**PROJECT DOCUMENTATION**

**Rhythemic Tunes: Your Melodic Companion**

**1. Introduction**

* **Project Title**: Rhythmic Tunes: Your Melodic Companion
* **Team ID**: NM2025TMID29908
* **Team Leader**: VIDHYA S (202400496@sigc.edu)
* **Team Members**:
  + OVIYA P ([202400973@sigc.edu](mailto:202400973@sigc.edu))
  + VANMATHY B ([202400322@sigc.edu](mailto:202400322@sigc.edu))
  + YUVASRI I ([202400719@sigc.edu](mailto:202400719@sigc.edu))

**2. Project Overview**

* **Purpose**: A music streaming platform that offers personalized recommendations using AI. Users can listen to music, create playlists, and interact with the community.
* **Features**:
  + Music streaming & playlist creation
  + Personalized recommendations
  + User authentication & profiles
  + Search by song, artist, or genre

**3. Architecture**

* **Frontend**: React.js, Redux (for state management), Material-UI/Tailwind CSS (for UI).
* **Backend**: Node.js, Express.js (RESTful APIs), MongoDB (or MySQL/PostgreSQL), JWT for authentication.
* **Database**: MongoDB/MySQL/PostgreSQL for dynamic content, Redis (optional) for caching.

**4. Setup Instructions**

* **Prerequisites**: Node.js, MongoDB (for NoSQL), Git, React.js, Express.js, Visual Studio Code.
* **Installation Steps**:
  1. Clone the repository:
  2. git clone https://github.com/yourusername/rhythmic-tunes.git
  3. cd rhythmic-tunes
  4. Install Backend Dependencies:
  5. cd backend
  6. npm install
  7. Install Frontend Dependencies:
  8. cd frontend
  9. npm install

**5. Folder Structure**

rhythmic-tunes/

│

├── backend/ # Server-side code

│ ├── controllers/ # API route handlers

│ ├── models/ # Mongoose models

│ ├── routes/ # API routes

│ ├── config/ # Configuration files

│ ├── middleware/ # Custom middleware

│ └── server.js # Main server file

│

├── frontend/ # Client-side code

│ ├── src/

│ │ ├── components/ # Reusable UI components

│ │ ├── pages/ # Views

│ │ ├── services/ # API calls

│ │ ├── context/ # State management

│ │ └── App.js # Main app entry

│

└── README.md # Documentation

**6. Running the Application**

* **Frontend**:
* npm start

Available at: http://localhost:5174

**Backend**:

* npm run dev

Available at: http://localhost:5173

* Ensure MongoDB is running locally or set up MongoDB Atlas for cloud database.

**7. API Documentation**

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

**User Authentication**:

* + POST /auth/register: Create a new user.
  + POST /auth/login: Login and return JWT.

**Music**:

* + GET /music/search? =keyword: Search for songs.
  + GET /music/recommend: Personalized recommendations.
  + GET /music/:id: Get song details.

**Playlists**:

* + POST /playlist: Create a playlist.
  + GET /playlist/:id: Fetch user playlist.
  + PUT /playlist/:id: Update playlist.
  + DELETE /playlist/:id: Delete playlist.

**8. Authentication**

* Uses **JWT** for secure login.
* Passwords are stored using **bcrypt**.
* Middleware verifies tokens before accessing protected routes.

**9. User Interface**

* **Pages**:
  + Login/Signup: Secure user authentication.
  + Home: Recommended music & trending playlists.
  + Search: Find songs by keyword.
  + Now Playing: Song details with playback options.
  + Playlist Manager: Create, edit, and delete playlists.
  + Profile: User details and settings.

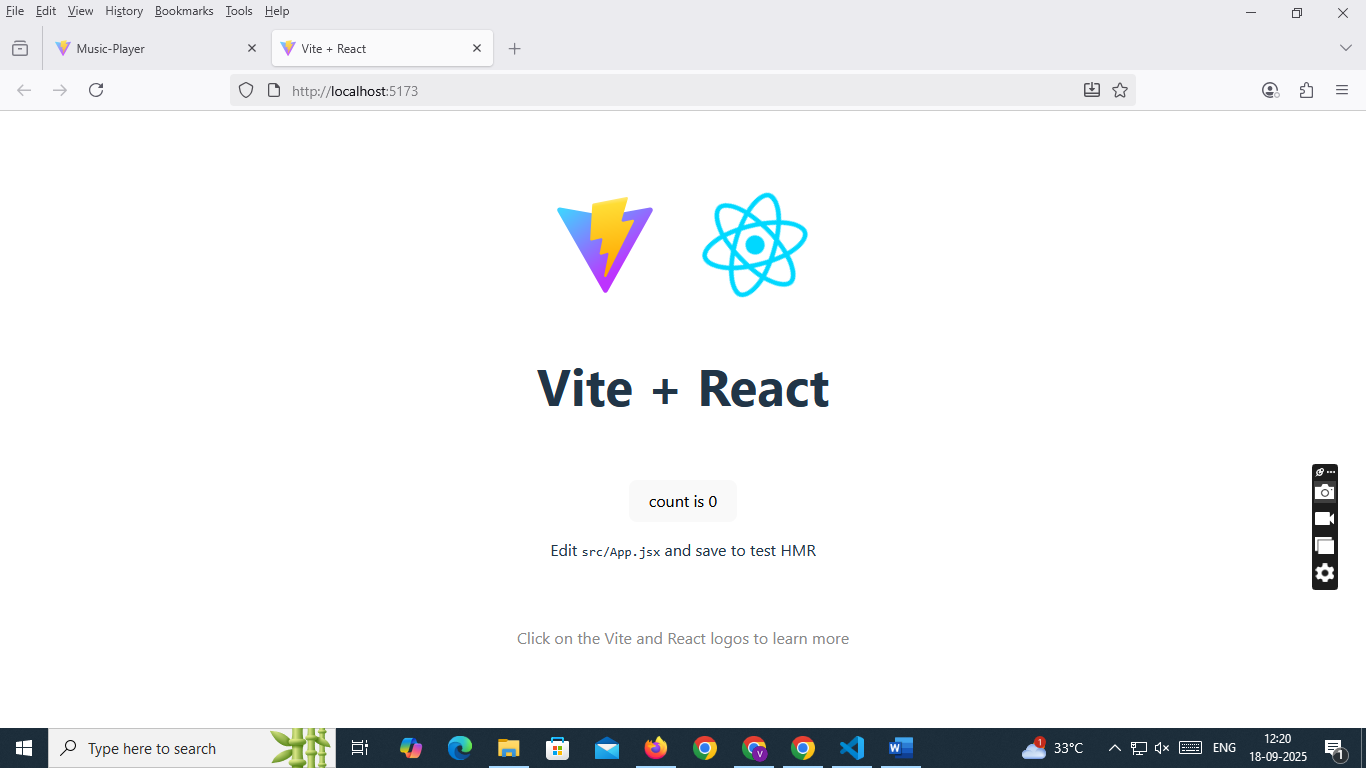
**10. Testing**

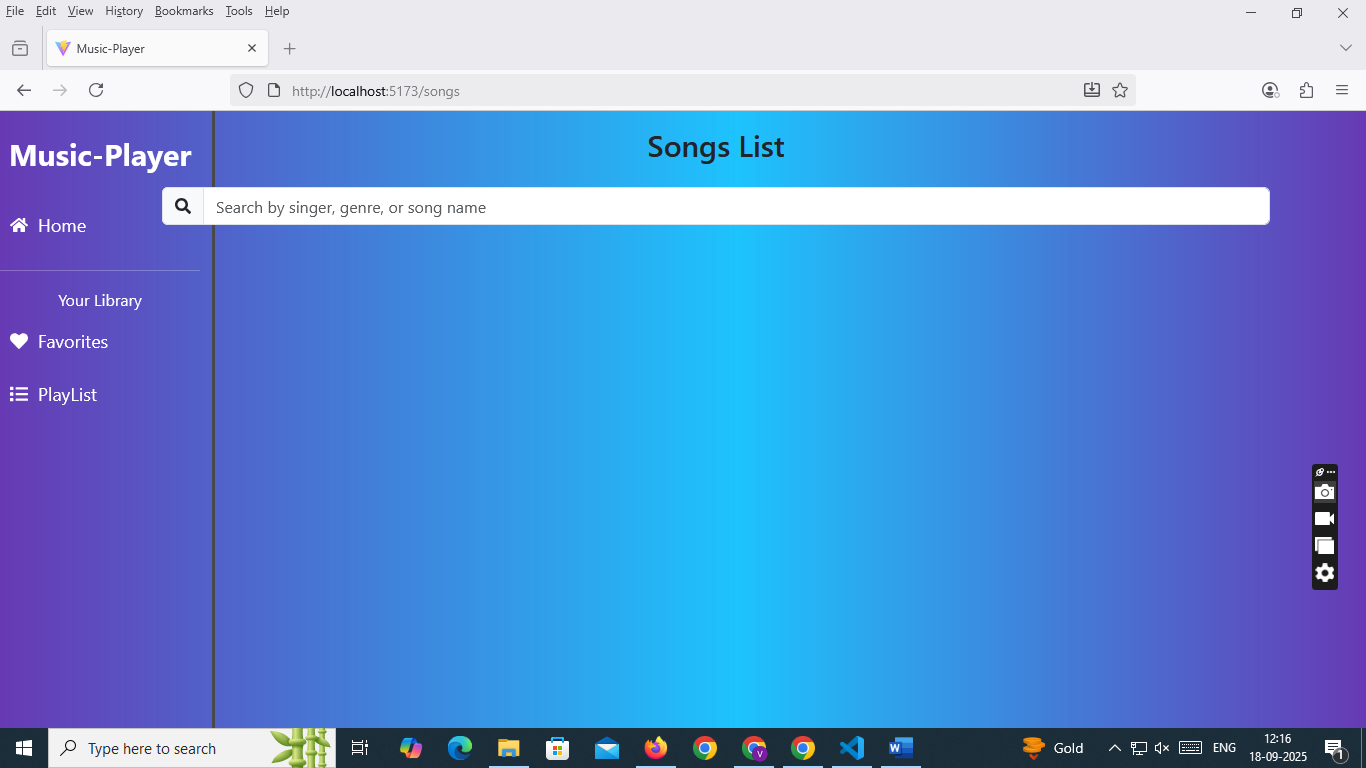
* **Unit Testing**: Jest/Mocha (backend), React Testing Library (frontend).
* **Integration Testing**: Supertest (API).
* **Manual Testing**: Cross-browser UI testing, API tested with Postman.

**11.Screenshots or Demo**

**Demo link:**

<https://drive.google.com/file/d/1Kty1x4Xs1T7rpY5qDkMir_Hwv0aKtK1s/view?usp=drivesdk>





**12. Known Issues**

1. **Limited Music API Access**: Some songs may not load if external API limits are exceeded.
2. **Playback Restrictions**: Only song previews (30-60s) available due to licensing.
3. **Cross-Browser Compatibility**: Minor UI inconsistencies in older browsers (e.g., IE11).
4. **Offline Mode**: Requires internet connection to play music.
5. **Search Delay**: Music search can be slower if API response is delayed.

**13. Future Enhancements**

1. **Offline Playback**: Allow downloading songs for offline listening.
2. **AI-based Recommendations**: Suggest music based on mood, voice input, or listening habits.
3. **Lyrics Integration**: Display synchronized lyrics during playback.
4. **Social Features**: Share playlists and create collaborative playlists.
5. **Voice Commands**: Add voice-controlled navigation (e.g., “Play top trending songs”).