**VALLIAMMAL COLLEGE FOR WOMEN (1363)**

**DEPARTMENT OF COMPUTER APPLICATION**

**PROJECT TITLE: RYTHIMIC TUNES**

**TEAM ID: 147633**

**TEAM LEADER: S.SARIGASRI**

**TEAM MEMBERS: HINDHUJA V.B, GAYATHRI.V, LAVANYA.L, SARIGASRI.S**

**GITHUB LINK:** **https**://**github**.**com**/**Sarigasri**/**Rythimic**\_**sarigasriNm**-

**DRIVE LINK:** **https://drive.google.com/file/d/1IL47LDmbgWwiYNCWtm4yA5DMaWTKEEl5/view**

**INTRODUCTION:-**

**This rhythmic composition incorporates a synchronized pattern of sounds, creating a harmonious and engaging melody The following tune showcases a deliberate arrangement of rhythmic elements, designed to evoke a specific emotional response This musical piece features a carefully crafted rhythm, blending [genre/style] influences with modern production techniques. This rhythmic tune is characterized by its infectious groove, propelled by a combination of percussive elements and melodic phrases The rhythm section of this composition provides a dynamic foundation, supporting the interplay between melody and harmony The tune's rhythmic structure is marked by a repeating pattern of strong and weak beats, creating a sense of tension and release.**

**Scenario-based intro:**

**Imagine** **yourself** **on** a **sun**-**kissed beach**, **surrounded** **by the vibrant rhythms of a** **tropical** **paradise** **Get** **ready** **to** **ignite** **the** **dance** **floor** **with** **a** **pulsating** **beat** **that'll** **keep** **you** **moving** **all** **night** **long** **Picture** **a** **bustling** **city**

**Target Audience:-**

Music Streaming is designed for a diverse audience, including:

● **Music Enthusiasts:** People passionate about enjoying and listening Music Through out there free time to relax themselves.

**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:-**

**Song Listings:** Display a comprehensive list of available songs with details such as title, artist, genre, and release date.

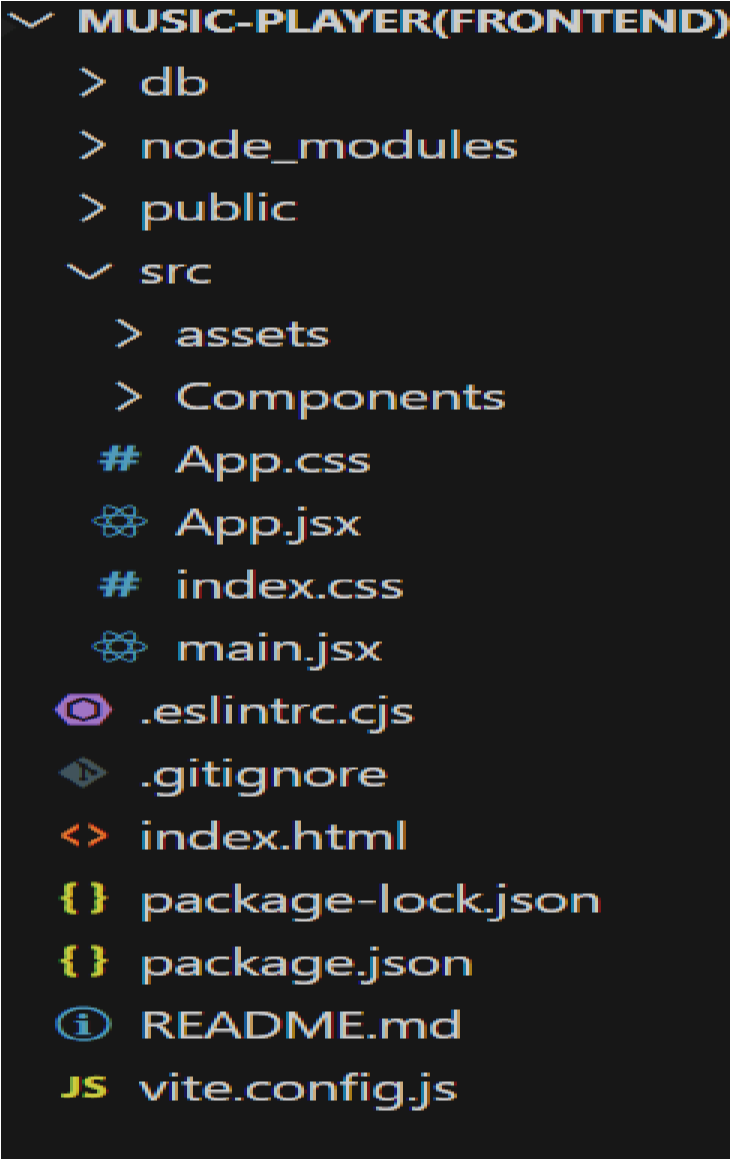
**Playlist Creation:** Empower users to create personalized playlists, adding and organizing songs based on their preferences.

**Playback Control:** Implement seamless playback control features, allowing users to play, pause, skip, and adjust volume during music playback.

**Offline Listening:** Allow users to download songs for offline listening, enhancing the app's accessibility and convenience.

**Search Functionality:** Implement a robust search feature for users to easily find specific songs, artists, or albums within the app.

**Project structure:**



The project s ructure may vary depending on the specific library, framework, programming language, or development approach used. It's essential to organize the files and directories in a logical and consistent manner to improve code maintainability and collaboration among developers.

app/app.component.css, src/app/app.component: These files are part of the main AppComponent, which serves as the root component for the React app. The component handles the overall layout and includes the router outlet for loading different components based on the current route.

**PROJECT FLOW:-**

# Project demo:

Before starting to work on this project, let’s see the demo.

Demo link[https://drive.google.com/file/d/1zZuq62lyYNV\_k5uu0SFjoWa35UgQ4LA9/view?usp=driv e\_link](https://drive.google.com/file/d/1zZuq62lyYNV_k5uu0SFjoWa35UgQ4LA9/view?usp=drive_link)

Use the code in:

[https://drive.google.com/drive/folders/1BkYWfW\_K3ek\_UgtXNTAsDqlhdCuqz6nT?usp= drive\_link](https://drive.google.com/drive/folders/1BkYWfW_K3ek_UgtXNTAsDqlhdCuqz6nT?usp=drive_link)

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

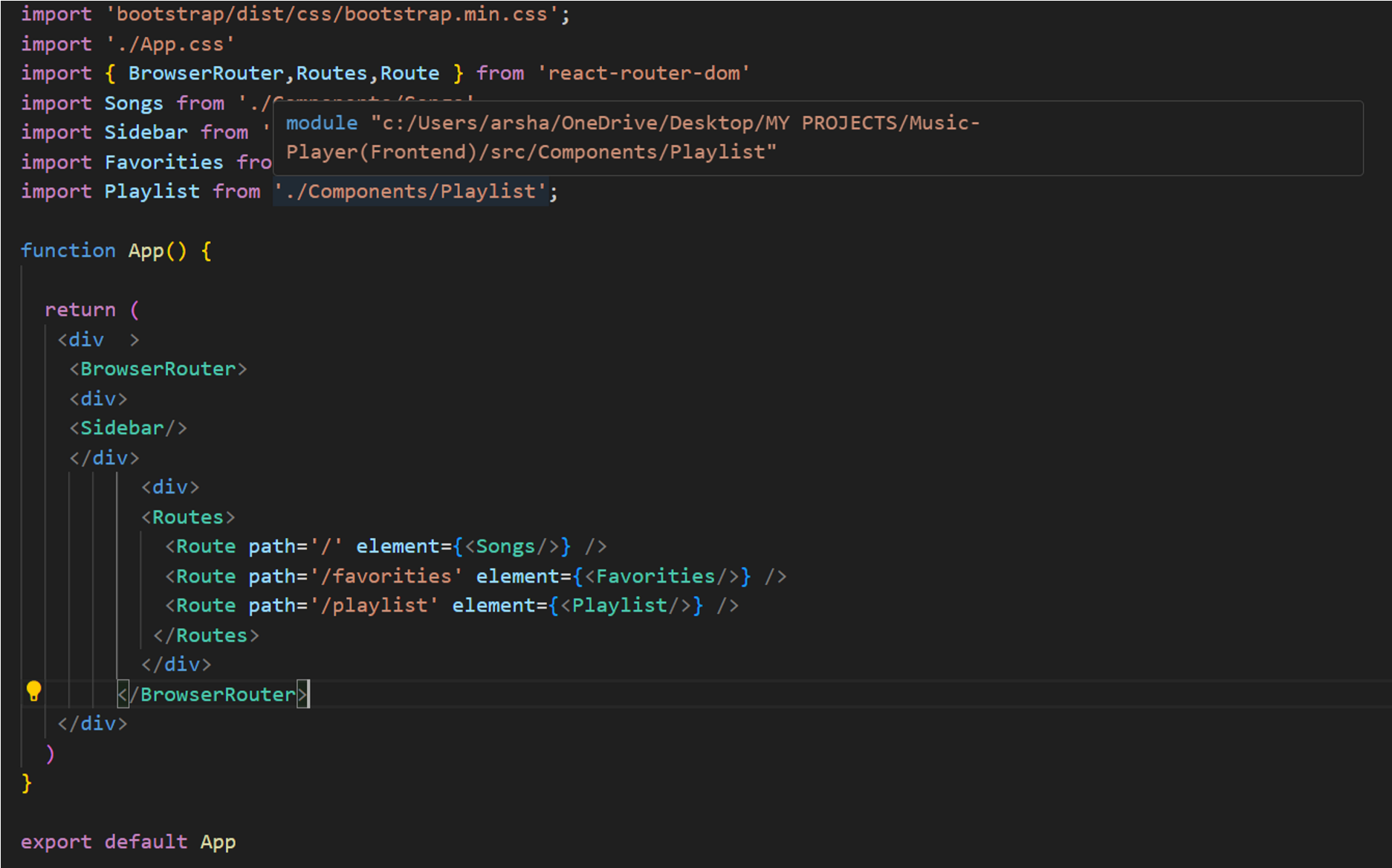
o <https://react-bootstrap-v4.netlify.app/getting-started/introduction/> o <https://axios-http.com/docs/intro> o <https://reactrouter.com/en/main/start/tutorial>

**Milestone 2: Project Development:**

# 1. Setup React Application:

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

Setting Up Routes:-



# Code Description:-

* Imports Bootstrap CSS (bootstrap/dist/css/bootstrap.min.css) for styling components.
* Imports custom CSS (./App.css) for additional styling.
* Impor s BrowserRouter, Routes, and Route from react-router-dom for setting up client-side routing in the application.
* Defines the App functional component that serves as the root component of the application.
* Uses BrowserRouter as the router container to enable routing functionality.
* Includes a div as the root container for the application.
* Within BrowserRouter, wraps components inside two div containers:
  + The first div contains the Sidebar component, likely serving navigation or additional content.
  + The second div contains the Routes component from React Router, which handles rendering components based on the current route.
  + Inside Routes, defines several Route components:
  + Route with path='/' renders the Songs component when the root path is accessed (/).
  + Route with path='/favorities' renders the Favorities component when the /favorities path is accessed.
  + Route with path='/playlist' renders the Playlist component when the /playlist path is accessed.
* Exports the App component as the default export, making it available for use in other parts of the application.

# Fetching Songs:-

**C**

**o**

**d**

**e**

**D**

**e**

**s**

**c**

**r**

**i**

**p**

**t**

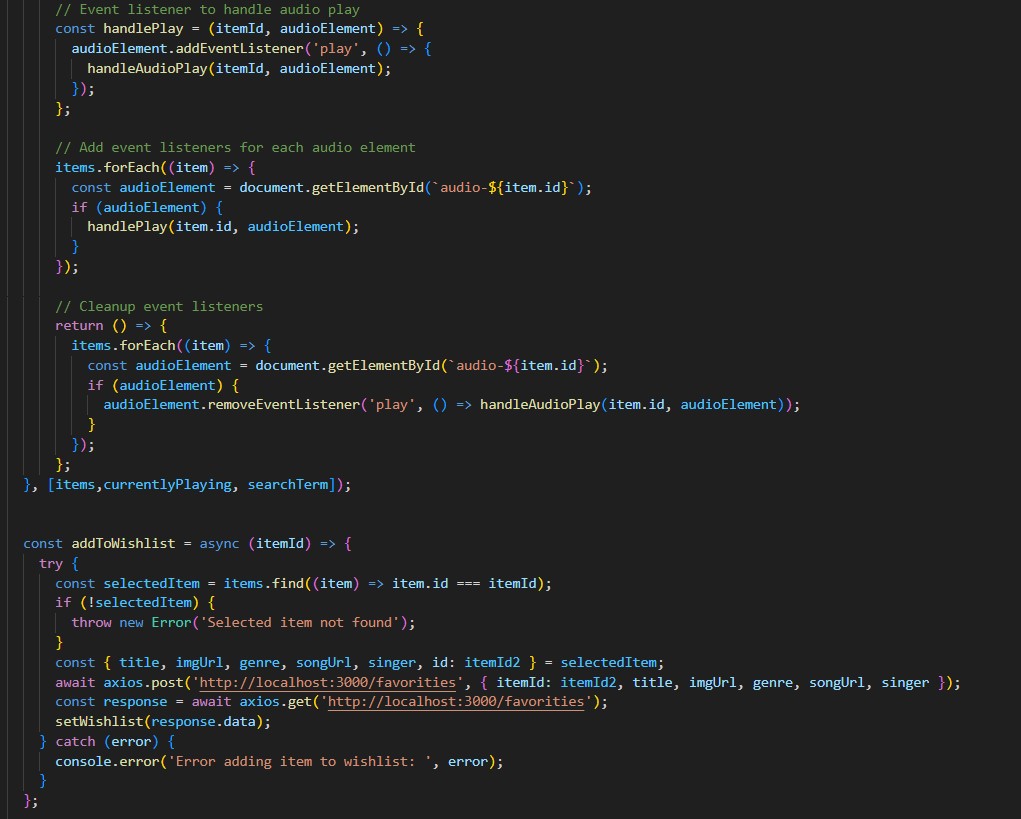
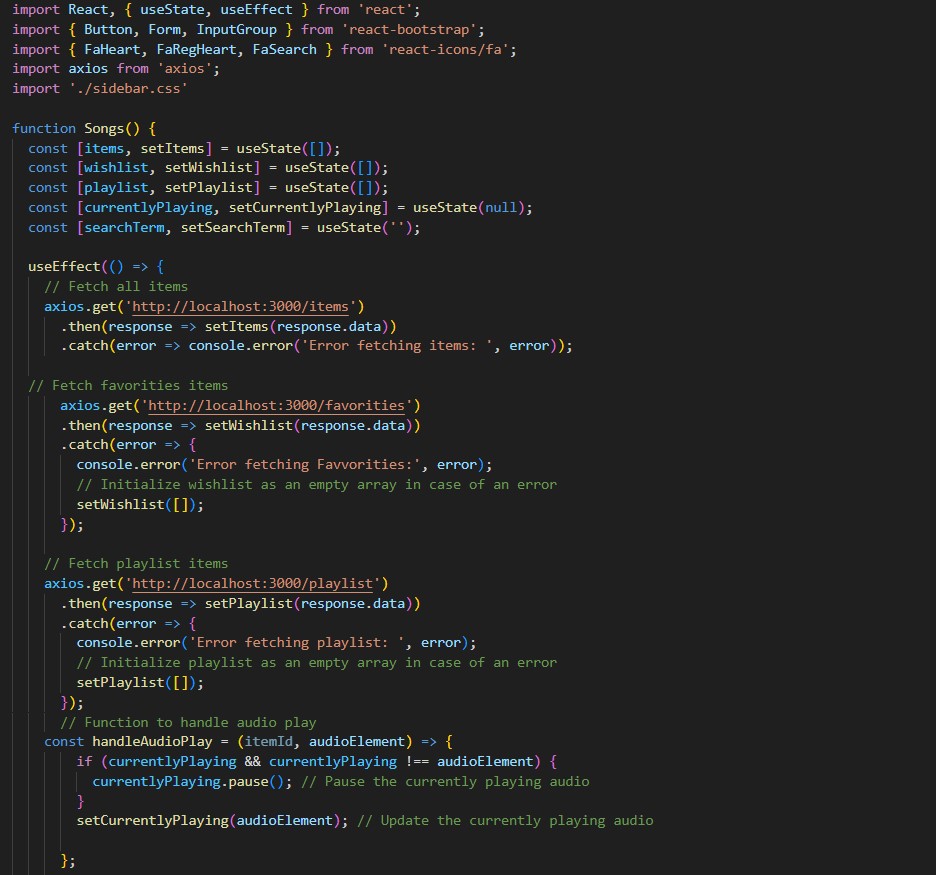
**i**

**o**

**n**

**:**

**-**



* **useState:**
  + 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.

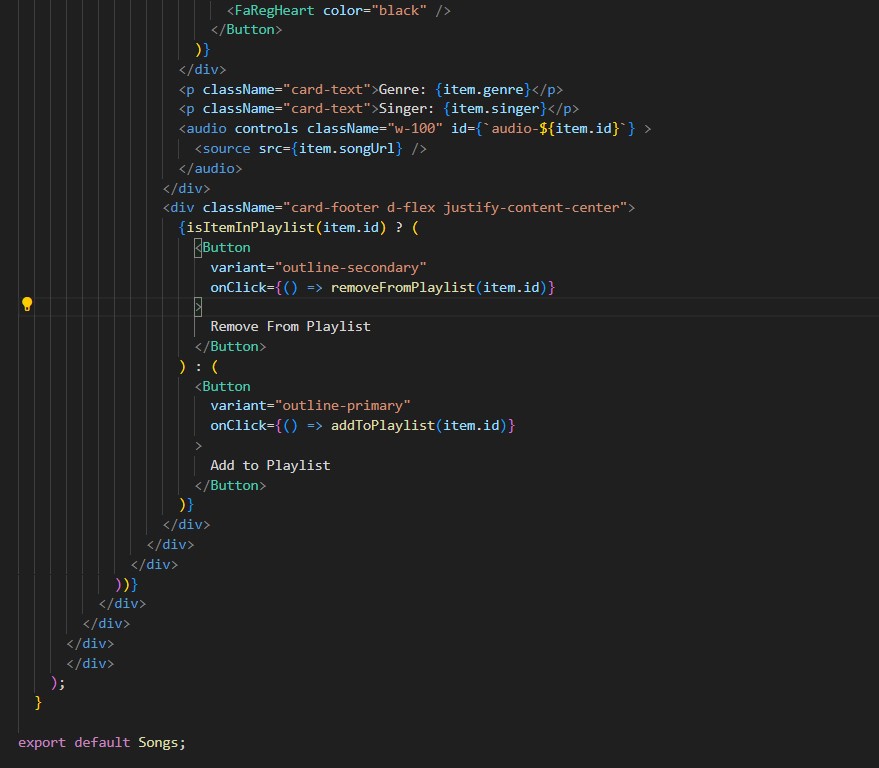
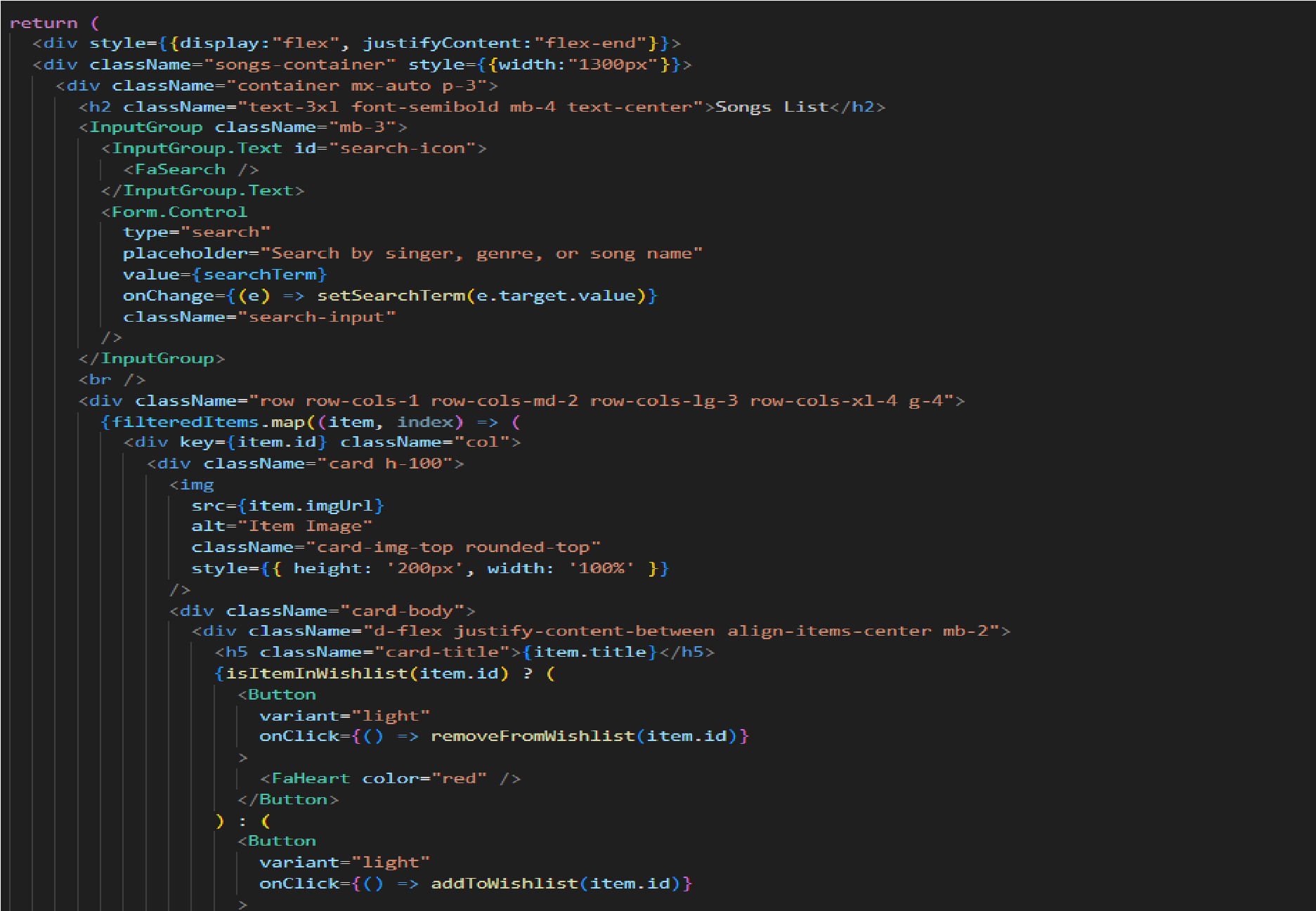
* **isItemInPlaylist(itemId):**

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:**
  + - Renders a form with an input field (Form, InputGroup, Button, FaSearch) for searching items.
    - Maps over filteredItems to render each item in the UI.
    - Includes buttons (FaHeart, FaRegHeart) to add/remove items from wishlist and playlist. o Renders audio elements for each item with play/pause functionality.
* **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:-

* **Container Setup:**
  + Uses a div with inline styles (style={{display:"flex", justifyContent:"flex-end"}}) to align the content to the right.
  + The main container (songs-container) has a fixed width

(width:"1300px") and contains all the UI elements related to songs.

* **Header:**
  + Displays a heading (<h2>) with text "Songs List" centered

(className="text-3xl font-semibold mb-4 text-center").

* **Search Input:**
  + Utilizes InputGroup from React Bootstrap for the search functionality.
  + Includes an input field (Form.Control) that allows users to search by singer, genre, or song name.
  + Binds the input field value to searchTerm state (value={searchTerm}) and updates it on change (onChange={(e) => setSearchTerm(e.target.value)}).
  + Styled with className="search-input".
* **Card Layout:**
  + Uses Bootstrap grid classes (row, col) to create a responsive card layout

(className="row row-cols-1 row-cols-md-2 row-cols-lg-3 row-cols-xl-4 g-4").

* + Maps over filteredItems array and renders each item as a Bootstrap card

(<div className="card h-100">). ● **Card Content:**

* + Displays the item's image (<img>), title (<h5

className="card-title">), genre (<p className="card-text">), and singer (<p className="card-text">).

* + Includes an audio player (<audio controls className="w-100" id={audio-${item.id}}>) for playing the song with a source (<source src={item.songUrl} />).
* **Wishlist and Playlist Buttons:**
  + Adds a heart icon button (<Button>) to add or remove items from the wishlist

(isItemInWishlist(item.id) determines which button to show).

* + Includes an "Add to Playlist" or "Remove From Playlist" button (<Button>) based on whether the item is already in the playlist

(isItemInPlaylist(item.id)).

* **Button Click Handlers:**
  + Handles adding/removing items from the wishlist

(addToWishlist(item.id), removeFromWishlist(item.id)).

* + Manages adding/removing items from the playlist

(addToPlaylist(item.id), removeFromPlaylist(item.id)).

* **Card Styling:**
  + - Applies Bootstrap classes (card, card-body, card-footer) for styling the card components.
    - Uses custom styles (rounded-top, w-100) for specific elements like images and audio players.

**Project Execution:**

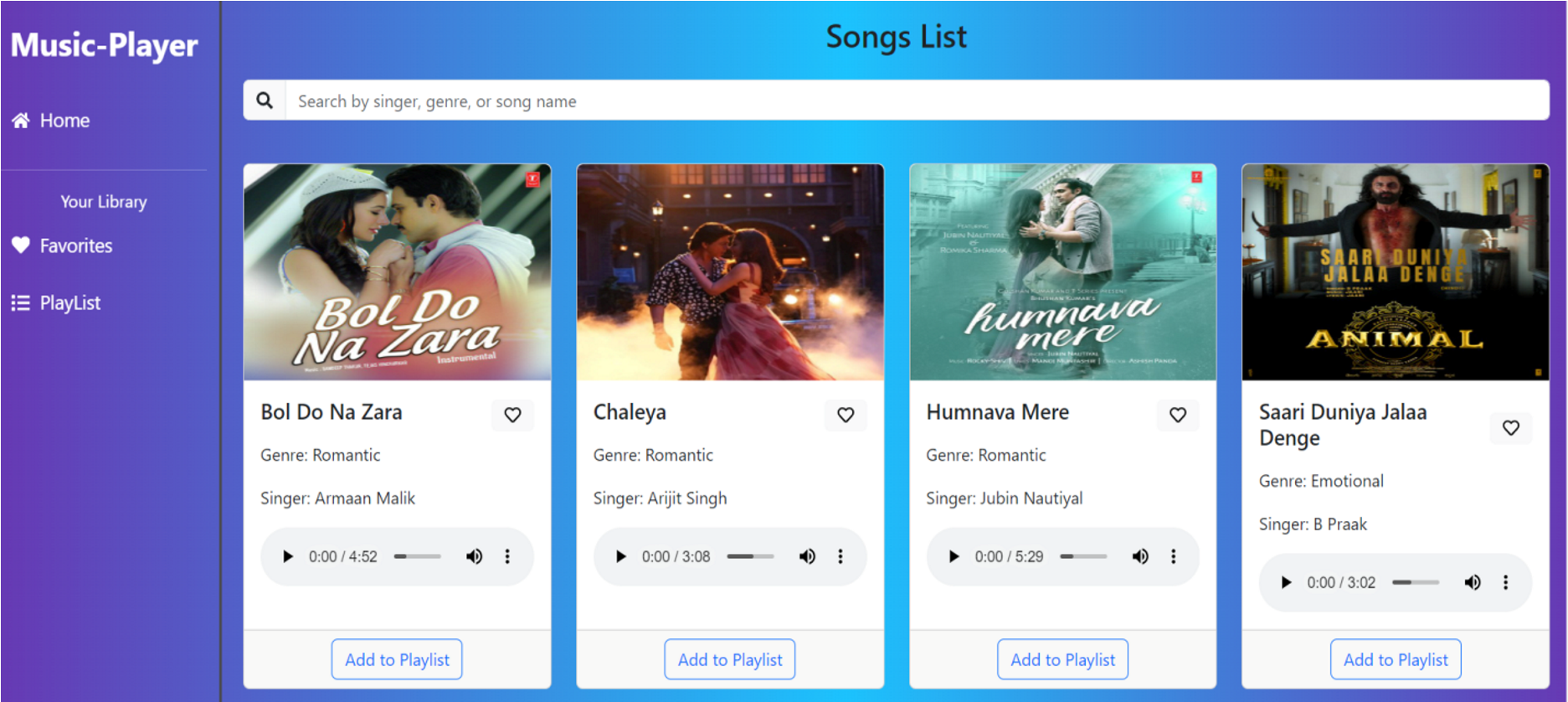
After completing the code, run the react application by using the command “npm start” or “npm run dev” if you are using vite.js

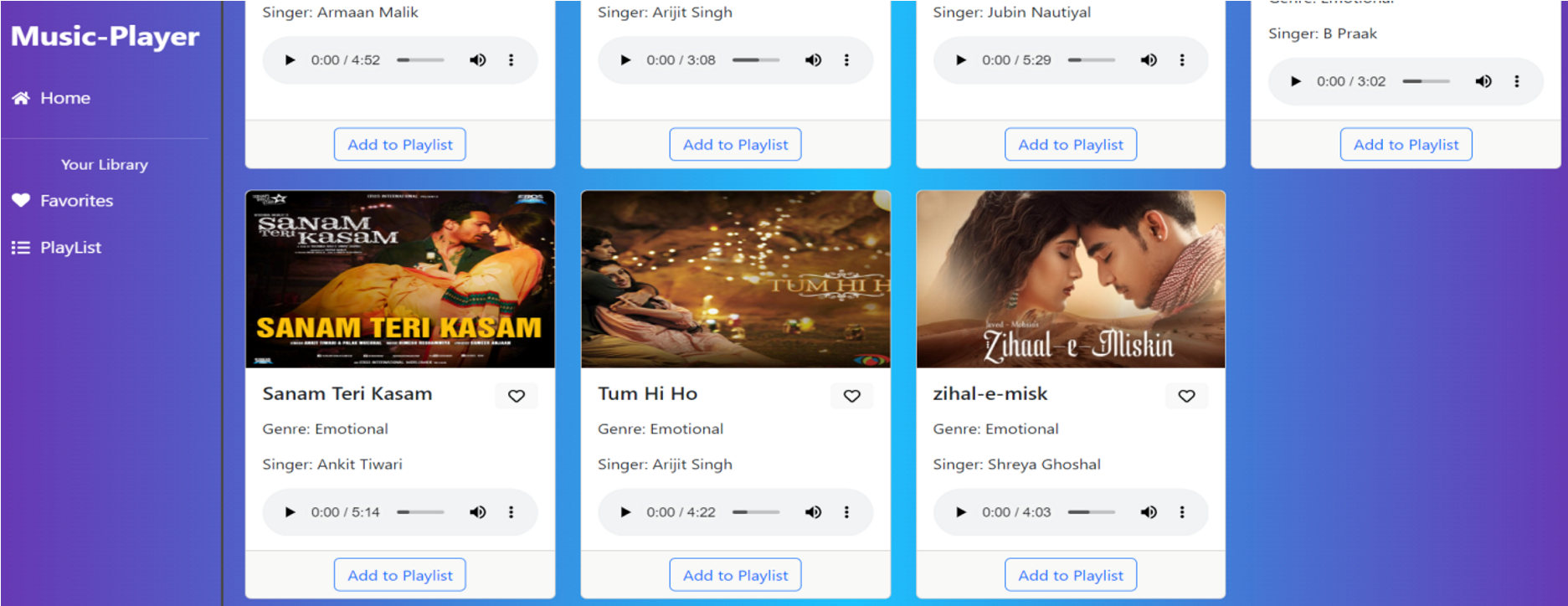
And the Open new Terminal type this command “json-server --watch ./db/db.json” to start the json server too.

After that launch the Rythimic Tunes.

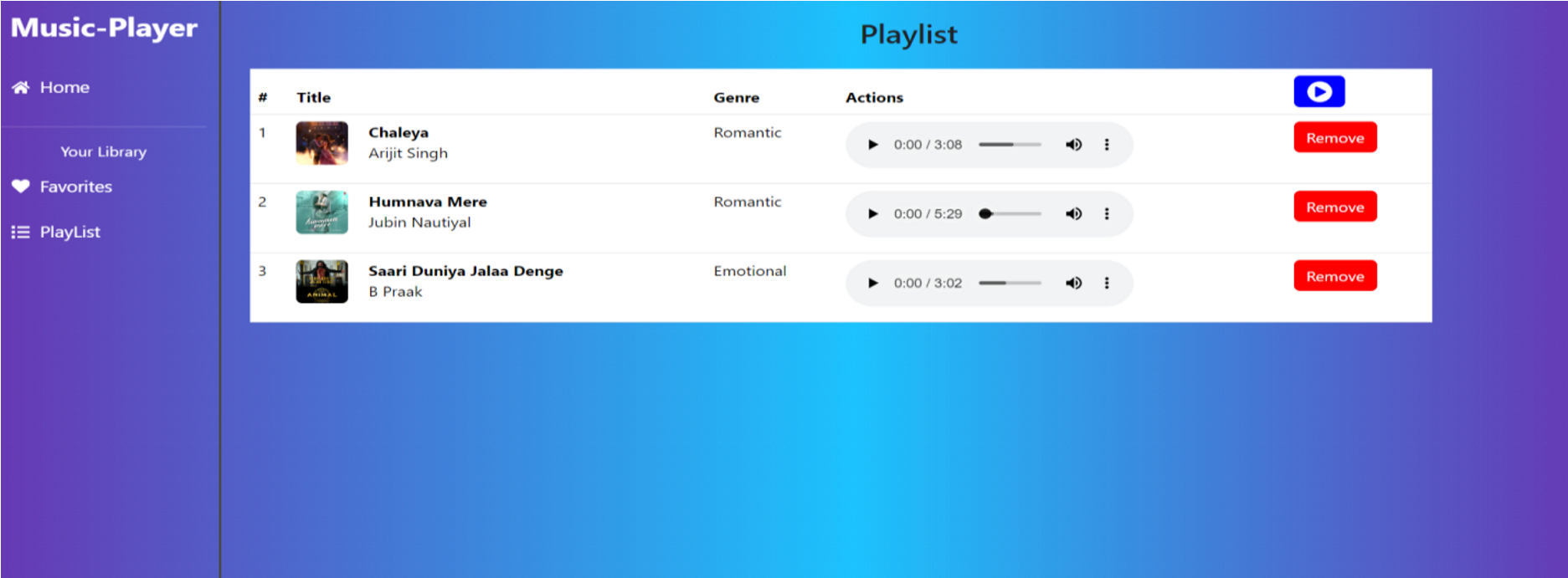
Here are some of the screenshots of the application.

**Hero components**

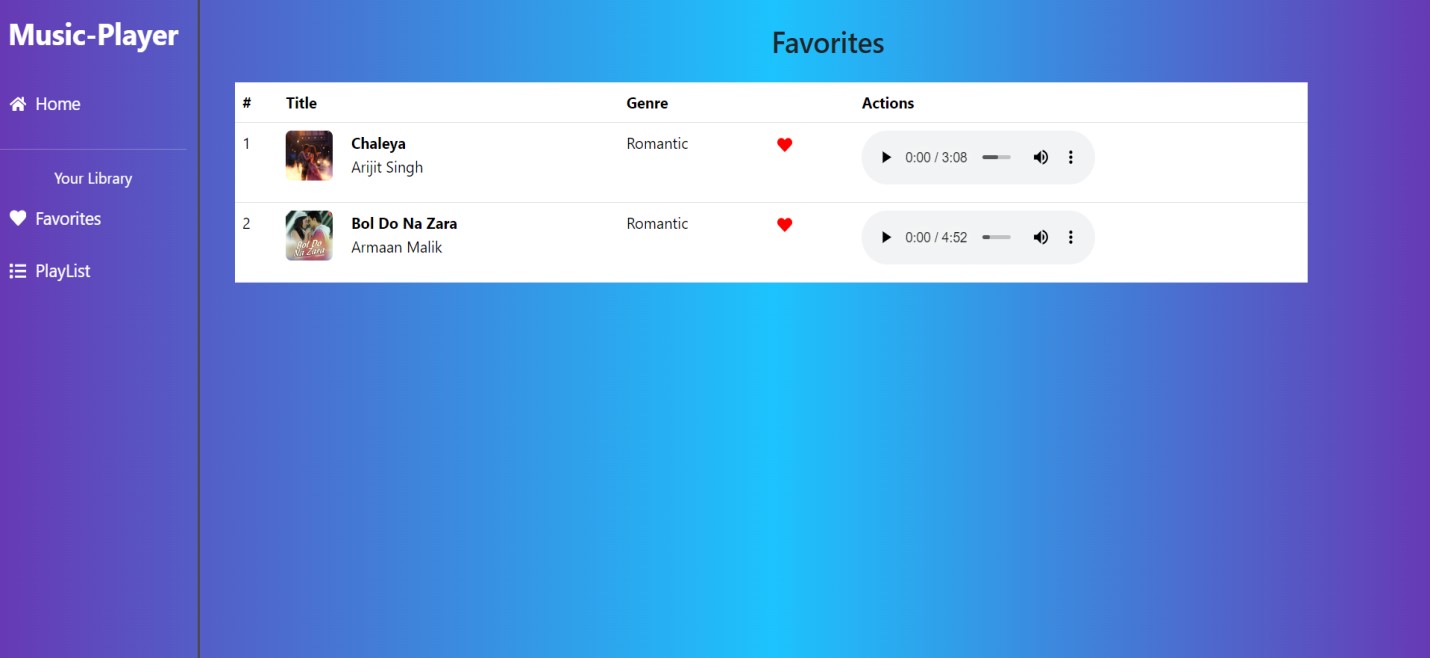




**Playlist**



**Favorites**



**Project Demo link:**

<https://drive.google.com/file/d/1zZuq62lyYNV_k5uu0SFjoWa35UgQ4LA9/view?usp=drive_link>

**\*\*\* Happy coding!! \*\*\***