Project Documentation

Rhythmic Tunes

1.Introduction

- **Project Title: Rhythmic Tunes**
- **Team ID :[NM2025TMID37595]**
- **Team Leader: [Swathika.R & swathikaswathika14@gmail.com]**
- **Team Members:**

[Upputuri pushpa & pushpayadavy01@gmail.com]

[Suriyadarshini ranjitham & suriyadharshinis4@gmail.com]

[Subhashini & subha11062007@gmail.com]

[Rathiya.c & rathiyakeerthivasan@gmail.com]

2.Project Overview

- **Purpose:** Rhythmic Tunes is a web application that helps users manage and enjoy music interactively. It provides a platform where users can explore, interact, and personalize their music experience.
- **Features:**

* Music library integration

* Real-time project execution with React

* JSON server backend for managing data

* Secure and scalable architectureProject posting and bidding

3.Architecture

- **Frontend**: the efficient, coordinated flow of data and automated tasks that speed up development and create a smoother user experience
- **Backend:** Node.js, with frameworks like Express.js, can serve as the backend for music streaming applications (e.g., "RhythmicTunes"). It handles user authentication, managing song libraries, creating and storing playlists, and serving audio files to the frontend.
- **Database:** To implement "rhythmic tunes of database using Node.js," you can build a music streaming or creation application where audio data is stored in a database and served to a client. A simple approach uses database records to trigger sounds, while a more complex one involves streaming actual audio files.

4.Setup Instructions

To set up "rhythmic tunes" using Node.js, you will create a full-stack application with a backend that manages musical data in a database and a frontend that uses the Web Audio API to play and interact with those rhythmic patterns. The following guide details how to build a step sequencer using Node.js, Express, MongoDB, and the Tone.js library.

- **Prerequisites:**

- Node.js and npm :Download and install Node.js, which includes the npm package manager.
- MongoDB: Install MongoDB. You can run it locally or use a cloud service like MongoDB Atlas.

Installation Steps:

```
# Clone the repository git clone

# Install client dependencies cd client npm
install

# Install server dependencies cd
../server npm install
```

5.Folder Structure

```
├── client/              # React frontend
│   ├── public/          # Static assets (favicon, index.html, images)
│   └── src/             # Main source code
│      ├── components/    # Reusable UI components
│      ├── pages/         # App pages (Home, Login, Dashboard, etc.)
│      ├── assets/        # Images, icons, styles
│      ├── hooks/         # Custom React hooks
│      ├── context/       # React Context (state management)
│      └── App.js         # Root component
│
├── server/              # Node.js backend
│   ├── routes/          # API routes
│   ├── controllers/       # Business logic
│   ├── models/          # Database models (MongoDB, SQL, etc.)
│   ├── middleware/        # Auth, logging, error handling
│   ├── config/          # Database & environment configs
```

```
│   ├── utils/            # Helper functions
│   └── index.js          # Entry point for backend
├── package.json          # Project metadata (root if monorepo setup)
└── README.md             # Documentation
```

6.Running the Application

- **Frontend:'npm start'(runs on http://localhost:3000)**
- **Backend:'npm run dev'(runs on http://localhost:3000)**

7.API Documentation:

User APIs

Register User

Endpoint:

POST /api/user/register

Request Body:

{ "username": "exampleUser", "email": "user@example.com", "password": "yourPassword" }

Response:

{ "success": true, "message": "User registered successfully", "user": { "id": "64f5a1c2e8b123", "username": "exampleUser", "email": "user@example.com" } }

Login User

Endpoint:

POST /api/user/login

Request Body:

{ "email": "user@example.com", "password": "yourPassword" }

Response:

{ "success": true, "message": "Login successful", "token": "jwt_token_here" }

Project APIs

Create Project

Endpoint:

POST /api/projects/create

Request Body:

{ "title": "My New Music Project", "description": "Collaboration for a Hip-Hop track", "category": "Hip-Hop", "createdBy": "64f5a1c2e8b123" }

Response:

{ "success": true, "message": "Project created successfully", "project": { "id": "6501b3c4f7d901", "title": "My New Music Project", "category": "Hip-Hop" } }

Get Project by ID

Endpoint:

GET /api/projects/:id

Response:

{ "success": true, "project": { "id": "6501b3c4f7d901", "title": "My New Music Project", "description": "Collaboration for a Hip-Hop track", "category": "Hip-Hop", "createdBy": "64f5a1c2e8b123" } }

Application APIs

Apply to Project

Endpoint:

POST /api/apply

Request Body:

{ "projectId": "6501b3c4f7d901", "applicantId": "64f5a1c2e8b123", "message": "I'd love to collaborate on this project!" }

Response:

{ "success": true, "message": "Application submitted successfully" }

Chat APIs

Send Message

Endpoint:

POST /api/chat/send

Request Body:

{ "senderId": "64f5a1c2e8b123", "receiverId": "64f5a9f7c3d456", "message": "Hey, I'm interested in your project!" }

Response:

{ "success": true, "message": "Message sent successfully", "chat": { "id": "6502d4e9a5d789", "senderId": "64f5a1c2e8b123", "receiverId": "64f5a9f7c3d456", "message": "Hey, I'm interested in your project!" } }

Get User Chats

Endpoint:

GET /api/chat/:userId

Response:

{ "success": true, "chats": [{ "id": "6502d4e9a5d789", "senderId": "64f5a1c2e8b123", "receiverId": "64f5a9f7c3d456", "message": "Hey, I’m interested in your project!", "timestamp": "2025-09-18T12:34:56Z" }] }

8.Authentication

- Rhythmic Pattern Creation: A user creates a personalized rhythmic pattern, which can be a combination of taps, holds, or swipes on a touchscreen.
- Enhanced Security: Rhythm-based authentication provides an additional layer of security, making it more difficult for unauthorized users to gain access.Rhythm-based authentication for rhythmic tunes is an innovative approach that uses a user's sense of rhythm to verify their identity. This method involves creating a personalized rhythmic pattern, which serves as a unique authentication credential [1].
- Pattern Recognition: The system recognizes and records the user's rhythmic pattern, which is then stored as their authentication credential.
- Authentication: When the user attempts to access a secure system or application, they are prompted to recreate their rhythmic pattern. If the input matches the stored pattern, the user is authenticated.
- Benefits:
  - ❖ - Enhanced Security: Rhythm-based authentication provides an additional layer of security, making it more difficult for unauthorized users to gain access.
  - ❖ - Improved User Experience: This method is more engaging and enjoyable than traditional authentication methods, such as passwords or PINs.
  - ❖ - Increased Accessibility: Rhythm-based authentication can be particularly beneficial for individuals with disabilities, such as those with visual or motor impairments.

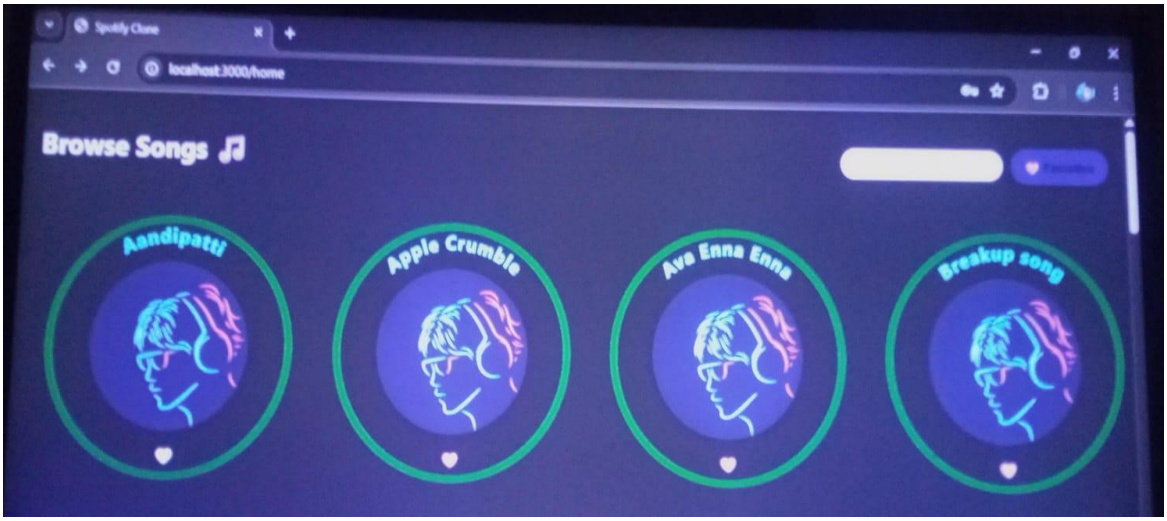
9.User Interface

1. Tune Library: A grid or list view of available rhythmic tunes.
2. Tune Player: A media player to play, pause, and loop rhythmic tunes.
3. Tempo Control: A slider or buttons to adjust the tempo of the tune.
4. Time Signature Display: A visual representation of the time signature.

10.Testing

- Manual testing during milestones
- Tools: Chrome Dev Tools

11.Screenshots :



12.Known Issues :

Known issues for rhythmic tunes include time signature ambiguity, meter changes, rhythmic conflict, and lack of clear structure, which can create confusion for musicians and listeners. Additionally, cultural or personal interpretations of rhythmic tunes can lead to varying performances and analyses, highlighting the complexity and nuance of rhythmic music.

13.Future Enhancements :

1. Social sharing: Enable users to share their favorite rhythmic tunes on social media platforms, with options for sharing as audio files or interactive experiences.

2. Community forums: Create community forums where users can discuss rhythmic tunes, share their own creations, and collaborate with others on new music projects.
   These enhancements can further enrich the rhythmic tunes experience, providing users with new ways to interact with music, learn, and connect with others.