# **RHYTHMIC TUNES**

**INTRODUCTION:**

Team Members:

Nivethitha G

Priyadharshini B

Prathisha R

Udanya P

**PROJECT OVERVIEW**

**PURPOSE:**

The Music Streaming App is a digital platform designed to provide users with instant access to a vast library of music from various artists, genres, and languages. It offers seamless streaming, personalized recommendations, playlist creation, offline listening, and social sharing features. The app aims to enhance the music experience through high-quality audio, AI-driven curation, and a user-friendly interface. With a freemium model that includes ads and premium subscriptions, it supports both user convenience and artist promotion while ensuring sustainable revenue generation.

.

**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 user**s** to easily find specific songs, artists, or albums within the app.

**ARCHITECTURE**

**COMPONENTS STRUCTURE:**

**1. App.tsx –** Root component managing navigation and global state

**2. Navigation –** Handles routing between Home, Search, Library, and Profile.

**3. Header –** Displays app logo, search bar, and user profile.

**4. Sidebar –** Contains navigation links and playlist shortcuts.

**5. AudioPlayer –** Manages music playback (play, pause, seek, volume).

**6. Home –** Displays featured playlists, top charts, and recommendations

**7. Search –** Provides song, album, and artist search functionality.

**8. Playlist –** Shows playlist details and track listing.

**9. Profile –** Manages user details and settings.

**STATE MANAGEMENT:**

State management in a music streaming app ensures smooth data flow and synchronization across components. It handles global states like user authentication, playback status, and playlists, as well as local UI states like theme settings and loading indicators. Popular libraries for state management include Redux Toolkit for scalability, Context API for lightweight sharing, and MobX or Zustand for reactive state handling. Efficient state management improves performance, enhances user experience, and ensures real-time updates, such as seamless song transitions and personalized recommendations.

**ROUTING:**

In a React-based music streaming app without authentication or subscription, React Router handles seamless navigation across key sections. The main routes include Home (/), where users discover featured playlists and recommendations, Search (/search), allowing users to find songs, albums, and artists, and Library (/library), where saved songs and playlists are displayed. Dynamic routes like Playlist (/playlist/:id), Album (/album/:id), Artist (/artist/:id), and Song Details (/song/:id) enable detailed content viewing. Additional routes like Settings (/settings) allow users to customize their experience. Using <BrowserRouter>, <Routes>, and <Route>, React Router ensures smooth navigation and an intuitive user experience.

**PREREQUISITES:**

NODE.JS INSTALLED(VERSION 19+RECOMMENDED)

**INSTALLATION:**

Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository

<https://git-scm.com/downloads>

2.Navigate to the project :

Cd code

3.install dependencies

npm install

4.start the application

npm start

**Folder Structure**

**Client:**

**●** src/components - Contains React components

● src/pages - Different pages of the application

● src/assets - Images, icons, and styles

**Utilities:**

● src/utils/api.js - Handles API calls

● src/utils/helpers.js - Contains helper functions

**Running the Application**

To start the application, run:

npm start

**Component Documentation**

**Key Components:**

● RecipeCard.js - Displays a summary of a recipe

● RecipeDetails.js - Shows detailed recipe information

● IngredientList.js - Manages shopping lists

**Reusable Components:**

● Button.js - Customizable button component

● Modal.js - Reusable modal for alerts and pop-ups

**State Management**

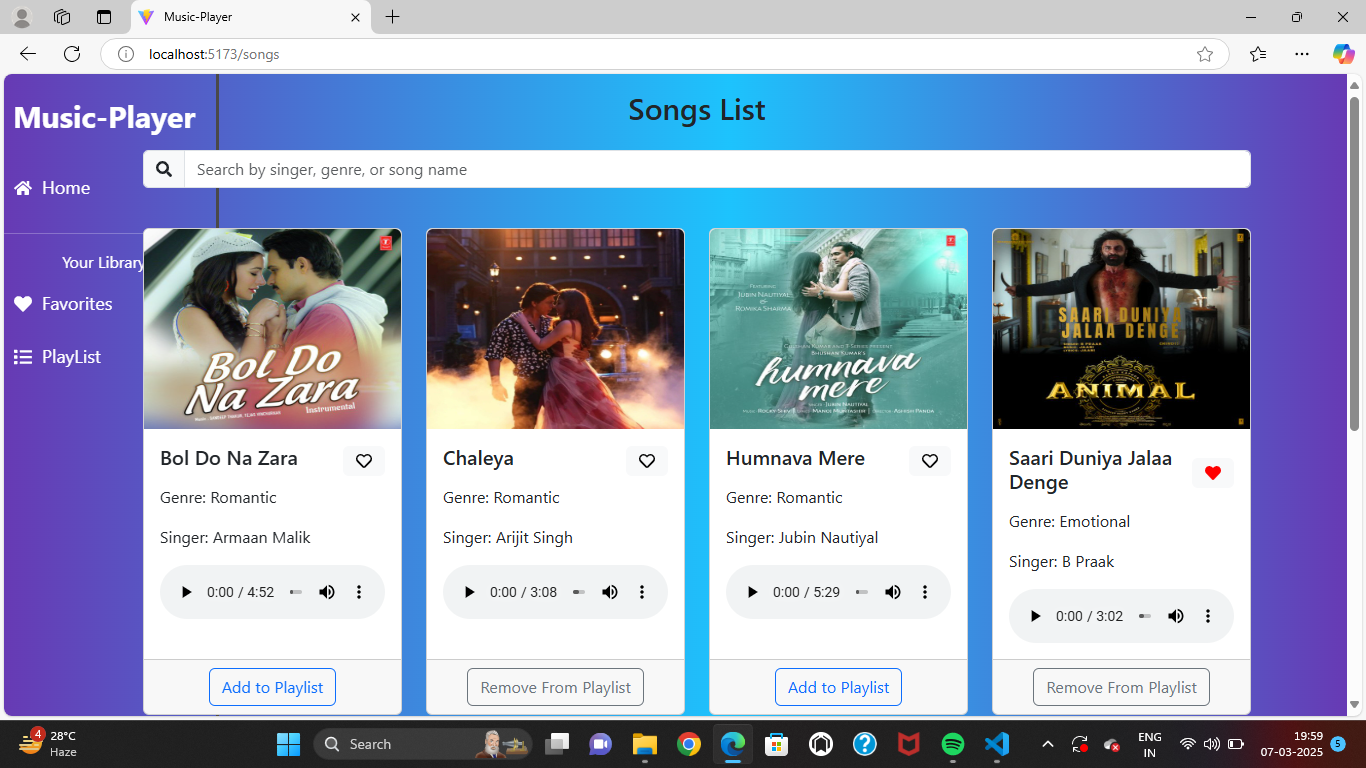
**Global State:**

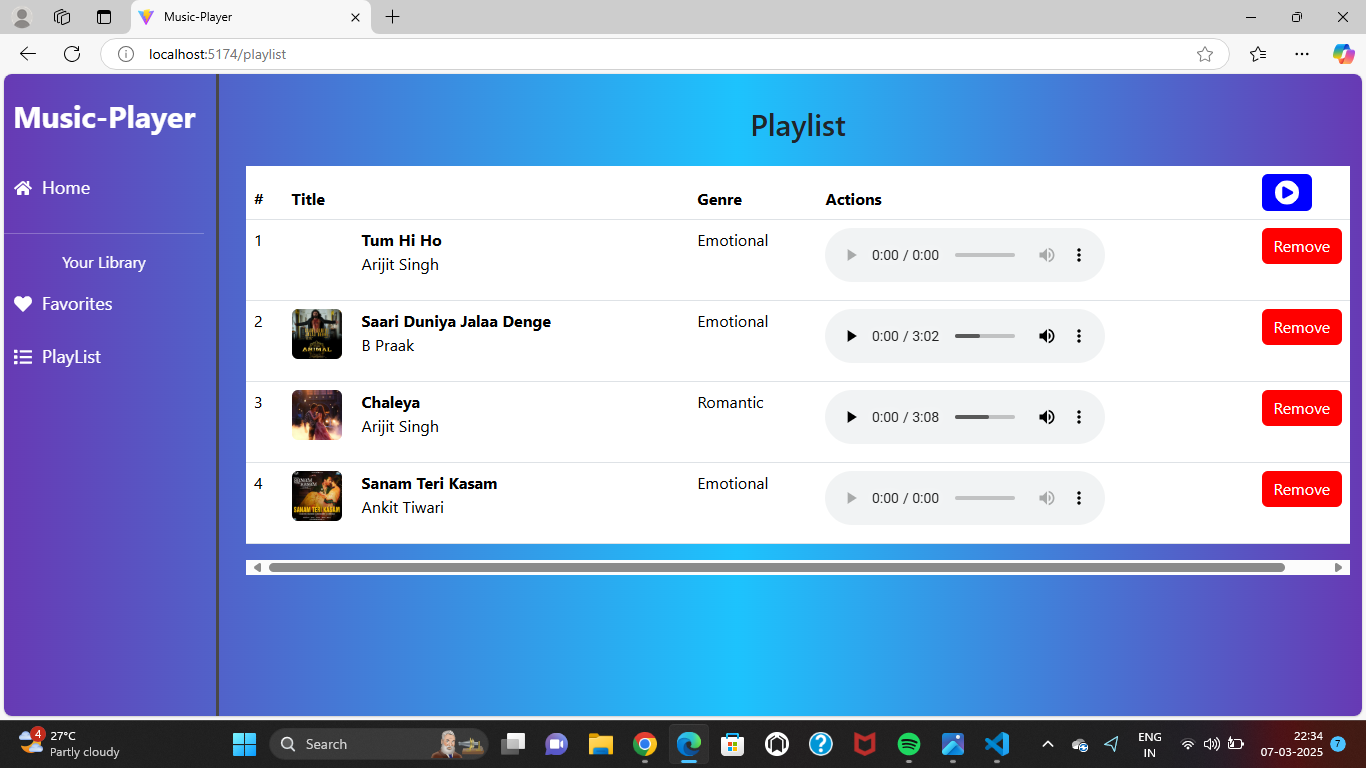
Global state (managed using Redux, Context API, or Zustand) stores app-wide data like current playback status, playlists, and user preferences, ensuring consistency across components

**Local State:**

Global state (managed using Redux, Context API, or Zustand) stores app-wide data like current playback status, playlists, and user preferences, ensuring consistency across components

**USER INTERFACE**

****

****

**Styling**

**CSS Frameworks/Libraries:**

● Using Styled-Components for modular styling

● Material UI for pre-designed components

**Theming:**

● Light and Dark mode support

● Custom color palette for branding

**Testing**

**Testing Strategy:**

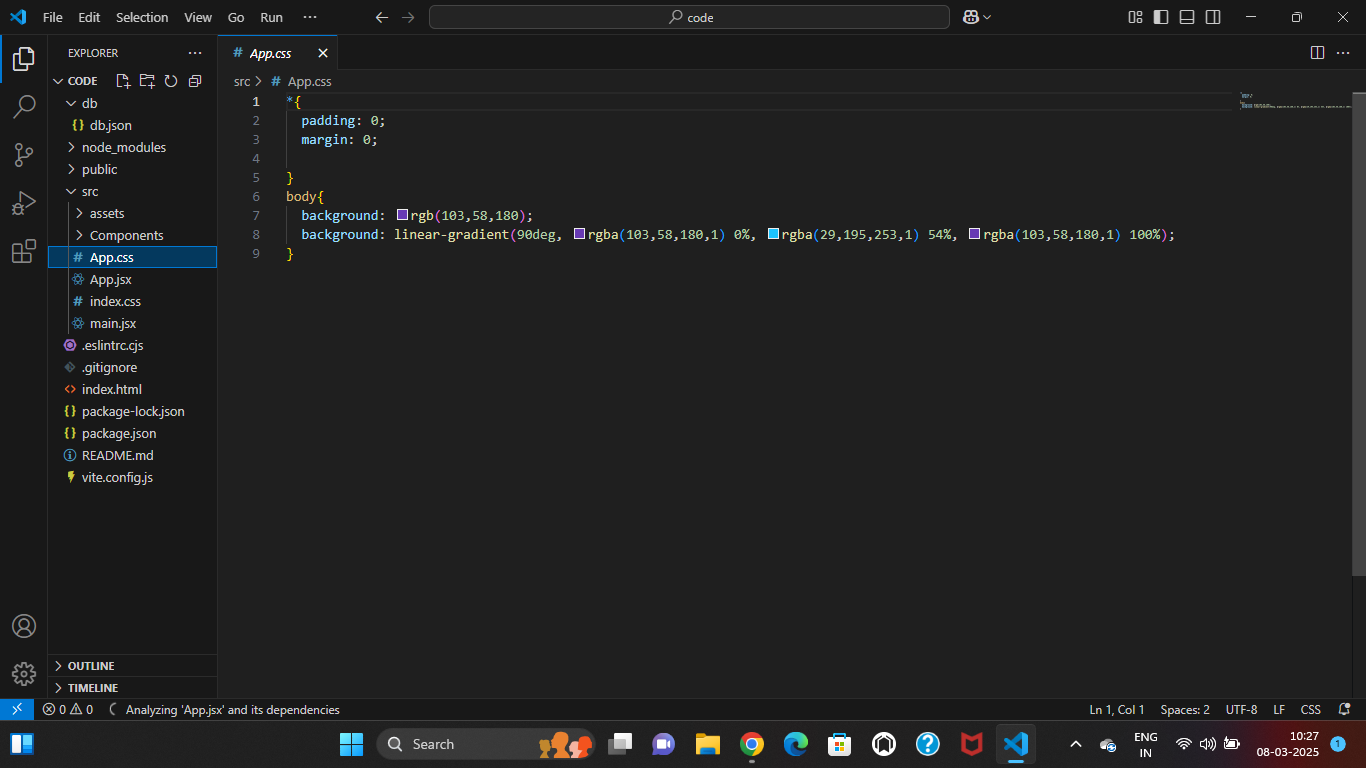
● Unit Testing using Jest

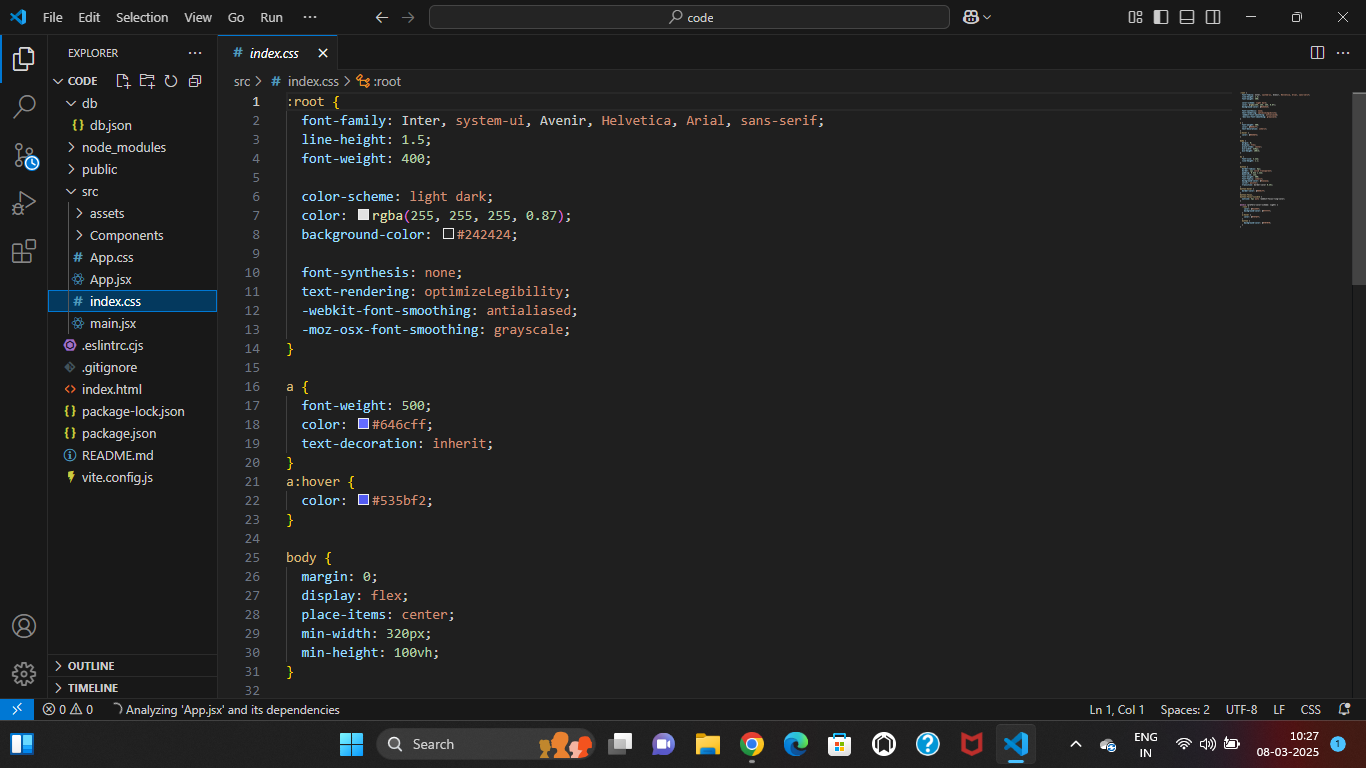
● Component Testing with React Testing Library

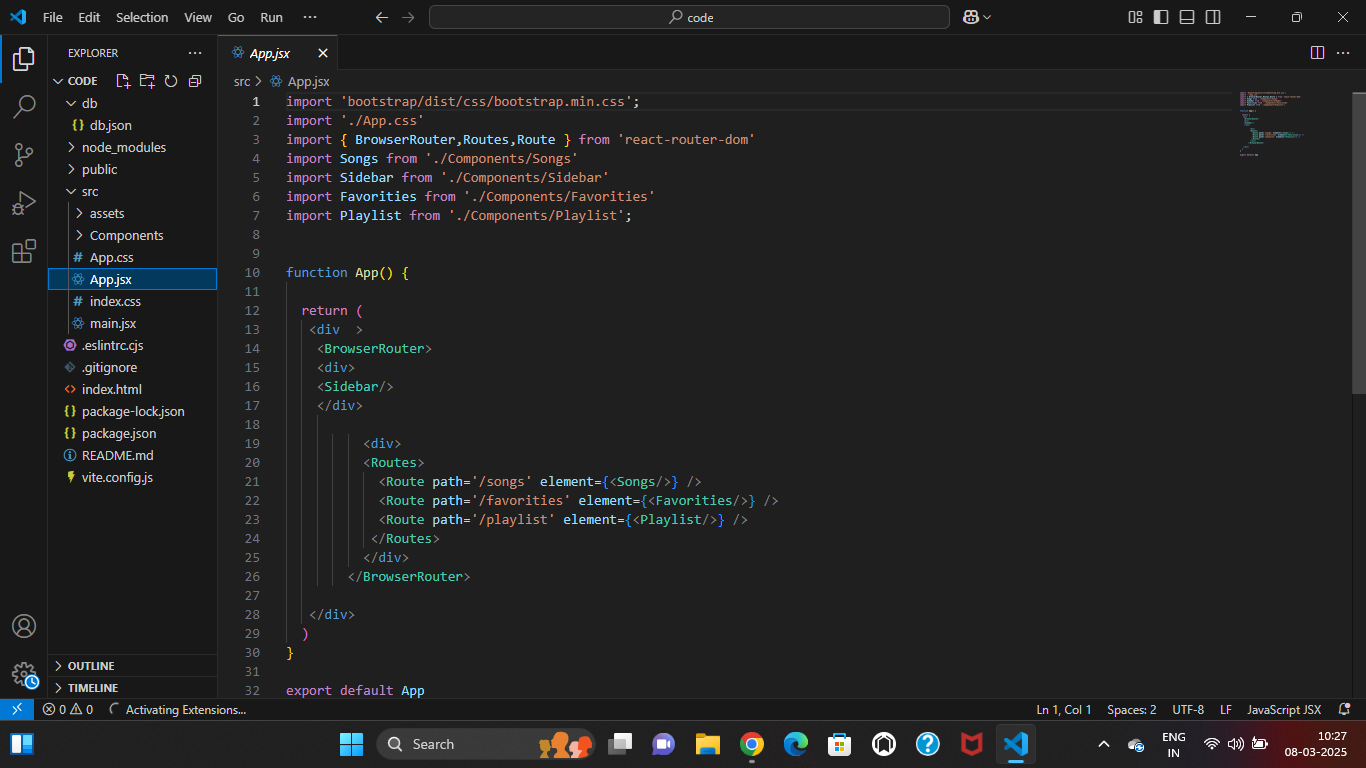
**Code Coverage:**

* Coverage reports generated using Test

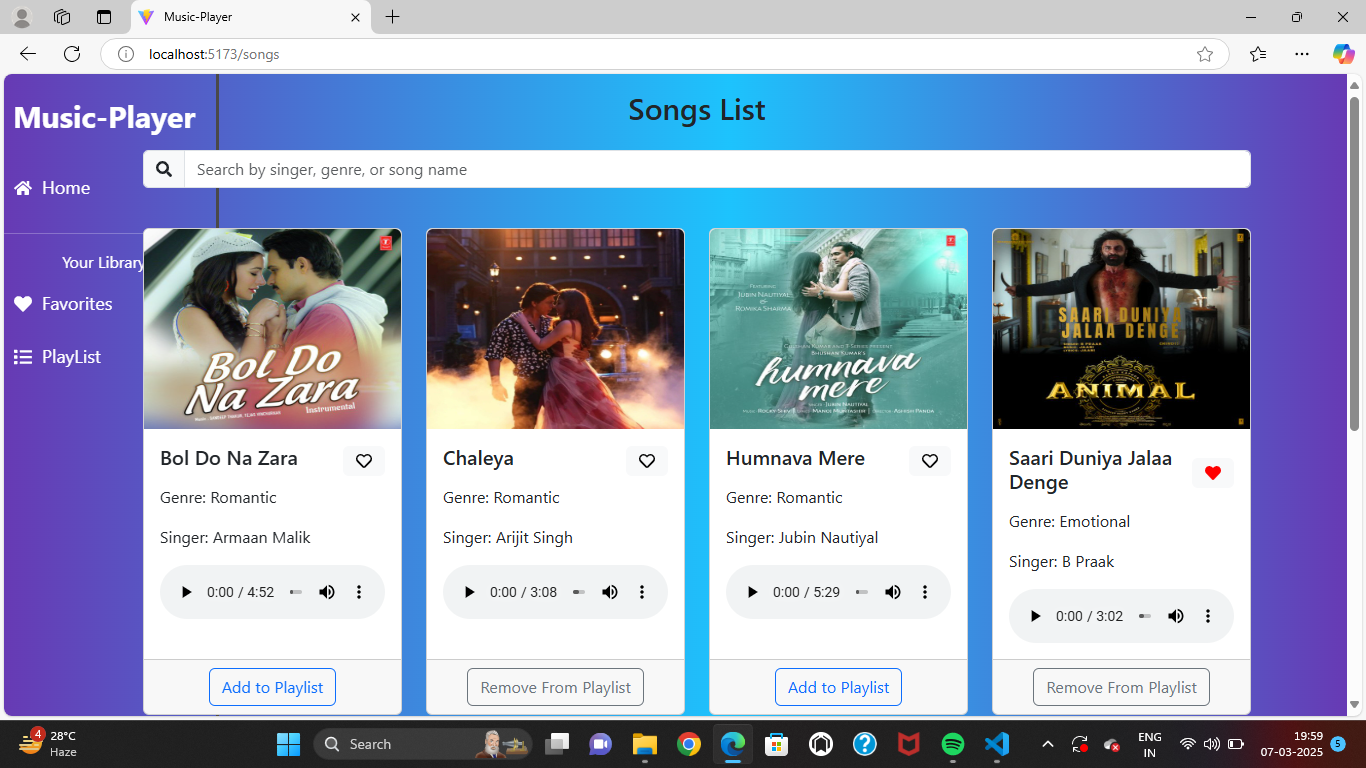
**SCREENSHOTS**

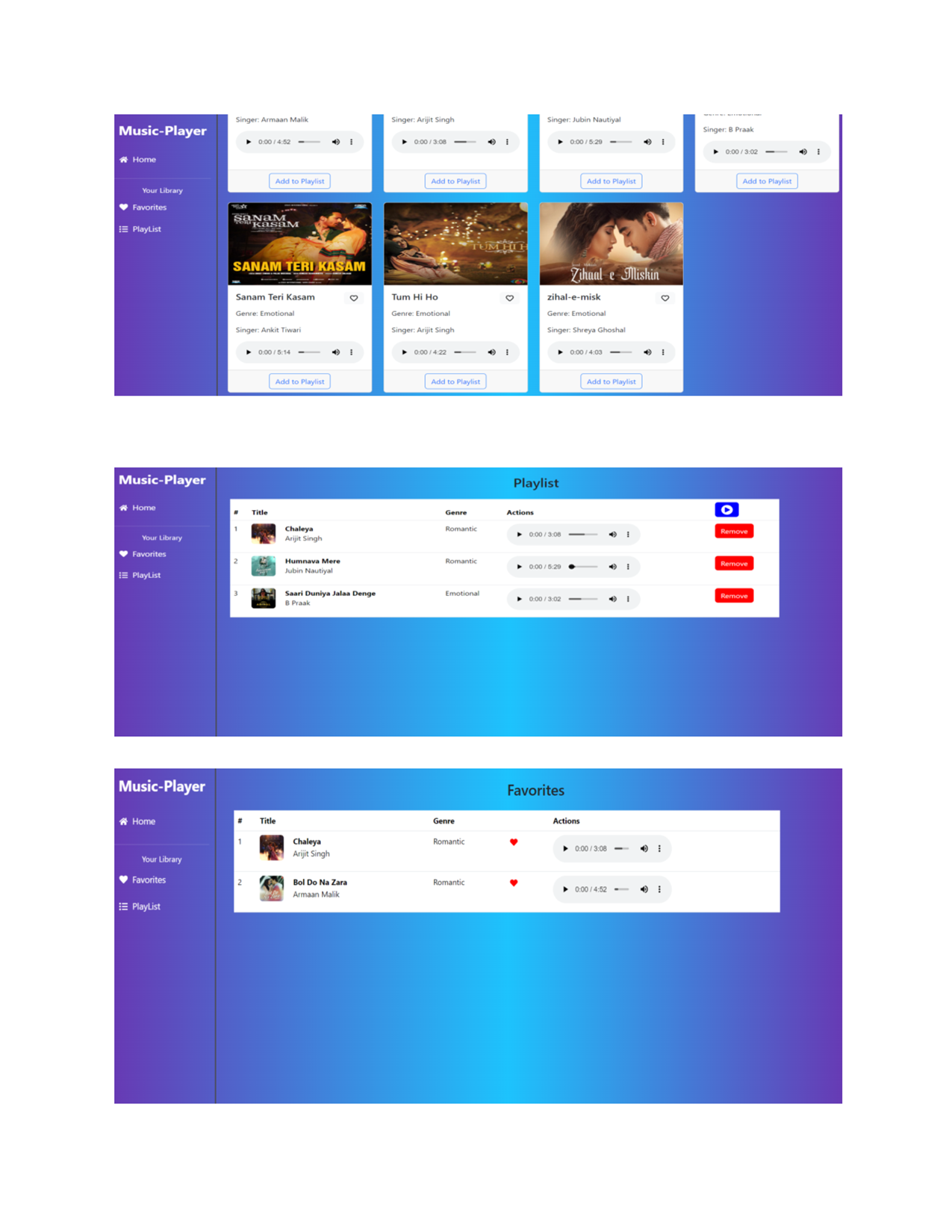






EXECUTED OUTPUT:





**Future Enhancements**

Future enhancements for the music streaming app could include AI-powered recommendations for personalized playlists, offline mode for downloading songs, and social features like playlist sharing and user interactions. Adding real-time lyrics synchronization, multi-device sync, and custom themes would enhance user experience. Expanding content with podcasts and radio integration, along with gesture and voice controls, would make navigation more intuitive and engaging. These improvements would ensure a more dynamic and interactive app.