**Rythimic Tunes** (React)

**Introduction: -**

Welcome to the future of music streaming with our cutting-edge React.js-based application. Designed for modern music lovers, it combines powerful functionality with an intuitive interface, offering a seamless experience across all devices. Discover new hits, revisit classics, and enjoy a visually stunning, interactive platform that redefines how you experience music. Elevate your auditory journey and embrace a new era of music streaming with us. Press play on innovation today!

**Scenario-Based Intro: -**

Picture this: You're walking down a busy city street, surrounded by the hum of traffic and chatter. To brighten your morning, you open **RythimicTunes**, your go-to music streaming app. With a few taps, you’re immersed in a personalized playlist. An upbeat pop song energizes your walk, while a calming indie track soothes your train ride. Every moment is perfectly scored to match your mood, thanks to smart, intuitive features. Welcome to **RythimicTunes** – where music adapts to your life, one beat at a time.

**Target Audience: -**

Music Streaming is designed for a diverse audience, including:

● **Music Enthusiasts:** People passionate about enjoying and listening Music Throughout their free time to relax themselves.

**Project Goals and Objectives: -**

The goal of our Music Streaming Application is to create a seamless platform for music lovers to explore, enjoy, and share music. Key objectives include:

* **User-Friendly Interface:** Design an intuitive interface for easy exploration, saving, and sharing of music.
* **Comprehensive Music Streaming:** Offer robust features like advanced search for simplified music discovery.
* **Modern Tech Stack:** Use cutting-edge technologies like React.js for a smooth and enjoyable user experience.

**Key Features:**

* Song Listings: Show song details like title, artist, genre, and release date.
* Playlist Creation: Let users create and customize playlists.
* Playback Control: Enable play, pause, skip, and volume adjustment.
* Offline Listening: Allow song downloads for offline use.
* Search Functionality: Offer a robust search tool for songs, artists, or albums.

**PRE-REQUISITES**: -

Here are the key prerequisites for developing a frontend application using React.js:

**Node.js and npm**:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the local environment. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

* Download: <https://nodejs.org/en/download/>
* Installation instructions: <https://nodejs.org/en/download/package-manager/> **React.js**:

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

* Create a new React app:

npm create vite@latest

Enter and then type project-name and select preferred frameworks and then enter

* Navigate to the project directory:

cd project-name npm install

* Running the React App:

With the React app created, you can now start the development server and see your React application in action.

* Start the development server:

npm run dev

This command launches the development server, and you can access your React app at [http://localhost:5173](http://localhost:5173/) in your web browser.

**HTML, CSS, and JavaScript**: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

* Git: Download and installation instructions can be found at:

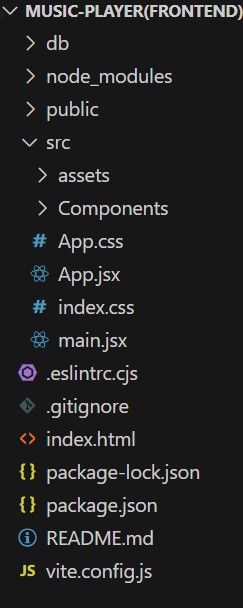
<https://git-scm.com/downloads>

**Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

* Visual Studio Code: Download from <https://code.visualstudio.com/download>
* Sublime Text: Download from <https://www.sublimetext.com/download>

• WebStorm: Download from [https://www.jetbrains.com/webstorm/download](https://www.jetbrains.com/webstorm/download%20)

**Project structure:**



The project structure varies based on the tools and frameworks used. Organizing files and directories logically ensures better maintainability and collaboration.

**AppComponent Files:**

app/app.component.css and src/app/app.component: Part of the main AppComponent, which acts as the root component. It manages the layout and includes the router outlet to load components based on the current route.

**PROJECT FLOW: -**

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

Demolink: [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:

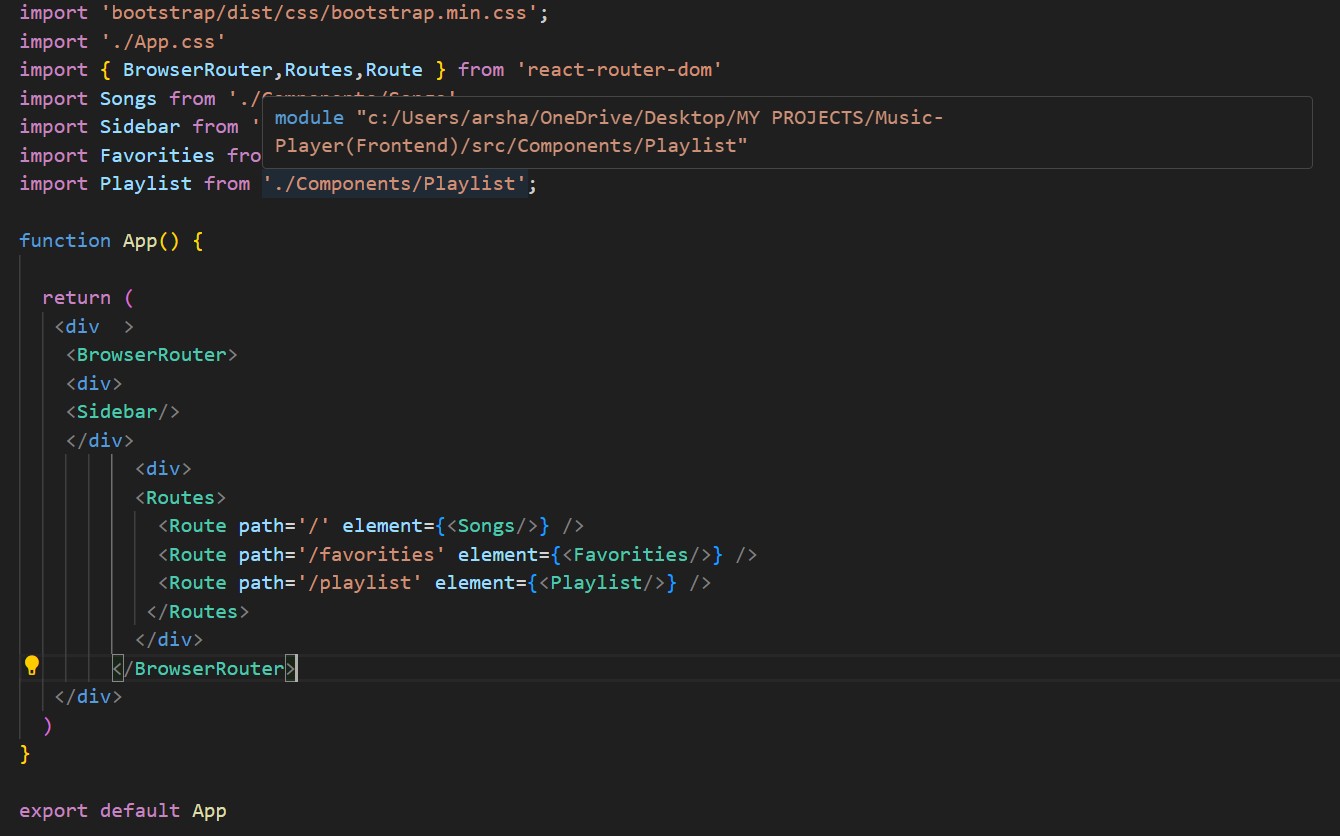
* + React Js o React Router Dom o React Icons o Bootstrap/tailwind css
  + Axios
* For further reference, use the following resources o <https://react.dev/learn/installation>

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:**

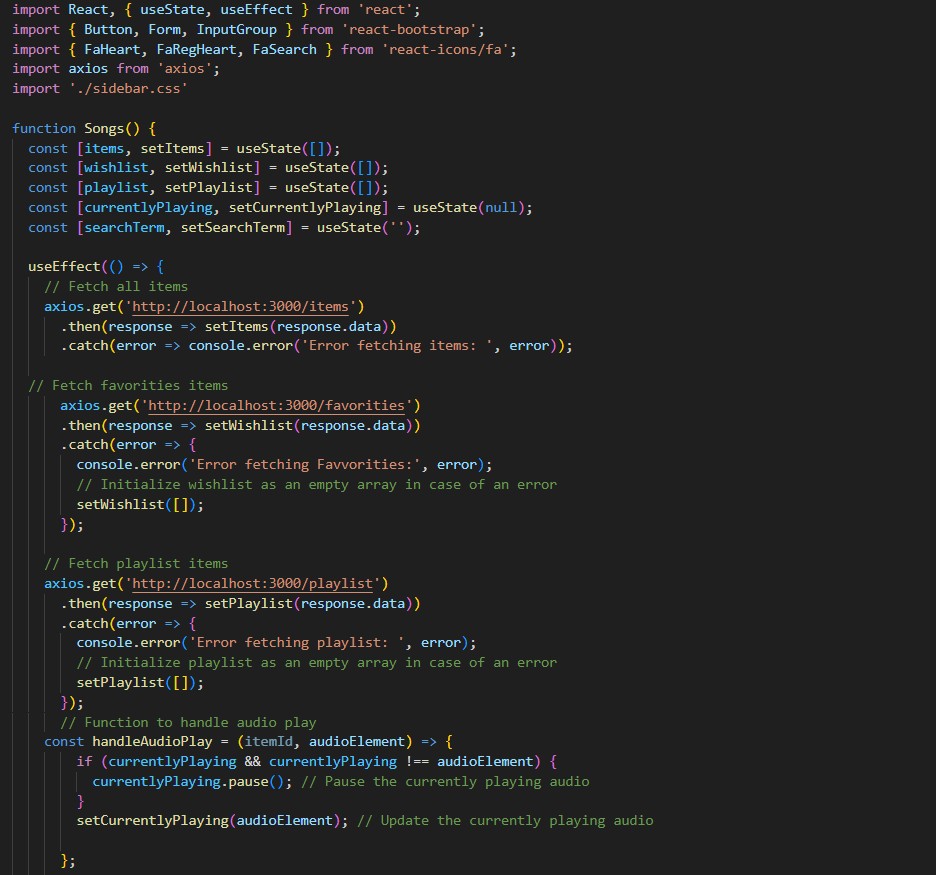
* Create React application.
* Configure Routing.
* Install required libraries.

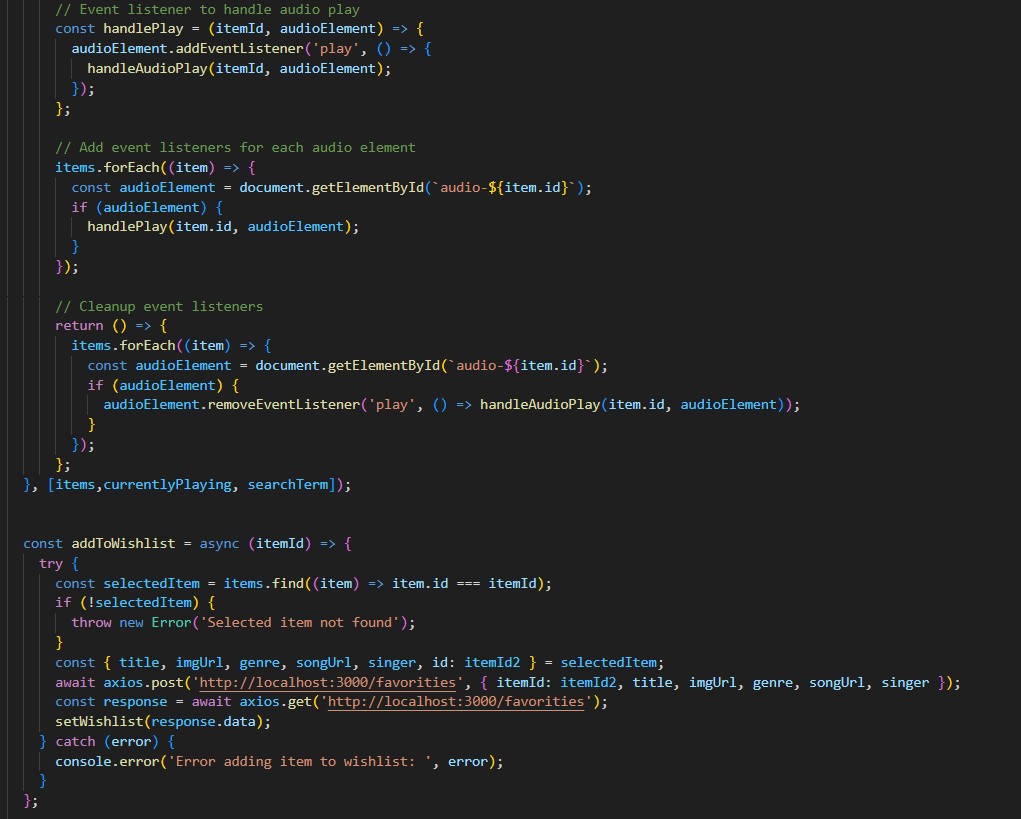
Setting Up Routes: - 

**Code Description: -**

* Imports **Bootstrap CSS** for styling and custom **App.css** for additional design.
* Uses **BrowserRouter, Routes, and Route** from react-router-dom for client-side routing.
* Defines the **App** component as the root, wrapped in **BrowserRouter**.
* Contains two main div containers:
  + The first holds the **Sidebar** for navigation.
  + The second uses **Routes** to render components based on the route:
    - / renders the **Songs** component.
    - /favorities renders the **Favorities** component.
    - /playlist renders the **Playlist** component.
* Exports **App** as the default for use across the application.

**Fetching Songs: -**

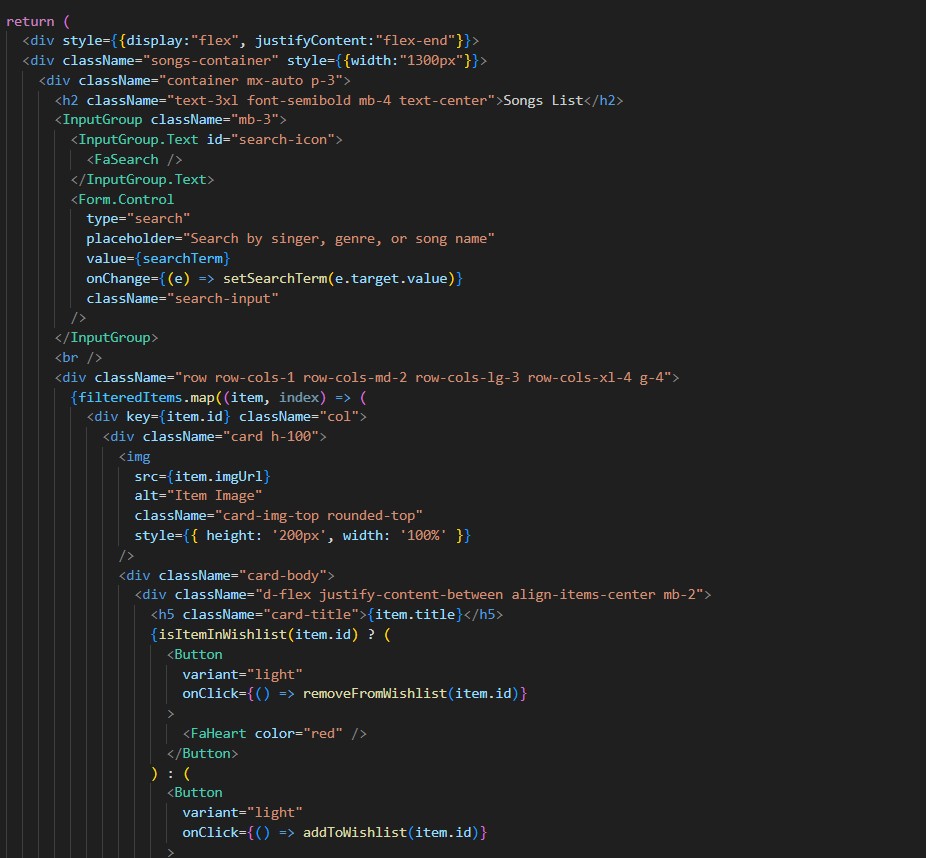


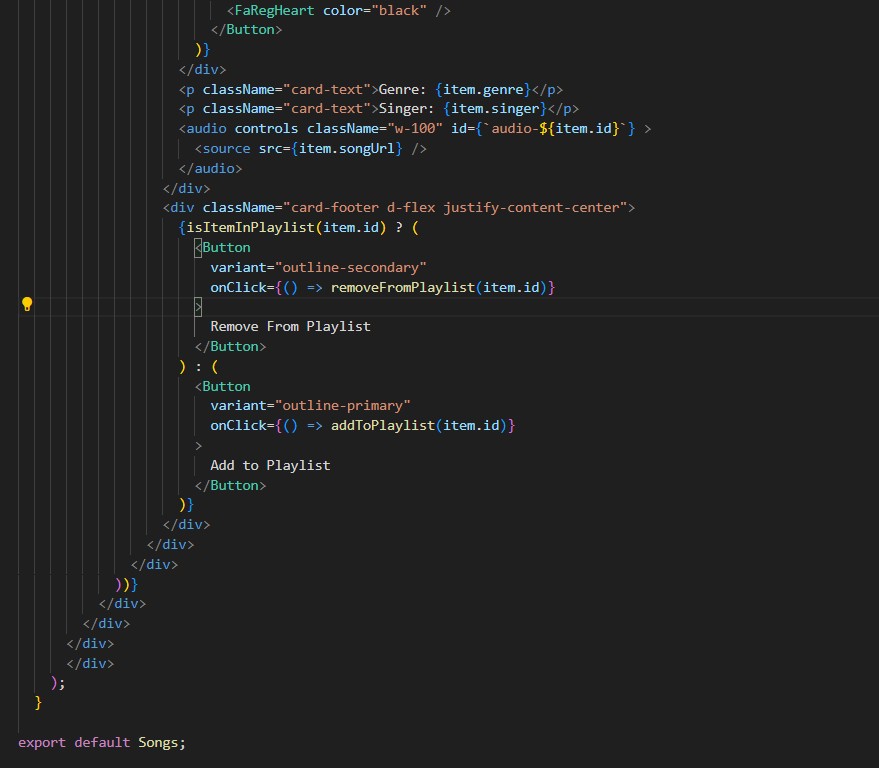


**Code Description:**

* State Management:
  + Uses useState to manage:
    - items, wishlist, playlist: Fetched from respective endpoints.
    - currentlyPlaying: Tracks active audio.
    - searchTerm: Stores user search input.
* Data Fetching:
  + Uses useEffect to fetch data for items, wishlist, and playlist, updating state.
* Audio Playback:
  + Manages playback with handleAudioPlay and handlePlay to ensure only one audio plays at a time.
* Wishlist & Playlist Functions:
  + addToWishlist/removeFromWishlist: Adds/removes items via POST/DELETE requests.
  + addToPlaylist/removeFromPlaylist: Adds/removes items via POST/DELETE requests.
  + isItemInWishlist/isItemInPlaylist: Checks if an item exists in the wishlist or playlist.
* Search Functionality:
  + Filters items based on searchTerm, matching title, singer, or genre.
* UI Rendering:
  + Renders a search form and maps filteredItems to display each item.
  + Includes buttons for wishlist/playlist actions and audio controls.
* Error Handling:
  + Catches and logs errors during data fetching and wishlist/playlist operations.

**Frontend Code for Displaying Songs: -**





**Code Description:**

* **Container Setup:**
  + Uses a div with inline styles to align content to the right.
  + Main container (songs-container) has a fixed width and holds all song-related UI elements.
* **Header:**
  + Displays a centered heading (<h2>) with the text "Songs List."
* **Search Input:**
  + Uses InputGroup for search functionality with an input field (Form.Control) bound to searchTerm.
  + Styled with className="search-input".
* **Card Layout:**
  + Uses Bootstrap grid classes (row, col) for a responsive layout.
  + Maps over filteredItems to render each item as a Bootstrap card (<div className="card h-100">).
* **Card Content:**
  + Displays the item's image, title, genre, and singer.
  + Includes an audio player (<audio controls>) with a source (<source src={item.songUrl} />).
* **Wishlist & Playlist Buttons:**
  + Adds a heart icon button to add/remove items from the wishlist (based on isItemInWishlist).
  + Includes an "Add to Playlist" or "Remove From Playlist" button (based on isItemInPlaylist).
* **Button Click Handlers:**
  + Manages adding/removing items from the wishlist and playlist.
* **Card Styling:**
  + Applies Bootstrap classes (card, card-body, card-footer) and custom styles (rounded-top, w-100) for design.

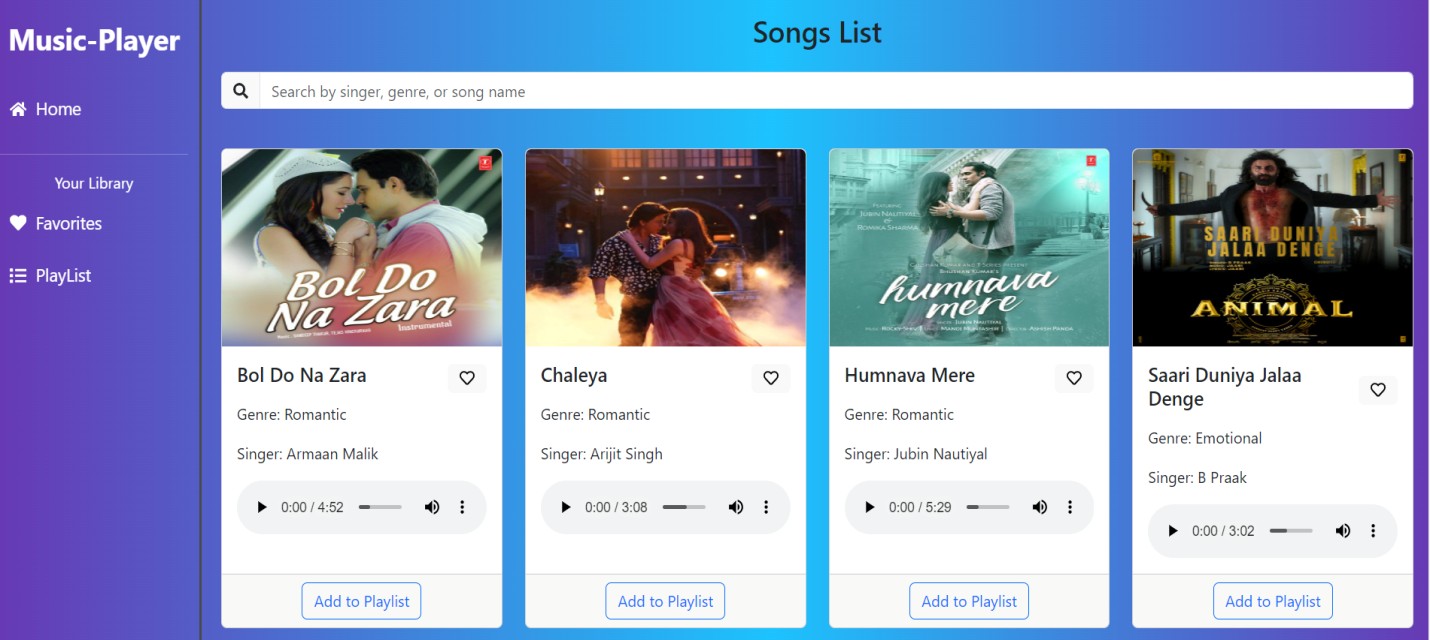
**Project Execution:**

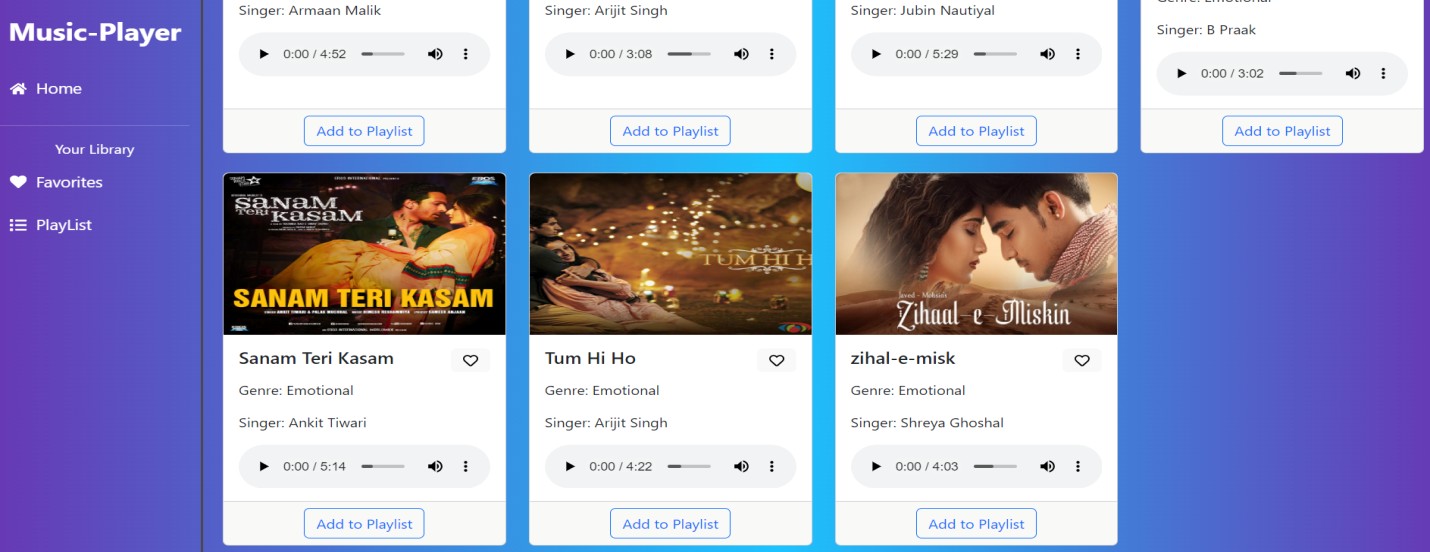
After completing the code, run the react application by using the command “npm start” or “npm run dev” if you are using vite.js And the Open new Terminal type this command “json-server --watch ./db/db.json” to start the json server too.

After that launch the Rythimic Tunes.

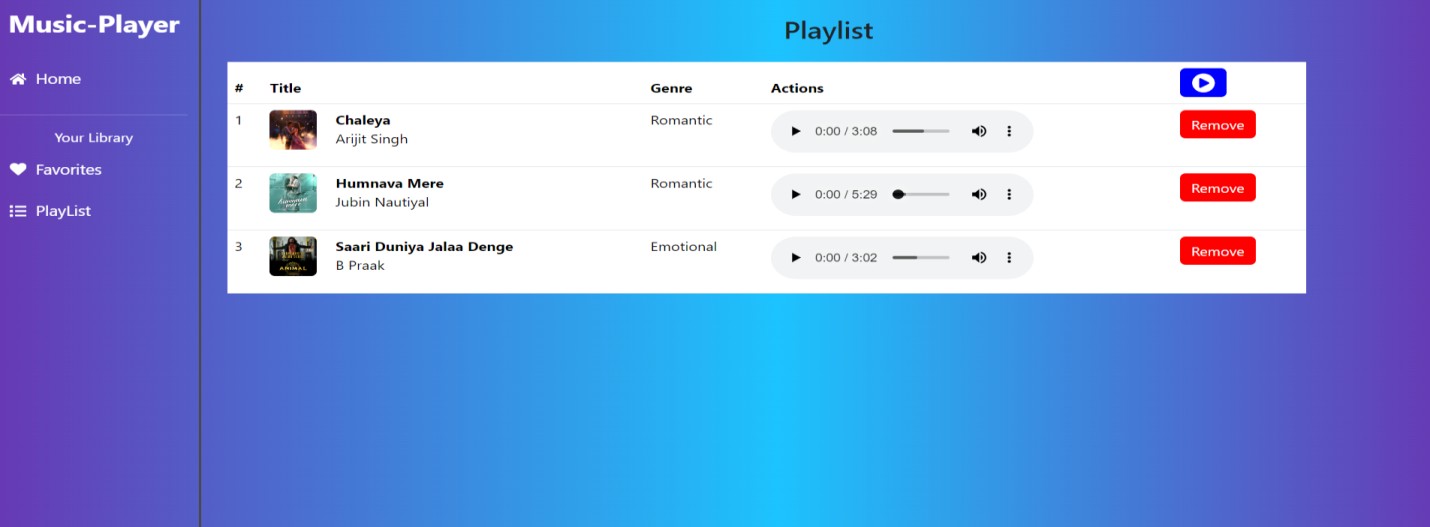
Here are some of the screenshots of the application.

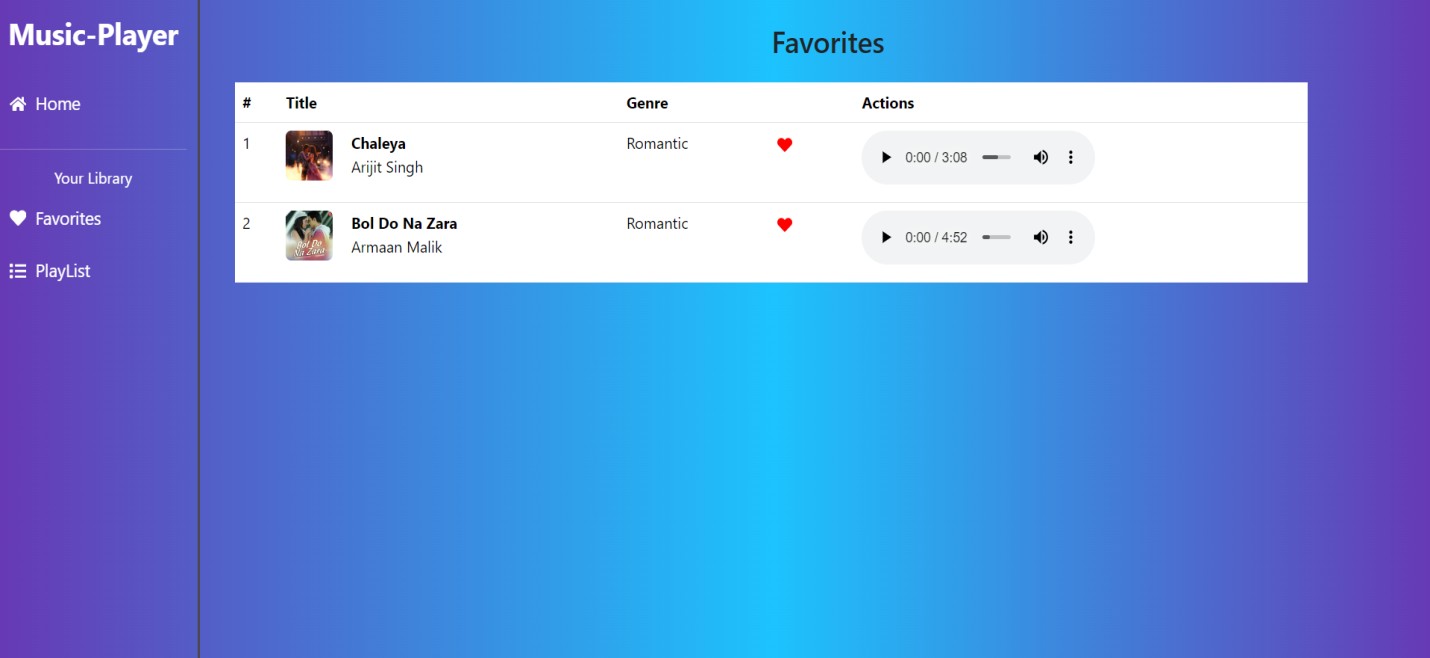
**Hero components**





**Playlist**

 **Favourites**



**Project Demo link:**

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