

# **RHYTHMIC TUNES**

## **(YOUR MELODY COMPANION)**

TEAM ID :SWTID1741155745

TEAM LEADER : PAVITHRA .R

: LOGESHWARI.V

TEAM MEMBERS LAVANYA .D  
POONGODI .P

CLASS : III-BSC COMPUTER SCIENCE

EMAIL ID'S : [pavithraratha56@gmail.com](mailto:pavithraratha56@gmail.com)  
[logeshwarivmurugan@gmail.com](mailto:logeshwarivmurugan@gmail.com)  
[lavanya.dd.1411@gmail.com](mailto:lavanya.dd.1411@gmail.com)  
[poongodipurushothaman83@gmail.com](mailto:poongodipurushothaman83@gmail.com)

## **Rhythmic Tunes (React)**

### **Introduction:-**

Welcome to the future of musical indulgence – an unparalleled audio experience awaits you with our cutting-edge Music Streaming Application, meticulously crafted using the power of React.js. Seamlessly blending innovation with user-centric design, our application is set to redefine how you interact with and immerse yourself in the world of music.

Designed for the modern music enthusiast, our React-based Music Streaming Application offers a harmonious fusion of robust functionality and an intuitive user interface. From discovering the latest chart-toppers to rediscovering timeless classics, our platform ensures an all-encompassing musical journey tailored to your unique taste.

The heart of our Music Streaming Application lies in React, a dynamic and feature-rich JavaScript library. Immerse yourself in a visually stunning and interactive interface, where every click, scroll, and playlist creation feels like a musical revelation. Whether you're

on a desktop, tablet, or smartphone, our responsive design ensures a consistent and enjoyable experience across all devices.

Say goodbye to the limitations of traditional music listening and welcome a world of possibilities with our React-based Music Streaming Application. Join us on this journey as we transform the way you connect with and savor the universal language of music. Get ready to elevate your auditory experience – it's time to press play on a new era of music streaming.

### **Scenario-Based Intro:-**

Imagine stepping onto a bustling city street, the sounds of cars honking, people chatting, and street performers playing in the background. You're on your way to work, and you need a little something to elevate your mood. You pull out your phone and open your favorite music streaming app, "Rhythmic Tunes" With just a few taps, you're transported to a world of music tailored to your tastes. As you walk, the app's smart playlist kicks in, starting with an upbeat pop song that gets your feet tapping. As you board the train, the music shifts to a relaxing indie track, perfectly matching your need to unwind during the commute.

### **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.

## 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 `https://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> **Project structure:**

The project structure 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=drive\\_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:

- o React Js
- o React Router Dom
- o React Icons
- o Bootstrap/tailwind css
- o Axios

- For further reference, use the following resources

<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.
- Imports 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:
  - o The first div contains the Sidebar component, likely serving navigation or additional content.
  - o The second div contains the Routes component from React Router, which handles rendering components based on the current route.
- o Inside Routes, defines several Route components:
  - o Route with path="/" renders the Songs component when the root path is accessed (/).
  - o Route with path="/favorites" renders the Favorites component when the /favorites path is accessed.
  - o 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:-

### Code Description:-

- **useState:**
  - o items: Holds an array of all items fetched from <http://localhost:3000/items>.
  - o wishlist: Stores items marked as favorites fetched from <http://localhost:3000/favorites>.
  - o playlist: Stores items added to the playlist fetched from <http://localhost:3000/playlist>.
  - o currently Playing: Keeps track of the currently playing audio element.
  - o searchTerm: Stores the current search term entered by the user.
- **Data Fetching:**
  - o Uses useEffect to fetch data:
    - Fetches all items (items) from <http://localhost:3000/items>.
    - Fetches favorite items (wishlist) from <http://localhost:3000/favorites>.
    - Fetches playlist items (playlist) from <http://localhost:3000/playlist>.
  - o Sets state variables (items, wishlist, playlist) based on the fetched data.
- **Audio Playback Management:**
  - o 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.
  - o Ensures that only one audio element plays at a time by pausing others when a new one starts playing.
- **addToWishlist(itemId):**
  - o Adds an item to the wishlist (favorites) by making a POST request to <http://localhost:3000/favorites>.
  - o Updates the wishlist state after adding an item.
- **removeFromWishlist(itemId):**
  - o Removes an item from the wishlist (favorites) by making a DELETE request to <http://localhost:3000/favorites/{itemId}>.
  - o Updates the wishlist state after removing an item.

- `isItemInWishlist(itemId)`:
  - Checks if an item exists in the wishlist (favorites) based on its `itemId`.
- `addToPlaylist(itemId)`:
  - Adds an item to the playlist (playlist) by making a POST request to <http://localhost:3000/playlist>.
  - Updates the playlist state after adding an item.
- `removeFromPlaylist(itemId)`:
  - Removes an item from the playlist (playlist) by making a DELETE request to <http://localhost:3000/playlist/{itemId}>.
  - Updates the playlist state after removing an item.
- `isItemInPlaylist(itemId)`:
  - 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.
  - Renders audio elements for each item with play/pause functionality.
- Error Handling:
  - Catches and logs errors during data fetching (`axios.get`).
  - Handles errors when adding/removing items from wishlist and playlist.

## Frontend Code For Displaying Songs:-

### Code Description:- ●

#### Container Setup:

- o Uses a div with inline styles (`style={{display:"flex", justifyContent:"flex-end"}})` to align the content to the right.

- o The main container (`songs-container`) has a fixed width (`width:"1300px"`) and contains all the UI elements related to songs.

#### ● Header:

- o Displays a heading (`<h2`) with text "Songs List" centered (`className="text- 3xl font-semibold mb-4 text-center"`).

#### ● Search Input:

- o Utilizes `InputGroup` from `React Bootstrap` for the search functionality.

- o Includes an input field (`Form.Control`) that allows users to search by singer, genre, or song name.

- o Binds the input field value to `searchTerm` state (`value={searchTerm}`) and updates it on change (`onChange={(e) => setSearchTerm(e.target.value)}`).

- o Styled with `className="search-input"`.

#### ● Card Layout:

- o 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"`).

- o Maps over `filteredItems` array and renders each item as a `Bootstrap card ()`. ● Card Content: o Displays the item's image (Error! Filename not specified.), `title ()`, `genre ()`, and `singer ()`.

- o Includes an audio player `()` for playing the song with a source `()`.

#### ● Wishlist and Playlist Buttons:

- o Adds a heart icon button `()` to add or remove items from the wishlist (`isItemInWishlist(item.id)` determines which button to show).
- o Includes an "Add to Playlist" or "Remove From Playlist" button `()` based on whether the item is already in the playlist (`isItemInPlaylist(item.id)`).



- Button Click Handlers:

- o Handles adding/removing items from the wishlist (addToWishlist(item.id), removeFromWishlist(item.id)).
  - o Manages adding/removing items from the playlist (addToPlaylist(item.id), removeFromPlaylist(item.id)).

- Card Styling:

- o Applies Bootstrap classes (card, card-body, card-footer) for styling the card components.
  - o Uses custom styles (rounded-top, w-100) for specific elements

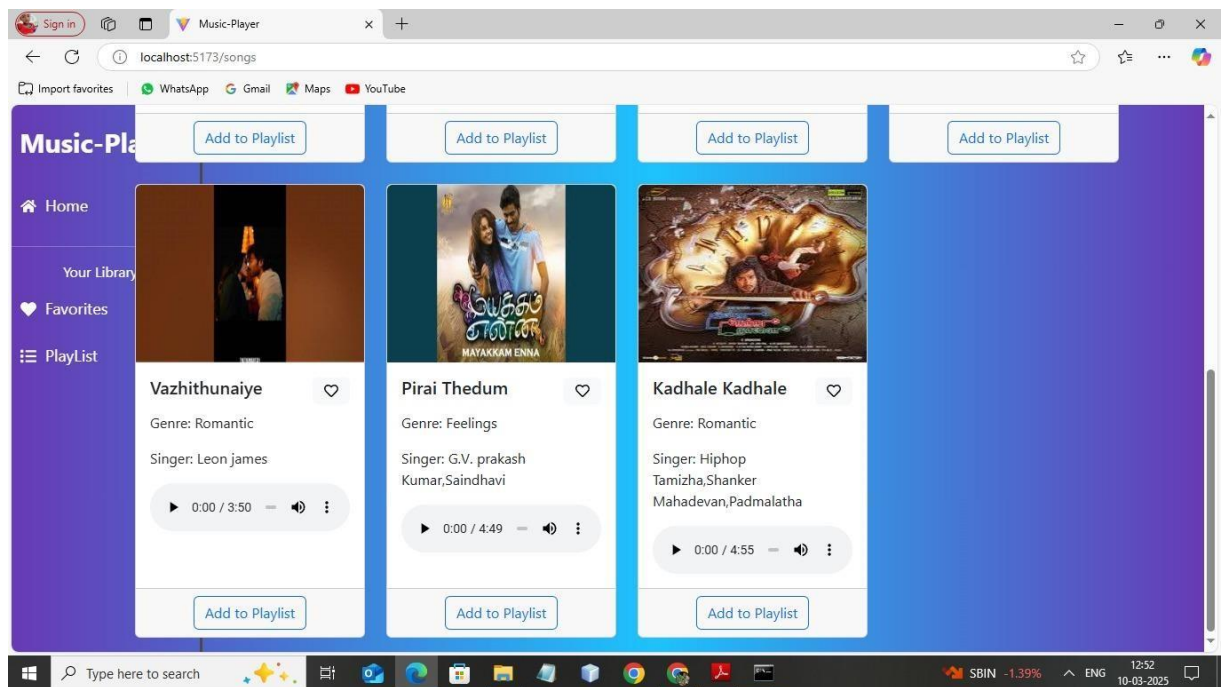
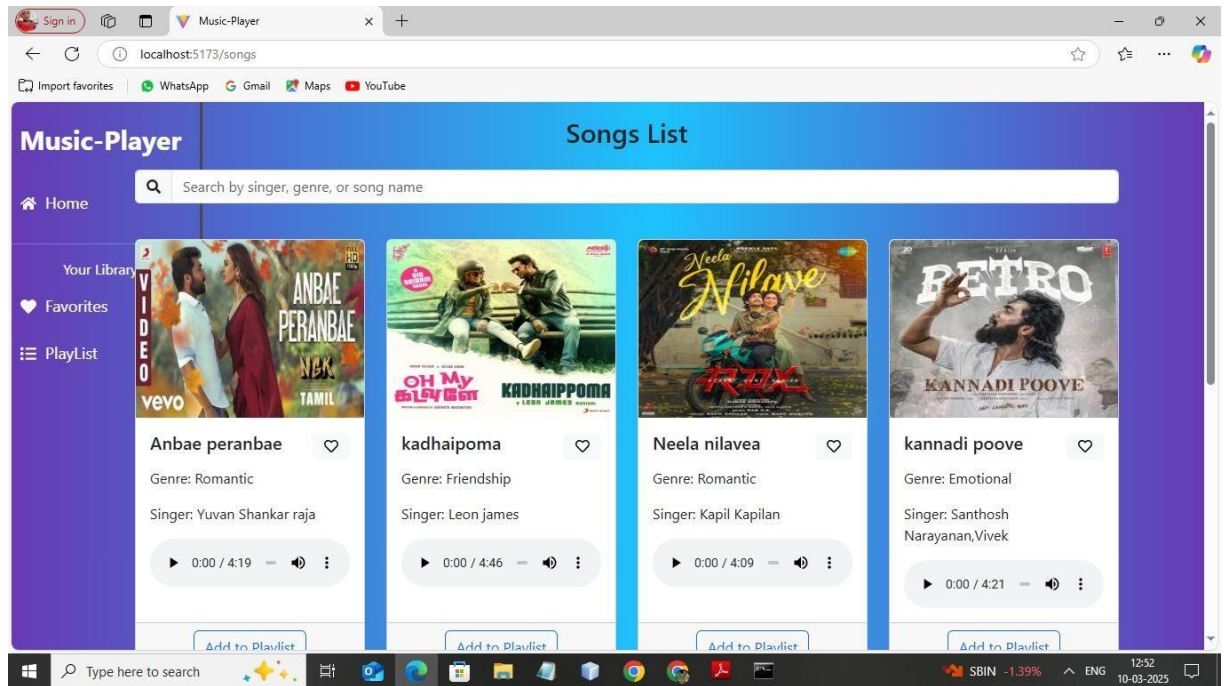
- like images

- and audio players.

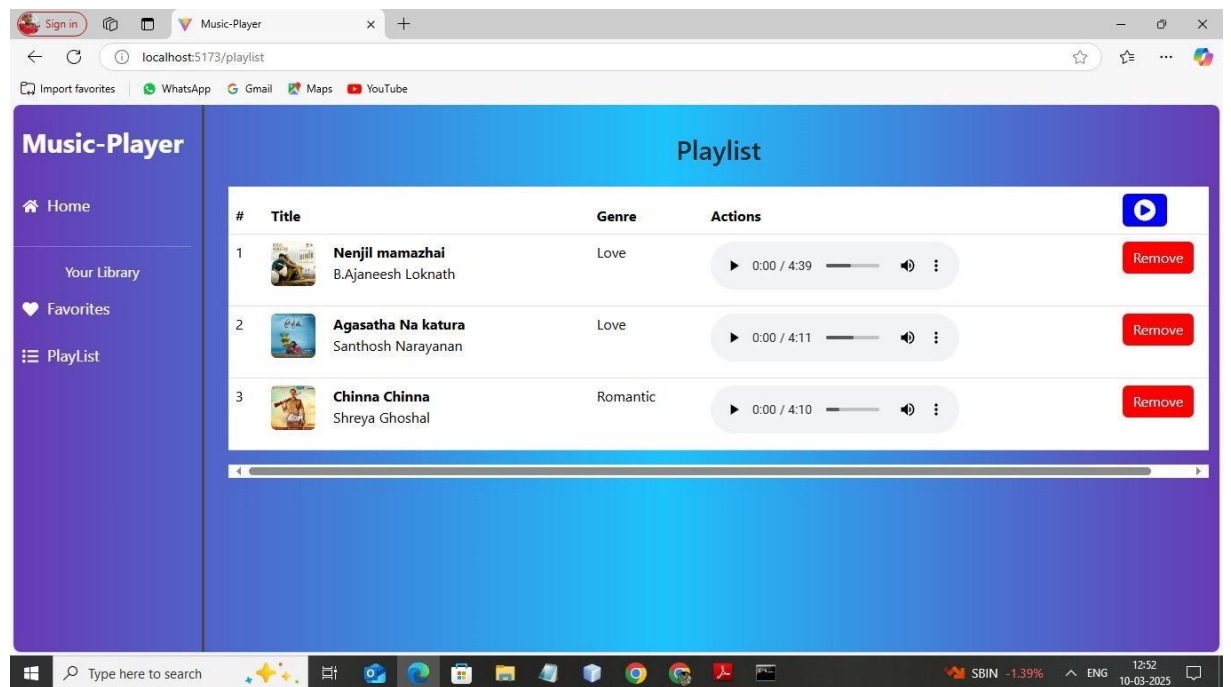
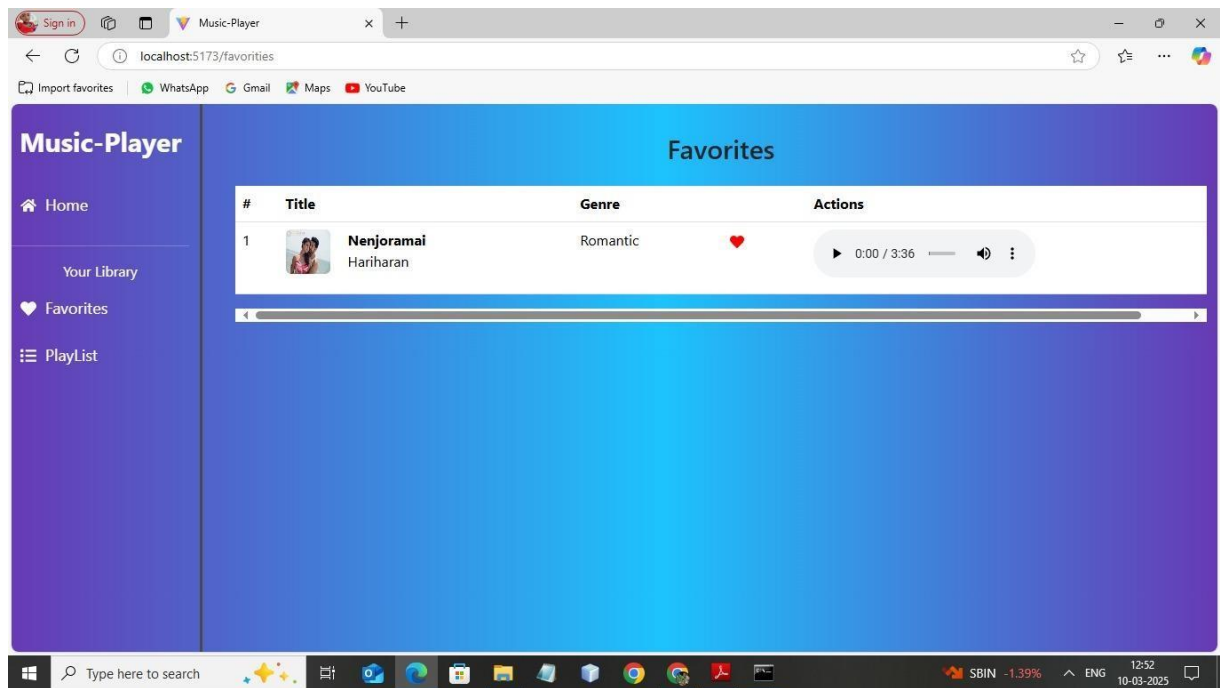
### **Project Execution:**

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 Rhythmic Tunes.



C



Project Demo Link:

[https://drive.google.com/file/d/1xg0tfEFVYL\\_sqBsqtyATol6gn0O299up/view?usp=drive\\_link](https://drive.google.com/file/d/1xg0tfEFVYL_sqBsqtyATol6gn0O299up/view?usp=drive_link)

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