RYTHIMIC TUNES PROJECT DOCUMENTATION INTRODUCTION

Project Title: RYTHIMIC TUNES :

Team Members:

Team ID : SWTID1741179481146382

TEAM MEMBERS	TEAM MEMBERS - EMAIL ID
TEAM LEADER : NISHANTH G	**sainishanth0045@gmail.com**
TEAM MEMBER : JOTHIKANNAN S	**jothikannans088@gmail.com**
TEAM MEMBER : DHAKSHANAMOORTHI K	**dharsanamoorthi9@gmail.com**
TEAM MEMBER : MUNIYANDI S	**muniyandis293@gmail.com**

Project Overview

Purpose:

Rhythmic Tunes is a music streaming web application designed to provide users with a seamless, engaging, and personalized music experience. Built using **React.js** for the frontend and **Node.js** for the backend, the platform enables users to explore, play, and manage their favorite music effortlessly.

Features:

User Authentication: Secure login and signup using JWT authentication.

Music Library: Browse an extensive collection of songs by artists, albums, and genres.

Playlists Management: Create, edit, and delete custom playlists

Favorites & Likes: Mark songs as favorites for easy access.

Architecture:

Component Structure

The application follows a modular component-based architecture:

App.jsx – The main entry component.

Header.jsx – Navigation bar with search and user authentication options.

Home.jsx – Displays a list of songs, albums, and playlists.

MusicPlayer.jsx – A reusable component for playing songs with controls like play, pause, shuffle, and repeat.

Playlist.jsx – Displays user-created playlists and allows adding/removing songs.

SongCard.jsx – A reusable component for individual song display.

LyricsDisplay.jsx – Shows lyrics in sync with the currently playing song.

ThemeToggle.jsx – Dark mode and light mode toggle component.

Footer.jsx – Footer section with links and copyright information.

State Management:

- Uses Context API for managing global state.
- Music Context.jsx handles all music-related data, including songs, playlists, and player controls.
- Local state is used within individual components for UI interactions, such as play/pause status and theme switching.

Routing:

Uses react-router-dom for seamless navigation between pages.

Example Routes:

- **/ → Home Page** (Displays trending and recommended songs)
- **/song/:id →** Song Details Page (Displays lyrics, artist info, and play options)
- **/playlist/:id →** Playlist Page (Shows user-created playlists and songs)
- **/upload →** Upload New Song Page (Admin feature for adding songs)
- **/login →** User Authentication Page

Setup Instructions for Rhythmic Tunes

Prerequisites

- Install **Node.js** (latest stable version recommended)

Installation

1. Clone the repository:

git clone https://github.com/yourusername/rhythmictunes.git

2. Navigate to the project directory:

```
cd cookbook
```

3. Install dependencies:

```
npm install
```

4. Start the development server:

```
npm run dev
```

Folder Structure

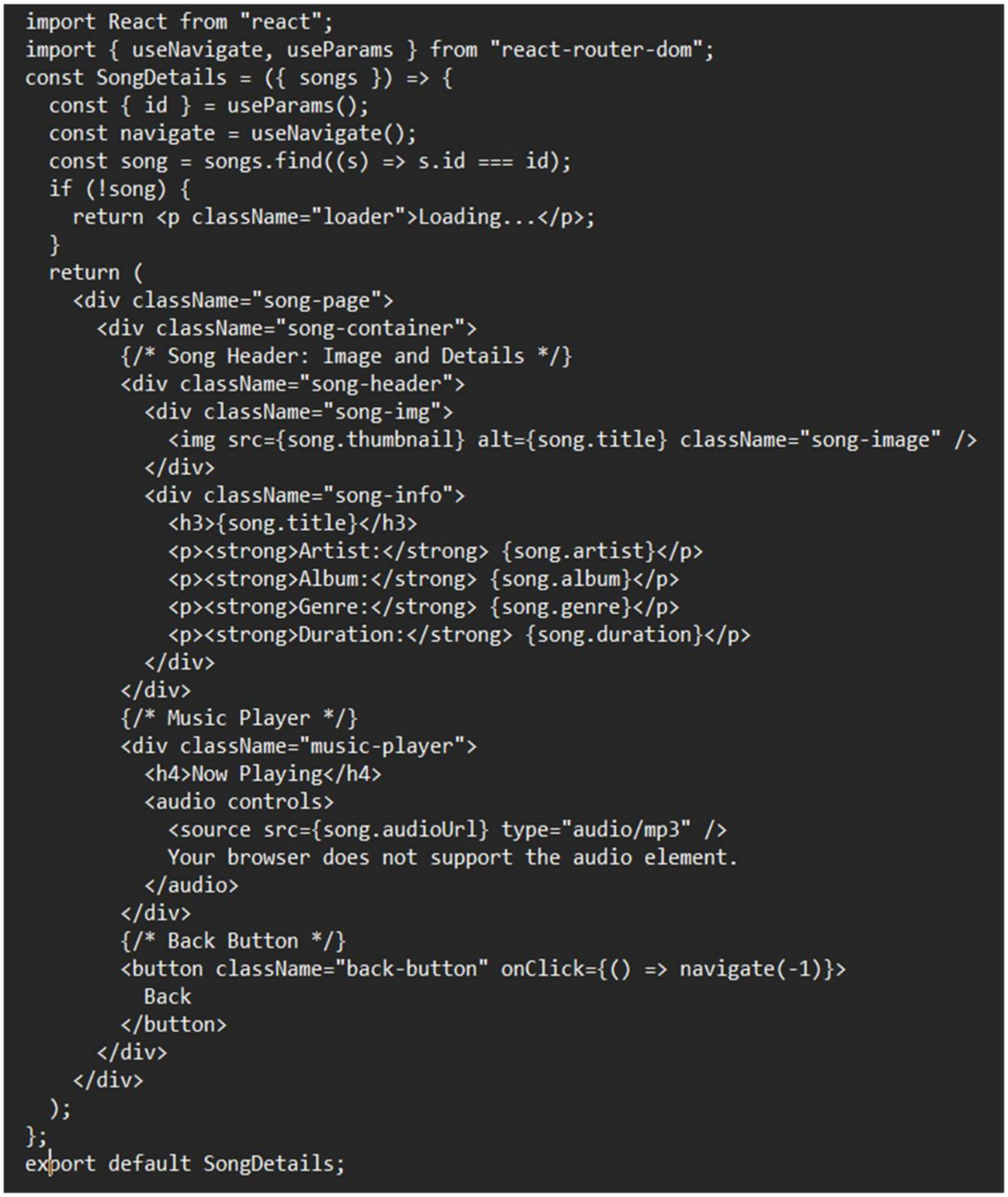
```
📁 RhythmicTunes
|   ├📁 src
|   |   ├📁 components
|   |   |   ├📄Header.jsx
|   |   |   ├📄Footer.jsx
|   |   |   ├📄MusicPlayer.jsx
|   |   |   ├📄Playlist.jsx
|   |   |   ├📄SongCard.jsx
|   |   |   └📄ThemeToggle.jsx
|   |   ├📁 pages
|   |   |   ├📄 Home.jsx
|   |   |   ├📄SongDetails.jsx
|   |   |   ├📄PlaylistDetails.jsx
|   |   |   └📄Login.jsx
|   |   ├📁 context
|   |   |   └📄 MusicContext.jsx
|   |   ├📄 App.jsx
|   |   ├📄main.jsx
|   |   ├📄App.css
|   |   ├📄index.css
|   |   └📄.eslintrc.cjs
|   ├📄 package.json
|   ├📄 package-lock.json
|   ├📄 README.md
|   └📄 vite.config.js
├📁 backend
|   ├📁 config
|   |   └📄db.js
|   ├📂 controllers
|   |   ├📄 authController.js
|   |   ├📄 songController.js
|   |   └📄 playlistController.js
|   ├📂 models
|   |   ├📄 User.js
|   |   ├📄 Song.js
|   |   └📄 Playlist.js
|   ├📂 routes
|   |   ├📄 authRoutes.js
|   |   ├📄 songRoutes.js
|   |   └📄 playlistRoutes.js
└📄 README.md
```

Running the Application

- Run the frontend locally:
- npm run dev

Component Documentation

Key Components



```
import React from "react";
import { useNavigate, useParams } from "react-router-dom";
const SongDetails = ({ songs }) => {
  const { id } = useParams();
  const navigate = useNavigate();
  const song = songs.find((s) => s.id === id);
  if (!song) {
    return <p className="loader">Loading...</p>;
  }
  return (
    <div className="song-page">
      <div className="song-container">
        {/* Song Header: Image and Details */}
        <div className="song-header">
          <div className="song-img">
            <img src={song.thumbnail} alt={song.title} className="song-image" />
          </div>
          <div className="song-info">
            <h3>{song.title}</h3>
            <p><strong>Artist:</strong> {song.artist}</p>
            <p><strong>Album:</strong> {song.album}</p>
            <p><strong>Genre:</strong> {song.genre}</p>
            <p><strong>Duration:</strong> {song.duration}</p>
          </div>
        </div>
        {/* Music Player */}
        <div className="music-player">
          <h4>Now Playing</h4>
          <audio controls>
            <source src={song.audioUrl} type="audio/mp3" />
            Your browser does not support the audio element.
          </audio>
        </div>
        {/* Back Button */}
        <button className="back-button" onClick={() => navigate(-1)}>
          Back
        </button>
      </div>
    </div>
  );
};
export default SongDetails;
```

Reusable Components for RhythmicTunes

Header.jsx – Navigation bar for easy access to different sections.
Footer.jsx – Bottom section with credits, links, or additional info.
SongCard.jsx – Displays a single song card (Title, Artist, Album, Image).
SongForm.jsx – Handles adding/editing songs in the playlist.

State Management & UI for RhythmicTunes

Global State:

- Managed using **Context API** in MusicContext.jsx.
- Provides **songs data** to all components for seamless access.

Local State:

- Used within components for **UI interactions**, such as:
- **Handling form inputs** in SongForm.jsx.
- **Managing play/pause state** in the audio player.

User Interface Styling:

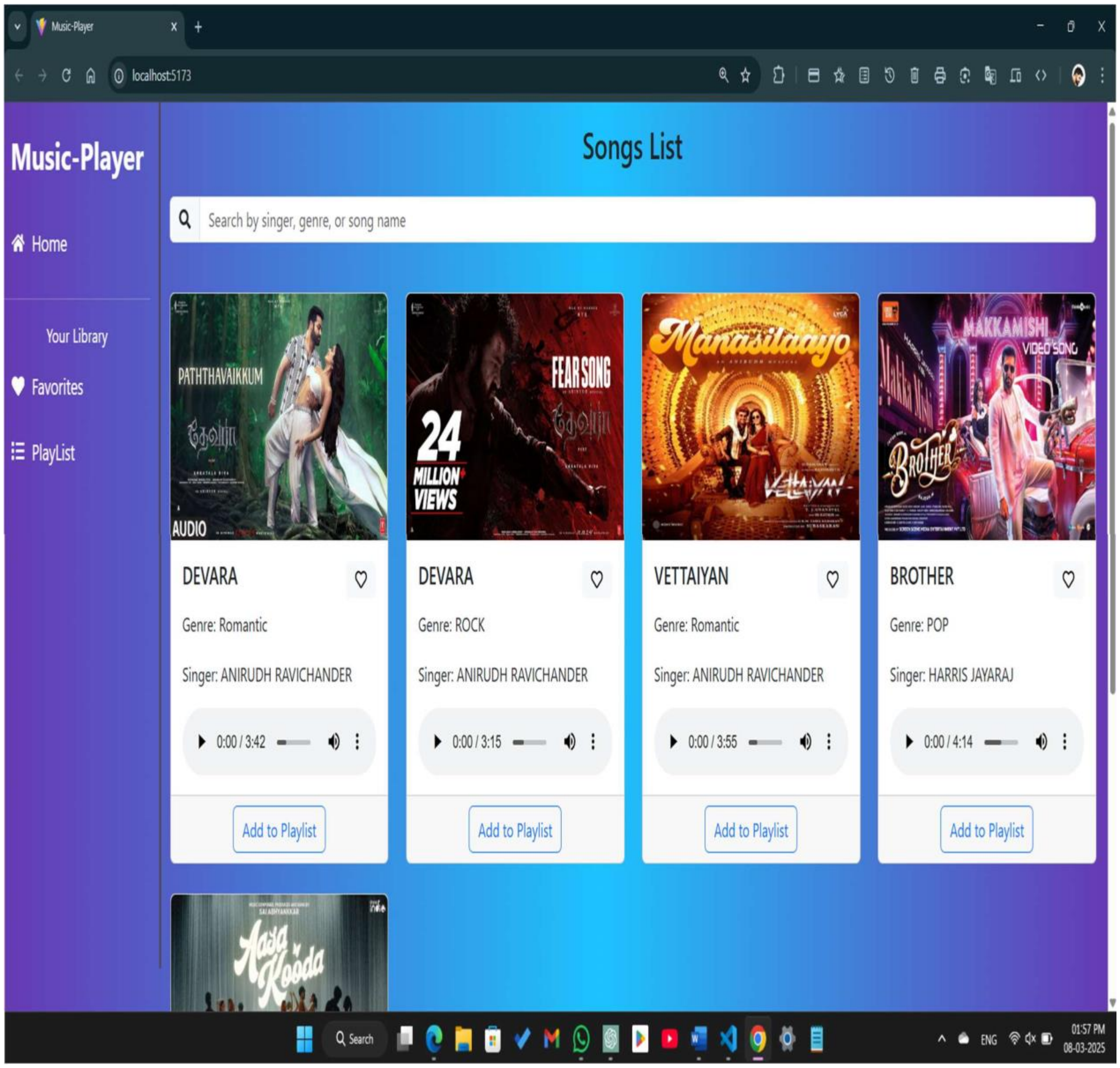
- **CSS:** Uses normal CSS for styling.
- **Theming:** Custom styles for buttons, song cards, and navigation.
- **Responsive Design:** Ensures a smooth experience across devices.

Testing Strategy:

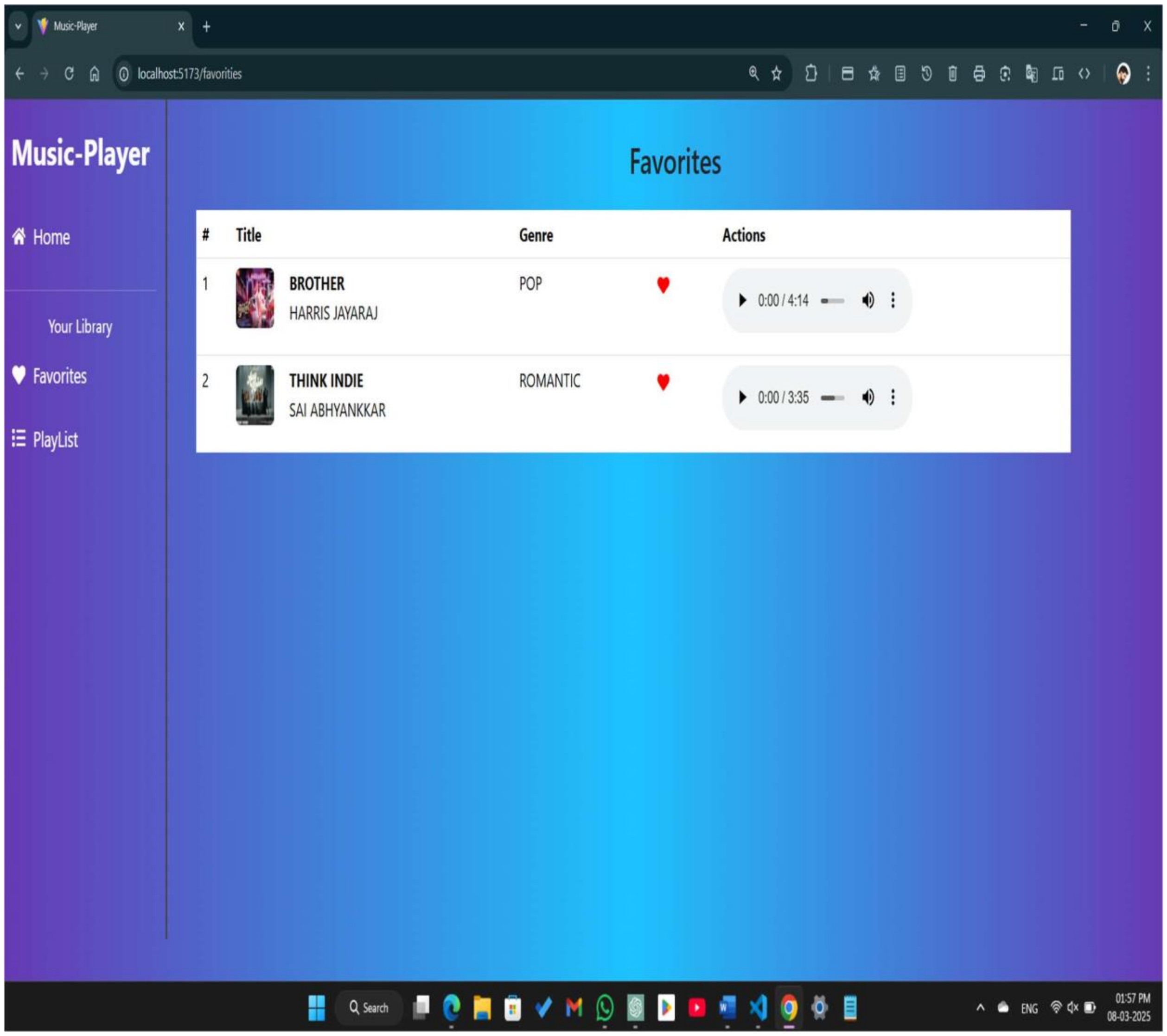
- **Testing Framework:** Uses **Jest & React Testing Library** for unit testing.
- **Code Coverage:** Ensures key components like SongCard.jsx and SongDetails.jsx are well-tested
-

Screenshots or Demo

HOME PAGE:


Music-Player
localhost:5173
Home
Your Library
Favorites
PlayList
Songs List
Search by singer, genre, or song name
DEVARA
Genre: Romantic
Singer: ANIRUDH RAVICHANDER
0:00 / 3:42
Add to Playlist
DEVARA
Genre: ROCK
Singer: ANIRUDH RAVICHANDER
0:00 / 3:15
Add to Playlist
VETTAIYAN
Genre: Romantic
Singer: ANIRUDH RAVICHANDER
0:00 / 3:55
Add to Playlist
BROTHER
Genre: POP
Singer: HARRIS JAYARAJ
0:00 / 4:14
Add to Playlist
01:57 PM
08-03-2025

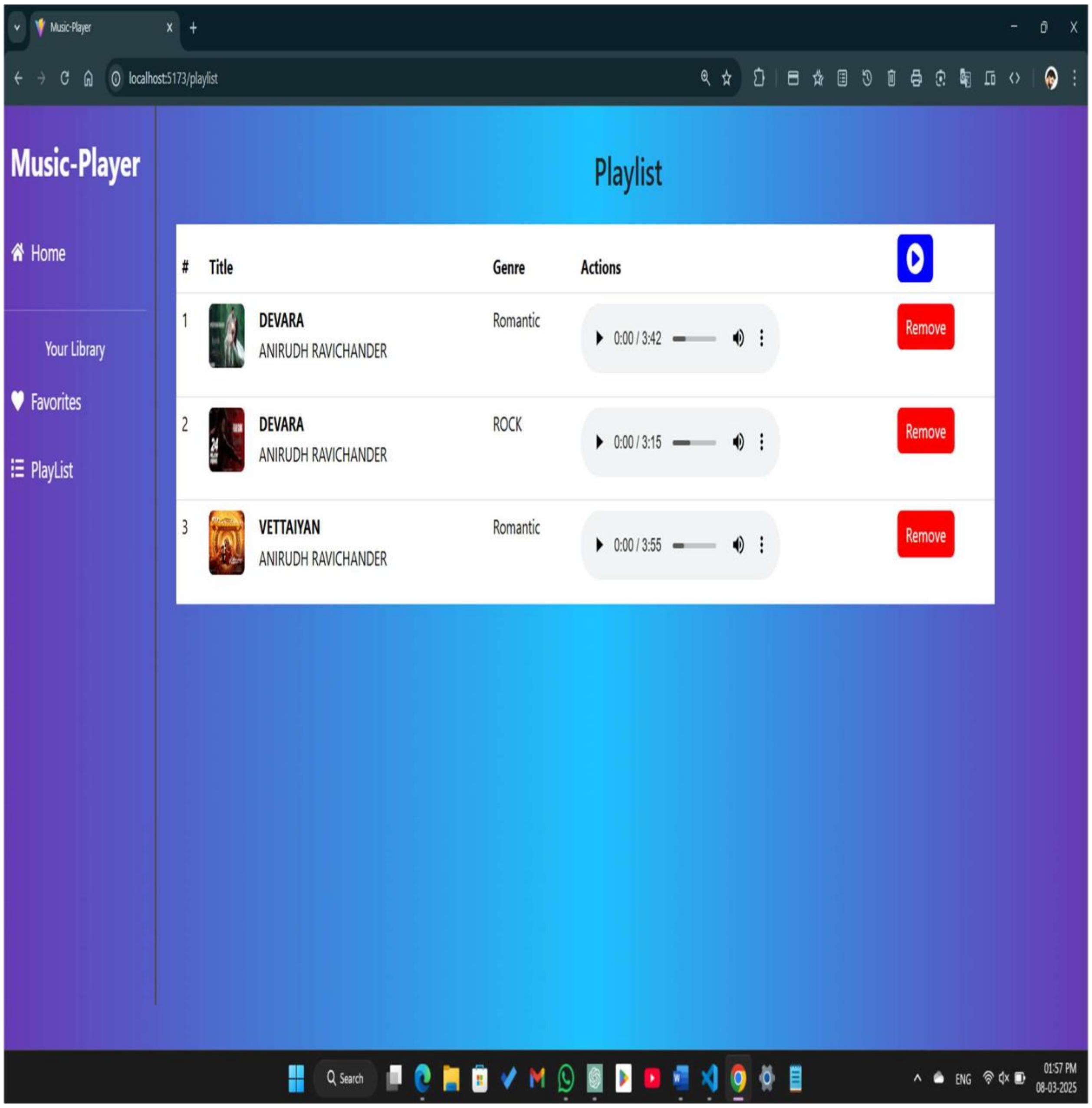
FAVORITES:



Favorites

#	Title	Genre		Actions
1	BROTHER HARRIS JAYARAJ	POP	♥	0:00 / 4:14
2	THINK INDIE SAI ABHYANKKAR	ROMANTIC	♥	0:00 / 3:35

PLAYLIST:



Playlist

#	Title	Genre	Actions	
1	DEVARA ANIRUDH RAVICHANDER	Romantic	0:00 / 3:42	Remove
2	DEVARA ANIRUDH RAVICHANDER	ROCK	0:00 / 3:15	Remove
3	VETTAIYAN ANIRUDH RAVICHANDER	Romantic	0:00 / 3:55	Remove

Demo Link:

https://drive.google.com/file/d/1J6JizhS4Jv_Vy5hGuQKc9iEFkfAy2I-J/view

Known Issues

- Song files are not stored persistently yet.
- Filtering options for songs need refinement (e.g., by artist, album, or genre).

Future Enhancements

- Add a favorites/playlist feature to allow users to save songs.