**Project Documentation**

**Rhythmic tunes: Your melodic companion**

**Team ID:**

NM2025TMID36551

**Team Leader:**

DEEPA DHARSHINI.S- 24csdeepadharshini015@gmail.com

**Team Members:**

CHARULATHA.S - 24cscharulatha013@gmail.com

DEENAKUMARI.D - 24csdeenakumari014@gmail.com

DEEPIKA.R - 24csdeepika017@gmail.com

**1. Introduction**

Rhythmic Tunes is a music-based project designed to provide users with an interactive platform for discovering, enjoying, and organizing music. The introduction covers the objectives, scope, and intended benefits of the system. The project envisions a world where music enthusiasts can seamlessly connect with their favourite tunes, explore rhythmic patterns, and build customized playlists with ease. This section also explains the motivation behind the project and its expected outcomes.

**2. Project Overview**

The purpose of Rhythmic Tunes is to act as a melodic companion for users. The project bridges the gap between conventional music players and modern intelligent systems. With advanced features like personalization, interactive UI, and real-time recommendations, users experience an engaging platform. The overview further elaborates on the vision, mission, and goals. It highlights the features like song categorization, playlist management, and smooth streaming experience.



Features:

* Music Discovery and Categorization: Allows users to explore songs by genres, moods, and trending playlists.
* Personalized Playlists: Suggests and auto-generates playlists based on user preferences and listening history.
* Interactive User Interface: A modern, responsive UI for smooth navigation across devices.
* Feedback and Review System: Users can rate songs, provide feedback, and influence recommendations.
* Admin Control Panel: A dedicated space for system administrators to manage users, content, and updates.

**3. Architecture**

Frontend:

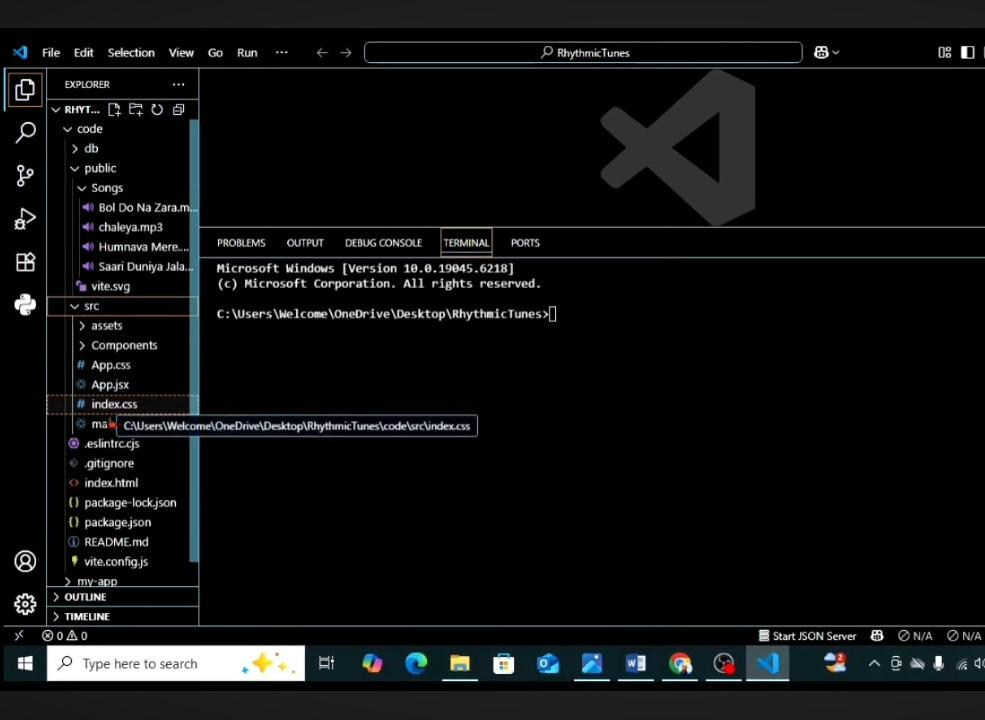
The frontend of Rhythmic Tunes is developed using React.js, enhanced with Bootstrap and Material UI. This combination ensures a responsive, interactive, and visually appealing user experience. Components such as navigation bars, music cards, playlists, and dashboards are modularly designed, making it easy to scale and maintain. The responsive design ensures smooth access on desktops, tablets, and mobile devices.

Backend:

The backend is built with Node.js and Express.js, which manage server logic, handle API requests, and maintain secure communication between the client and server. It supports authentication, playlist management, music search, and recommendation services. The modular structure of controllers and routes ensures organized code and efficient request handling.

Database:

MongoDB is used as the database to store user details, playlists, preferences, feedback, and system logs. Its flexibility and scalability make it ideal for handling diverse data such as user profiles, music metadata, and activity history. Mongoose ORM is used for schema definition and database interaction.



**4. Setup Instructions**

Prerequisites:

To run Rhythmic Tunes successfully, the following tools must be installed on your system:

1. Node.js – Required to run the backend server and manage dependencies.

2. MongoDB – For database storage and management of user data, playlists, and feedback.

3. Git – To clone the project repository and manage version control.

4. React.js – For building and running the frontend interface.

5. Visual Studio Code (VS Code) – Recommended IDE for editing and debugging the project.

Installation Steps:

1. Install Visual Studio Code and open it on your system.

2. Download and install Node.js (latest stable version).

3. Install MongoDB and set up the local database service.

4. Clone the project repository using Git:

git clone <repository\_link>

5. Open the project in VS Code and verify that dependencies are properly installed.

**5. Folder Structure**

The project is organized into a clean and modular folder structure to ensure clarity, maintainability, and scalability.

Rhythmic-Tunes/

│-- client/               # React frontend

│   │-- components/       # Reusable UI components (buttons, cards, music players)

│   │-- pages/            # Complete pages (Landing, Dashboard, Admin, Project Details)

│   │-- assets/           # Images, icons, and styling files

│   │-- utils/            # Helper functions for frontend logic

│

│-- server/               # Node.js backend

│   │-- routes/           # API route definitions (user, playlist, chat, feedback)

│   │-- models/           # MongoDB schemas (user, music, playlist, reviews)

│   │-- controllers/      # Business logic for handling API requests

│   │-- middleware/       # Authentication and error-handling middlewares

│   │-- config/           # Database configuration and environment settings

│

│-- docs/                 # Documentation files

│-- README.md             # Project description and instructions

│-- package.json          # Project dependencies and scripts

client/ → Contains all frontend code built with React.js, organized into modular components and pages for easy reuse.

server/ → Backend implementation using Node.js and Express.js. Handles APIs, authentication, and database interactions.

docs/ → Stores documentation-related materials such as user guides or screenshots.

package.json → Maintains dependency details for both frontend and backend.

**6. Running the Application**

Once the prerequisites and installation steps are completed, Rhythmic Tunes can be executed by starting both the frontend and backend servers.

Frontend:

1. Open a terminal and navigate to the client folder:

cd client

2. Start the frontend React application using:

npm start

3. This will launch the frontend in your default browser, usually accessible at:

http://localhost:3000.

Backend:

1. Open another terminal and navigate to the server folder:

cd server

2. Start the backend Node.js server using:

npm start

3. The backend server will begin running, ready to handle API requests and connect with the frontend.

Access:

Once both servers are running, users can interact with Rhythmic Tunes through the web interface at http://localhost:3000.

The system allows login, music browsing, playlist creation, and other features in a local development environment.



**7. API Documentation**

The backend of Rhythmic Tunes provides a set of RESTful APIs that enable communication between the frontend and the server. These APIs handle user management, project (music/playlist) handling, applications, and chat features.

User APIs:

* POST /api/user/register → Allows a new user to register by providing details such as name, email, and password.
* POST /api/user/login → Authenticates users and returns a JWT token for secure access.
* GET /api/user/profile → Fetches the profile details of a logged-in user.
* Projects (Playlists) APIs:
* POST /api/projects/create → Creates a new playlist or music project entry.
* GET /api/projects/:id → Retrieves details of a specific project/playlist by its ID.
* GET /api/projects/all → Fetches all available playlists or music collections.
* Applications (User Actions):
* POST /api/apply → Represents user actions such as adding songs to a playlist or subscribing to updates.

Chats APIs:

* POST /api/chat/send → Sends a message between users in real time.
* GET /api/chat/:userId → Fetches chat history between two users.
* API Characteristics:
* All APIs follow REST principles (stateless communication, structured endpoints).
* JSON format is used for request and response bodies.
* APIs are protected using JWT authentication middleware to ensure secure access.
* Errors are handled gracefully with appropriate HTTP status codes (200, 400, 401, 404, 500).

**8. Authentication**

Authentication in Rhythmic Tunes ensures that only authorized users can access private resources and perform secure operations. The system uses JWT (JSON Web Token)-based authentication, which is a widely adopted mechanism for modern web applications.

JWT-based Authentication:

* When a user logs in, the server generates a JWT token that contains encrypted user details.
* This token is returned to the client and stored securely (usually in local storage or cookies).
* For each subsequent request to private APIs, the client must include this token in the request header.
* The server verifies the token before granting access to the requested resource.
* Middleware Protection: Middleware functions are implemented in the backend to validate tokens automatically. Any request without a valid JWT is rejected with an appropriate error code (401 Unauthorized).
* This ensures sensitive operations such as creating playlists, sending chats, or managing user profiles are secured.

Password Security:

* User passwords are hashed before being stored in the database using modern encryption algorithms (e.g., bcrypt).
* Plain-text passwords are never stored, ensuring maximum safety even if the database is compromised.

Access Control:

* Different roles (e.g., User, Admin) have specific permissions.
* Admins can manage overall system activities, while users are restricted to their own accounts and playlists.

**9. User Interface**

The User Interface (UI) of Rhythmic Tunes is designed to provide a seamless, engaging, and intuitive experience for all users. The system uses React.js with Material UI and Bootstrap to deliver a modern, responsive design that adapts to multiple devices.

Landing Page:

* The first page users encounter, showcasing the project’s purpose and features.
* Includes a navigation menu, login/register buttons, and a quick overview of services.
* Designed to attract users with visually appealing elements such as banners and animations.
* Freelancer Dashboard (User Dashboard in this case):
* Personalized dashboard for users to manage their playlists, discover music, and track activity.
* Displays recommended songs, trending playlists, and recently played tracks.
* Provides quick access to actions like creating a new playlist or editing existing ones.

Admin Panel:

* Accessible only by system administrators.
* Used for monitoring overall platform activity, managing users, handling reported content, and updating music collections.
* Ensures smooth functioning of the system with role-based access.
* Project Details Page (Playlist Details Page):
* Provides detailed information about a selected playlist or music collection.
* Includes options such as play, add to favorites, provide feedback, and share.
* Displays user reviews, ratings, and related recommendations.

UI Characteristics:

Responsive Design: Works smoothly on desktops, tablets, and mobile devices.

Interactive Elements: Includes dynamic music players, search bars, and filtering options.

User-Friendly Layout: Minimal clicks required to access key features, reducing complexity.

Accessibility: Designed with color contrasts and clear navigation to support diverse user needs.

**10. Testing**

Manual Testing

Manual testing was employed during each development milestone to validate the core functionalities of the Rhythmic Tunes application. This approach allowed the team to interact directly with the user interface and APIs, simulating real-world usage scenarios.

Key aspects covered under manual testing:

User Workflows

Verified the end-to-end experience for different user types:

Listeners: account registration, searching and playing songs, creating playlists

Artists: uploading tracks, managing profile, viewing listener feedback

Admins: logging in, reviewing uploads, moderating content

Form Validation

Checked form behaviors with both valid and invalid inputs (e.g., empty fields, wrong email format, short passwords).

Navigation and UI

Ensured all buttons, links, and tabs navigated to the correct pages. Verified UI responsiveness across screen sizes.

Audio Playback

Tested that uploaded tracks played correctly, with working controls (play/pause/seek), volume adjustment, and error handling for broken audio links.

Error and Success Messages

Confirmed that user feedback (e.g., "Track uploaded successfully", "Invalid login credentials") displayed correctly and helped guide user behavior.

Tools Used

To support manual testing, the following tools were utilized:

Postman

Used to test and validate all backend API endpoints such as user registration, login, track upload, playlist creation, and music streaming.

Checked HTTP methods (GET, POST, PUT, DELETE), response payloads, and status codes.

Helped simulate error scenarios by sending malformed or unauthorized requests.

Chrome Developer Tools

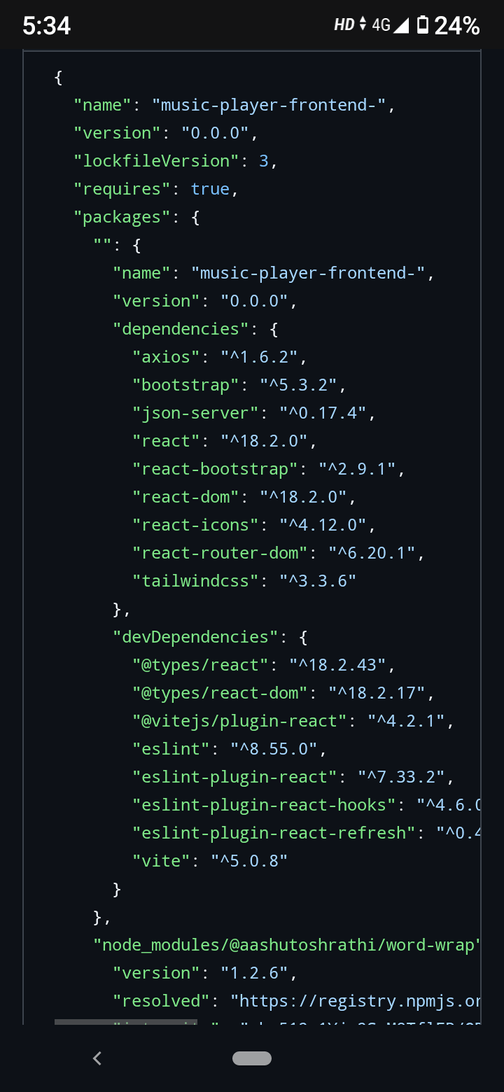
Inspected HTML, CSS, and JavaScript behavior during frontend testing.

Tested responsive design using the device emulator.

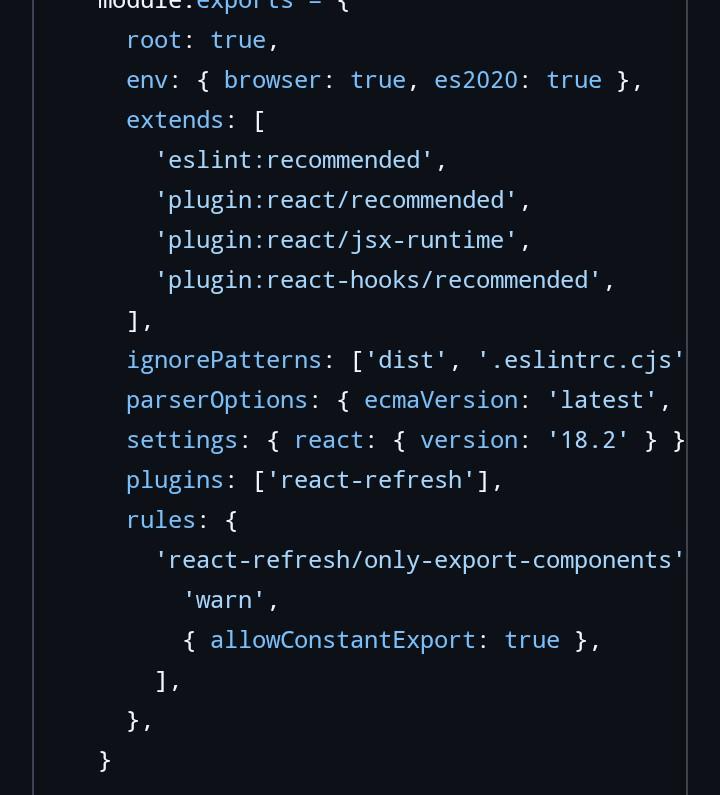
Monitored console logs, network requests, and performance insights to identify UI bugs and slow-loading resources.

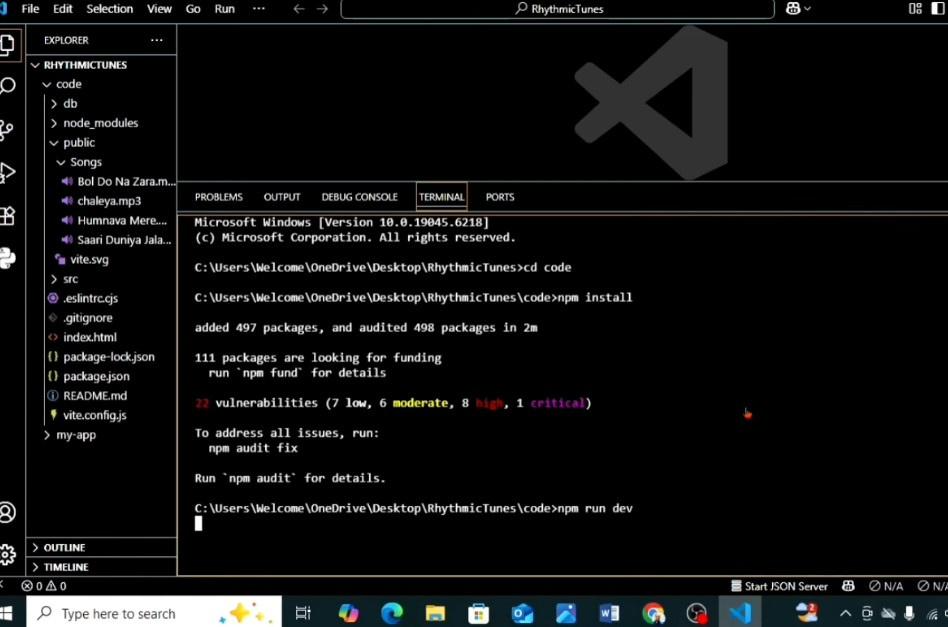
These tools complemented manual efforts and allowed for fast debugging and validation during development.

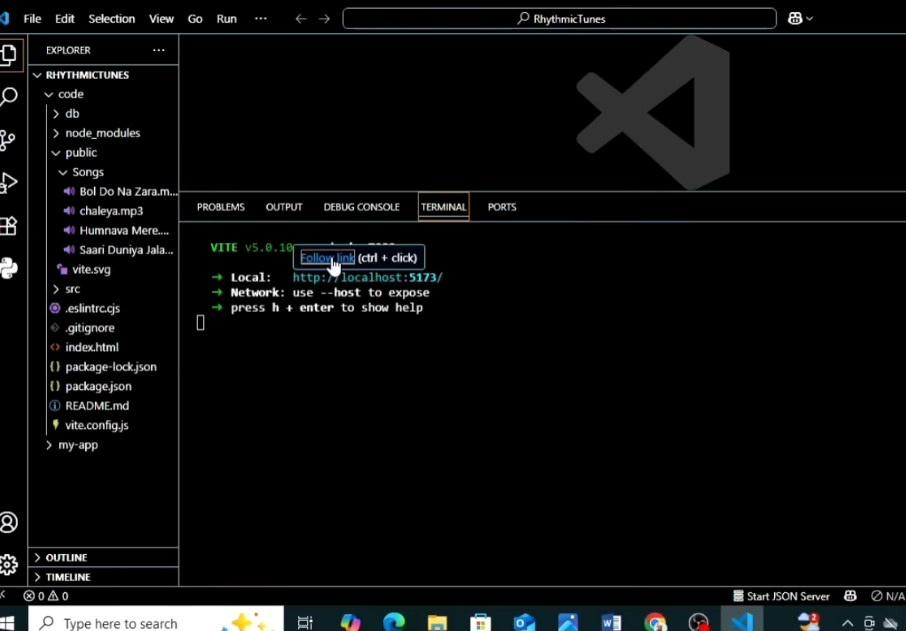
**11. Screenshot or Demo**

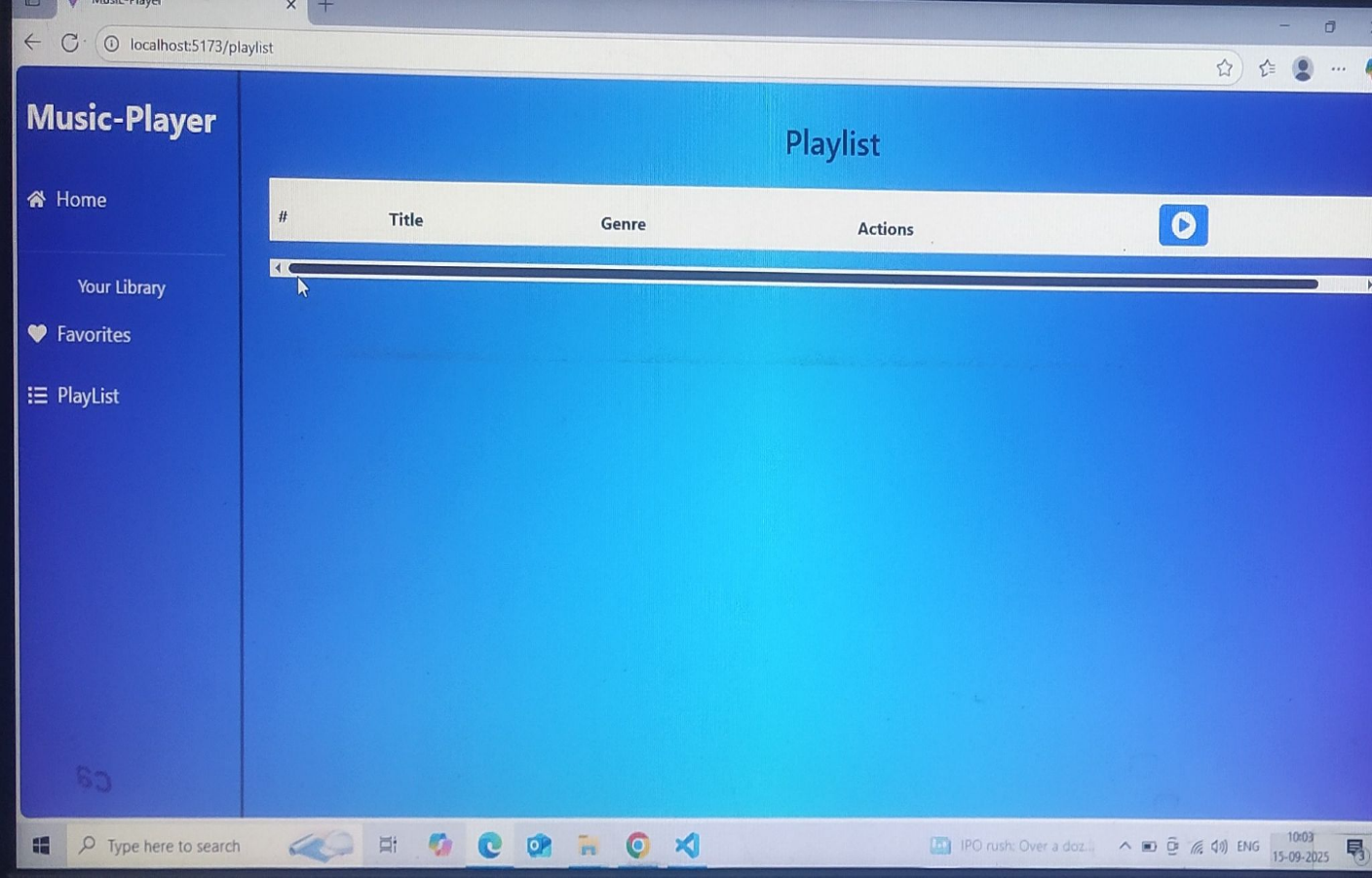












**12. Future Improvement**

Future plans include integrating AI-based  music recommendations, adding voice recognition features, enabling offline mode,and supporting global user communities. These enhancements will ensure that Rhythmic Tunes remains innovative and user-centric.

