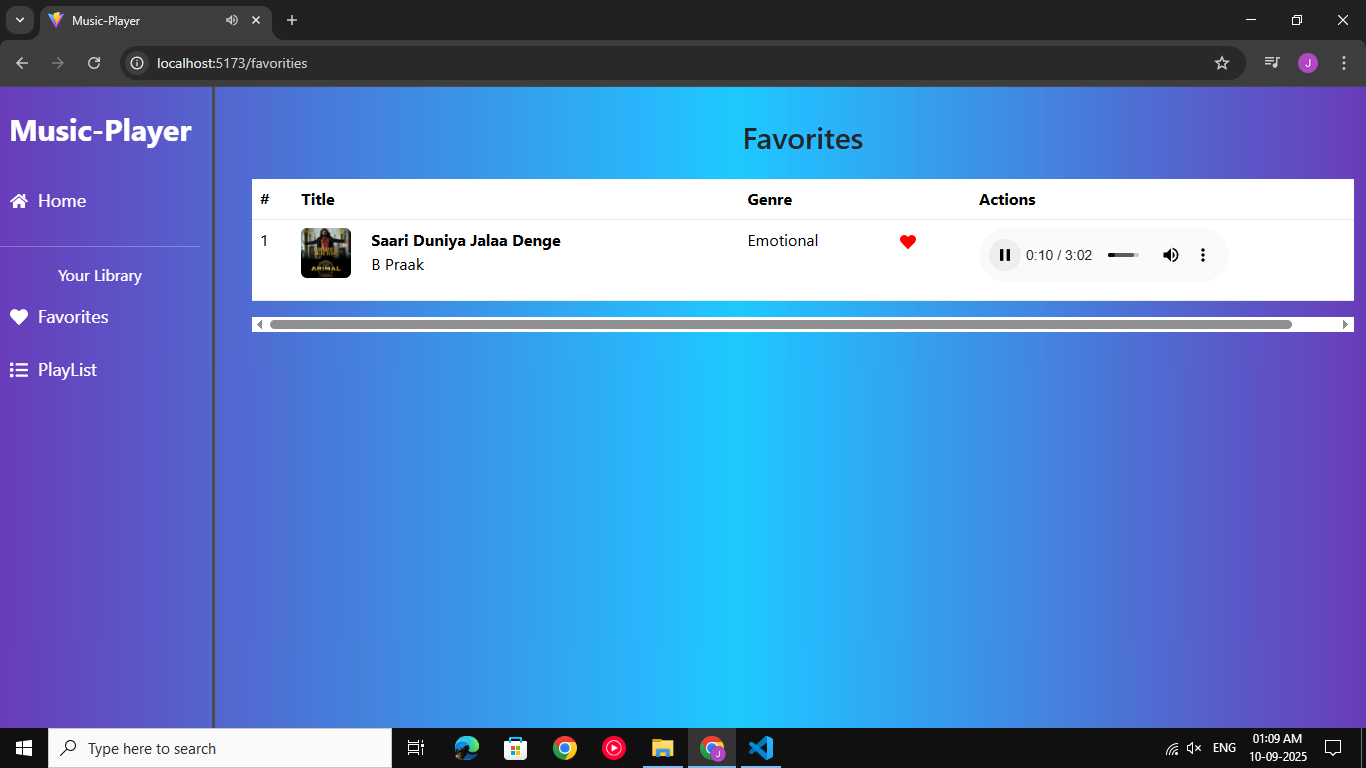
**\*** currentlyPlaying → Keeps track of active audio player.

🎨 **User Interface:**

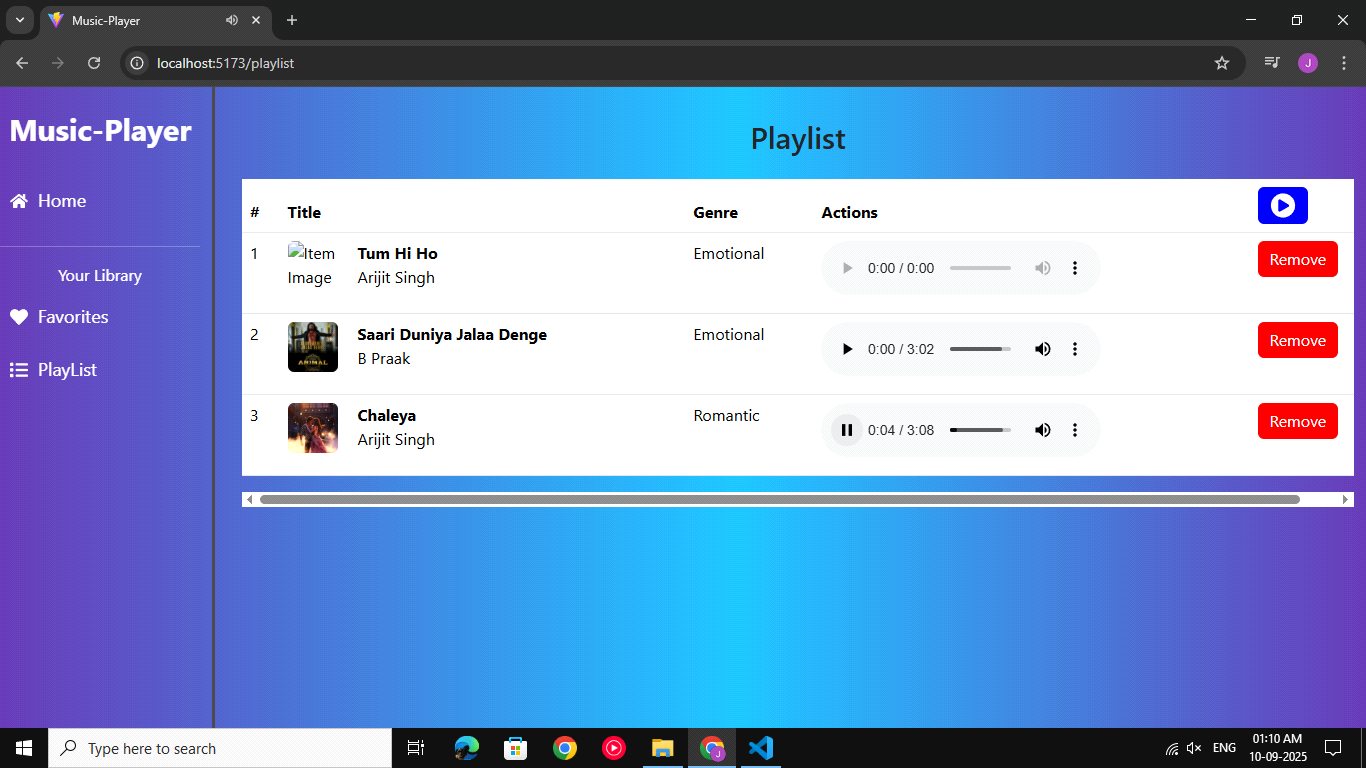
🖼 **Screenshots added here :**

**Homepage (Songs List):**  


**Favorites Page:**



**Playlist Page:**



🎨 **Styling:**

**CSS Frameworks/Libraries:**

**\*** Bootstrap (for responsive layout and grid system).

**\*** Tailwind CSS (for modern utility-first styling).

**Theming:**

**\*** Dark/Light mode not implemented (future enhancement).

**\*** Consistent music-themed design with icons and audio player styling.

🧪 **Testing:**

**Testing Strategy:**

**\*** Unit Tests: For reusable components (Buttons, Cards, Search Input).

**\*** Integration Tests: Ensuring Songs component interacts correctly with Favorites/Playlist.

**\*** End-to-End Testing: Can be implemented using Cypress for user flows.

**Code Coverage:**

Basic tests can be implemented with Jest + React Testing Library.

🐞 **Known Issues:**

**\*** Songs may overlap if multiple audio players are triggered.

**\*** No authentication system yet (data resets on refresh).

**\*** No API integration (currently using JSON server mock data).

🚀 **Future Enhancements:**

**\*** User authentication & profile management.

**\*** Integration with real music APIs (Spotify/Apple Music).

**\*** Offline mode for downloaded tracks.

**\*** Dark/Light theme support.

**\*** Advanced recommendations powered by AI.