**Rythmic Tunes**

**1. Introduction**

* **Project Title**: Rythmic Tunes
* **Team Members**:

Tamilselvan G

Praveen Kumar M (TEAM LEADER)

Prasanth S

Sathish S

**2. Project Overview**

* **Purpose**: Rythmic Tunes is a music streaming web application designed to allow users to discover, play, and manage playlists with ease.[flatlogic+2](https://flatlogic.com/blog/writing-documentation-for-your-javascript-project/)
* **Features**:
  + Secure user authentication and profile management
  + Playlist creation and editing
  + Real-time music streaming
  + Search for tracks, albums, and artists
  + Responsive, modern UI (if frontend is present)

**3. Architecture**

* **Component Structure**: Organized using controllers (handle routes), models (manage data), services (logic), and routes (API endpoints).
* **State Management**: Application state persists via a database (e.g., MongoDB). Session or JWT handles user authentication.
* **Routing**: Express.js manages API routing; RESTful endpoints exposed for frontend integration.

**4. Setup Instructions**

* **Prerequisites**:
  + Node.js (v18+ recommended)
  + npm or yarn
  + MongoDB (or other supported DB)
* **Installation**:
  + Clone the repository:

bash

git clone https://github.com/your-username/rythmic-tunes.git

* + Navigate to the project:

bash

cd rythmic-tunes

* + Install dependencies:

bash

npm install

* + Set environment variables in a .env file (see Configuration).

.

**5. Running the Application**

* Start the server:

bash

npm start

* Visit http://localhost:3000 or configured port in .env.

**6. Component Documentation**

* **Controllers**: Authenticate user, manage playlists, handle track playback.
* **Models**: User, Track, Playlist schemas.
* **Routes**: /api/login, /api/register, /api/tracks, /api/playlists, /api/play.
* **Reusable Functions**: Token verification, input validation.

**7. User Interface**

* The backend provides RESTful APIs; a separate frontend (e.g., React) can consume these endpoints. Screenshots/UI details apply if a frontend exists.

**8. Styling**

* Not applicable to backend. For frontend, document any CSS frameworks if present.

**9. Testing**

* **Testing Strategy**: Mocha and Chai used for API endpoint tests. Mock data is utilized for unit tests on business logic.
* **Code Coverage**: Istanbul (nyc) tracks test coverage.

**10. Screenshots or Demo**

* [Include screenshots of API responses or provide a demo link if available.]

**11. Known Issues**

* [List known issues, e.g., limited track support, database scaling limits, etc.]

**12. Future Enhancements**

* Add social sharing features
* Integrate third-party music APIs
* Expand analytical and recommendation features
* Provide public playlists and collaborative editing