

Rhythmic Tunes

(React)

DR.MGR JANAKI COLLEGE OF ARTS AND SCIENCE FOR WOMEN

(B.Sc.,COMPUTER SCIENCE- FINAL YEAR)

Presented By:

Monika S(asunm1423222208006)

Email ID: s8020962@gmail.com

Sureka T (asunm1423222208024)

Email ID: samsureka2@gmail.com

Divya S (asunm1423222207989)

Email ID: sudhasudhakarsudhasudhakar5@gmail.com

Devadharshini R(asunm1423222207986)

Email ID: devadharshini20122004@gmail.com

Introduction:-

Rhythmic tunes are the heartbeat of music, driving energy and movement through carefully crafted patterns of sound and silence. At their core, rhythmic tunes involve a sequence of beats, tempos, and time signatures that create a sense of pulse and flow. Whether it's the steady groove of a drumbeat, the syncopated rhythms of jazz, or the energetic tempo of dance music, rhythmic tunes play an essential role in shaping how we feel and respond to music.

These patterns often serve as the foundation for melodies and harmonies, providing structure and coherence to the musical composition. From the rhythmic complexity of classical music to the infectious beats of contemporary pop and hip-hop, rhythmic tunes have the power to captivate, energize, and even heal listeners. Exploring the world of rhythmic tunes opens up a deeper understanding of how music communicates beyond words, connecting people through its universal language of rhythm.

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.

Scenario-Based Intro:

Now, imagine sitting in a quiet room, listening to a soft, flowing melody on a classical piano. The rhythm is slower, more deliberate, with each note unfolding in a gentle waltz. The calming, consistent pulse of the piece soothes your mind, lulling you into a reflective state. Even without words, the rhythm speaks to your emotions, guiding you through the rise and fall of the music, evoking a deep sense of tranquility.

In each of these moments, rhythmic tunes do more than just provide structure—they create connection. They move us, shape our emotions, and even synchronize our bodies to the music. Whether it's a powerful drumbeat, a delicate piano rhythm, or a festive march, rhythmic tunes are the invisible threads weaving through every style of music, drawing listeners into a shared experience, and reminding us that rhythm is the heartbeat of all sound.

Target Audience:-

Musicians & Composers:

- **Who they are:** Aspiring or professional musicians, composers, and producers who focus on crafting music with attention to rhythm.
- **What they enjoy:** This group analyzes and experiments with rhythm in different ways, from creating rhythmic patterns and grooves to exploring time signatures and polyrhythms. Rhythmic tunes are studied as part of the creative process to build more dynamic compositions.

Project Goals and Objectives:-

Promote the understanding and appreciation of Rhythmic patterns in music, educate listeners, musicians, and dancers on the importance of rhythm as the foundational element of all musical genres.

□ **Master Rhythm Complexity:** To create music with a wide range of rhythms, moving from simple to complex patterns, to challenge both the composer and the listener.

□ **Enhance Groove and Feel:** To develop rhythmic tunes that create a strong, engaging groove, capturing the listener's attention through infectious beats and rhythmic structures.

□ **Experiment with Time Signatures:** Explore a variety of time signatures, from 4/4 and 3/4 to more unconventional ones like 7/8 or 5/4, to create unique rhythmic textures.

□ **Incorporate Rhythmic Diversity:** Mix different rhythmic techniques (e.g., polyrhythms, syncopation, and swing) within a single tune to add interest and complexity.

Key Features:-

- **Basic Patterns:** These are simple, repeating beats that form the backbone of the tune, such as quarter notes, eighth notes, or triplets.

- **Syncopation:** Emphasizing off-beats or unexpected accents, which creates tension and excitement.
- **Polyrhythms:** Layering different rhythmic patterns on top of each other, often involving two or more contrasting meters or time signatures, to create a complex and dynamic sound.
- **Changes in Tempo:** Adjusting the tempo throughout the track can add variety and change the emotional intensity.
- **Unconventional Time Signatures:** Using odd or compound time signatures (e.g., 5/4, 7/8) can add uniqueness and a distinct rhythmic feel to the music.
- **Backbeat:** Common in genres like rock, pop, and funk, where the emphasis is placed on the 2nd and 4th beats of a 4/4 measure.
- **Breakdowns:** Temporary reductions in rhythm or accompaniment that allow for tension-building before returning to full intensity.
- **Syncopated Melody:** A melody that works in tandem with the rhythm to create tension by emphasizing off-beats or using rhythmic complexity.

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:

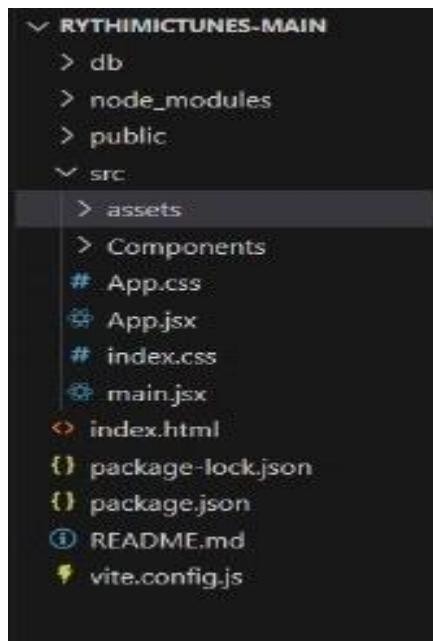
```
npm run dev
```

This command launches the development server, and you can access your React app at <http://localhost:5173> 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.

Demolink:

https://drive.google.com/file/d/1nii6agymKUYUhAdKS7_-GarehX0Lu8Fa/view?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**

- 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:-

```
dbjson > ...
{
  "items": [
    {
      "id": 1,
      "title": "Bol Do Na Zara",
      "singer": "Armaan Malik",
      "genre": "Romantic",
      "imgUrl": "https://c.saaavncdn.com/709/Bol-Do-Na-Zara-Instrumental-Hindi-2018-20181221231527-500x500.jpg",
      "songUrl": "Songs/Bol Do Na Zara.mp3"
    },
    {
      "id": 2,
      "title": "Chaleya",
      "singer": "Arijit Singh",
      "genre": "Romantic",
      "imgUrl": "https://akm-img-a-in.tosshub.com/indiatoday/images/media_bank/202309/chaleya-song-141618193-1x1.jpg?VersionId=4H90EXybp0zu2rGuku",
      "songUrl": "Songs/chaleya.mp3"
    },
    {
      "id": 3,
      "title": "Humna Mere",
      "singer": "Jubin Nautiyal",
      "genre": "Romantic",
      "imgUrl": "https://c.saaavncdn.com/259/Humna-Mere-Hindi-2018-20180522-500x500.jpg",
      "songUrl": "Songs/Humna Mere.mp3"
    },
    {
      "id": 4,
      "title": "Saari Duniya Jalaa Denge",
      "singer": "B Praak",
      "genre": "Emotional",
      "imgUrl": "https://c.saaavncdn.com/623/Saari-Duniya-Jalaa-Denge-From-AMIMAL-Hindi-2023-20231124191004-500x500.jpg",
      "songUrl": "Songs/Saari Duniya Jalaa Denge.mp3"
    },
    {
      "id": 5,
      "title": "Sanam Teri Kasam",

```

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:
 - 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:

- 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:-

```

son  X
() dbjson > [ ] items > {} 3
  "items": [
    {
      "title": "Saari Duniya Jalaa Denge",
      "singer": "B Praak",
      "genre": "Emotional",
      "imgUrl": "https://c.saavncdn.com/623/Saari-Duniya-Jalaa-Denge-From-ANIMAL-Hindi-2023-20231124191004-500x500.jpg",
      "songUrl": "Songs/Saari Duniya Jalaa Denge.mp3"
    },
    {
      "id": 5,
      "title": "Sanam Teri Kasam",
      "singer": "Ankit Tiwari",
      "genre": "Emotional",
      "imgUrl": "https://i.scdn.co/image/ab67616d0000b2731c79ed9daa0ef22e185c71cd",
      "songUrl": "Songs/Sanam Teri Kasam.mp3"
    },
    {
      "id": 6,
      "title": "Tum Hi Ho",
      "singer": "Arijit Singh",
      "genre": "Emotional",
      "imgUrl": "https://lyrichindi.com/wp-content/uploads/2023/09/desktop-wallpaper-meri-aashiqui-tum-hi-ho-tum-hi-ho.jpg",
      "songUrl": "Songs/Tum Hi Ho.mp3"
    },
    {
      "id": 7,
      "title": "Zihaal-e-Miskin",
      "singer": "Shreya Ghoshal",
      "genre": "Emotional",
      "imgUrl": "https://imgnew.outlookindia.com/uploadimage/library/16_9/16_9_5/IMAGE_1685023719.jpg",
      "songUrl": "Songs/Zihaal e Miskin.mp3"
    }
  ],
  "favorites": [],
  "playlist": []
}

```

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 currentlyPlaying: 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):**

- o Checks if an item exists in the wishlist (favorites) 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:**

- o Filters items based on the searchTerm.
- o Matches title, singer, or genre with the lowercase version of searchTerm.

- **JSX:**

- o Renders a form with an input field (Form, InputGroup, Button, FaSearch) for searching items.
- o Maps over filteredItems to render each item in the UI.
- o 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:-

```

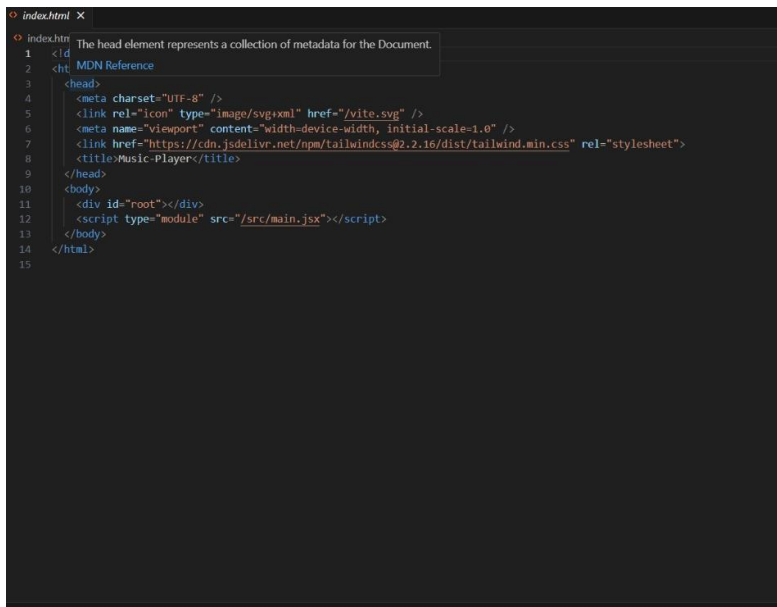
src > App.jsx > ...
1 import 'bootstrap/dist/css/bootstrap.min.css';
2 import './App.css'
3 import { BrowserRouter,Routes,Route } from 'react-router-dom'
4 import Songs from './components/Songs'
5 import Sidebar from './components/Sidebar'
6 import Favorites from './components/Favorites'
7 import Playlist from './components/Playlist';
8
9 (alias) function BrowserRouter({ basename, children, future, window, }: BrowserRouterProps):
10 React.JSX.Element
11   import BrowserRouter
12   ret
13   <d A <Router> for use in web browsers. Provides the cleanest URLs.
14     <BrowserRouter>
15       <div>
16         <Sidebar/>
17       </div>
18       <div>
19         <Routes>
20           <Route path="/" element={}<Songs/> } />
21           <Route path="/songs" element={}<Songs/> } />
22           <Route path="/favorites" element={}<Favorites/> } />
23           <Route path="/playlist" element={}<Playlist/> } />
24         </Routes>
25       </div>
26     </BrowserRouter>
27   </div>
28   )
29 }
30
31 export default App
32
33
34

```

```

# App.css X
src > # App.css > *
1 {
2   padding: 0;
3   margin: 0;
4 }
5
6 body{
7   background: rgb(103,58,180);
8   background: linear-gradient(90deg, rgba(103,58,180,1) 0%, rgba(29,195,253,1) 54%, rgba(103,58,180,1) 100%);
9 }

```



```
1 <!-- The head element represents a collection of metadata for the Document. -->
2 <!-- MDN Reference -->
3 <head>
4   <meta charset="UTF-8" />
5   <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7   <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.16/dist/tailwind.min.css" rel="stylesheet">
8   <title>Music-Player</title>
9 </head>
10 <body>
11   <div id="root"></div>
12   <script type="module" src="/src/main.jsx"></script>
13 </body>
14 </html>
15
```

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 () 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 (<div className="card h-100">).

- **Card Content:**

- o Displays the item's image (img), title (<h5 className="card-title">), genre (<p className="card-text">), and singer (<p className="card-text">).
 - o 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:**

- o Adds a heart icon button (<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 (<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:

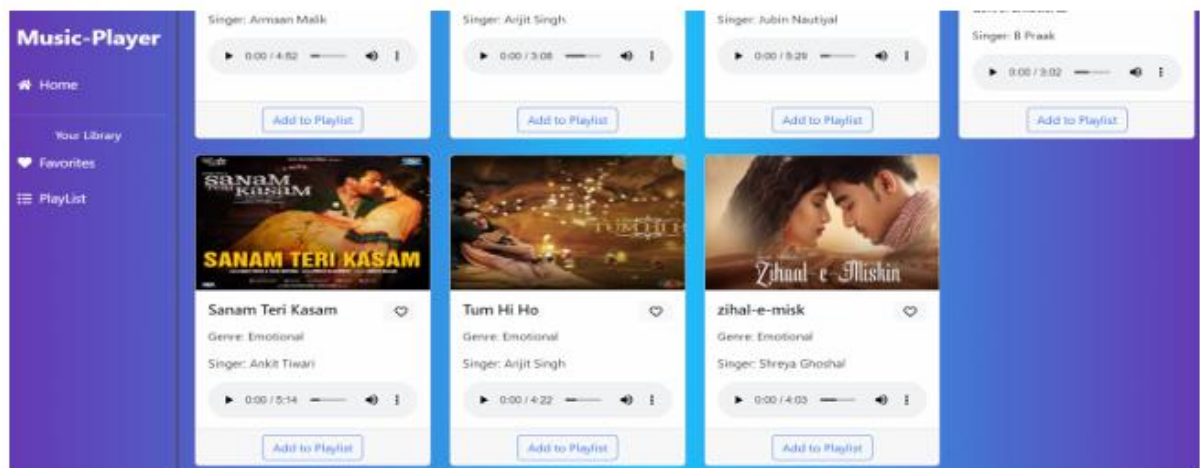
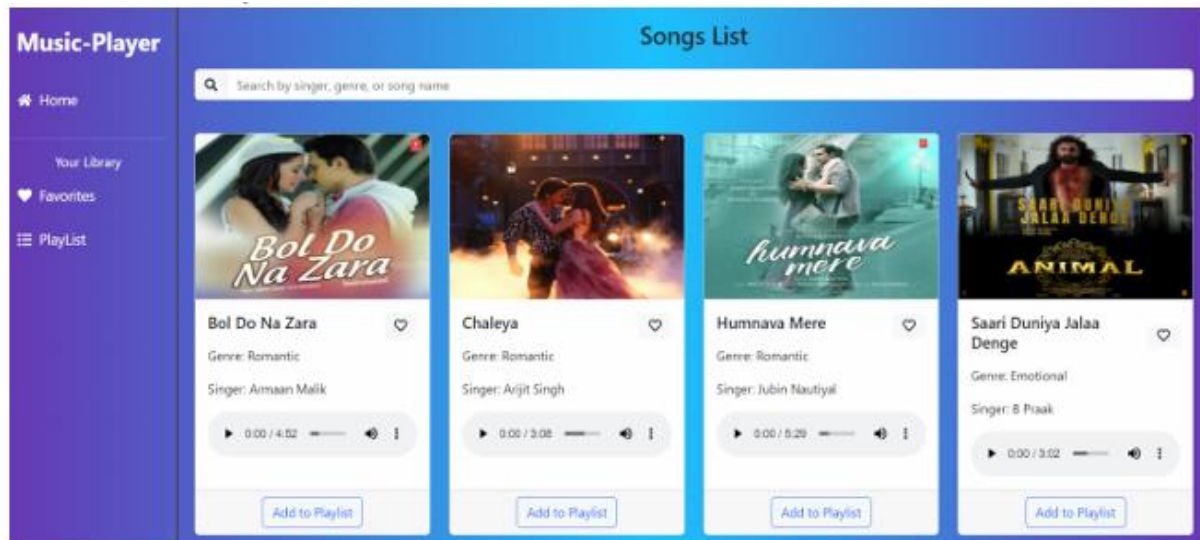
After completing the code, run the react application by using the command “npm start” or “npm run dev” if you are using vite.js

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

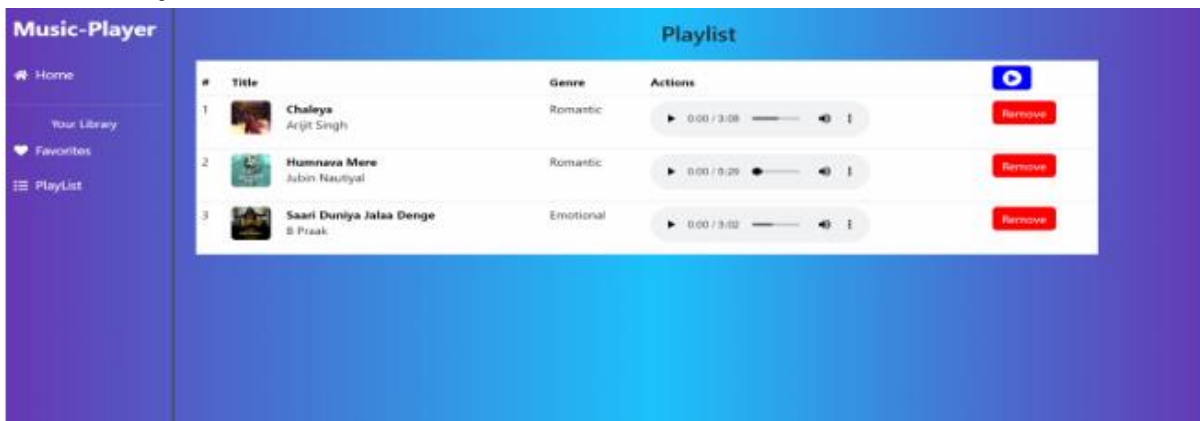
After that launch the Rythimic Tunes.

Here are some of the screenshots of the application.

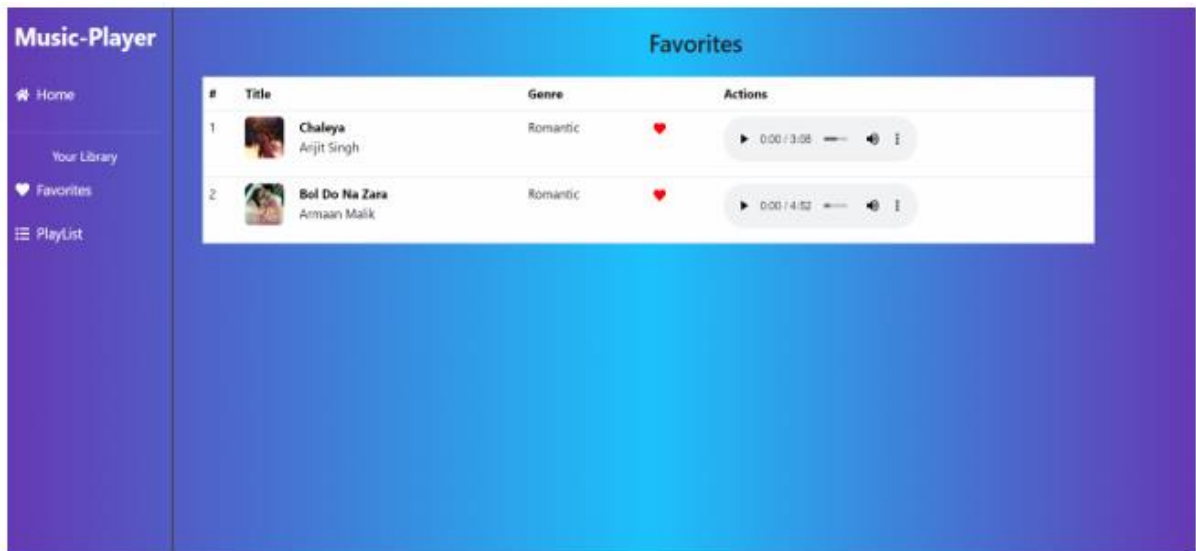
Hero components



Playlist



Favourites



Project Demo link:

https://drive.google.com/drive/folders/1q3187PkwLptPGRrwAXTCflpU2_DhcLEo?usp=drive_link