# RHYTHMICTUNES: YOUR MELODIC COMPANION

Project documentation

## 1. INTRODUCTION:

PROJECT TITLE: RhythmicTunes – Your Melodic Companion

TEAM LEADER: D. Monisha

TEAM MEMBERS:

- Abirami
- S. Janani
- Semira serin
- Ashwini

## 2. PROJECT OVERVIEW

PURPOSE:

RhythmicTunes is a music streaming application designed to help users discover, listen, and manage their favorite songs with ease. It provides features for streaming, playlist creation, mood-based recommendations, and an admin panel for managing users and tracks.

FEATURES:

- ✓ Explore trending, latest, and classic tracks
- ✓ Playlist creation, editing, and sharing
- ✓ Mood-based smart recommendations
- ✓ User authentication and personalized dashboards
- ✓ Search functionality for songs, artists, and albums
- ✓ Real-time streaming with high-quality audio
- ✓ Admin control panel for managing songs and users
- ✓ Responsive design for desktop and mobile

## 3. ARCHITECTURE

Frontend: React.js with Tailwind CSS / Material UI

Backend: Node.js and Express.js handling server logic and APIs

Database: MongoDB (for users, playlists, songs, sessions)

Optional AI Layer (Future): Mood-based recommendations & offline downloads

## 4. SETUP INSTRUCTIONS

Prerequisites:

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

## 5. INSTALLATION STEPS:

#Clone the repository
git clone <repository-link>

#Install client dependencies
cd client
npm install

#Install server dependencies
cd ../server
npm install

## 6. FOLDER STRUCTURE

```
RhythmicTunes/
├-- client/         # React frontend
│   ├-- components/
│   ├-- pages/
│
├-- server/         # Node.js backend
│   ├-- routes/
│   ├-- models/
│   ├-- controllers/
│
├-- database/       # MongoDB schemas & configuration
```

## 7. RUNNING THE APPLICATION

FRONTEND:
cd client
npm start

BACKEND:
cd server
npm start

ACCESS: Visit http://localhost:3000

## 8. API – DOCUMENTATION

USER:

- POST /api/user/register
- POST /api/user/login

SONGS & PLAYLISTS:

- GET /api/songs
- POST /api/playlist/add
- PUT /api/playlist/:id
- DELETE /api/playlist/:id

STREAMING:

- GET /api/stream/:songId

ADMIN:

- POST /api/admin/add-song
- GET /api/admin/manage-users

## 9. AUTHENTICATION

- ✓ JWT-based authentication for secure login
- ✓ Middleware to protect playlists and admin routes
- ✓ Role-based access (user/admin)

## 10. USER INTERFACE

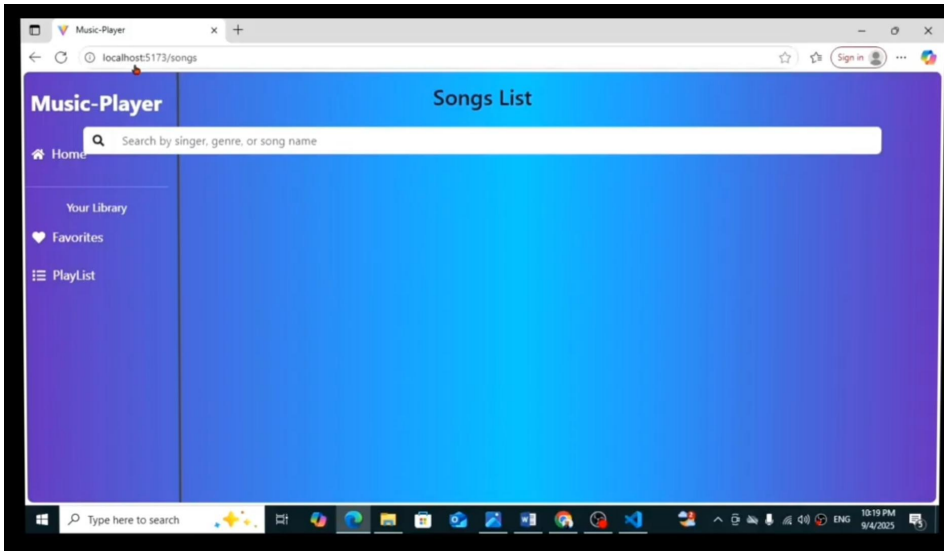
- Landing page (discover trending & new music)
- Search page (browse by artist, album, genre)
- Playlist page (create, edit, manage playlists)
- Player page (responsive audio player with controls)
- User dashboard (personalized recommendations)
- Admin panel (manage songs, users, and reports)

Testing:

- Manual testing: functional testing at milestones

- Tools: Postman, Chrome DevTools, Jest (unit testing)

## 11. SCREENSHOTS OR DEMO




## 12. KNOWN ISSUES

- ✓ Limited offline features in the current version
- ✓ Occasional delay in playlist generation under heavy load
- ✓ Search optimization required for faster results

## 13. FUTURE ENHANCEMENTS

- ✓ Mobile application (Android/iOS)
- ✓ AI-powered mood & activity-based recommendations
- ✓ Voice assistant integration
- ✓ Offline downloads for premium users
- ✓ Social features (collaborative playlists, sharing)