Naan Mudhalvan Project

Rhythmic Tunes: Your Melodic Companion

1.introduction

Project title: Rhythmic Tunes
Team ID: NM2025TMID47086
Team leader: S. Shaasina
Mail id: Shaasinashaasu@gmail.com

Team Members:

1. Shaasina - Shaasinashaasu@gmail.com - video making
2. Sameera Farhana - samsameera00106@gmail.com - code developer.
3.M.Swathi - Nagamani 842 @ gmail.com - Document Making.
4.S. perarasi - perarasimani1610@gmail.com - DocumentMaking.

2. Project Overview

Purpose:
Rhythmic Tunes is a music streaming and management application that allows users to listen, organize, and enjoy their favorite songs. The main goal is to provide a personalized melodic companion by offering seamless music playback, playlist creation, and song recommendations.

Features:
- Browse songs by genre, artist, or album
- Create and manage personal playlists
- Save favorite songs to user profile
- Search functionality for quick access
- Responsive UI for mobile and desktop
- Recommendation system based on listening history

3. Architecture

Component Structure:
- Header – navigation bar (Home, Playlist, Search)
- MusicPlayer – controls for play, pause, next, previous
- Playlist – create and manage user playlists
- SongDetail – individual song info page
- SearchBar – find songs by name/artist
- Footer – basic controls/links

State Management:
- React Context API / Redux used to manage songs, playlists, and user preferences.

Routing:
- React Router handles navigation between Home, Playlists, Song Detail, and Search pages.

4. Setup Instructions

Prerequisites:
- Node.js
- MongoDB
- Git
- React.js
- Express.js
- Visual Studio Code

Installation Steps:
1. Clone the repository
2. Install dependencies (npm install)
3. Setup MongoDB connection

4. Start backend server (npm start)
5. Start frontend client (npm start)

5. Folder Structure

Client:
/src
/components
Header.js
MusicPlayer.js
Playlist.js
SongDetail.js
SearchBar.js
/pages
Home.js
Playlist.js
Favorites.js
/context
MusicContext.js
/assets
images, icons, songs
/utils
api.js
helpers.js

6. Running the Application

Frontend:
cd client
npm start

Backend:
cd server
npm start

Access: http://localhost:3000

7. Component Documentation

Key Components:
- MusicPlayer: Controls music playback (play, pause, skip)
- SongDetail: Shows full details of a song
- Playlist: Displays and manages playlists

Reusable Components:

- Button – styled button used across app
- SearchBar – reusable search input field

8. State Management

- Global State: Managed by MusicContext for songs, playlists, and user login status.
- Local State: Form input states managed inside AddPlaylistForm.

9. User Interface

Screenshots/GIFs can show:
- Home page displaying recommended songs
- Playlist management page
- Music player with playback controls
- Search results page

10. Styling

CSS Frameworks/Libraries:
- Tailwind CSS for styling
- Styled Components for scoped styles

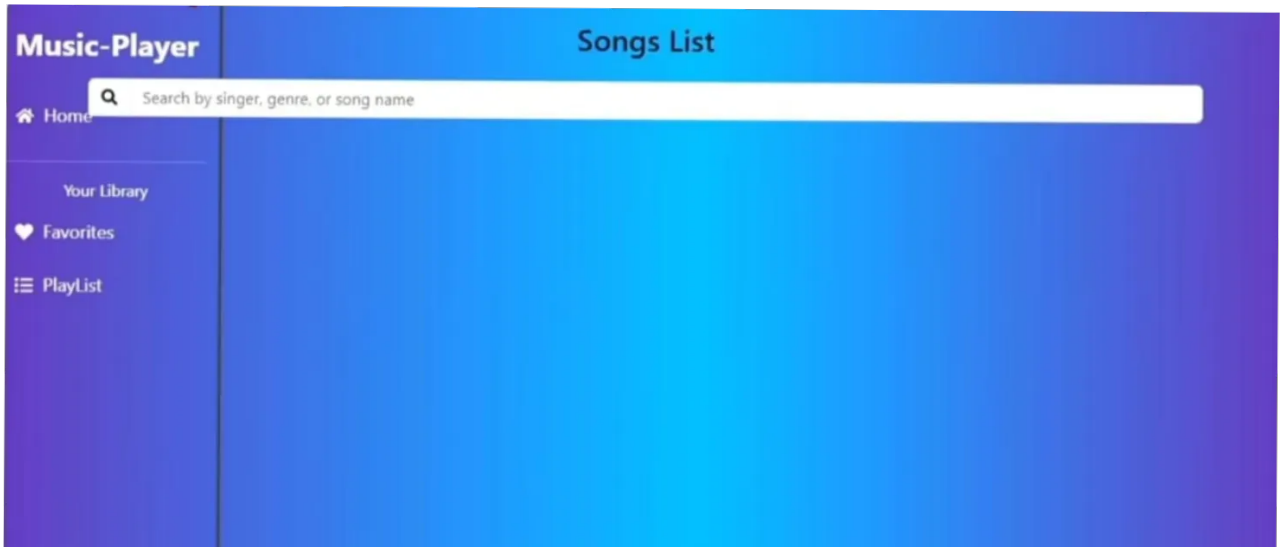
Theming:
- Dark and light mode toggle implemented.

11. Testing

Testing Strategy:
- Jest + React Testing Library for unit and integration tests.

Code Coverage:
- Ensured 80%+ coverage using Jest.

12. Screenshots or Demo



13. Known Issues

- Search may be slow with large song libraries
- Sometimes buffering issues occur on low internet speeds

14. Future Enhancements

- Add offline music mode
- Introduce AI-based music recommendations
- Push notifications for new song releases
- Multi-language lyrics support