**RYTHIMIC TUNES**

**PROJECT REPORT**

**SUBMITTED TO NAAN MUDHALVAN IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE MONGODB DATABASE ADMIN PATH(Self-Managed) For SMARTBRIDGE**

**BY**

**TEAM LEADER:**

**R. VIMAL ADITHAN (vimaladhithanvimal@gmail.com)**

**TEAM MEMBERS:**

**D. MADHAN (maraa76ammu@gmail.com)**

**P. YOGESH (yogesharul100@gmail.com)**

**M. NIRMAL KUMAR (nirmalkumar1812004@gmail.com)**

**M. SWETHA (swethamoorthy2020@gmail.com)**

**Department of Computer Science**

****

**DEPARTMENT OF COMPUTER SCIENCE**

**PROF.DHANAPALAN COLLEGE OF**

**SCIENCE & MANAGEMENT – KELAMBAKKAM.**

**MARCH - 2025**

**Rythimic Tunes** (React)

**Introduction:-**

Immerse yourself in the future of music with our state-of-the-art streaming platform, meticulously crafted using React.js. We’ve prioritized a seamless blend of groundbreaking technology and an intuitively designed interface, aiming to revolutionize how you engage with sound and personalize your listening experience.

Explore the vibrant world of music, from the latest chart-topping sensations to the cherished melodies of the past, all within our dynamic React-powered application. Its responsive design ensures a visually engaging and consistently smooth experience across all your devices, turning every click and scroll into a moment of musical discovery.

Break free from the constraints of traditional music listening. Our platform unlocks a vast universe of sonic possibilities, empowering you to effortlessly curate personalized playlists, delve into diverse genres, and rediscover the joy of music tailored to your unique preferences.

Join us on a journey to redefine your connection with music. Elevate your auditory experience and embrace the future of streaming – it's time to press play and embark on a new era of musical exploration.

**Target Audience:-**

Your music streaming app targets tech-savvy young adults who prioritize personalized digital experiences. You'll also reach music enthusiasts seeking high-quality audio and discovery. Students and young professionals need convenient, mobile listening, while others value customized entertainment and intuitive interfaces.

# Social Media Engagers: These users actively share their musical discoveries and engage with artists and playlists on social platforms. They value integration with social media and the ability to share their listening experiences.

# Fitness and Wellness Focused: Individuals who use music to enhance their workouts, meditation, or relaxation routines. They appreciate curated playlists for specific activities and seamless integration with fitness trackers or wellness apps.

# Local Music Scene Supporters: Users who are passionate about discovering and supporting local artists and independent musicians. They seek platforms that showcase emerging talent and provide opportunities to connect with their local music scene.

# Project Goals and Objectives:-

The primary goal of Music Streaming is to provide a seamless platform for music enthusiasts, enjoying, and sharing diverse musical experiences. Our objectives include:

**User-Friendly Interface:** Develop an intuitive interface that allows users to effortlessly explore, save, and share their favorite music tracks and playlists.

**Comprehensive Music Streaming:** Provide robust features for organizing and managing music content, including advanced search options for easy discovery.

**Modern Tech Stack:** Harness cutting-edge web development technologies, such as React.js, to ensure an efficient and enjoyable user experience while navigating and interacting with the music streaming application.

# Key Features:-

**Play Music:**

* Play, pause, skip, and adjust volume.
* Display a simple now playing screen with song title and artist.

**Find Music:**

* Search for songs, artists, and albums.
* Display a list of all 10 songs in the library view.

**Organize Music:**

* Create playlists and sort your music.
* A default playlist containing all 10 songs.

**User Friendly:**

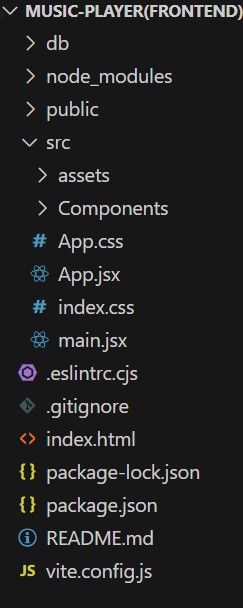
* Easy to use on any device.
* Simple and clean user interface.

**PRE-REQUISITES**:-

**Essential Tools & Knowledge for React Development:**

1. **Node.js & npm (Package Manager):**
   * Required for running JavaScript locally and managing project dependencies.
   * Download & Install: [nodejs.org/en/download/](https://nodejs.org/en/download/)
   * Installation Guides: [nodejs.org/en/download/package-manager/](https://nodejs.org/en/download/package-manager/)
2. **React.js (JavaScript Library):**
   * For building interactive and reusable UI components.
   * Create a new React project: npm create vite@latest (follow prompts).
   * Navigate to project: cd project-name, then install dependencies: npm install.
   * Run development server: npm run dev (access at http://localhost:5173).
3. **Fundamental Web Technologies:**
   * **HTML:** Structure your application's layout.
   * **CSS:** Style and design your user interface.
   * **JavaScript:** Implement client-side interactivity.
4. **Git (Version Control):**
   * Track changes, collaborate, and manage your project's history.
   * Download & Install: [git-scm.com/downloads](https://git-scm.com/downloads)
   * Use platforms like GitHub or Bitbucket for remote repositories.
5. **Code Editor/IDE:**
   * Choose a development environment for writing and managing code.
   * **Examples:** 
     + Visual Studio Code: [code.visualstudio.com/download](https://code.visualstudio.com/download)
     + Sublime Text: [sublimetext.com/download](https://www.sublimetext.com/download)
     + WebStorm: [jetbrains.com/webstorm/download](https://www.jetbrains.com/webstorm/download)

**Project structure:**



**Project Organization: Why it Matters**

* Think of your project like a well-organized toolbox. Everything has its place.
* A good structure makes your code easier to find, fix, and share with others.
* It helps you keep your project clean and manageable as it grows.

**Example: React App Structure and the Main Component**

* In a React project, you'll have folders and files that work together.
* The src folder is where most of your code lives.
* The app folder (or similar) will contain your main component.
* app.component.css: This file contains the style rules for the main component. This controls the look of the root component.
* app.component.js (or .jsx or .tsx): This file contains the javascript code that defines the main component. This main component is what is loaded first, and it handles the overall layout of the application.
* The main component is like the "container" for your entire app. It sets up the basic layout and decides which "pages" (other components) to show.

**Milestone 1: Project Setup and Configuration:**

**1. Install required tools and software:**

* **Installation of required tools**:

1. Open the project folder to install necessary tools In this project, we use:

* + React Js

o React Router Dom

o React Icons o Bootstrap/tailwind css

* + Axios

* For further reference, use the following resources o <https://react.dev/learn/installation>

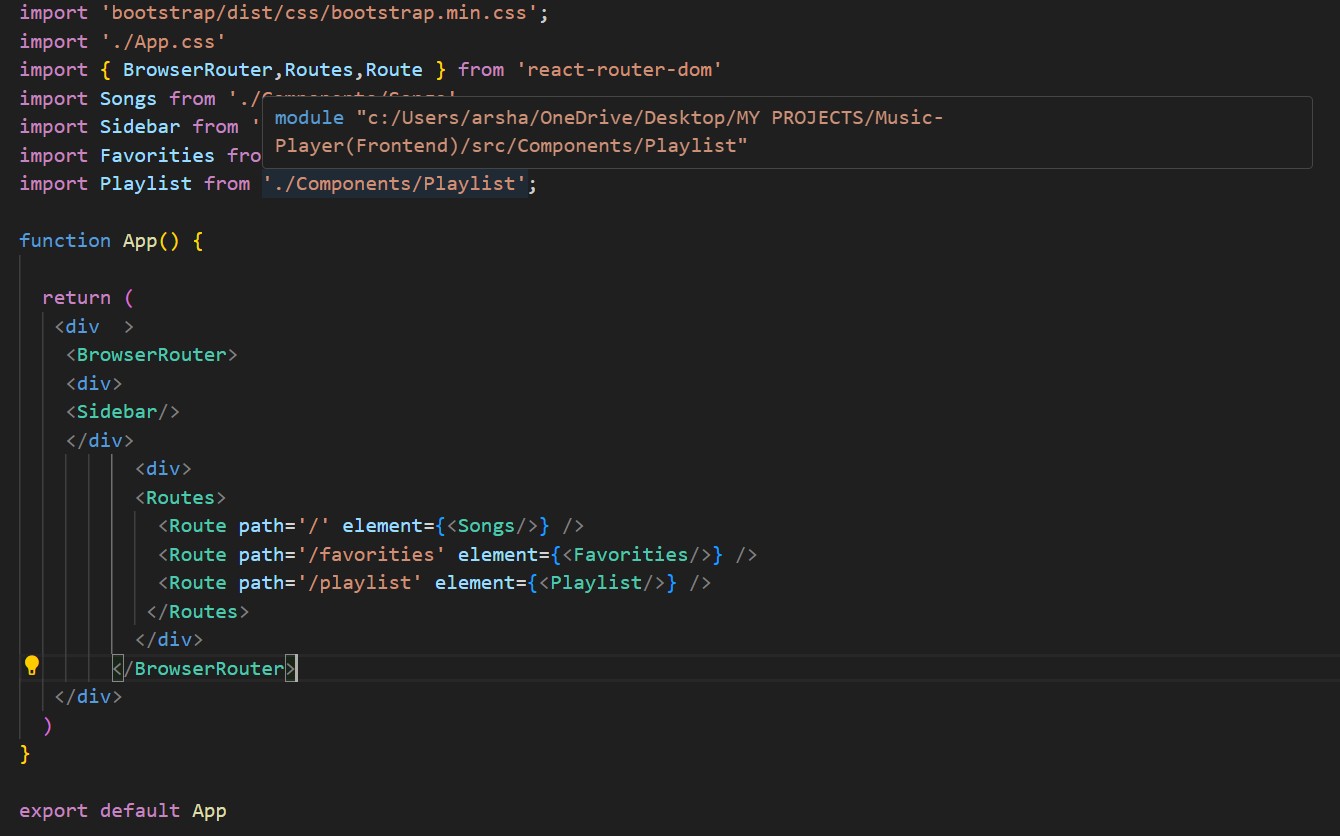
o <https://react-bootstrap-v4.netlify.app/getting-started/introduction/> o <https://axios-http.com/docs/intro> o <https://reactrouter.com/en/main/start/tutorial>

**Milestone 2: Project Development:**

**1. Setup React Application:**

* Create React application.
* Configure Routing.
* Install required libraries.

Setting Up Routes:-

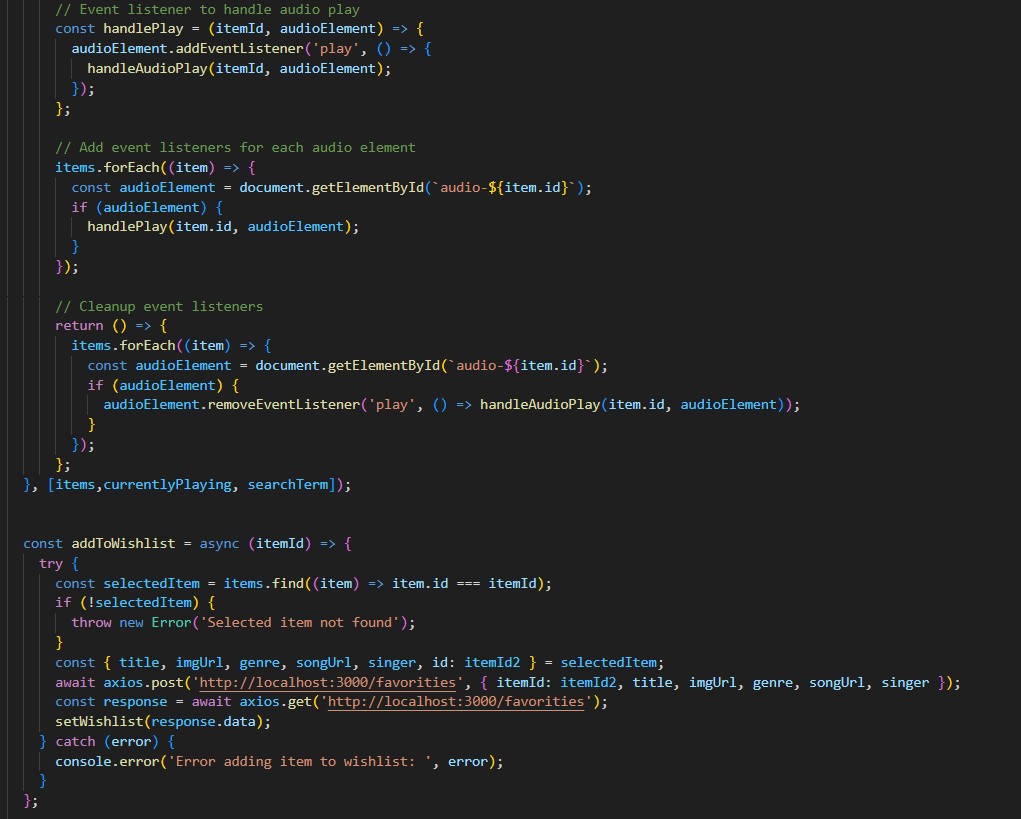
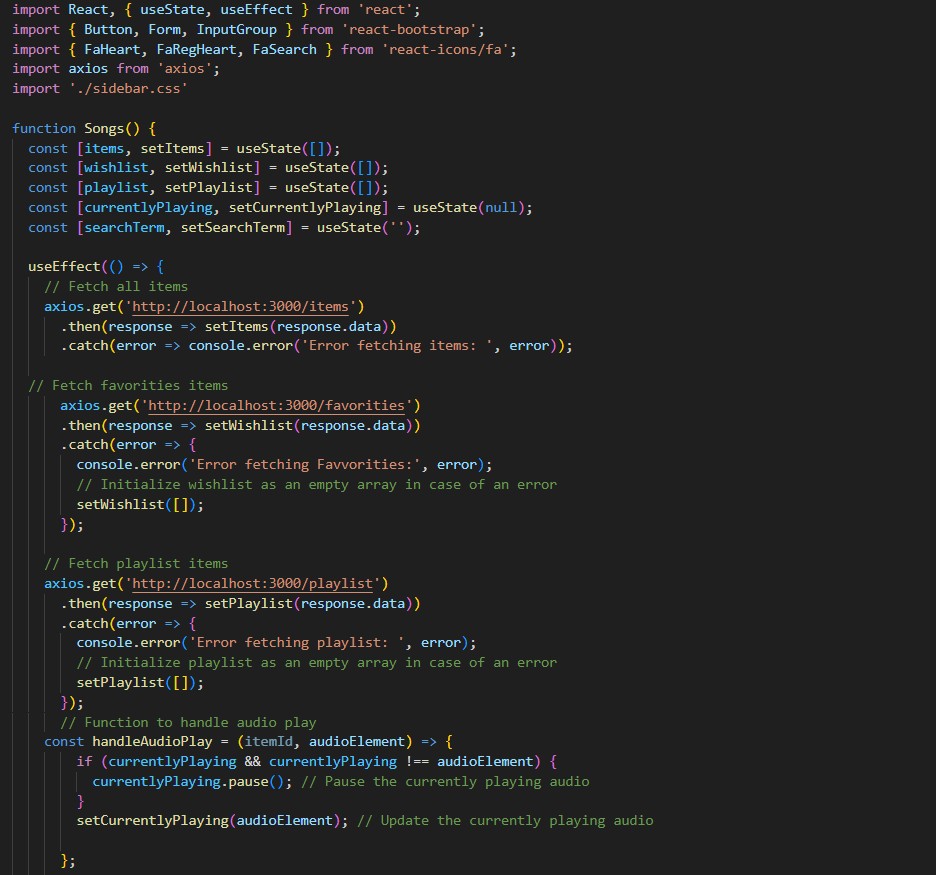


**Code Description:-**

**The App Component: Your Project's Main Hub**

* **Styling:**
  + Uses Bootstrap's CSS for basic styling.
  + Adds custom styles from App.css for further customization.
* **Routing (Navigation):**
  + Uses react-router-dom to manage different "pages" within the app.
  + BrowserRouter sets up the routing system.
  + Routes and Route define which components show up at which URLs.
* **Structure:**
  + The App component is the main container for the entire app.
  + It includes a Sidebar component (for navigation or other content).
  + It also includes Routes to show different content based on the URL.
* **Pages (Routes):**
  + / shows the Songs component.
  + /favorities shows the Favorities component.
  + /playlist shows the Playlist component.
* **Export:**
  + The App component is exported, so other parts of the app can use it.

**Fetching Songs:-**



**Code Description:-**

* + **Use State:**
    - items: Holds an array of all items fetched from http://localhost:3000/items.
    - wishlist: Stores items marked as favorites fetched from http://localhost:3000/favorities.
    - playlist: Stores items added to the playlist fetched from http://localhost:3000/playlist. o currentlyPlaying: Keeps track of the currently playing audio element. o searchTerm: Stores the current search term entered by the user.
* **Data Fetching:**
  + Uses useEffect to fetch data:

▪ Fetches all items (items) from http://localhost:3000/items.

▪ Fetches favorite items (wishlist) from

http://localhost:3000/favorities.

▪ Fetches playlist items (playlist) from

http://localhost:3000/playlist. o Sets state variables (items, wishlist, playlist) based on the fetched data.

* **Audio Playback Management:**
  + Sets up audio play event listeners and cleanup for each item:

▪ handleAudioPlay: Manages audio playback by pausing the currently playing audio when a new one starts.

▪ handlePlay: Adds event listeners to each audio element to trigger handleAudioPlay.

* + Ensures that only one audio element plays at a time by pausing others when a new one starts playing.
* **addToWishlist(itemId):**
  + Adds an item to the wishlist (favorities) by making a POST request to http://localhost:3000/favorities. o Updates the wishlist state after adding an item.
* **removeFromWishlist(itemId):**
  + Removes an item from the wishlist (favorities) by making a DELETE request to http://localhost:3000/favorities/{itemId}. o Updates the wishlist state after removing an item.
* **isItemInWishlist(itemId):**
  + Checks if an item exists in the wishlist (favorities) based on its itemId.
* **addToPlaylist(itemId):**

o Adds an item to the playlist (playlist) by making a POST request to http://localhost:3000/playlist. o Updates the playlist state after adding an item.

* **removeFromPlaylist(itemId):**

o Removes an item from the playlist (playlist) by making a DELETE request to http://localhost:3000/playlist/{itemId}.

o Updates the playlist state after removing an item.

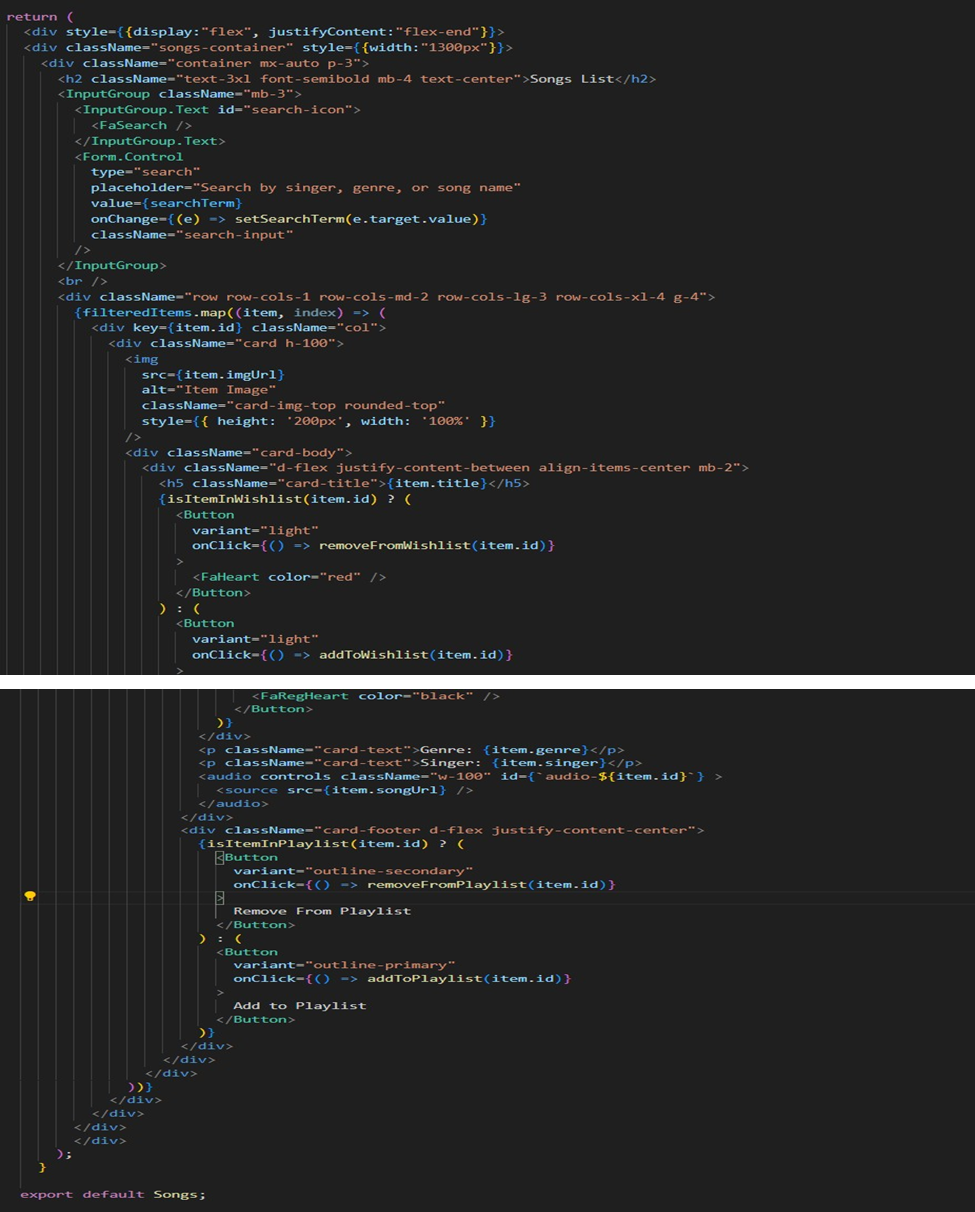
* **isItemInPlaylist(itemId):**

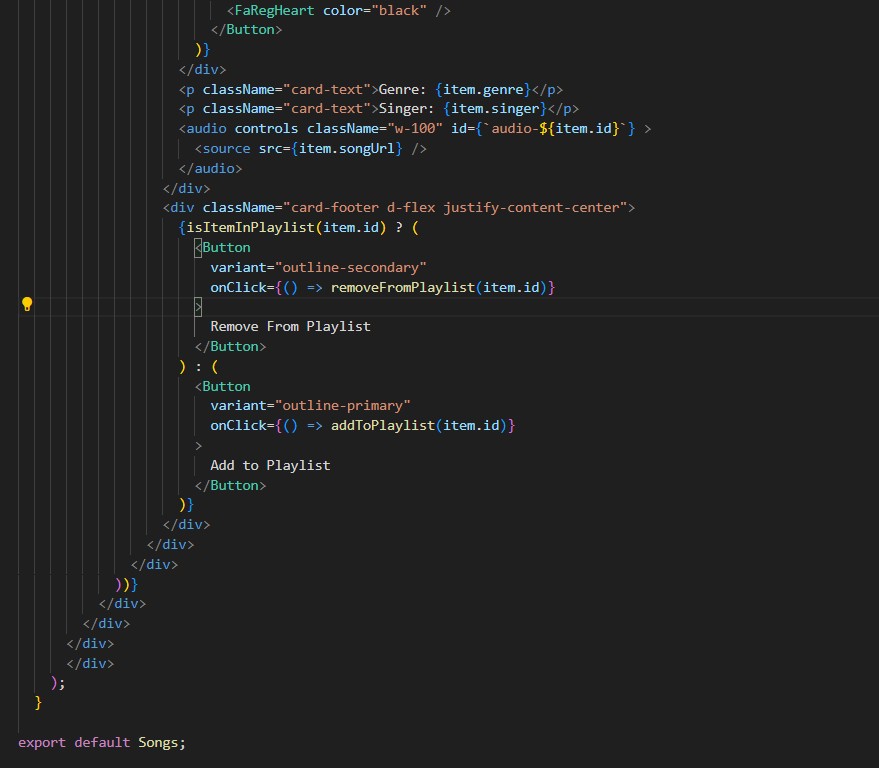
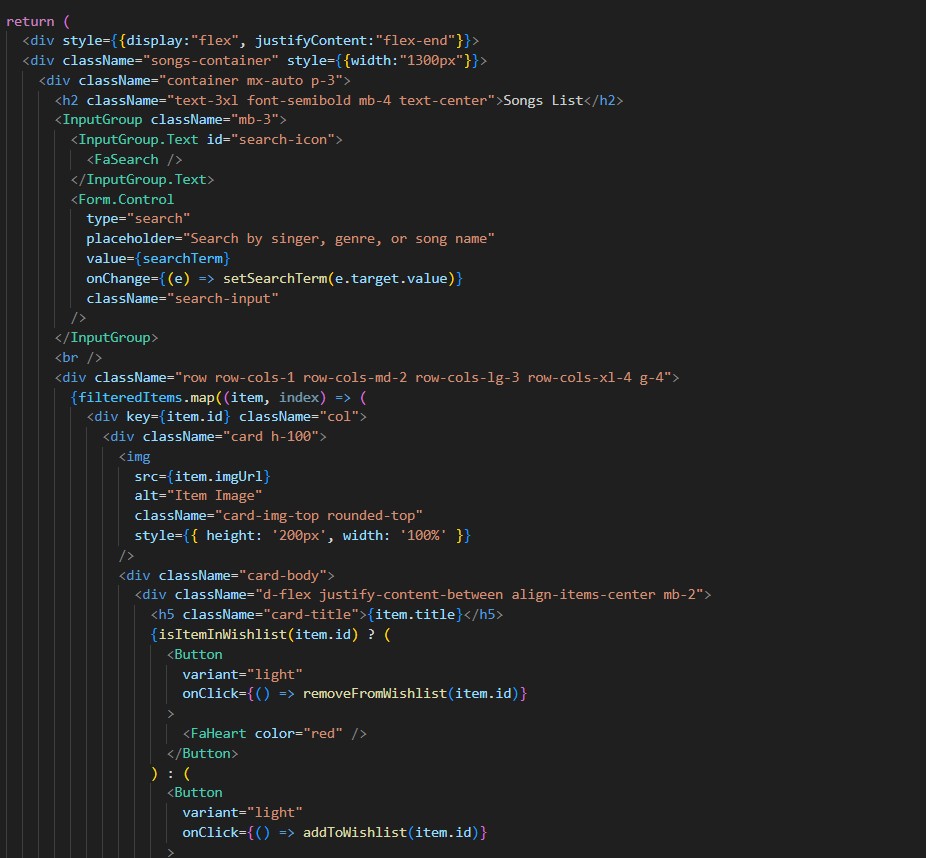
o Checks if an item exists in the playlist (playlist) based on its itemId.

* **filteredItems:**
  + - Filters items based on the searchTerm.
    - Matches title, singer, or genre with the lowercase version of searchTerm.
* **JSX:**
* Search: Search bar for finding items.
* List: Displays matching items.
* Actions: Add/remove from favorites/playlists.
* Audio: Play/pause audio for each item.**Error Handling:**

o Catches and logs errors during data fetching (axios.get).

o Handles errors when adding/removing items from wishlist and playlist.

**Frontend Code For Displaying Songs:-   
  
**



**Code Description:-**

 **Layout:**

* Right-aligned container with a fixed-width section for song display.
* Responsive grid layout using Bootstrap cards.

 **Header & Search:**

* "Songs List" heading.
* Search bar (singer, genre, song name) using React Bootstrap's InputGroup.

 **Song Cards:**

* Displays song information (image, title, genre, singer) in cards.
* Includes an audio player for each song.

 **Wishlist & Playlist:**

* Heart icon button to add/remove songs from the wishlist.
* "Add/Remove from Playlist" buttons.

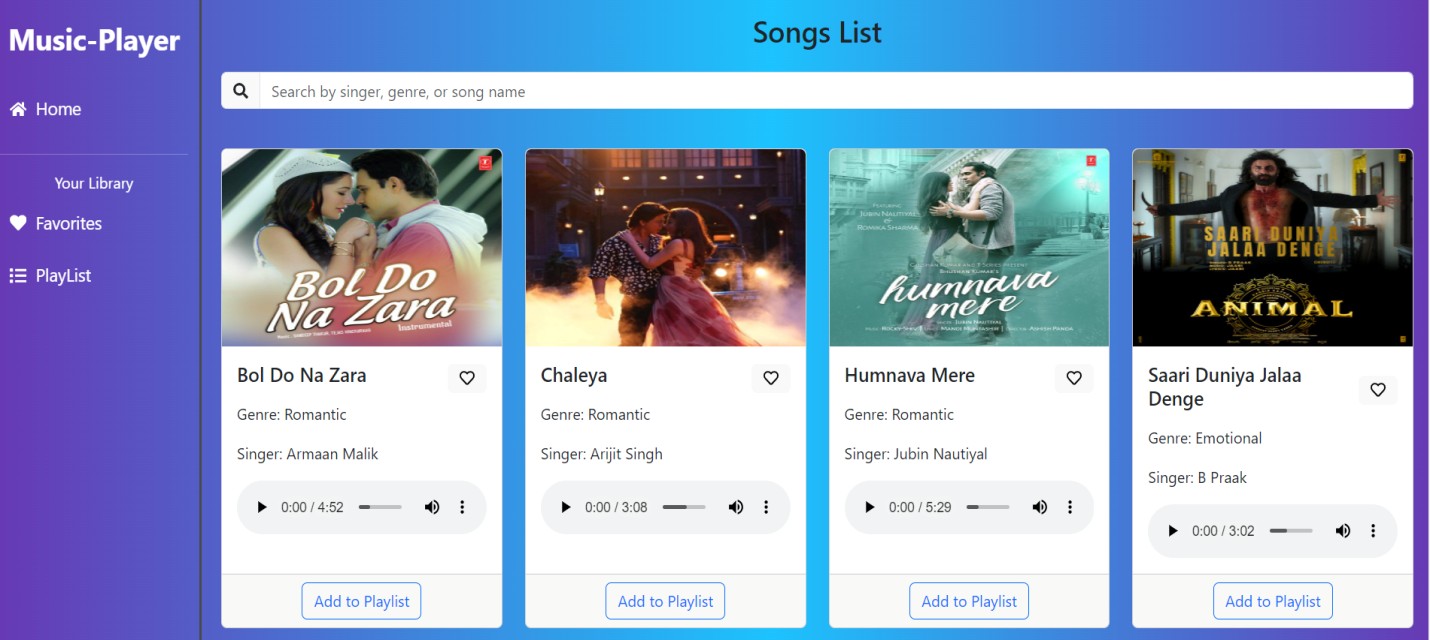
 **Functionality:**

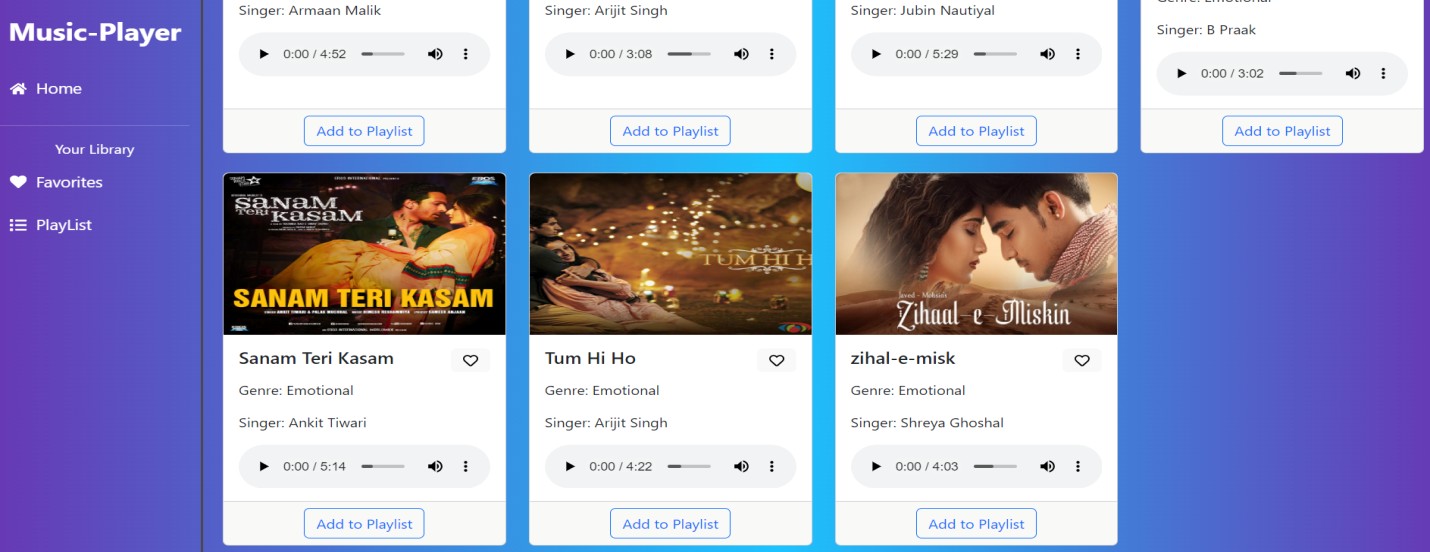
* Handles adding/removing songs from wishlist and playlist.
* Uses Bootstrap and custom styles for card and element appearance.

**Project Execution:**

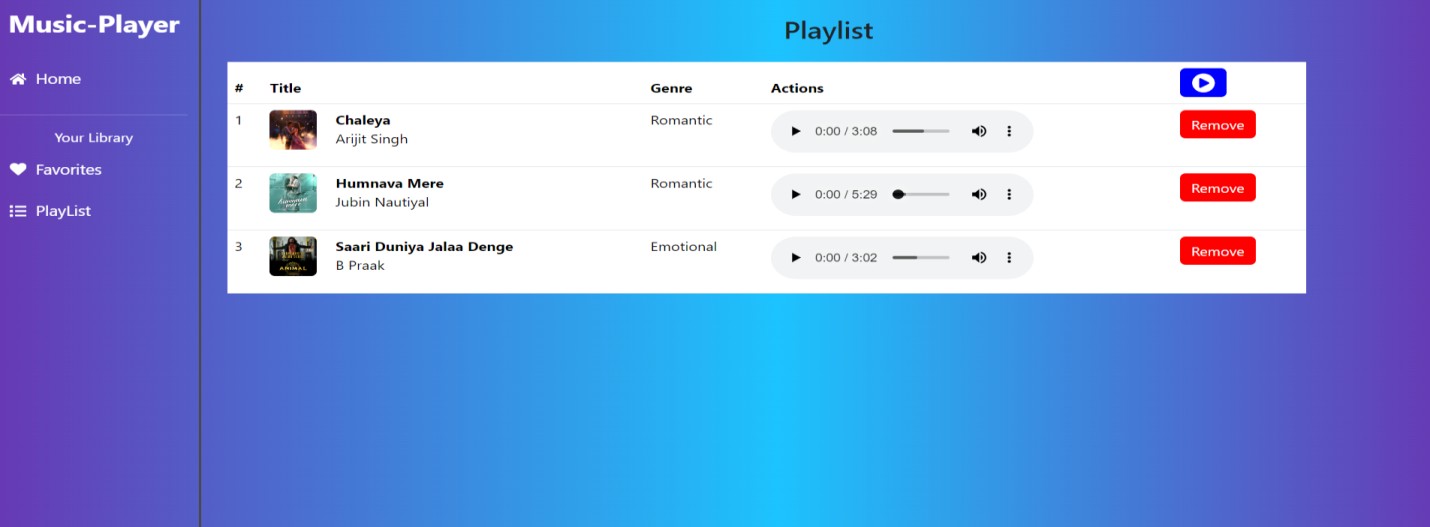
* To run your React music application, you'll need to start two servers. First, open your terminal and use npm start (or npm run dev for Vite) to launch the React development server. This will make your application accessible in your web browser.
* Next, open a separate terminal window and run json-server --watch ./db/db.json. This command starts a JSON server, which provides the data your music app needs from the db.json file. This ensures your app has access to the song information and other necessary data.
* Finally, open your web browser and navigate to the address provided by the React development server (usually http://localhost:3000 or http://localhost:5173). You should now see your "Rythimic Tunes" application running, and you can refer to the provided screenshots to verify it's working correctly.

## Hero components

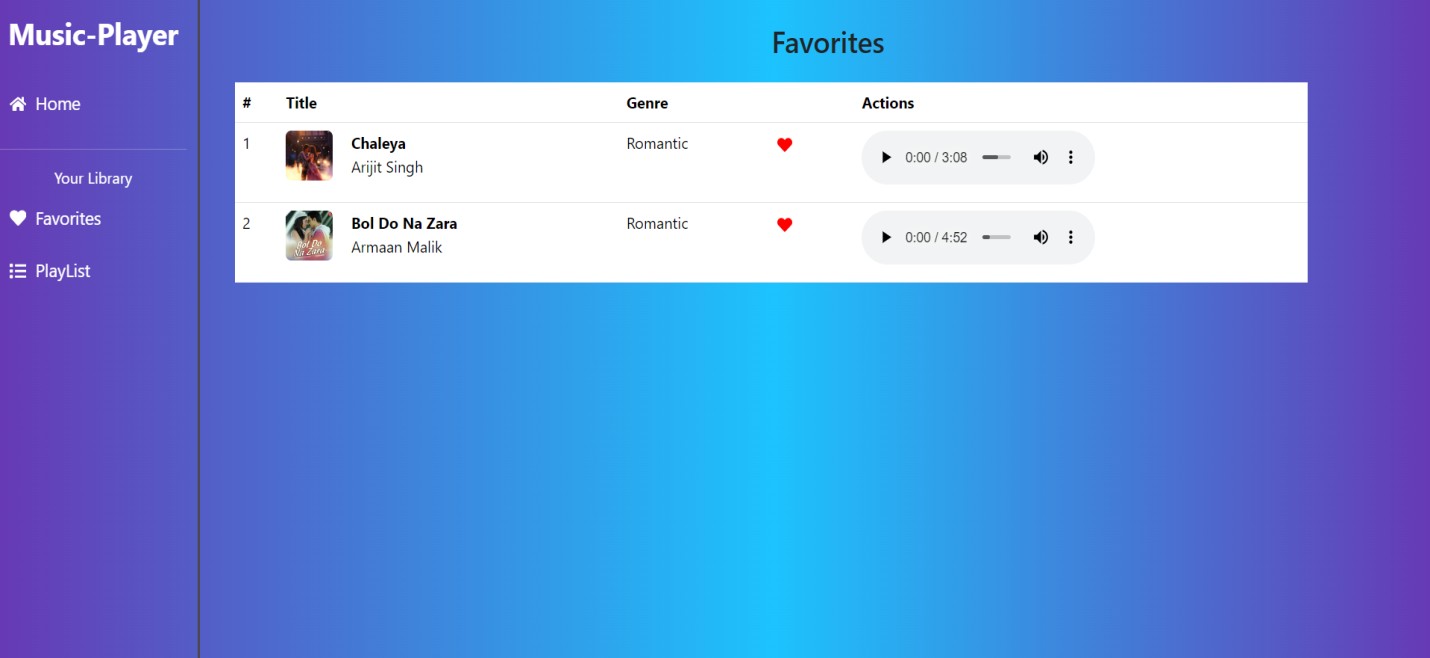




## Playlist



## Favorites



## Thank You