

project documentation

RYTHAMIC TUNES



1.INTRODUCTION

* Rythamic Tunes Rythamic Tunes is a freelancing web application designed to connect clients with skilled freelancers through a secure and user-friendly platform. It simplifies the hiring, project management, and collaboration process by offering dedicated spaces for clients, freelancers, and administrators.
* **Secure Authentication** – Uses JWT-based login with middleware to protect private data.
* **User Interface** – Includes a landing page, freelancer dashboard, admin panel, and project details page for smooth navigation.
* **Project Management** – Clients can post projects, freelancers can apply, and both can track progress.
* **Collaboration Tools** – Real-time communication through messaging and notifications.
* **Database** – Stores user profiles, projects, applications, and chat messages in MongoDB.
* **Testing** – Manual testing with Postman and Chrome Dev Tools ensures system reliability.
* **Project Title** : Rythamic Tunes
* **Team Id:** NM2025TMID39843
* **Team Leader**:
* POOJA. K - poojakarthiban2006@gmail.com
* **Team Member:**
  + CHANDIYA. S - sandyqueen1126@gmail.com
  + DIVIYASREE. P – divyaparthiban357@gmail.com
  + JEEVITHA. V - jeevithajeevi038@gmail.com

**2.PROJECT OVERVIEW**

* **PURPOSE:** 
  + **Temporal Framework**: Establishes a sense of time and organizes musical sounds and silences.
  + **Unity and Coherence**: Ties together melodic, harmonic, and other musical elements.
  + **Emotion and Movement:** Evokes feelings and creates a sense of forward momentum.
  + **Character and Feel:** Shapes the overall personality and impression of a piece.
  + **Engagement:** Motivates movement and creates a connection with the listener.
* **FEATURES:**
* **Project Bidding –** Post and bid on music projects.
* **Secure Chat –** Safe user communication.
* **Reviews –** Feedback after project completion.
* **Admin Panel** – Manage users and platform settings.

**3.ARCHITECTURE**

* **Frontend**: Built with React.js, styled using Bootstrap and Material UI. This is the part users interact with — like pages, dashboards, and forms.
* **Backend**: Powered by Node.js with Express.js to handle the server logic and manage API endpoints for communication between frontend and database.
* **Database**: MongoDB is used to store all important data, such as user details, project information, freelancer applications, and chat messages.

**4. SETUP INSTRUCTION**

* **Prerequisites:**
  + **Node.js →** runtime, npm, backend, event-driven, asynchronous
  + **MongoDB →** NoSQL, collections, documents, CRUD, database
  + **Git →** version control, commit, push, pull, branch, merge
  + **React.js →** components, JSX, props, state, hooks, virtual DOM
  + **Express.js →** middleware, routing, REST API, server, request/response.
  + **Mongoose →** schema, model, validation, query, ODM
  + **Visual Studio Code (VS Code) →** extensions, IntelliSense, debugger, terminal, workspace

**Installation Steps:**

* **Clone the repository**

git clone <repository-url>

* **Install client dependencies**

cd client

npm install

* **Install server dependencies**

cd ../server

npm install

* **Start the development servers**

**For client**: npm start

**For server**: npm run dev

**5. FOLDER STRUCTURE**

* **Client folder** – This contains the frontend built with React. Inside it, the components folder has reusable UI elements, while the pages folder holds complete screens like the landing page or dashboard.
* **Server folder** – This contains the backend built with Node.js and Express. The routes folder defines API endpoints, the controllers folder contains the business logic for those endpoints, and the models folder defines how data is stored in MongoDB.

**6. RUNNING THE APPLICATION**

* **Frontend (React Client)**
* Open your terminal.
* Navigate into the client folder: **cd client**
* Start the frontend React app with: **npm start**
* **Backend (Node.js Server)**
* Open a new terminal (keep the frontend running in the other one).
* Navigate into the server folder: **cd server**
* Start the backend server with: **npm start**

**Access:** Visit <http://localhost:3000>

**7. API DOCUMENTATION**

* **User APIs**
* **/api/user/register →** Used when a new user wants to sign up by providing details like name, email, and password.
* **/api/user/login →** Used for logging in. It checks the user’s credentials and returns a token for authentication.
* **Project APIs**
* **/api/projects/create →** Allows a client (or admin) to create a new projectwith details such as title, description, budget, etc.
* **/api/projects/:id** → Fetches or manages details of a specific project using its unique ID (like viewing or updating that project).
* **Application APIs**
* **/api/apply →** Lets freelancers apply for a project by sending their proposal or application details.
* **Chat APIs**
* **/api/chat/send** → Sends a message from one user to another.
* **/api/chat/:userId →** Retrieves the chat history with a particular user using their ID.

**8.AUTHENTICATION**

* **JWT-**based Authentication: Rythamic Tunes uses JSON Web Tokens (JWT) to provide secure login. After a user logs in, a token is issued, which is required for accessing protected resources.
* **Middleware Protection**: Middleware checks the token before allowing access to private routes (like user profile, projects, or chats), ensuring only authorized users can enter.

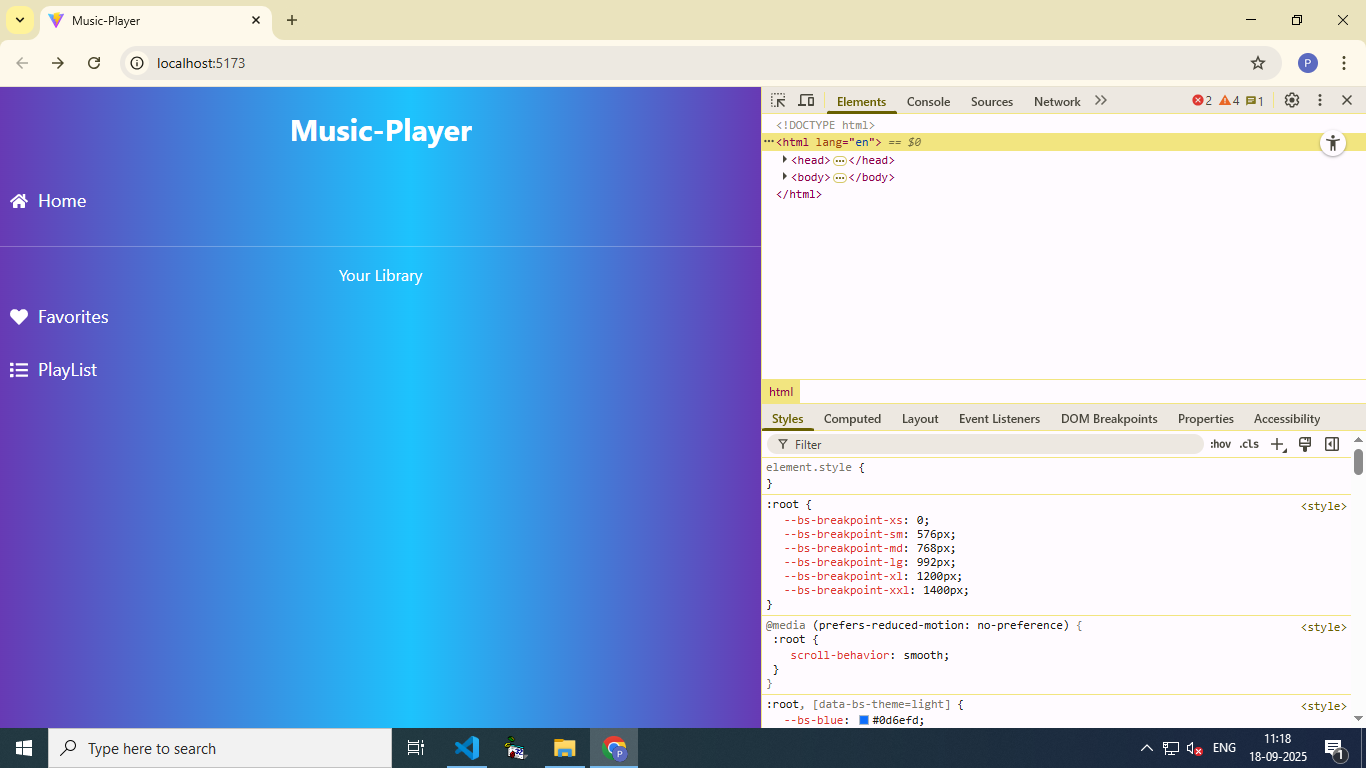
**9.USER INTERFACE**

* **Landing Page :**The homepage that introduces the platform and its features**.**
* **Freelancer Dashboard** : A personal workspace where freelancers can view and manage their projects, profile, and tasks.
* **Admin Panel :** A control area for admins to manage users, projects, and overall system settings.
* **Project Details Page :** A page showing full information about a specific project (description, status, assigned freelancer, etc.).

**10.TESTING**

* **Manual Testing during milestones :** The app is checked by hand at different stages of development to make sure each feature works correctly.
* **Postman :** Used to test backend APIs (sending requests like login, fetch data, etc.).
* **Chrome Dev Tools** : Used to test/debug the frontend, check UI, console errors, and network activity.

**11.SCREENSHORTS OR DEMO**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**12.KNOWN ISSUES**

1. **Authentication Delays –** Sometimes login or token validation may take extra time due to server response.

2. **UI Responsiveness** – Some frontend pages might not display perfectly on smaller screens.

3**. API Errors** – If incorrect data is sent, error messages may not always be clear.

4. **Chat Synchronization** – Messages may not update instantly without refreshing in some cases.

5. **Database Dependencies** – Requires MongoDB to be running; otherwise, APIs fail.

6**. Cross-Browser Compatibility** – Best optimized for Chrome; minor glitches may appear on other browsers.

7. **Testing Coverage** – Only manual testing done for milestones; full automated test coverage is not yet implemented.

**13.FUTURE ENHANCEMENT**

* **AI Recommendations** – Suggest projects, freelancers, or music collaborations using AI.
* **Advanced Chat Features** – Voice/video calls, group chats, and file sharing.
* **Payment Integration** – Secure online payments and wallet system.
* **Analytics Dashboard** – Insights for users, freelancers, and admins.
* **Scalability** – Optimize system for more users and larger data.
* **Enhanced Security** – Stronger authentication and data protection.