RYTHIMIC TUNES(REACT)

(Music Streaming Application)

Team ID: SWTID1741423241154402

Team Leader:

Kamini .K

kaminitamil17@gmail.com

Team Members :

Kaviya .K

kkaviya6862@gmail.com

Nivetha .M

nivethamani714@gmail.com

Thenmozhi .R

thenmozhiranganathan30@gmail.com

Arun kumar .M

arunvijay7743@gmail.com

Introduction:-

Welcome to the future of musical indulgence – anunparalleledaudioexperienceawaits 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 redefinehowyouinteractwithandimmerseyourselfintheworldof 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 journeyas 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-BasedIntro:-

Imaginesteppingontoabustlingcitystreet,thesoundsofcarshonking,peoplechatting, 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, "RythimicTunes.

Withjustafewtaps,you'retransportedtoaworldofmusictailoredtoyourtastes.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.

TargetAudience:-

MusicStreamingisdesignedforadiverseaudience,including:

- **MusicEnthusiasts:**PeoplepassionateaboutenjoyingandlisteningMusicThrough out there free time to relax themselves.

ProjectGoalsandObjectives:-

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.

KeyFeatures:-

- ? **Song Listings:** Display a comprehensive list of available songswithdetailssuchas title, artist, genre, and release date.
- ? **PlaylistCreation:**Empoweruserstocreatepersonalizedplaylists,addingand 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:-

HerearethekeyprerequisitesfordevelopingafrontendapplicationusingReact.js:

? **Node.jsandnpm**:

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.

InstallNode.jsandnpmonyourdevelopmentmachine,astheyarerequiredto run JavaScript on the server-side.

- Download:https://nodejs.org/en/download/
- Installationinstructions: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 iteasierto build dynamic and responsive web applications.

InstallReact.js,aJavaScriptlibraryforbuildinguserinterfaces.

- CreateanewReactapp:

```
npmcreatevite@latest
```

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

- Navigatetotheprojectdirectory:

```
cdproject-name

npm install
```

- RunningtheReactApp:

WiththeReactappcreated,youcannow start the development server and see your React application in action.

- Startthedevelopmentserver:

```
npmrundev
```

Thiscommandlaunchesthedevelopmentserver,andyoucanaccessyourReactappat http://localhost:5173 inyourwebbrowser.

?**HTML, CSS,andJavaScript**:BasicknowledgeofHTMLforcreatingthestructureof your app, CSS for styling, and JavaScript for client-side interactivity is essential.

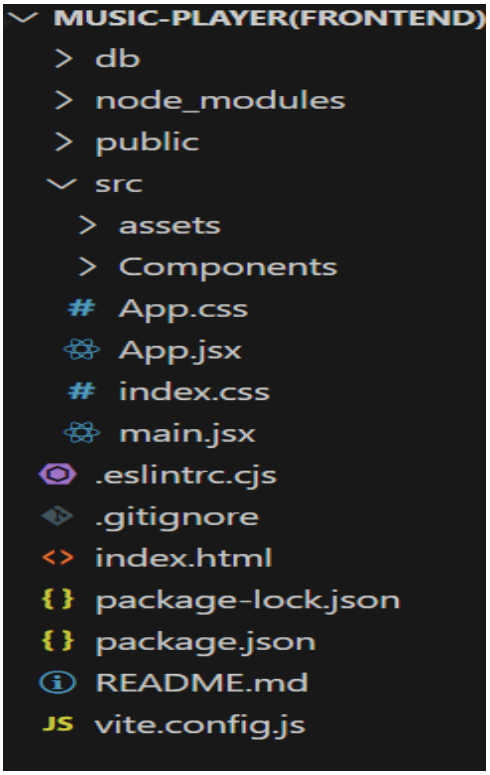
?**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. PlatformslikeGitHuborBitbucket can host your repository.

- Git:Downloadandinstallationinstructionscanbefoundat: 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.

- VisualStudioCode:Downloadfrom https://code.visualstudio.com/download
- SublimeText:Downloadfrom https://www.sublimetext.com/download
- WebStorm:Downloadfrom https://www.jetbrains.com/webstorm/download

Projectstructure:



The project structure may vary depending on the specific library, framework, programming language, or development approach used. It's essential to organize the filesanddirectoriesinalogicalandconsistentmannertoimprovecodemaintainability and collaboration among developers.

app/app.component.css, src/app/app.component: These files are part of the main AppComponent,whichservesastherootcomponentfortheReactapp.Thecomponent handles the overall layout and includes the router outlet for loading different components based on the current route.

PROJECTFLOW:-

Projectdemo:

Beforestartingtoworkonthisproject,let'sseethedemo.

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

Usethecode in:

https://drive.google.com/drive/folders/1BkYWfW_K3ek_UgtXNTAsDqlhdCuqz6nT?usp=drive_link

Milestone1:ProjectSetupandConfiguration:

1.Installrequiredtoolsandsoftware:

- **Installationofrequiredtools**:

1. Opentheprojectfoldertoinstallnecessarytools In this project, we use:
    - ReactJs
    - ReactRouterDom
    - ReactIcons
    - Bootstrap/tailwindcss
    - Axios

- Forfurtherreference,usethefollowingresources
    - https://react.dev/learn/installation
    - https://react-bootstrap-v4.netlify.app/getting-started/introduction/
    - https://axios-http.com/docs/intro
    - https://reactrouter.com/en/main/start/tutorial

Milestone2:ProjectDevelopment:

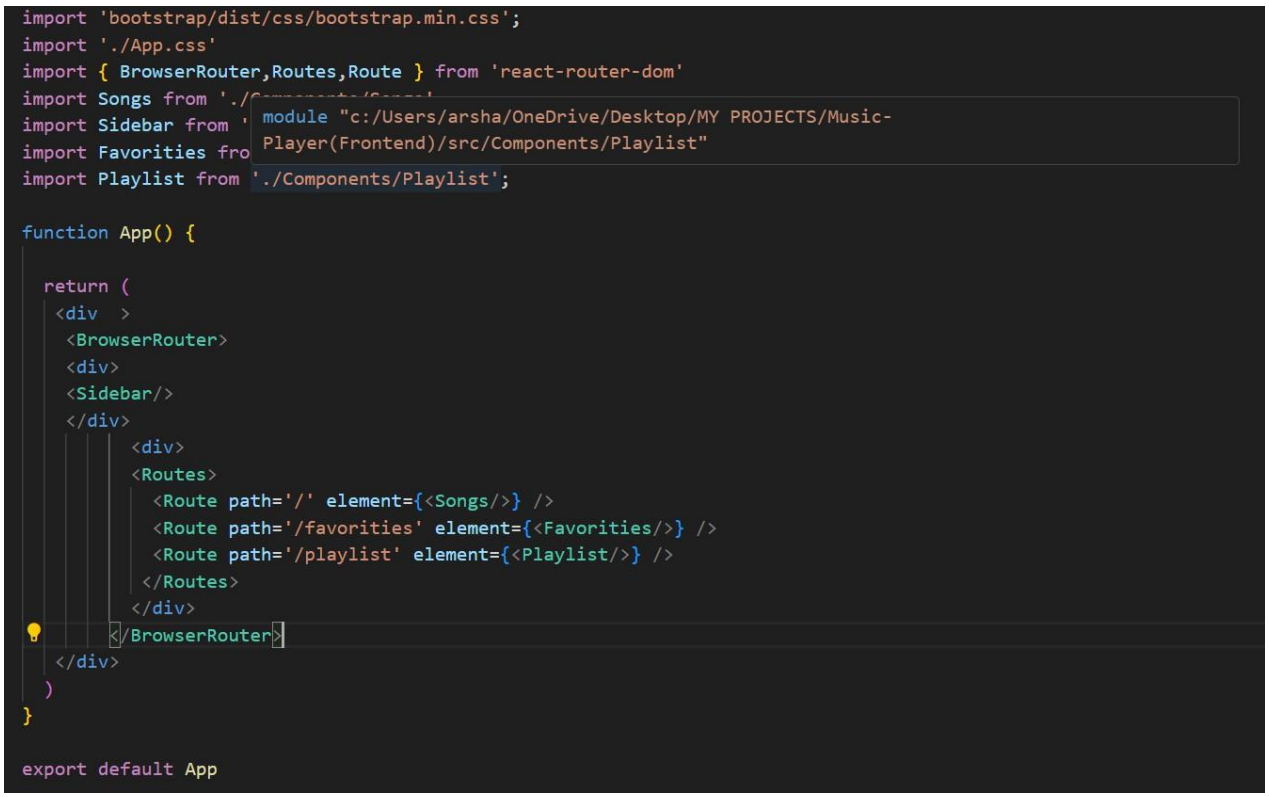
1. SetupReactApplication:

• CreateReactapplication.

• ConfigureRouting.

• Installrequiredlibraries.

Setting Up Routes:-



CodeDescription:-

- Imports Bootstrap CSS (bootstrap/dist/css/bootstrap.min.css) for styling components.
- ImportscustomCSS(./App.css)foradditionalstyling.

- 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 ofthe application.
- UsesBrowserRouterastheroutercontainertoenableroutingfunctionality.
- Includesadivastherootcontainerfortheapplication.
- WithinBrowserRouter,wrapscomponentsinsidetwodivcontainers:
    - 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.
    - InsideRoutes,definesseveralRoutecomponents:
        - Routewithpath='/'renderstheSongscomponent when the root path is accessed (/).
        - Routewithpath='/favorities'renderstheFavoritiescomponent when the /favorities path is accessed.
        - Routewithpath='/playlist'rendersthePlaylistcomponentwhenthe /playlistpathisaccessed.
- Exports the App component as the default export, making it available for use in other parts of the application.

FetchingSongs:-

```
import React, { useState, useEffect } from 'react';
import { Button, Form, InputGroup } from 'react-bootstrap';
import { FaHeart, FaRegHeart, FaSearch } from 'react-icons/fa';
import axios from 'axios';
import './sidebar.css'

function Songs() {
  const [items, setItems] = useState([]);
  const [wishlist, setWishlist] = useState([]);
  const [playlist, setPlaylist] = useState([]);
  const [currentlyPlaying, setCurrentlyPlaying] = useState(null);
  const [searchTerm, setSearchTerm] = useState('');

  useEffect(() => {
    // Fetch all items
    axios.get('http://localhost:3000/items')
      .then(response => setItems(response.data))
      .catch(error => console.error('Error fetching items: ', error));

  // Fetch favorities items
     axios.get('http://localhost:3000/favorities')
     .then(response => setWishlist(response.data))
     .catch(error => {
       console.error('Error fetching Favvorities:', error);
       // Initialize wishlist as an empty array in case of an error
       setWishlist([]);
     });

    // Fetch playlist items
    axios.get('http://localhost:3000/playlist')
      .then(response => setPlaylist(response.data))
      .catch(error => {
        console.error('Error fetching playlist: ', error);
        // Initialize playlist as an empty array in case of an error
        setPlaylist([]);
      });
      // Function to handle audio play
    const handleAudioPlay = (itemId, audioElement) => {
        if (currentlyPlaying && currentlyPlaying !== audioElement) {
          currentlyPlaying.pause(); // Pause the currently playing audio
        }
        setCurrentlyPlaying(audioElement); // Update the currently playing audio

      };
```

```
    // Event listener to handle audio play
    const handlePlay = (itemId, audioElement) => {
      audioElement.addEventListener('play', () => {
        handleAudioPlay(itemId, audioElement);
      });
    };

    // Add event listeners for each audio element
    items.forEach((item) => {
      const audioElement = document.getElementById(`audio-${item.id}`);
      if (audioElement) {
        handlePlay(item.id, audioElement);
      }
    });

    // Cleanup event listeners
    return () => {
      items.forEach((item) => {
        const audioElement = document.getElementById(`audio-${item.id}`);
        if (audioElement) {
          audioElement.removeEventListener('play', () => handleAudioPlay(item.id, audioElement));
        }
      });
    };
  }, [items,currentlyPlaying, searchTerm]);

  const addToWishlist = async (itemId) => {
    try {
      const selectedItem = items.find((item) => item.id === itemId);
      if (!selectedItem) {
        throw new Error('Selected item not found');
      }
      const { title, imgUrl, genre, songUrl, singer, id: itemId2 } = selectedItem;
      await axios.post('http://localhost:3000/favorities', { itemId: itemId2, title, imgUrl, genre, songUrl, singer });
      const response = await axios.get('http://localhost:3000/favorities');
      setWishlist(response.data);
    } catch (error) {
      console.error('Error adding item to wishlist: ', error);
    }
  };
```

CodeDescription:-

- **useState:**
  - `items`:Holdsanarrayofallitemsfetchedfrom `http://localhost:3000/items`.

- `wishlist`:Storesitemsmarkedasfavoritesfetchedfrom `http://localhost:3000/favorities`.
- `playlist`:Storesitemsaddedtotheplaylistfetched from `http://localhost:3000/playlist`.
- `currentlyPlaying`:Keepstrackofthecurrentlyplayingaudioelement.
- `searchTerm`:Storesthecurrentsearchtermenteredbytheuser.

- **DataFetching:**
  - Uses`useEffect`tofetchdata:
    - Fetchesallitems(`items`)from`http://localhost:3000/items`.
    - Fetchesfavoriteitems(`wishlist`)from `http://localhost:3000/favorities`.
    - Fetchesplaylistitems(`playlist`)from `http://localhost:3000/playlist`.
  - Setsstatevariables(`items`,`wishlist`,`playlist`)basedonthefetched data.
- **AudioPlayback Management:**
  - Setsupaudioplayeventlistenersandcleanupforeachitem:
    - `handleAudioPlay`: Manages audio playback by pausing the currently playing audio when a new one starts.
    - `handlePlay`:Addseventlistenerstoeachaudioelementtotrigger `handleAudioPlay`.
  - Ensuresthatonlyoneaudioelementplaysata time by pausing others when a new one starts playing.

- **addToWishlist(itemId):**
  - Addsanitemtothewishlist(`favorities`)bymakingaPOSTrequestto `http://localhost:3000/favorities`.
  - Updatesthe`wishlist`stateafteraddinganitem.
- **removeFromWishlist(itemId):**
  - Removes an item from the wishlist (`favorities`) by makingaDELETErequest to `http://localhost:3000/favorities/{itemId}`.
  - Updatesthe`wishlist`stateafterremovinganitem.
- **isItemInWishlist(itemId):**
  - Checksifanitemexistsinthewishlist(`favorities`)basedonits`itemId`.
- **addToPlaylist(itemId):**

- Addsanitemtotheplaylist(`playlist`)bymakingaPOSTrequestto `http://localhost:3000/playlist`.
- Updatestheplaylistafteraddinganitem.

- **removeFromPlaylist(itemId):**
  - Removesanitemfromtheplaylist(`playlist`)bymakingaDELETErequestto `http://localhost:3000/playlist/{itemId}`.
  - Updatestheplaylistafterremovinganitem.
- **isItemInPlaylist(itemId):**
  - Checksifanitemexistsintheplaylist(`playlist`)basedonits`itemId`.

- **filteredItems:**
  - Filters`items`basedonthe`searchTerm`.
  - Matches `title`, `singer`, or `genre`with thelowercaseversionof `searchTerm`.

- **JSX:**
  - Renders aformwithaninputfield(`Form`,`InputGroup`,`Button`,`FaSearch`) for searching items.
  - Mapsover`filteredItems`torendereachitemintheUI.
  - Includesbuttons(`FaHeart`,`FaRegHeart`)toadd/removeitemsfrom `wishlist`and `playlist`.
  - Rendersaudioelementsforeachitemwithplay/pausefunctionality.

- **ErrorHandling:**
  - Catchesandlogserrorsduringdatafetching(`axios.get`).
  - Handleserrorswhenadding/removingitemsfrom`wishlist`and`playlist`.

FrontendCodeForDisplayingSongs:-

```
return (
  <div style={{display:"flex", justifyContent:"flex-end"}}>
  <div className="songs-container" style={{width:"1300px"}}>
    <div className="container mx-auto p-3">
      <h2 className="text-3xl font-semibold mb-4 text-center">Songs List</h2>
      <InputGroup className="mb-3">
        <InputGroup.Text id="search-icon">
          <FaSearch />
        </InputGroup.Text>
        <Form.Control
          type="search"
          placeholder="Search by singer, genre, or song name"
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          className="search-input"
        />
      </InputGroup>
      <br />
      <div className="row row-cols-1 row-cols-md-2 row-cols-lg-3 row-cols-xl-4 g-4">
        {filteredItems.map((item, index) => (
          <div key={item.id} className="col">
            <div className="card h-100">
              <img
                src={item.imgUrl}
                alt="Item Image"
                className="card-img-top rounded-top"
                style={{ height: '200px', width: '100%' }}
              />
              <div className="card-body">
                <div className="d-flex justify-content-between align-items-center mb-2">
                  <h5 className="card-title">{item.title}</h5>
                  {isItemInWishlist(item.id) ? (
                    <Button
                      variant="light"
                      onClick={() => removeFromWishlist(item.id)}
                    >
                      <FaHeart color="red" />
                    </Button>
                  ) : (
                    <Button
                      variant="light"
                      onClick={() => addToWishlist(item.id)}
                    >
```

```
                        <FaRegHeart color="black" />
                      </Button>
                    )}
                  </div>
                  <p className="card-text">Genre: {item.genre}</p>
                  <p className="card-text">Singer: {item.singer}</p>
                  <audio controls className="w-100" id={`audio-${item.id}`} >
                    <source src={item.songUrl} />
                  </audio>
                </div>
                <div className="card-footer d-flex justify-content-center">
                  {isItemInPlaylist(item.id) ? (
                    <Button
                      variant="outline-secondary"
                      onClick={() => removeFromPlaylist(item.id)}
                    >
                      Remove From Playlist
                    </Button>
                  ) : (
                    <Button
                      variant="outline-primary"
                      onClick={() => addToPlaylist(item.id)}
                    >
                      Add to Playlist
                    </Button>
                  )}
                </div>
              </div>
            </div>
          ))}
        </div>
      </div>
      </div>
    );
  }

export default Songs;
```

CodeDescription:-

- **ContainerSetup:**
  - 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"`).
- **SearchInput:**
  - 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"`.

- **CardLayout:**
  - 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">`).
- **CardContent:**
  - 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} />`).
- **WishlistandPlaylistButtons:**
  - 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)`).

- **ButtonClickHandlers:**
  - 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)`).

- **CardStyling:**
  - AppliesBootstrapclasses(`card`,`card-body`,`card-footer`)forstylingthe card components.
  - Usescustomstyles(`rounded-top`,`w-100`)forspecificelementslikeimages and audio players.

ProjectExecution:

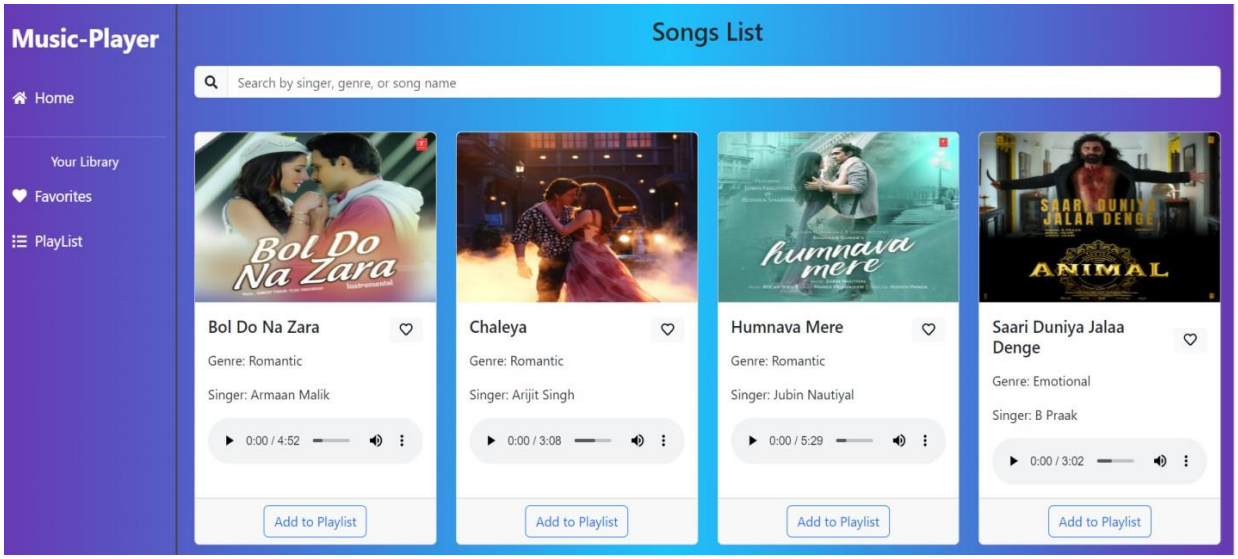
Aftercompletingthecode,runthereactapplicationbyusingthecommand“npmstart”or “npm run dev” if you are using vite.js

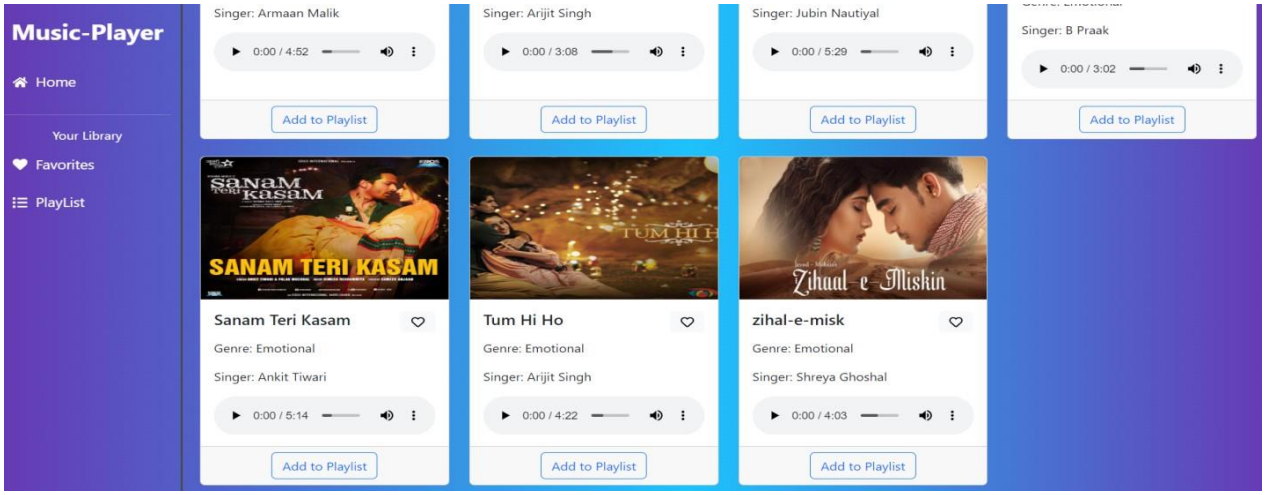
And the Open new Terminal type this command “json-server --watch./db/db.json” to start the json server too.

AfterthatlaunchtheRythimicTunes.

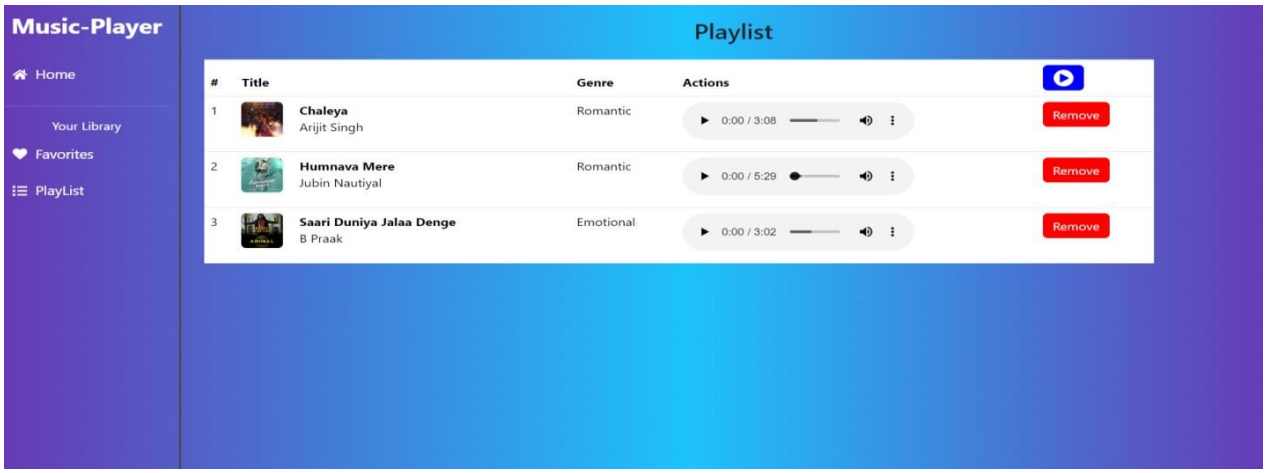
Herearesomeofthescreenshotsoftheapplication.

? **Herocomponents**

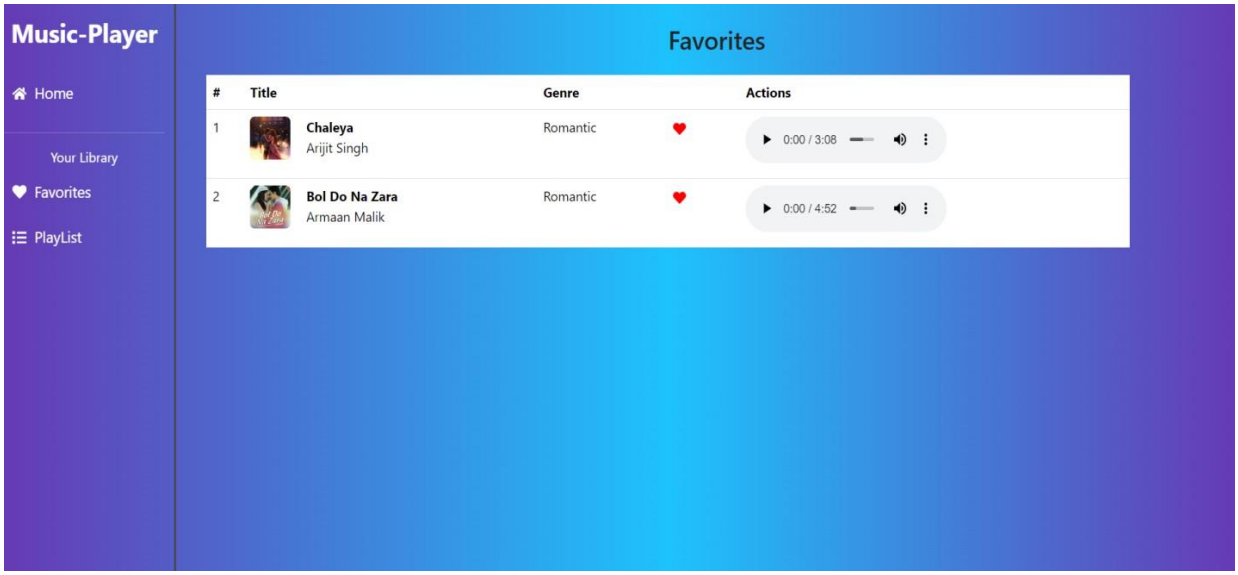




? Playlist



? Favorites



ProjectDemolink:

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

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