Project Documentation

**Rhythemic Tunes: Your melodic companion**

# Introduction

## Project Title: Rhythemic Tunes: Your Melodic Companion

* + **Team ID:** NM2025TMID29908
  + **Team Leader:** POOJA R & [202400441@sigc.edu](mailto:202400441@sigc.edu)
  + **Team Members:**

- KALAIVANI V & [202400314@sigc.edu](mailto:202400314@sigc.edu)

- MAYURI P & [202400932@sigc.edu](mailto:202400932@sigc.edu)

-PRIYA DHARASHINI S & [202400323@sigc.edu](mailto:202400323@sigc.edu)

# Project Overview

* + **Purpose: "**Rhythmic Tunes" is a music streaming and recommendation platform designed to offer users a personalized and seamless experience in discovering, playing, and enjoying music. The application integrates AI-powered music recommendations based on the user’s listening history and preferences. In addition to the music player, the app features a community-driven space where users can share their favorite tracks, create playlists, and discuss music**.**

## Features:

Music streaming and playlist cration

Personalized recommendations

User authentication & profiles

Search by song, artist, or genre

# Architecture

* + **Frontend:**
  + Built using React.js (or any other modern JavaScript framework like Vue.js or Angular).
  + Redux for state management.
  + Material-UI or Tailwind CSS for a sleek and responsive UI design.
  + **Backend:**
  + Node.js with Express.js for building RESTful APIs.
  + MongoDB as the database for storing user data, playlists, tracks, and other dynamic content.

JWT (JSON Web Tokens) for secure user authentication and authorization**.**

* + **Database:**
  + MongoDB/Mysql/PostgreSQL for storing users, playlist, and preferences
  + Redis (optional) for caching frequently accessed music data

# Setup Instructions

## Prerequisites:

#### **- Node.js (>= 16.x)**

#### **-MongoDB (if using NoSQL)**

#### **-Git for version control >**

-React.js

-Express.js – Mongoose-visual studio code

## Installation Steps:

**Clone the Repository**

Start by cloning the project repository from GitHub:

git clone https://github.com/yourusername/rhythmic-tunes.git

cd rhythmic-tunes

Replace yourusername with your actual GitHub username.

**Install Backend Dependencies**

Navigate to the **backend** directory and install all necessary dependencies:

cd backend

npm install

This will install the required packages such as Express, Mongoose, JWT, etc., in the **backend** folder.

**Install Frontend Dependencies**

Navigate to the frontend directory and install all the dependencies for the React app:

Cd ../frontend

npm install

This will install packages like React, React Router, Material-UI (or Tailwind CSS), Redux (if used), etc., for the frontend.

# Folder Structure

rhythmic-tunes/

│

├── backend/ # Server-side code

│ ├── controllers/ # API route handlers

│ ├── models/ # Mongoose models

│ ├── routes/ # API routes

│ ├── config/ # Configuration files (e.g., database connection)

│ ├── middleware/ # Custom middleware (e.g., for auth)

│ └── server.js # Main server file

│

├── frontend/ # Client-side code

│ ├── src/

│ │ ├── components/ # Reusable UI components

│ │ ├── pages/ # Application views

│ │ ├── services/ # API calls and logic

│ │ ├── context/ # State management (Redux or Context API)

│ │ └── App.js # Main application entry point

│

└── README.md # Project Documentation

# Running the Application

**Frontend**: The React frontend provides a sleek interface to interact with the music player, playlist management, and recommendations.

* + **Start the frontend**:npm start
  + The app will be available at http://localhost:5173

**Backend**: The backend is responsible for user authentication, storing and retrieving music data, handling playlists, and managing APIs.

npm run dev

* + The server will run at http://localhost:5173

**Access:** MongoDB should be running locally or you should configure MongoDB Atlas if you're using a cloud database.

# API Documentation

Base URL: http://localhost:5000/api

**User Authentication**

POST /auth/register → Create new user

POST /auth/login → Login & return JWT

**Music**

GET /music/search?q=keyword → Search for songs

GET /music/recommend → Get personalized recommendations

GET /music/:id → Get song details

**Playlists**

POST /playlist → Create playlist

GET /playlist/:id → Fetch user playlist

PUT /playlist/:id → Update playlist

DELETE /playlist/:id → Delete playlist

# Authentication

Uses JWT (JSON Web Token) for secure login sessions

Passwords stored using bcrypt hashing

Middleware verifies tokens before accessing protected routes (e.g., playlists, favorites)

# User Interface

Login/Signup Page – Secure entry

Home Page – Recommended music, trending playlists

Search Page – Find songs by keyword

Now Playing Screen – Song details with play/pause/next

Playlist Manager – Create, edit, and delete playlists

Profile Page – User details, liked songs, settings

(UI can be made with React + Tailwind CSS for modern look)

# Testing

**Unit Testing:**

Jest / Mocha for backend logic

React Testing Library for frontend components

**Integration Testing:**

Supertest for API endpoints

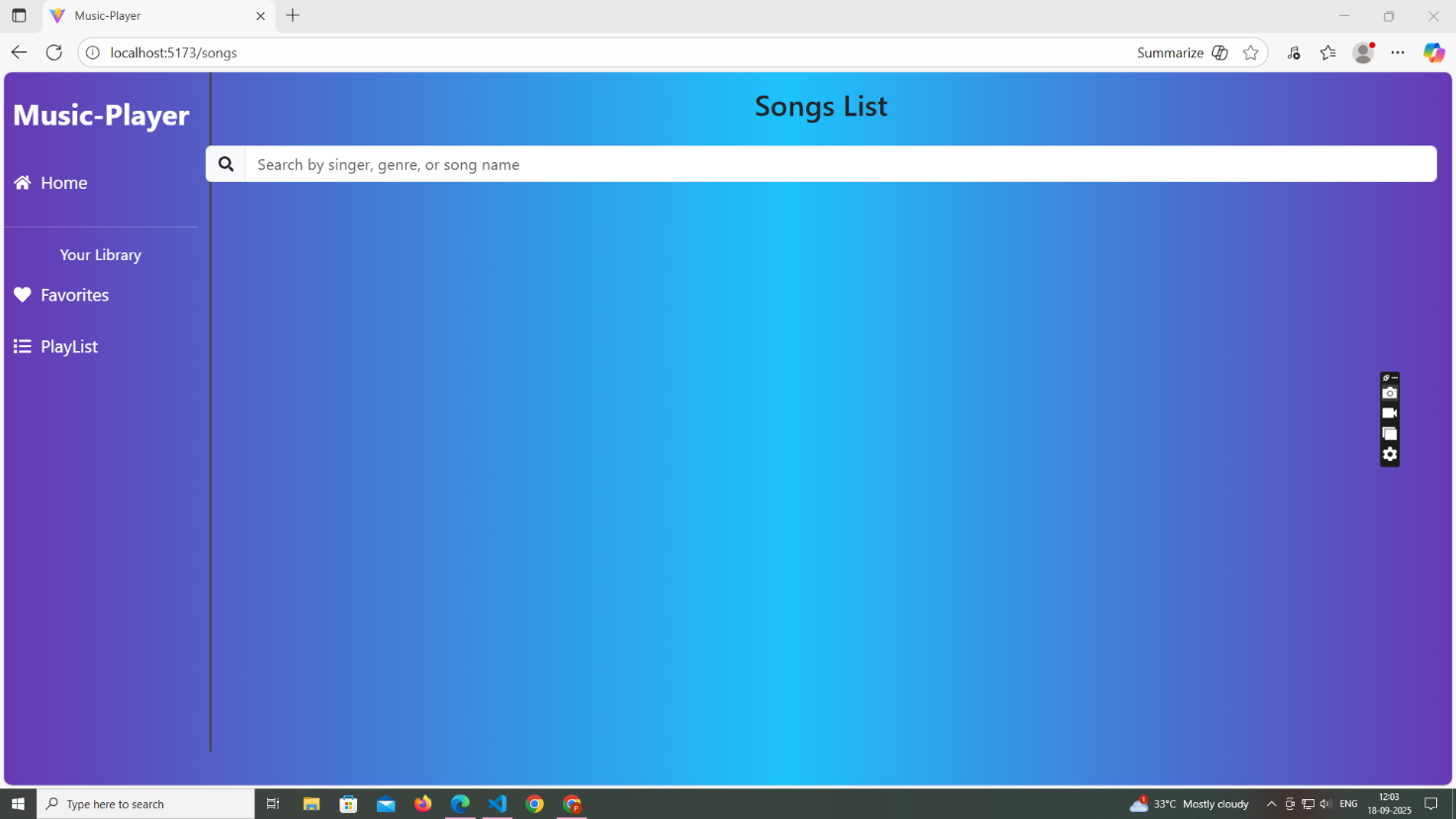
**Manual Testing:**

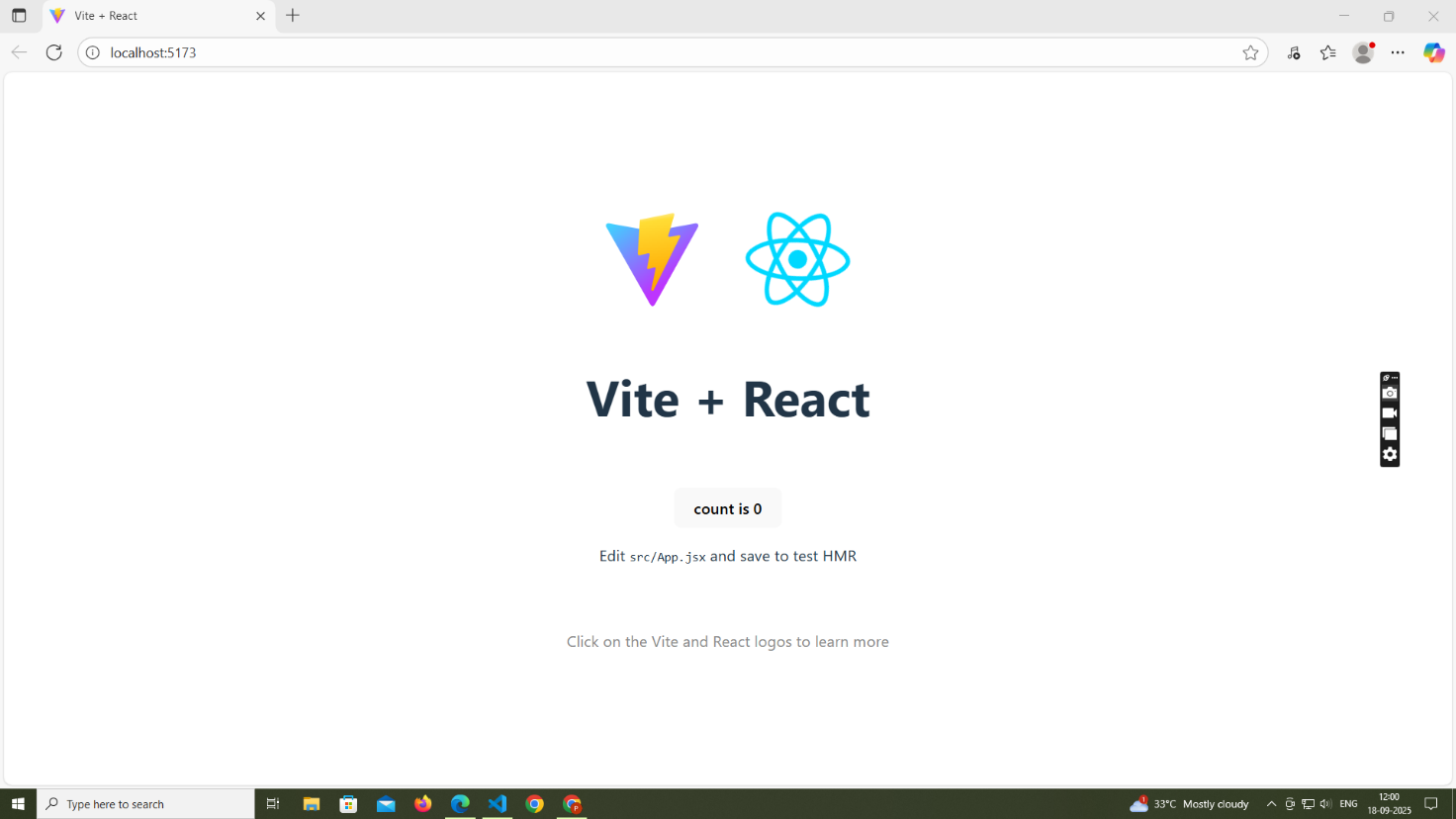
Cross-browser UI testing

API tested with Postman

1. **Screenshots or Demo**

Live Demo: [**https://drive.google.com/file/d/1Ad8UGnKUn3tAjyqPy58BYllkj9mRAi6L/view?usp=drivesdk**](https://drive.google.com/file/d/1Ad8UGnKUn3tAjyqPy58BYllkj9mRAi6L/view?usp=drivesdk)





**12.Known Issues**

1**.**Limited Music API Access – If the external API (Spotify/YouTube) rate limit is exceeded, some songs may not load.

2. Playback Restrictions – Due to licensing, only song previews (30–60s) may be available instead of full tracks.

3. Cross-Browser Compatibility – Minor UI inconsistencies may occur in older browsers (e.g., IE11)

4. Offline Mode Not Supported – Users must have an active internet connection to play music.

5. Search Delay – Music search may be slower if API response is delayed

**13.Future enhancements**

1. Offline Playback – Allow users to download and listen to songs without internet.

2. AI-based Recommendations – Use machine learning to suggest music based on mood, voice input, or listening habits.

3. Lyrics Integration – Display synchronized lyrics during playback.

4Social Features – Enable sharing playlists with friends and collaborative playlist creation.

5. Voice Commands – Add voice-controlled navigation (e.g., "Play top trending songs").