**Rhythmic Music**

**1.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.

**Project Title:**

**RhythmicTunes**: Your Melodic Companion

* **Team Members:** Team id: SWTID1741230277156128

1. **Team Leader:**S.Sandeya

**Email id :** <sandeyasana2004@gmail.com>

2.S.Rajeshwari

**Email id:** <rajisrinivasan3101@gmail.com>

**3.**R.Rathi Priya

**Email id:** <rathipriya5322@gmail.com>

**4.**V.Sandhiya

**Email id:** <sandhiyavenkatesan79@gmail.com>

**2.Project Overview**

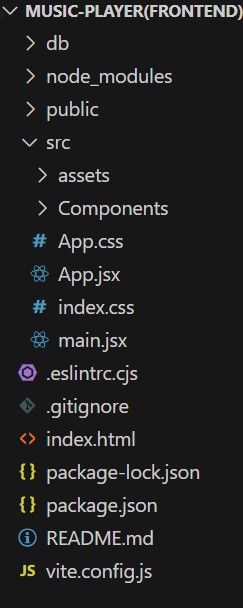
* **Purpose:**

Rhythmic Tunes is a music streaming application aimed at delivering a user-friendly platform for music discovery, playlist management, and easy access to a wide range of tracks. The goal is to provide a seamless and engaging music experience by offering personalized playlists, top charts, and music recommendations.

* **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.

**3.Architecture**

* **Component Structure:**
* **Header:** Displays the logo, navigation links, and user account information.
* **Sidebar:** Contains links to different playlists and categories.
* **Playlist:** Represents a list of tracks in a given playlist.
* **Player:** Controls the playback of songs.

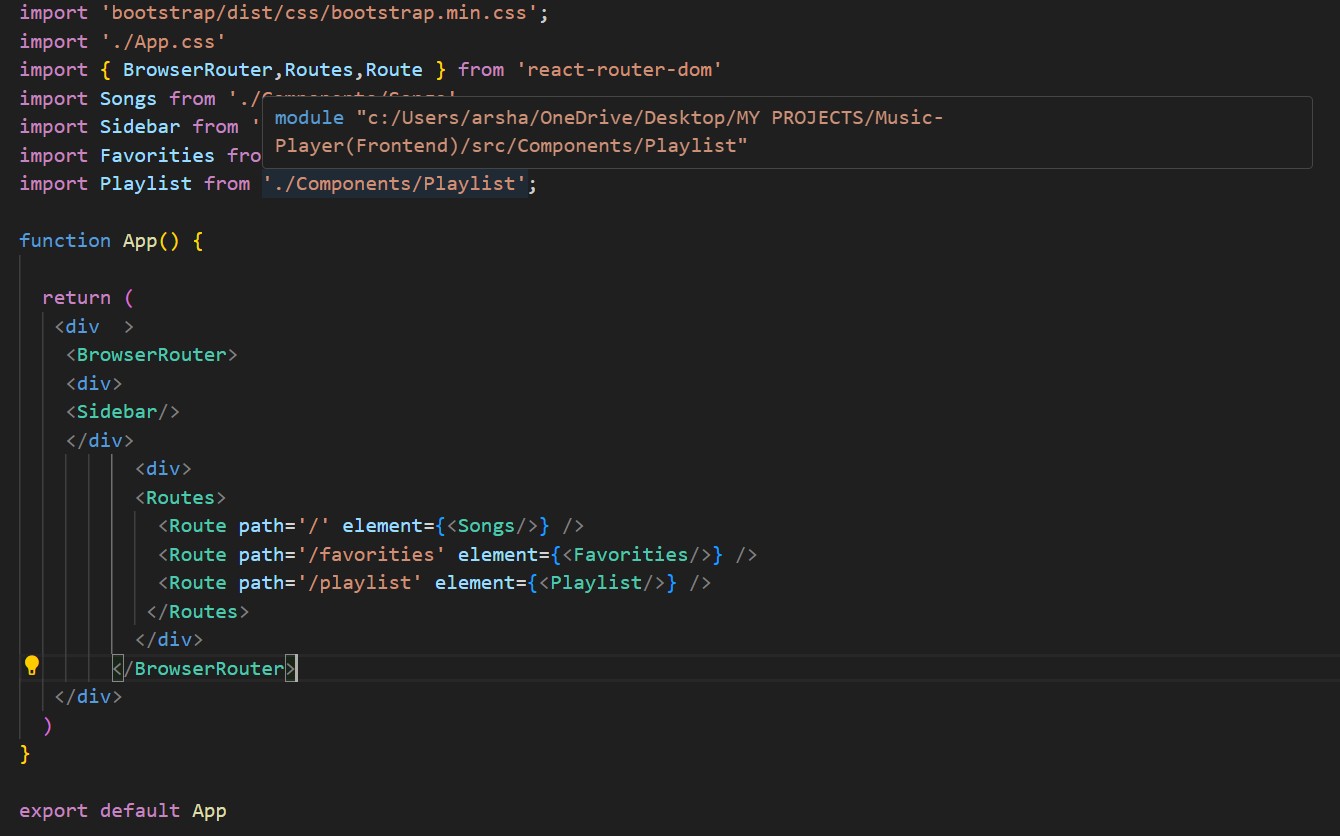


* **State Management:**

State is managed using Context API for global state like authentication status, current playlist, and active track. Local state is handled within individual components (e.g., play/pause status in the Player component).

* **Routing:**

Setting Up Routes:-



* + 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:
      * Route with path='/' renders the Songs component when the root path is accessed (/).
      * Route with path='/favorities' renders the Favorities component when the /favorities 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 application.

**4. Setup Instructions**

* **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.

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

* **Installation:**

**Installation of required tools**:

1. Open the project folder to install necessarytools In this project, we use:

* + React Js o React Router Dom o React Icons o Bootstrap/tailwind css
  + Axios

**5.Folder Structure**

**Client:**

* **/components:** Contains all the React components (e.g., Header, Sidebar, Player).
* **/pages:** Contains the main pages of the app (e.g., Home, Library, Discover).
* **/assets:** Contains static files such as images and icons.
* **/hooks:** Custom React hooks (e.g., useAuth, usePlayer).

**Utilities:**

* Helper functions like formatDuration and fetchTracks to simplify data processing.

**6.Running Application**

**Frontend:**  
 After installing dependencies, run the following command to start the React development server:

npm start

* The application should now be accessible at <http://localhost:3000>.

**7.Component Documentation**

* **Key Components:** 
  + **Header:** Displays the app name, navigation, and user account info.
    - Props: None.
  + **Player:** Controls the music player interface (play/pause, skip).
    - Props: track, isPlaying, onPlayPause.
  + **Sidebar:** Contains the list of playlists and categories.
    - Props: playlists (array of playlist objects).
* **Reusable Components:** 
  + Track: A reusable component to display track information.
    - Props: trackName, artist, albumCover.

**8.State Management**

**Global State:**

The app's state is managed using the Context API. The PlayerContext keeps track of the current track and playback state across components.

Global State Variables:

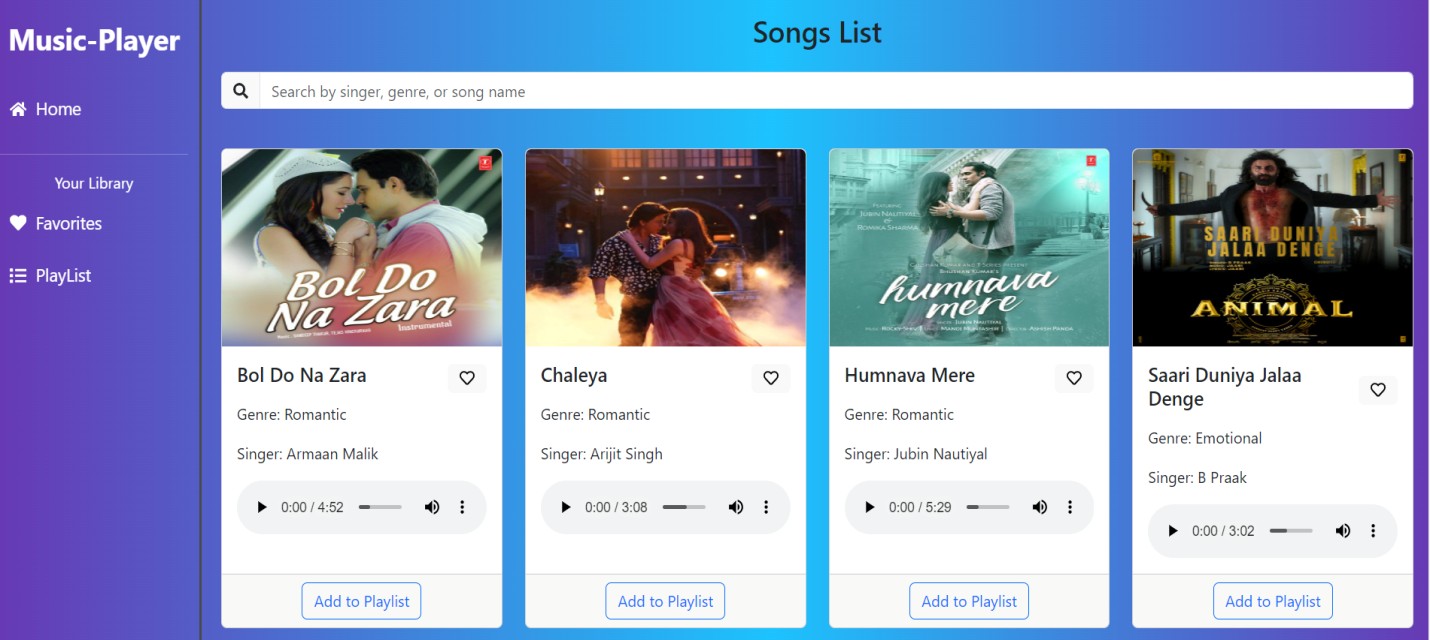
* + currentTrack: The currently playing song.
  + isPlaying: Boolean indicating whether music is playing.

**Local State:**  
 Local state is handled within components, e.g., in the Player component, where the play/pause button toggles the isPlaying state.

**9.User Interface**

**Screenshots:**

1. **Homepage:** Shows featured playlists and music recommendations.
2. **Player Controls:** Displays track name, artist, and basic controls (play, pause, skip).
3. **Library:** User's saved playlists and liked songs.



**10.Styling**

* **CSS Frameworks/Libraries:** 
  + Styled-Components: For styling React components with scoped, dynamic styles.
  + React-Bootstrap: For UI elements like buttons and modals.
* **Theming:** 
  + Custom light and dark themes are implemented using Styled-Components.

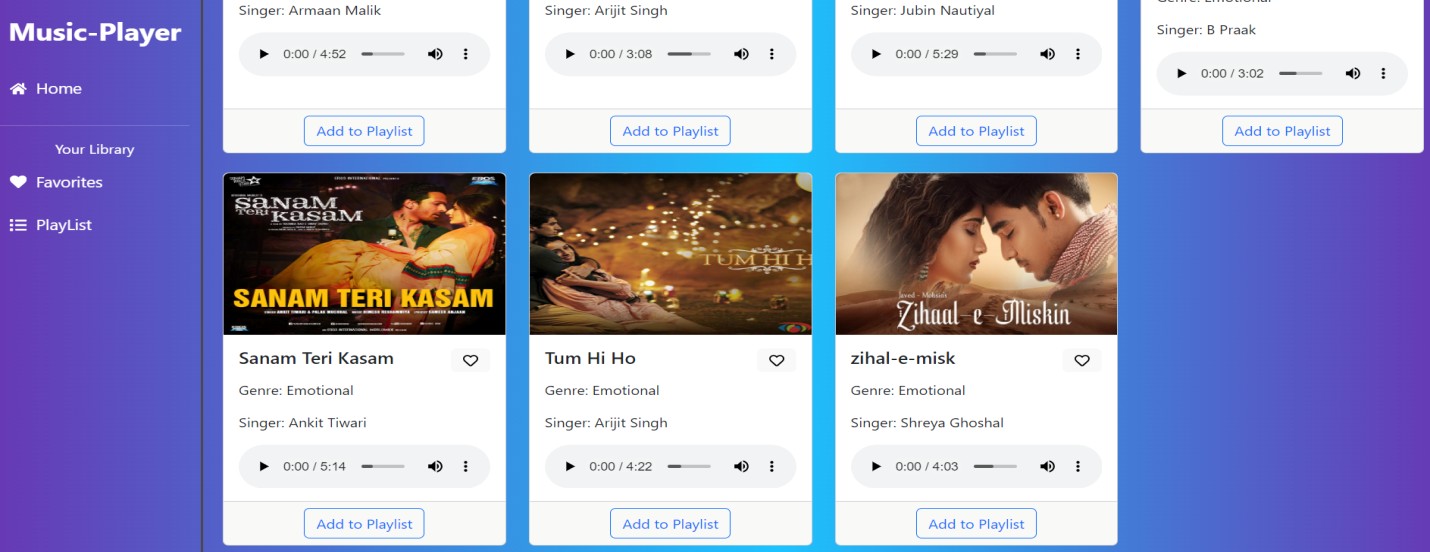
**11.Testing**

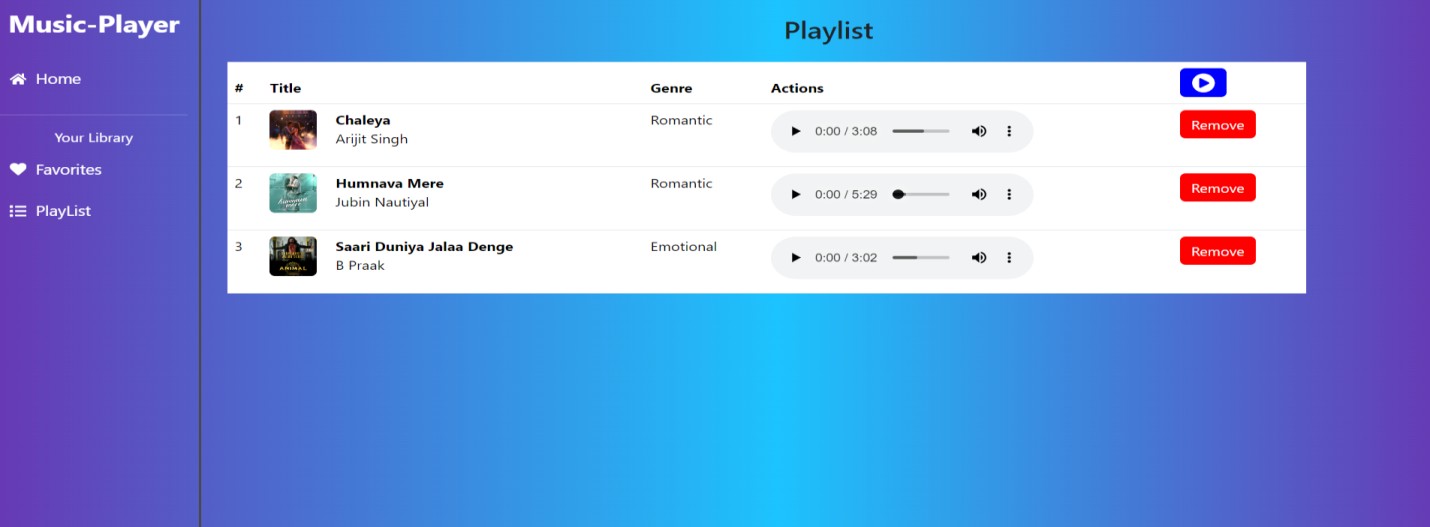
**Testing Strategy:**

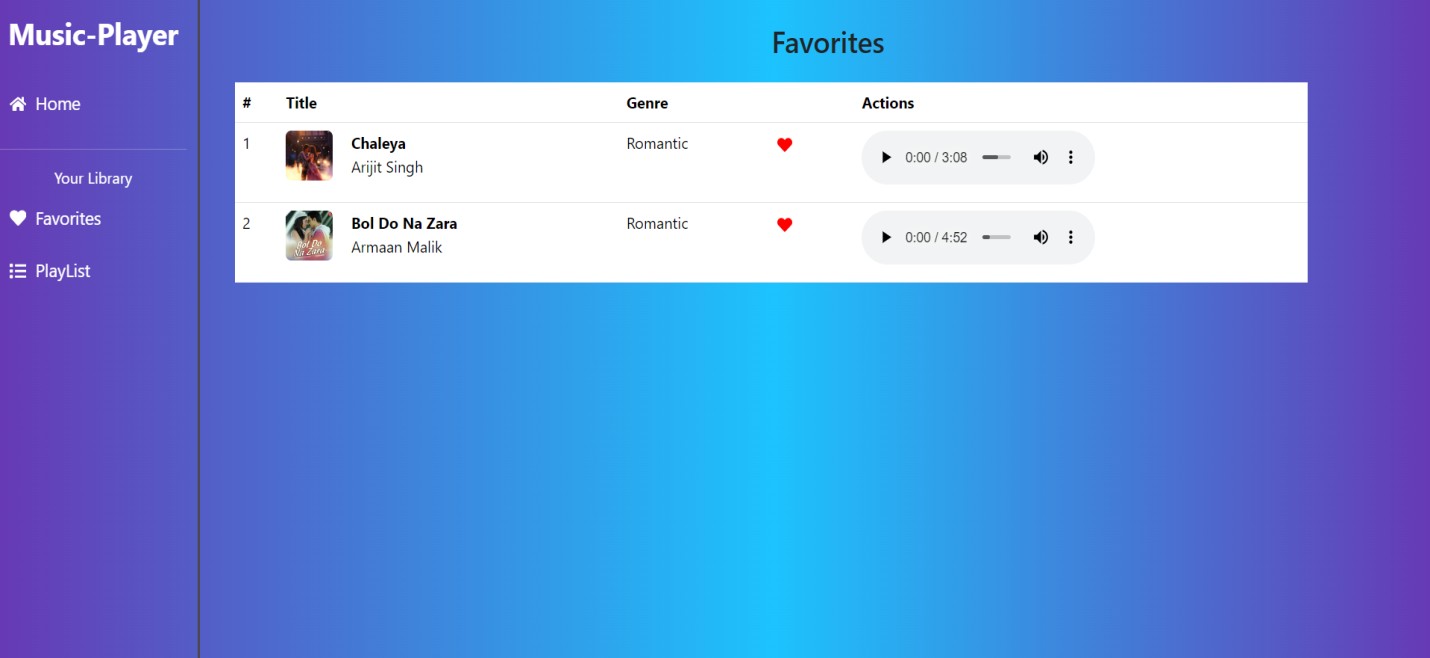
* Unit Tests: Used for individual components, especially for button clicks and track loading.
* Integration Tests: Ensure that components work together (e.g., Player and Sidebar).
* End-to-End Testing: Using Cypress to simulate user interactions and test the entire flow (e.g., logging in and playing a track).

**Code Coverage:**  
The project uses Jest with React Testing Library for unit tests, ensuring high code coverage (above 90%).

**12.Screenshots or Demo**

**Home**

**Playlist** 

**Favorites**

**Project demo:**

let’s see the Demo Link and Code Link.

**Demo Video Link:**

[**https://drive.google.com/file/d/1cQV8PxpFeVKz4ryPxOBKArb33bUoOAn2/view?usp=drivesdk**](https://drive.google.com/file/d/1cQV8PxpFeVKz4ryPxOBKArb33bUoOAn2/view?usp=drivesdk)

**Use the code in:**

[**https://drive.google.com/folderview?id=1MWNufgE9URG7l\_sWDglirs60DFn0Sk6P**](https://drive.google.com/folderview?id=1MWNufgE9URG7l_sWDglirs60DFn0Sk6P)

**13. Known Issues**

* **Known Bugs:** 
  + Playlist Sorting: Some playlists may not be sorted properly based on the date added.
  + Responsive Issues: The mobile view may have alignment issues with the player controls on smaller devices.

**14. Future Enhancements**

* **Future Features**:
  + **Offline Playback**: Allow users to download and listen to music offline.
  + **Social Sharing**: Enable users to share their favorite tracks and playlists on social media.
  + **Lyrics Integration**: Display lyrics in sync with the track.
  + **Enhanced Playlist Management**: Allow users to create and modify custom playlists with more flexibility.

**Thank You**