**Project Documentation**

**Project Title: Rhythmic Tunes**

**1.Introduction**

**• Project Title: Rhythmic Tunes**

**• Team ID: NM2025TMID30057**

**• Team Leader: [NANDHINI.L-202400515@sigc.edu](mailto:NANDHINI.L-202400515@sigc.edu)**

**• Team Members:**

**[THARANI.B-202400688@sigc.edu](mailto:THARANI.B-202400688@sigc.edu)**

**[MOHANAPRIYA.B-202400127@sigc.edu](mailto:MOHANAPRIYA.B-202400127@sigc.edu)**

**[NANDHINI.K-202400115@sigc.edu](mailto:NANDHINI.K-202400115@sigc.edu)**

**2. project overview**

**• Purpose: Provide a platform for browsing music with an interactive UI and backend simulation.**

**• Features:**

**- Interactive user interface for music browsing**

**- Local server setup with npm and Vite**

**- JSON Server for backend simulation**

**- Real-time data fetching from db.json**

**3. Architecture**

**• Frontend: React.js (Vite) with Node.js runtime**

**• Backend: JSON Server (mock backend API)**

**• Database: db.json file acting as a mock database**

**4. Setup Instructions**

**• Prerequisites: Node.js, Visual Studio Code, Web Browser, JSON Server**

**• Installation Steps:**

**1. npm install**

**2. npm run dev**

**3. cd db**

**4. json-server --watch db.json --port 3000**

**5. Folder Structure**

**package.json: This is the manifest file for a Node.js project. It contains metadata about the project, such as its name, version, and a list of all its dependencies (libraries and packages it needs to run) and development dependencies. It also defines scripts that can be run to automate tasks like starting the development server or building the project.**

**package-lock.json: This file is automatically generated by npm (Node Package Manager). It records the exact versions of all dependencies and their sub-dependencies. This ensures that anyone who installs the project will get the exact same environment and a reproducible build, preventing issues caused by package updates.**

**.eslintrc.cjs: This is a configuration file for ESLint, a popular JavaScript linter. It defines the rules for code style and quality. ESLint helps developers write consistent code and catch potential errors early by flagging issues like unused variables or improper indentation.**

**vite.config.js: This is the configuration file for Vite, a modern frontend build tool. It specifies how the project should be bundled and served during development and how it should be built for production. Vite is known for its speed and efficient development server.**

**index.html: This is the main entry point for the web application. It's the core HTML file that the web browser loads.**

**README.md: This file is a markdown-formatted text file that provides essential information about the project. It typically includes details like what the project is, how to install it, how to use it, and any other relevant information for users and contributors.**

**.gitignore: This file tells Git (a version control system) which files and folders to ignore and not track. This is crucial for keeping the repository clean by excluding temporary files, build outputs, and sensitive information like user-specific configuration.**

**6. Running the Application**

**• Frontend: npm run dev (default at http://localhost:5173)**

**• Backend: json-server --watch db.json --port 30007. API Documentation**

**• Users: /api/users (GET, POST)**

**• Tracks: /api/tracks (GET)**

**• Playlists: /api/playlists (GET, POST)**

**8. Authentication**

**Rhythmic Tunes uses a basic mock authentication system with JSON Server.**

**• User login/logout is simulated.**

**• Private routes are protected through conditional rendering**

**9.User Interface**

**The application provides a clean interface for browsing music tracks, with navigation and playlist**

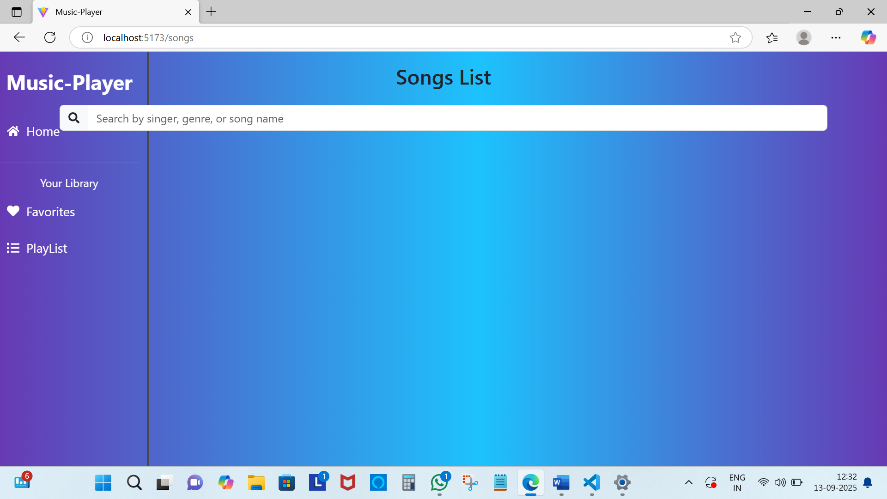
**management features for a smooth experience.**

**10.Testing**

**• Manual testing is performed using Postman and Chrome DevTools.**

**• The app is validated by checking UI responsiveness and API data fetching.**

**11.Screenshot**

****

**12. Known Issues**

**Currently, the project is a mock simulation. Real-time streaming and persistent authentication are**

**not implemented.**

**13. Future Enhancements**

**• Add real authentication (JWT, OAuth).**

**• Integrate real music streaming APIs.**

**• Implement playlist sharing and collaboration.**

**• Enhance UI/UX with animations and responsive design.**

**14. Demo Link:**

**<https://drive.google.com/file/d/1kqfrmbtzgHd5RXGYvJwr48T5XeGq60rU/view?usp=drivesdk>**