RHYTHMIC TUNES

–YOUR MELODIC COMPANION

1. Introduction

* Project Title: Rhythmic Tunes: Your Melodic Companion
* Team ID: NM2025TMID47765
* Team leader:

NAME: Deepalakshmi.R

MAIL ID: [sivanpeace2007@gmail.com](mailto:sivanpeace2007@gmail.com)

* Team Members:

NAME: Ashmita.M

MAIL ID: [ashmitaashmita227@gmail.com](mailto:ashmitaashmita227@gmail.com)

NAME: Bhuwaneshwari.M

MAIL ID: [bhuvanamadhu06@gmail.com](mailto:bhuvanamadhu06@gmail.com)

NAME: Charupriya.R

MAIL ID: [charupriya559@gmail.com](mailto:charupriya559@gmail.com)

1. Project overview

* Purpose: The purpose of the Rhythmic-Tunes Music Player is to provide a user-friendly application for playing music which makes our mind relax.
* Features: Key features include:

**music playback,**

**Playlist creation,**

**search functionality**

1. Architecture
   * **Component Structure**: Key features includes:

Music playback

playlist creation

search functionality.

* + **Now Playing**: Displays metadata like song title, artist, album artwork, and duration while a track is playing.
  + **Playlist View**: Displays songs grouped under a particular playlist, allowing users to edit or reorder the tracks.
  + **State Management**: The application uses the Context API for global state management to handle the current song playback status and user-created playlists.
  + **Search Bar Component**: Provides real-time filtering of available songs as the user types a query.
  + **Routing**: React Router is used to manage navigation between different views, such as the home page and the playlist page.
  + **HOME**:
* Displays featured playlists, recommended songs, and recent activity.
* Provides quick access to playback and playlists.
  + SEARCH:
* Dedicated search view where users can search and explore the full music library.

# PLAYLIST:

# Shows songs in a specific playlist

# Setup Instructions

* + **Pre requisites**: The required software dependencies are Node.js and npm

, React.js.

* + **Installation**: Provide a step-by-step guide to Clone the repository from GitHub. Navigate to the client directory.
* Run npm install to install dependencies.
* Configure environment variables for any API keys

# Folder structure

* 1. **Client**: The main React application is organized into folders such as components, pages, and assets for images and audio files.
  2. **Utilities** : Helper functions for music play back and custom hooks for managing audio state are located in the folder

# Code:

# Project-name/

# code/

# ├── db/

# │ └── db.json

# ├── node\_modules/

# ├── public/

# │ ├── Songs/

# │ └── vite.svg

# ├── src/

# │ ├── assets/

# │ ├── Components/

# │ ├── App.css

# │ ├── App.jsx

# │ ├── index.css

# │ ├── main.jsx

# ├── .eslintrc.cjs

# ├── .gitignore

# ├── db.json

# ├── index.html

# ├── package-lock.json

# ├── package.json

# ├── README.md

# ├── vite.config.js

# SAMPLE CODE: (App.jsx)

# import 'bootstrap/dist/css/bootstrap.min.css';

# import './App.css'

# import { BrowserRouter,Routes,Route } from 'react-router-dom'

# import Songs from './Components/Songs'

# import Sidebar from './Components/Sidebar'

# import Favorities from './Components/Favorities'

# import Playlist from './Components/Playlist';

# import './App.css';

# function App() {

# 

# return (

# <div >

# <BrowserRouter>

# <div>

# <Sidebar/>

# </div>

# 

# <div>

# <Routes>

# <Route path='/songs' element={<Songs/>} />

# <Route path='/favorities' element={<Favorities/>} />

# <Route path='/playlist' element={<Playlist/>} />

# </Routes>

# </div>

# </BrowserRouter>

# 

# </div>

# )

# }

# export default App

# SAMPLE CODE:

# <!doctype html>

# <html lang="en">

# <head>

# <meta charset="UTF-8" />

# <link rel="icon" type="image/svg+xml" href="/vite.svg" />

# <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

# <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.16/dist/tailwind.min.css" rel="stylesheet">

# <title>Music-Player</title>

# </head>

# <body>

# <div id="root"></div>

# <script type="module" src="/src/main.jsx"></script>

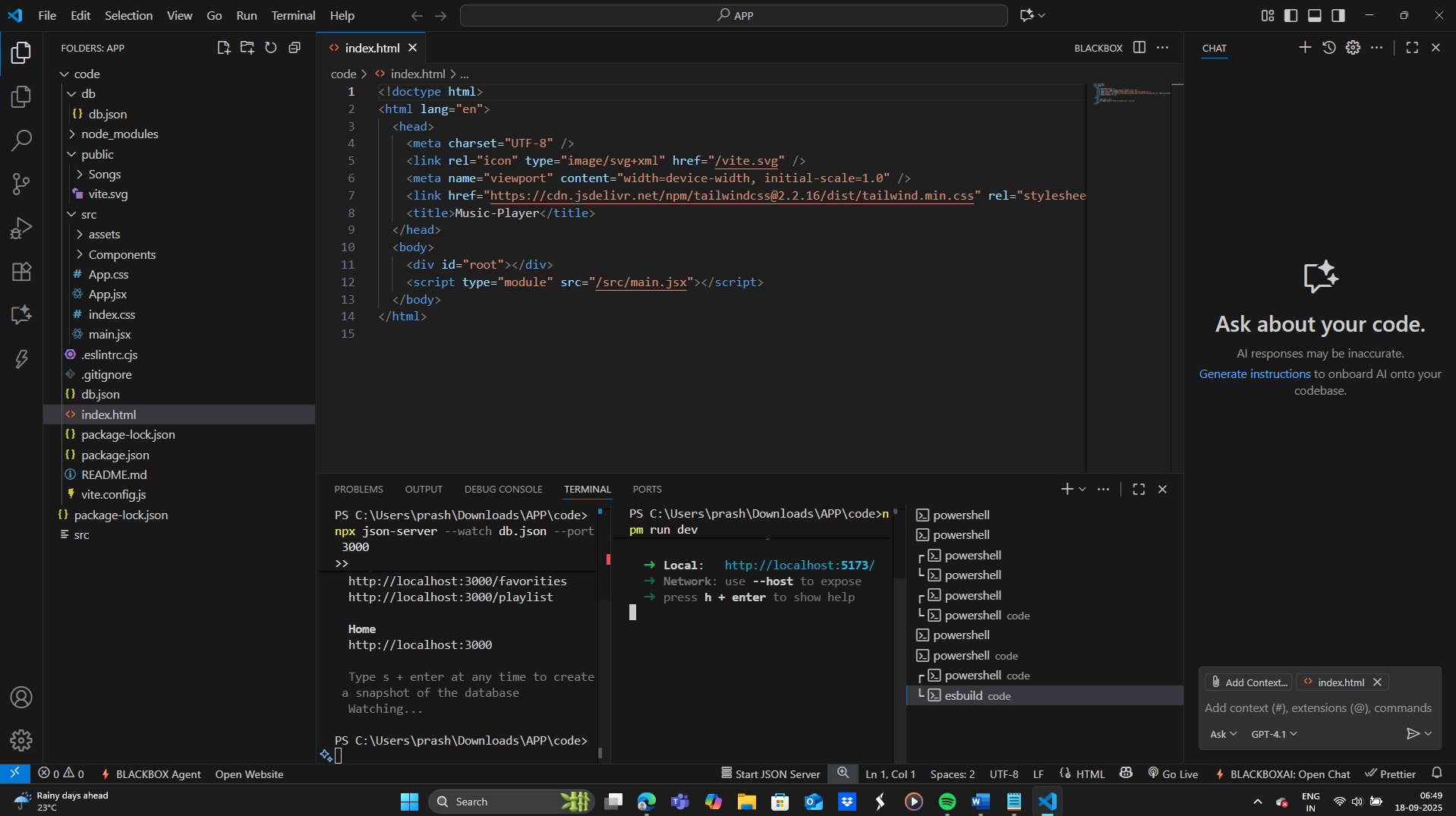
# </body>

# </html>

# Running the application

* cd -code
* npx json-server --watch db.json --port 3000
* open second powershell in visual studio
* then run the command in second powershell : npm run dev

you can get an clear idea about the below image this image will make clear of you that we need to open double powershell 1st powershell is used open a cd command with server and in second power shell we can use 2nd command to run the app the link will be displayed you can use ctrl key by pressing it continuously and move your cursor to the link then click on it and it will get opened.



# 7.Component Documentation

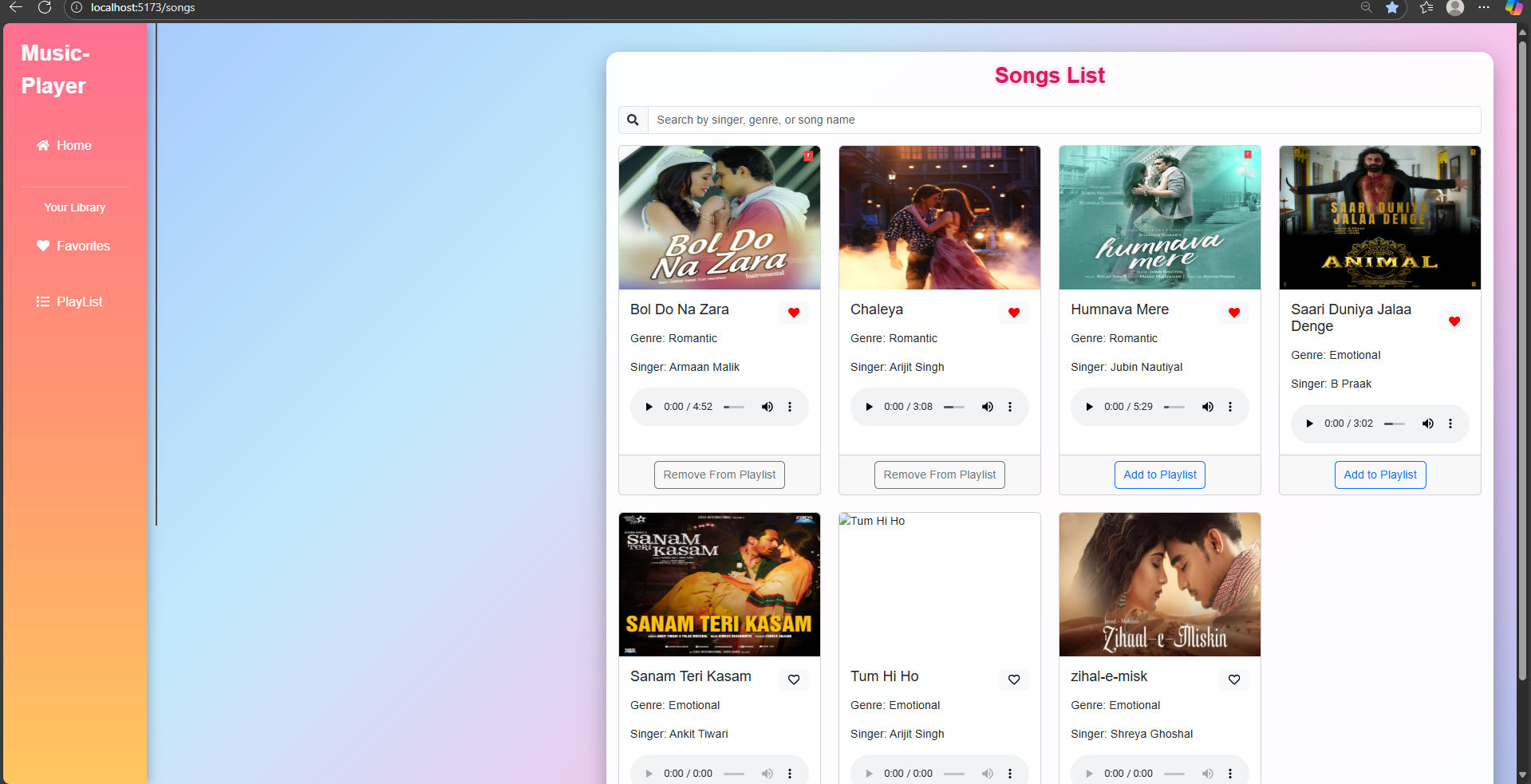
* + **Components**: Manages music play back controls. It receives a song prop.
  + Playlist Component: Displays the list of songs and manages playlist creation.
  + Search Component: Handles searching for songs
  + **Reusable Components**: Button Component: A configurable button component used throughout the application.
  + \*Song Card Component: A component for displaying individual songs in lists.

# 8.State Management

* + **Global State**: The Context API is used to manage the global state, ensuring that the current song and playback status are accessible to all components.
  + **Local State**: use State hooks to manage their own local state, such has form inputs or UI toggles.

# User Interface

* 1. Provide screen shots or GIFs show casing different UI features, such as pages, forms, or interactions.



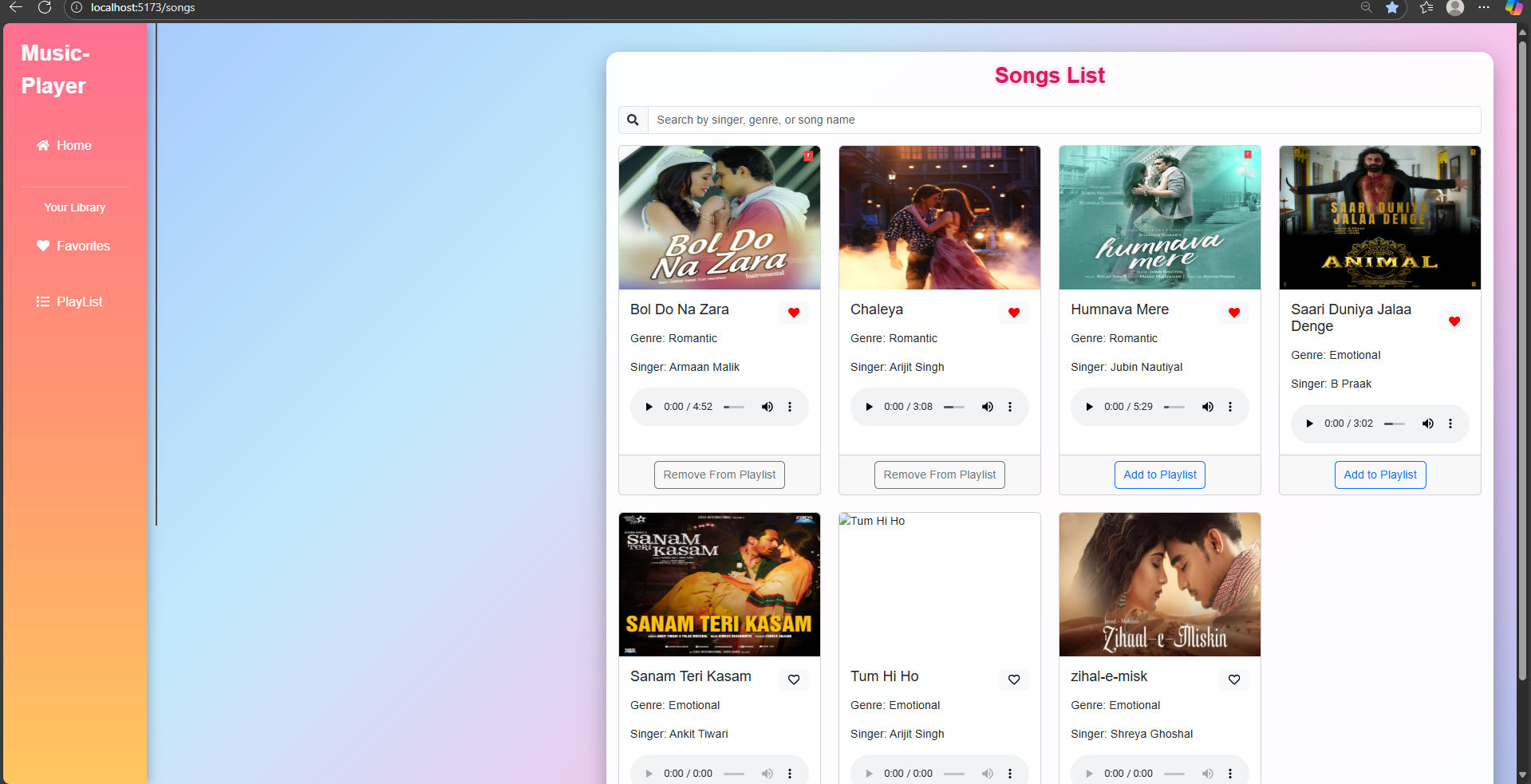
# Styling

* **CSS Frame works/Libraries**: Describe any CSS frame works, libraries, or pre- processors (e.g., Sass, Styled-Components) used.
* **Theming**: A custom design system with light and dark themes is implemented

# Testing

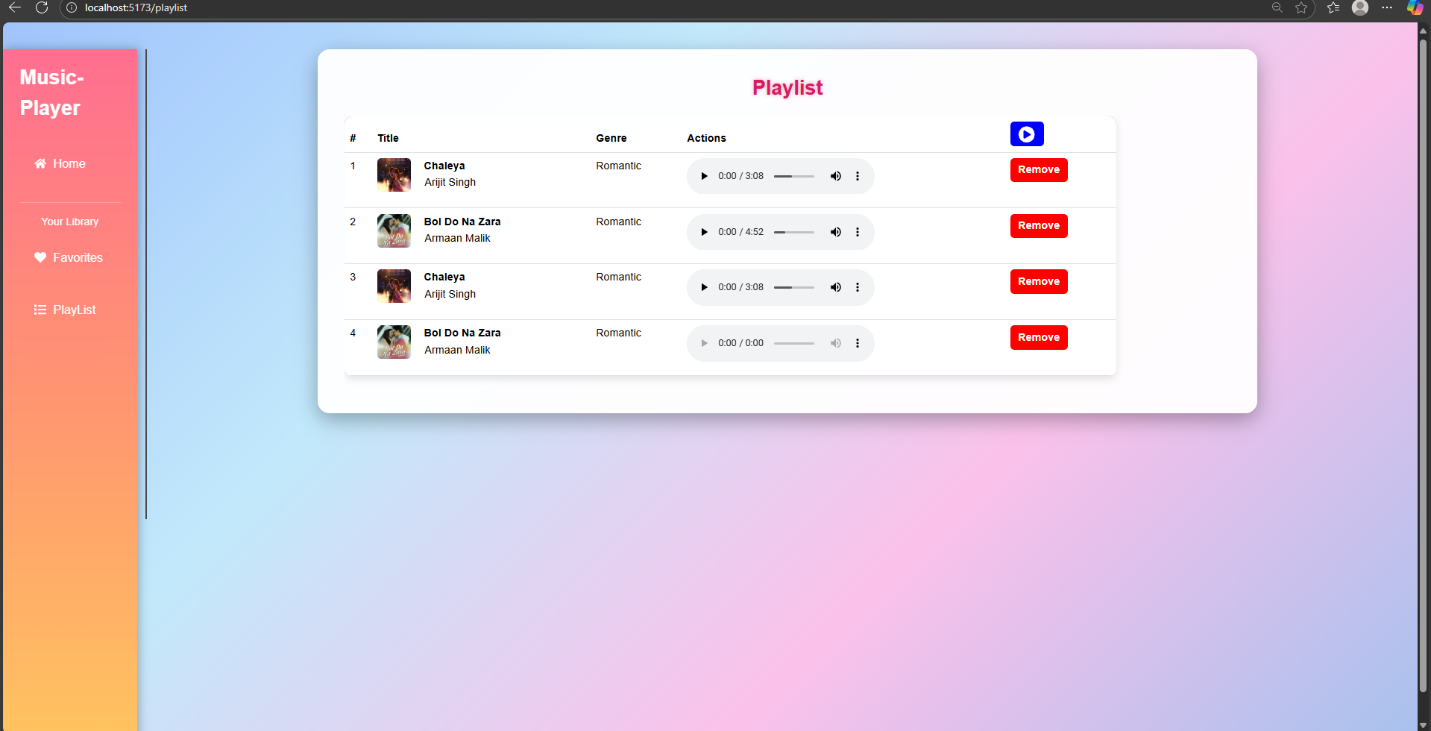
* **Testing Strategy**: The application uses Jest and React Testing Library for unit and integration testing of components.

1. **Screen shots or Demo:**

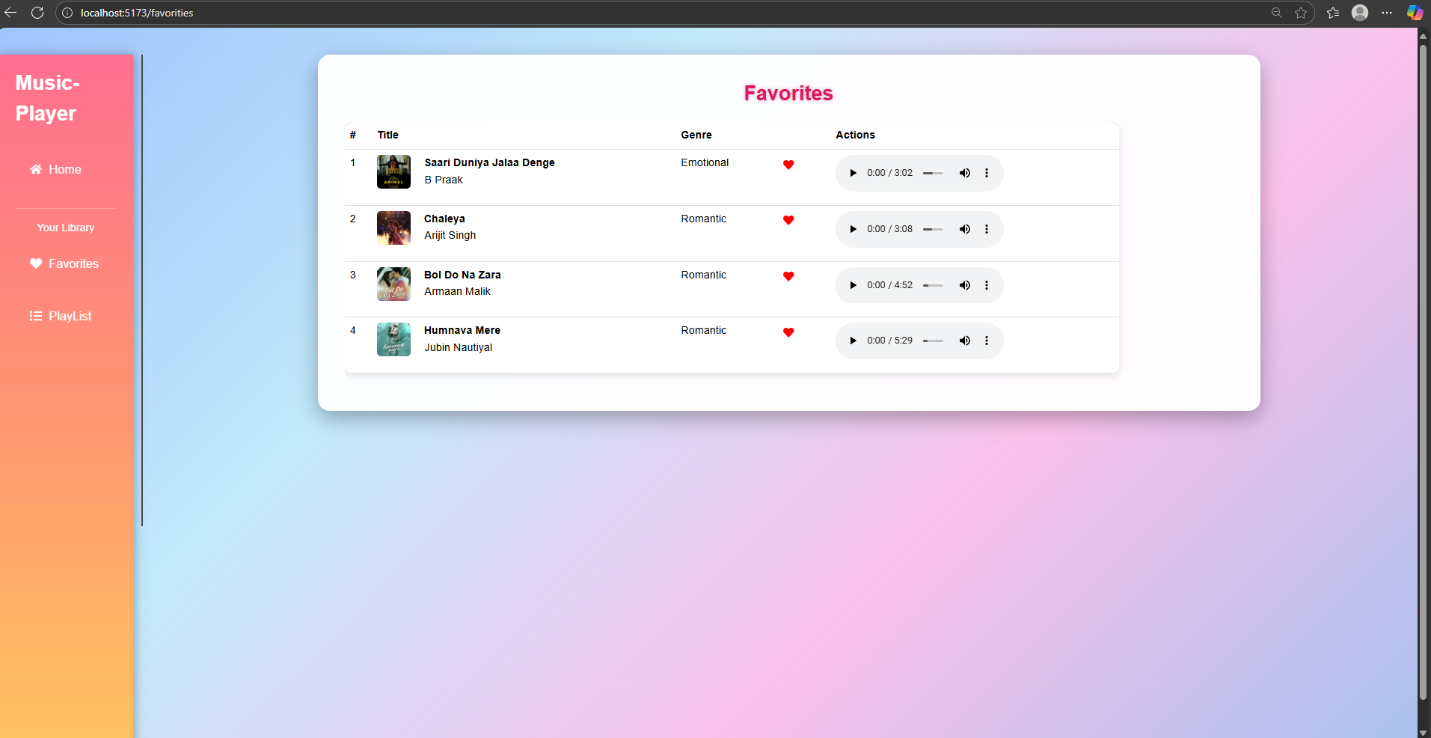
****

**The above image make you clear that the front page of the app MUSIC-PLAYER display as.**

**Playlist:**

****

**Favorites:**

****

# Future Enhancements

 **User Authentication System**: Implementing secure login and registration features to support personalized user experiences and data protection.

 **Enhanced Search Functionality**: Improving the search mechanism to support filters, keyword suggestions, and faster response times.

 **UI Animations**: Adding more dynamic animations and transitions to create a more engaging and modern user interface.

* feature: Potential future features could include a user authentication system, enhanced search functionality, and adding more animations to the user interface.

# DEMOLINK:

# <https://drive.google.com/drive/folders/1ThrwjF-e5HHkXvOJwhnnefzcBvBBzsaZ>