**RYTHMIC TUNES**

**Project Documentation**

**Introduction**

**Project Title**: RYTHMIC TUNES



**Team Members**

|  |  |  |  |
| --- | --- | --- | --- |
| Team details | Gmail address details: | Name of the team members |  |
|  |  |  |  |
| Team leader | gopisukash22@gmai.com | G SUKASH |  |
|  |  |  |
|  |  |  |  |
| Team member 1 | gowrikarthik1412@gmail.com | V KARTHIK |  |
|  |  |
|  |  |  |  |
| Team member 2 | bennybenny3659@gmail.com | A BENNY |  |
|  |  |
|  |  |  |  |
| Team member 3 | prakashvivek1008@gmail.com | V JAYAPRAKASH |  |
|  |  |  |
|  |  |  |  |



1. **Project Overview**

 **Purpose**

The primary purpose of this music application is to provide users with a comprehensive and engaging platform for discovering, listening to, and sharing music. The app aims to achieve the following objectives:



**Music Discovery**: To help users discover new artists, genres, and songs tailored to their personal tastes through accurate playlists, recommendations, and trending music features.



**Seamless Listening Experience**: To ofer a high-quality audio playback experience with an intuitive music player that allows users to easily navigate their music library, create playlists, and enjoy their favourite tracks without interruptions.



**Social Interaction**: To foster a community of music lovers by enabling users to share their playlists, follow friends, and engage with others through social media integration, thereby enhancing the overall music experience.



**Personalization**: To provide personalized music recommendations based on users' listening habits, preferences, and interactions within the app, ensuring that each user feels a unique connection to the music they love.



**Accessibility**: To ensure that users can access their music library anytime and anywhere, whether they are using a mobile device or a web browser, making music an integral part of their daily lives. **User Empowerment**: To empower users to take control of their music experience by allowing them to create, manage, and share their playlists, thus encouraging creativity and self-expression through music.



**Ofine Listening**: To provide users with the option to download their favorite songs and playlists for ofine listening, ensuring that they can enjoy music without needing an internet connection. **User Engagement**: To keep users engaged through features like music quizzes, challenges, and interactive content that encourages exploration and interaction with the app.



**Artist Support**: To create a platform that supports emerging and independent artists by providing them with a space to showcase their music and connect with fans, thereby promoting diversity in the music industry.



**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.

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

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

1. **Architecture**

**Component Structure**:



Here are some components 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 efcient 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.



**Routing**:

1. **Overview of Routing**

Routing allows users to navigate between diferent parts of the application without reloading the page.

It helps in creating a single-page application (SPA) experience, where diferent views are rendered based on the URL.

1. **Key Routes**

Here are some common routes you might implement in a music application:

**Home Route (/)**

Displays featured playlists, trending songs, and recommended artists.

**Search Route (/search)**

Provides a search interface for users to fnd songs, artists, and albums.

**Library Route (/library)**

Shows the user's music library, including their songs, albums, and playlists.

**Playlist Route (/playlist/:id)**

Displays a specifc playlist based on the playlist ID in the URL.

**Song Route (/song/:id)**

Shows detailed information about a specifc song, including playback options.

**User Profle Route (/profle)**

Displays the user's profle, including their playlists and listening history.

**Settings Route (/settings)**

Allows users to manage their account settings and preferences.

* 1. **Setting Up Routing in React**

1. **Dynamic Routing**

Use dynamic routing to handle routes that require parameters, such as playlist or song IDs.

For example, the route /playlist/:id allows you to access a specifc playlist based on its ID.

**5. Navigation Links**

Implement navigation links to allow users to navigate between diferent routes easily. You can use the Link component from React Router:



**6. Handling Not Found Routes**

Implement a catch-all route to handle 404 errors for undefned routes:



1. **Setup Instructions**
   * Installation of required tools:

Open the project folder to install necessary tools.In this project, we use:

React Js



React Router Dom



React Icons



Bootsrap/tailwind css



Axios



* For further reference, use the following resources

<https://react-bootstrap-v4.netlify.app/getting-started/introduction/>



<https://axios-http.com/docs/intro>

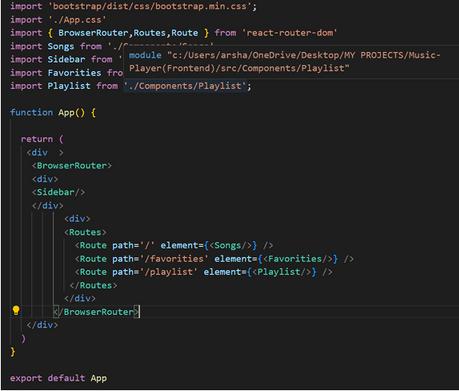


<https://reactrouter.com/en/main/start/tutorial>



●Setup React Application:

* Create React application.
* Confgure Routing.
* Install required libraries. Setting Up Routes:-



**Code Descripton:-**

* Imports Bootstrap CSS (bootstrap/dist/css /bootstrap.min.css) for styling components.
* Imports custom CSS (./App.css) for additonal styling.
* Imports BrowserRouter, Routes, and Route from react-router-dom for setng up client-side routng in the applicaton.

* Defnes the App functonal component that serves as the root component of the applicaton.
* Uses BrowserRouter as the router container to enable routng functonality.
* Includes a div as the root container for the applicaton.
* Within BrowserRouter, wraps components inside two div containers:

\*The frst div contains the Sidebar component, likely serving navigaton or additonal content.

\*The second div contains the Routes component from React Router, which handles rendering components based on the current route.

\*Inside Routes, defnes several Route components:

* Route with path='/' renders the Songs component when the root path is accessed (/).
* Route with path='/favorites' renders the Favorites component when the /favorites 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 applicaton.

**Fetching Songs:-**

1.Two Two

2.Chaleya

3.Humnava Mere

4.Saari Duniya Jalaa Denge

5.kissik

6.Mattikinaru Mattikinaru

7.Kannadi prove

**Code Descripton:-**

● useState:

●items: Holds an array of all items fetched from

htp://localhost:3000/items.

* + wishlist: Stores items marked as favorites fetched from htp://localhost:3000/favorites.
* playlist: Stores items added to the playlist fetched from htp://localhost:3000/playlist.
* currentlyPlaying: Keeps track of the currently playing audio element. ●searchTerm: Stores the current search term entered by the user.
* Data Fetching:

●Uses useEfect to fetch data:

* Fetches all items (items) from htp://localhost:3000/items.
* Fetches favorite items (wishlist) from htp://localhost:3000/favorites.
* Fetches playlist items (playlist) from htp://localhost:3000/playlist.

●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 tme by pausing others when a new one starts playing.

* addToWishlist(itemId):
* Adds an item to the wishlist (favorites) by making a POST request to htp://localhost:3000/favorites.
* Updates the wishlist state afer adding an item.
* removeFromWishlist(itemId):

●Removes an item from the wishlist (favorites) by making a DELETE request to htp://localhost:3000/favorites/{itemId}.

●Updates the wishlist state afer 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 htp://localhost:3000/playlist.

●Updates the playlist state afer adding an item.

● removeFromPlaylist(itemId):

●Removes an item from the playlist (playlist) by making a DELETE request to htp://localhost:3000/playlist/{itemId}.

●Updates the playlist state afer removing an item.

● isItemInPlaylist(itemId):

●Checks if an item exists in the playlist (playlist) based on its itemId.

* flteredItems:
* Filters items based on the searchTerm.
* Matches ttle, singer, or genre with the lowercase version of searchTerm.
* JSX:
* Renders a form with an input feld (Form, InputGroup, Buton, FaSearch) for searching items.

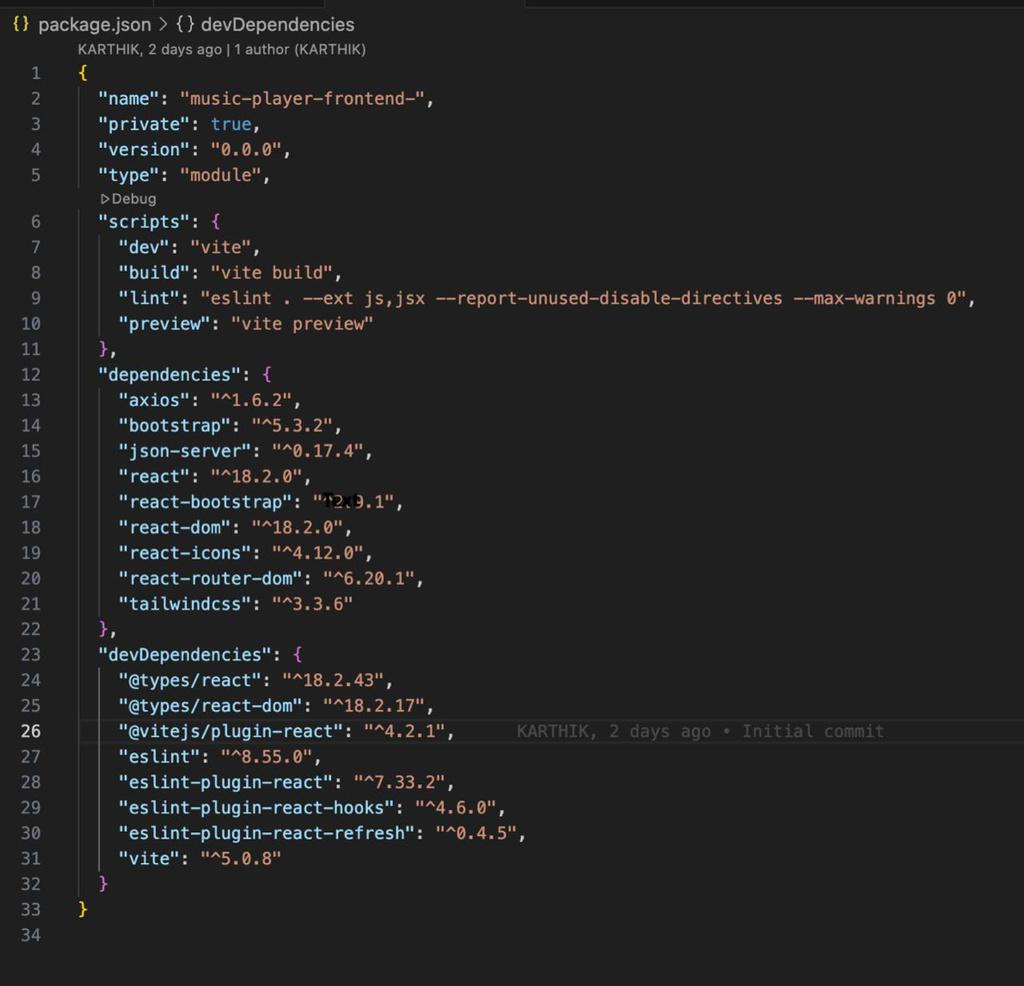
●Maps over flteredItems to render each item in the UI.

* Includes butons (FaHeart, FaRegHeart) to add/remove items from wishlist and playlist.
* Renders audio elements for each item with play/pause functonality.
* 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 Descripton:-**

* Container Setup:

●Uses a div with inline styles (style={{display:"fex", justfyContent:"fex-end"}}) to align the content to the right.

* The main container (songs-container) has a fxed 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:
* Utlizes InputGroup from React Bootstrap for the search functonality.
* Includes an input feld (Form.Control) that allows users to search by singer, genre, or song name.
* Binds the input feld 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 flteredItems array and renders each item as a Bootstrap card (<div className="card h-100">).
* Card Content:
* Displays the item's image (<img>), ttle (<h5 className="card-ttle">), 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 Butons:
* Adds a heart icon buton (<Buton>) to add or remove items from the wishlist (isItemInWishlist(item.id) determines which buton to show).
* Includes an "Add to Playlist" or "Remove From Playlist" buton (<Buton>) based on whether the item is already in the playlist (isItemInPlaylist(item.id)).
* Buton 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 specifc elements like images and audio players.

**Project Executon:**

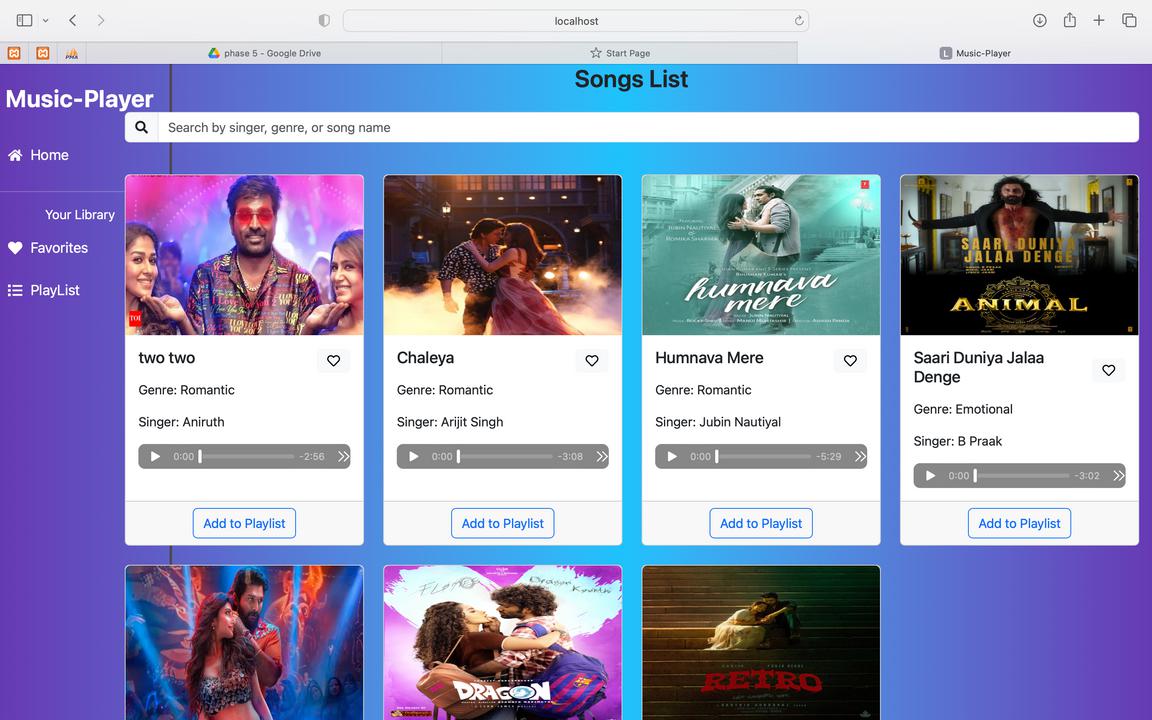
●Afer completng the code, run the react applicaton by using the command “npm start” or “npm run dev” if you are using vite.js

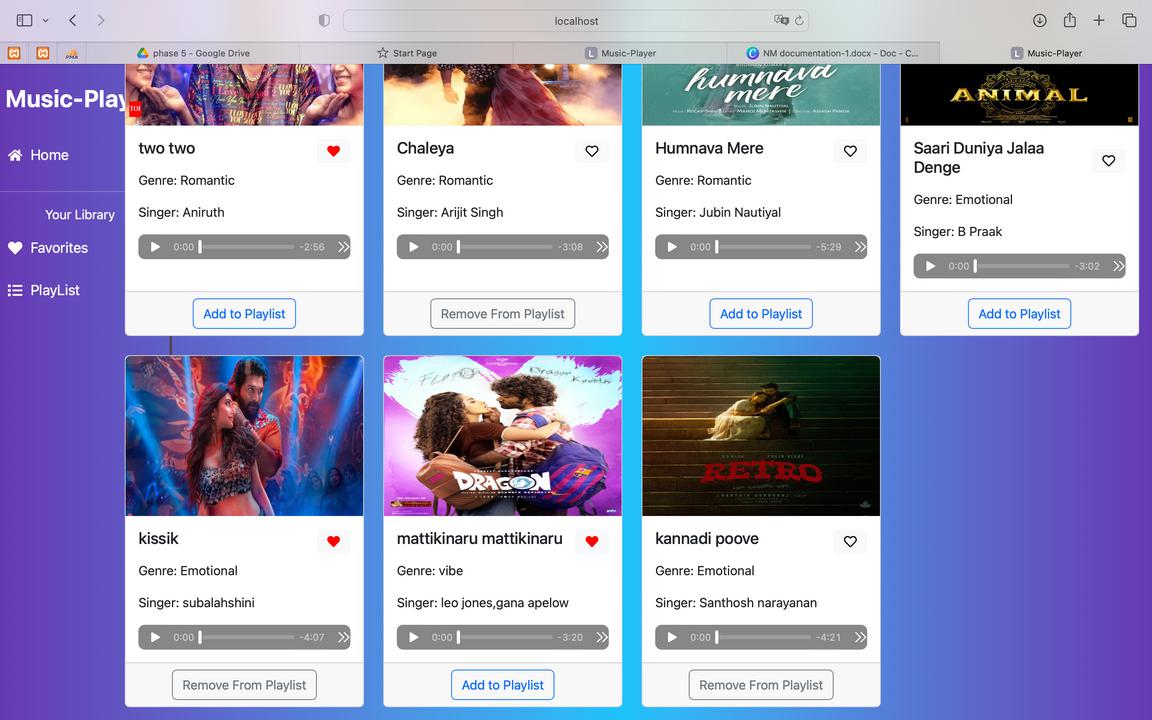
●And the Open new Terminal type this command “json-server --watch ./db/db.json” to start the json server too.

●Afer that launch the Rythimic Tunes.

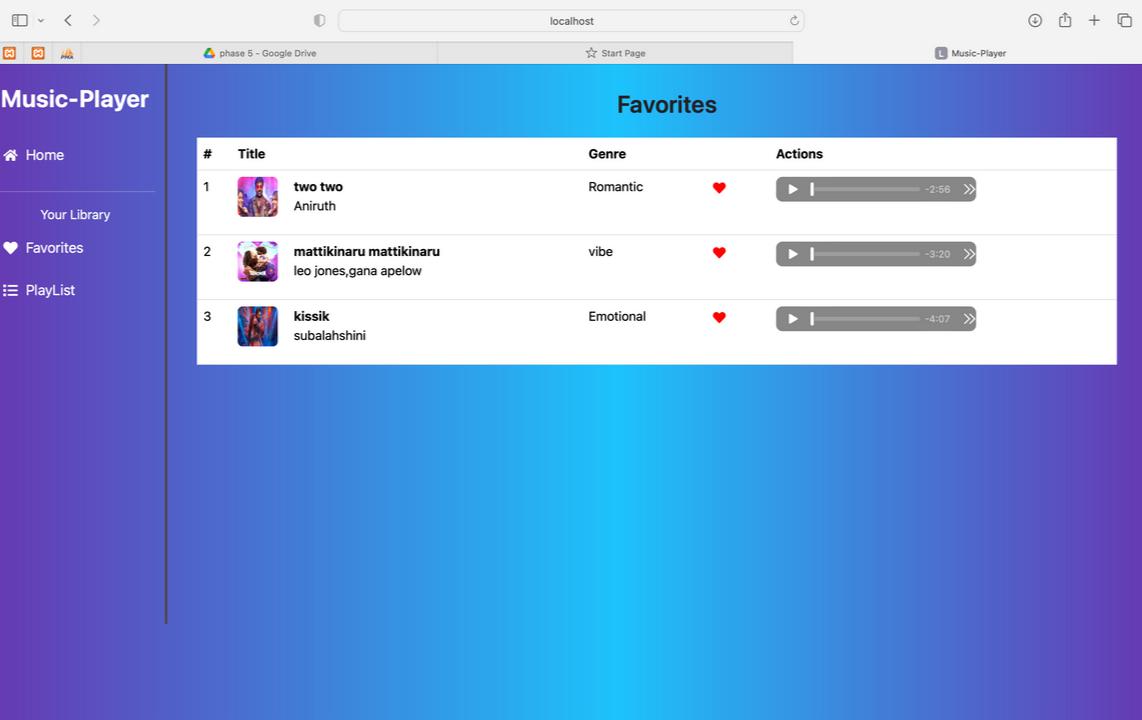
Here are some of the screenshots of the applicaton.

**Hero components :**

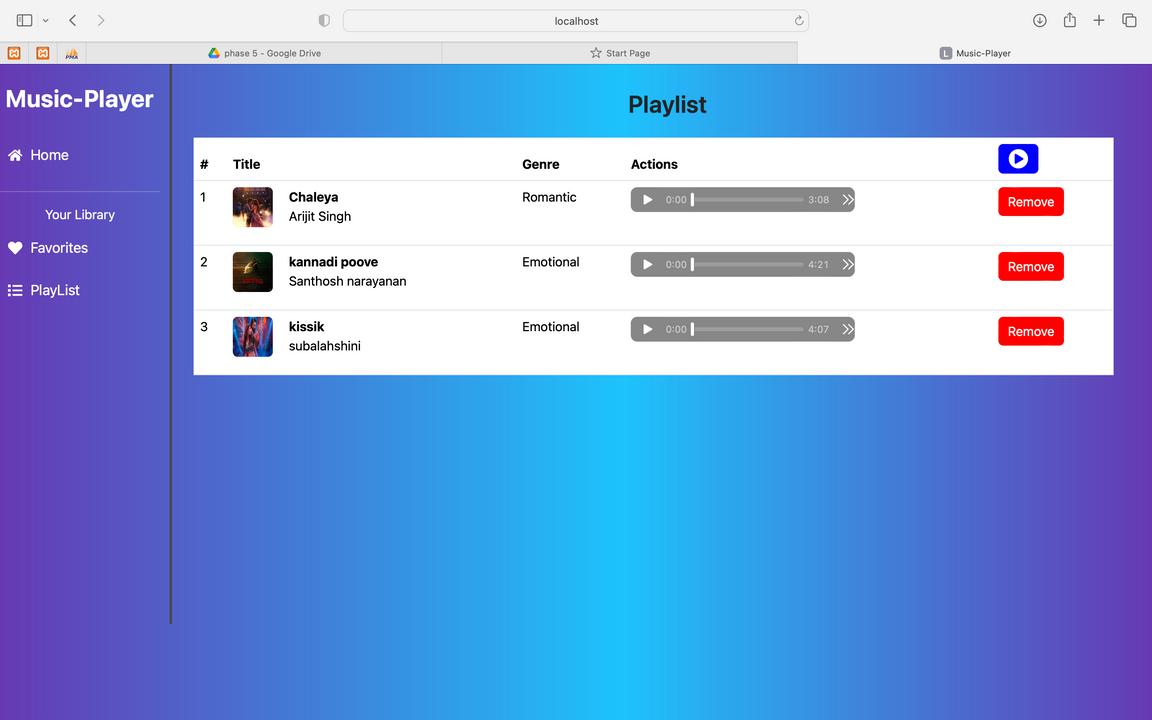




**Favourites :**



**Playlist :**



**Demo link:**[**https://drive.google.com/drive/folders/1tsWvxphZh-qwtBjAH98bMW0n13aJBeJP?**](https://drive.google.com/drive/folders/1tsWvxphZh-qwtBjAH98bMW0n13aJBeJP?usp=share_link)[**usp=share\_link**](https://drive.google.com/drive/folders/1tsWvxphZh-qwtBjAH98bMW0n13aJBeJP?usp=share_link)

**Source code drive**

**link:**[**https://drive.google.com/drive/folders/19pgXw0dlsBolRf6ExpIntpjPgcGM-47j?**](https://drive.google.com/drive/folders/19pgXw0dlsBolRf6ExpIntpjPgcGM-47j?usp=share_link)[**usp=share\_link**](https://drive.google.com/drive/folders/19pgXw0dlsBolRf6ExpIntpjPgcGM-47j?usp=share_link)