Frontend Development with React.js

Project Documentation for Rhythmic Tunes

**1.Introduction**

* Project Title: **Rhythmic Tunes**
* Team Members:
* **Mounika M** **(Team Leader)**

[Email Id: [mounika.m5007@gmail.com](mailto:mounika.m5007@gmail.com) ]

* **Swathi N**

[Email Id: [swathiswathi63165@gmail.com](mailto:swathiswathi63165@gmail.com) ]

* **Tamizharasi M**

[Email Id: [tamizhm286@gmail.com](mailto:tamizhm286@gmail.com) ]

* **Madhumitha M**

[Email Id: [mmadhumitha1905@gmail.com](mailto:mmadhumitha1905@gmail.com) ]

**2. Project Overview**

* **Purpose:**
* Rhythmic Tunes is a web application designed to provide users with a seamless music listening experience. The application allows users to search, play music tracks, create playlists, and discover new music based on their preferences.
* **Features:**
* Music player with play, pause, skip, and volume control.
* Search functionality to find songs, albums, and artists.
* User authentication (login/signup).
* Playlist creation and management.
* Responsive design for mobile and desktop.

**3. Architecture**

* **Component Structure:**

The application is built using React.js with a component-based architecture.

**Major components include:**

* **Header:** Contains the navigation bar and search bar.
* **Player:** Music player controls (play, pause, volume, etc.).
* **Sidebar:** Displays user playlists and navigation links.
* **Home Page:** Display featured tracks, recommended playlists, and new releases.
* **Search page:** Allows users to search for songs, albums, and artists.
* **Playlist Page:** Displays user-created playlists and allows playlists management.
* **State Management:**

The application uses **Redux** for global state management. The Redux store manages user authentication, current playing track, playlist data, and search results.

* **Routing:**

The application uses **React Router for navigation.**

**Routes include:**

* /: Home page
* /search: Search page
* /playlist/:id: playlist detail page
* /login: User login page

**4.Setup Instructions**

* **Prerequisites:**
* Node.js (v16 or higher)
* npm (v8 or higher)
* Git
* **Installation:**

1. Clone the repository: git clone <https://github.com/Swathinadesan/RHYTHMICTUNES>
2. Navigate to the client directory: cd rhythmic-tunes/client
3. Install dependencies: npm install
4. Configure environment variables: Create a .env file in the directory and add the necessary variable (e.g., API keys )
5. Start the development server: npm start

**5.Folder Structure**

* **Client:**
* **src/components:** # Reusable components (Header, player, etc.)
* **src/pages: #** Page Components (HomePage, SearchPage,etc.)
* **src/assets: #** Images, icons and other static files.
* **src/redux: #** Redux store, actions, and reducers.
* **src/utils: #** Utility functions and helpers.
* **App.js: #** Main application component.
* **Index.js: #** Entry point.
* **Utilities:**

* **api.js:** Handle API requests to the backend.
* **Auth.js:** Manages user authentication and token storage.
* **Hook/usePlayer.js:** Custom hook for managing the music player state.

**6.Running the Application**

**Frontend:**

* To start the frontend server, run the following command in the client directory:
* npm start
* npm install
* npx json-server ./ db/db.json
* npm run dev
* The application will be available at <http://localhost:5173>

**7.Component Documentation**

**Key Components:**

* **Header:** Display the navigation bar and search bar.
* **Props:** Onsearch (function to handle search queries).
* **Player:** Controls the muic playback
* **Prop:** CurrentTrack (Object containing track details), onplay, onpause and onskip.
* **PlaylistCard:** Display a palylist with its name and cover image.
* **Props:** Playlist (Object containing playlist deatails), onclick (function to handle palylist selection).

**Reusable Components:**

* **Button:** A customizable button component.
* **Props**: text, onclick, disabled.
* **Input:** A reusable input field for forms and search.
* **Props:** Type, placeholder, value, onchange.

**8.State Management**

* **Global State:**

The Redux store manages the following global states:

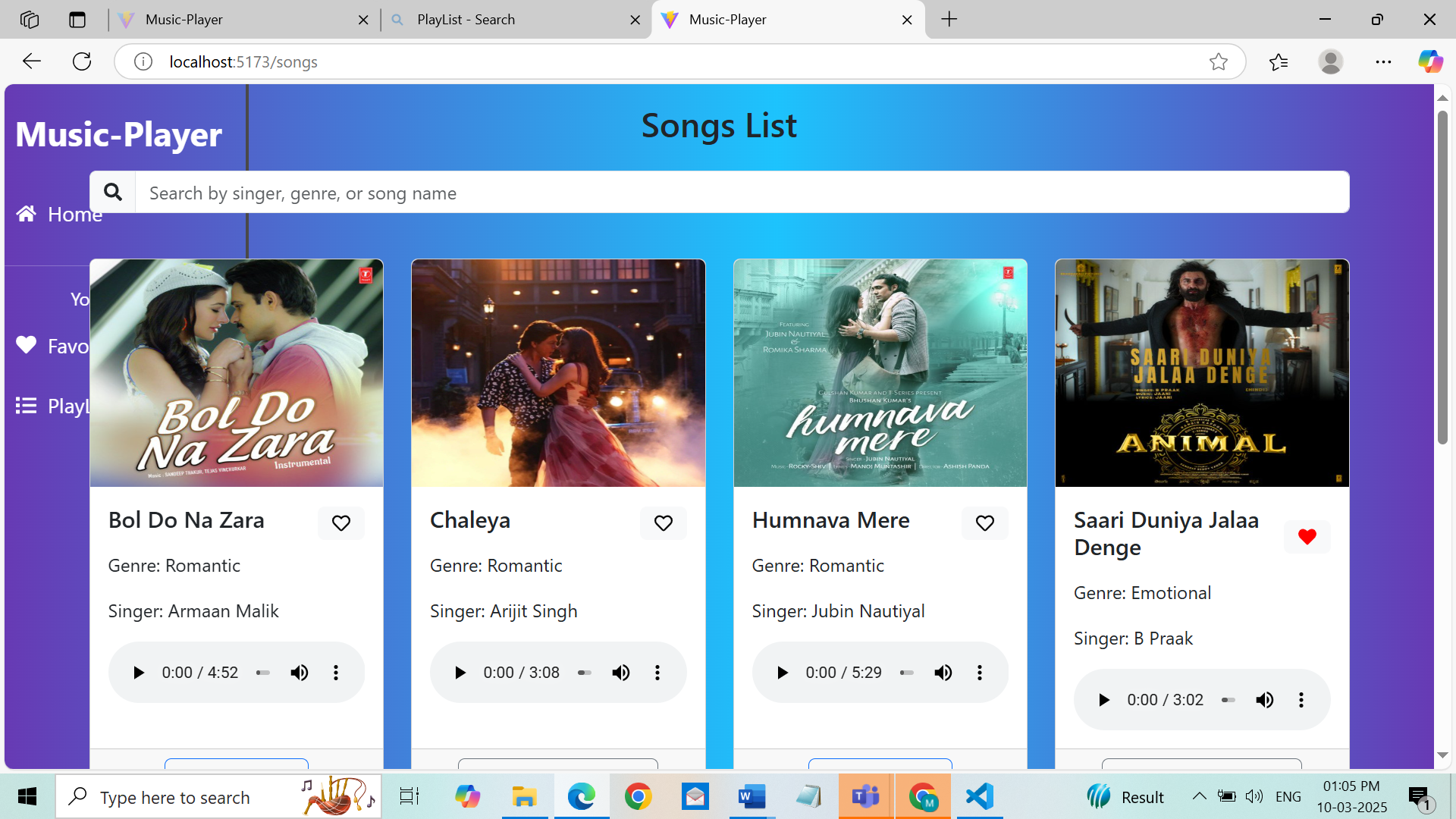
* **User:** Current authentication user.
* **Player:** Current playing track, playback status (playing/pause) and volume.
* **Playlists:** User-Created playlists.
* **Search Results:** Results from the search functionality.
* **Local State:**

Local state is managed using React’s usestate hook within components. For example, the search component manages the search query input locally.

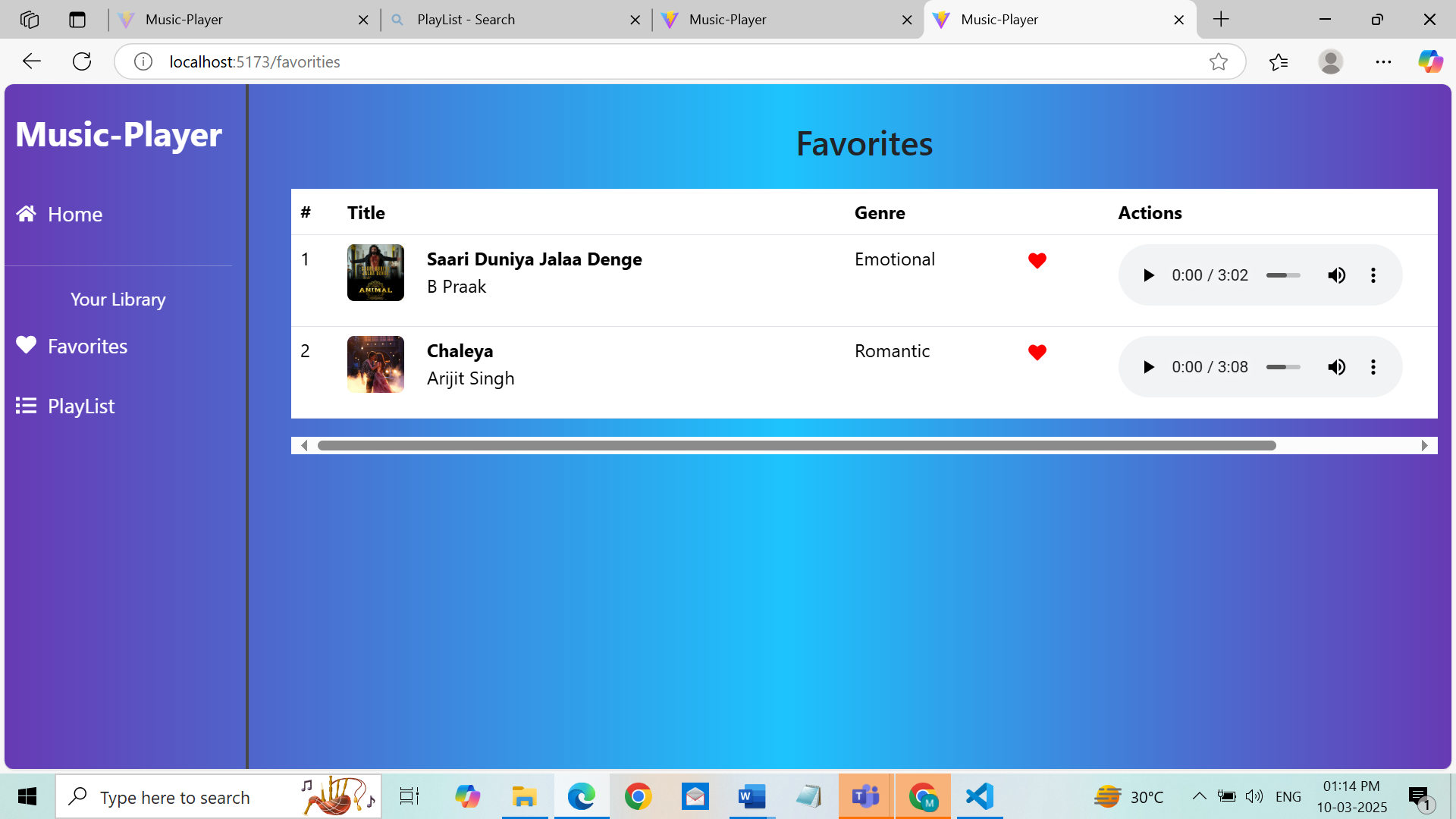
**9.User Interface:**

**Screenschot**

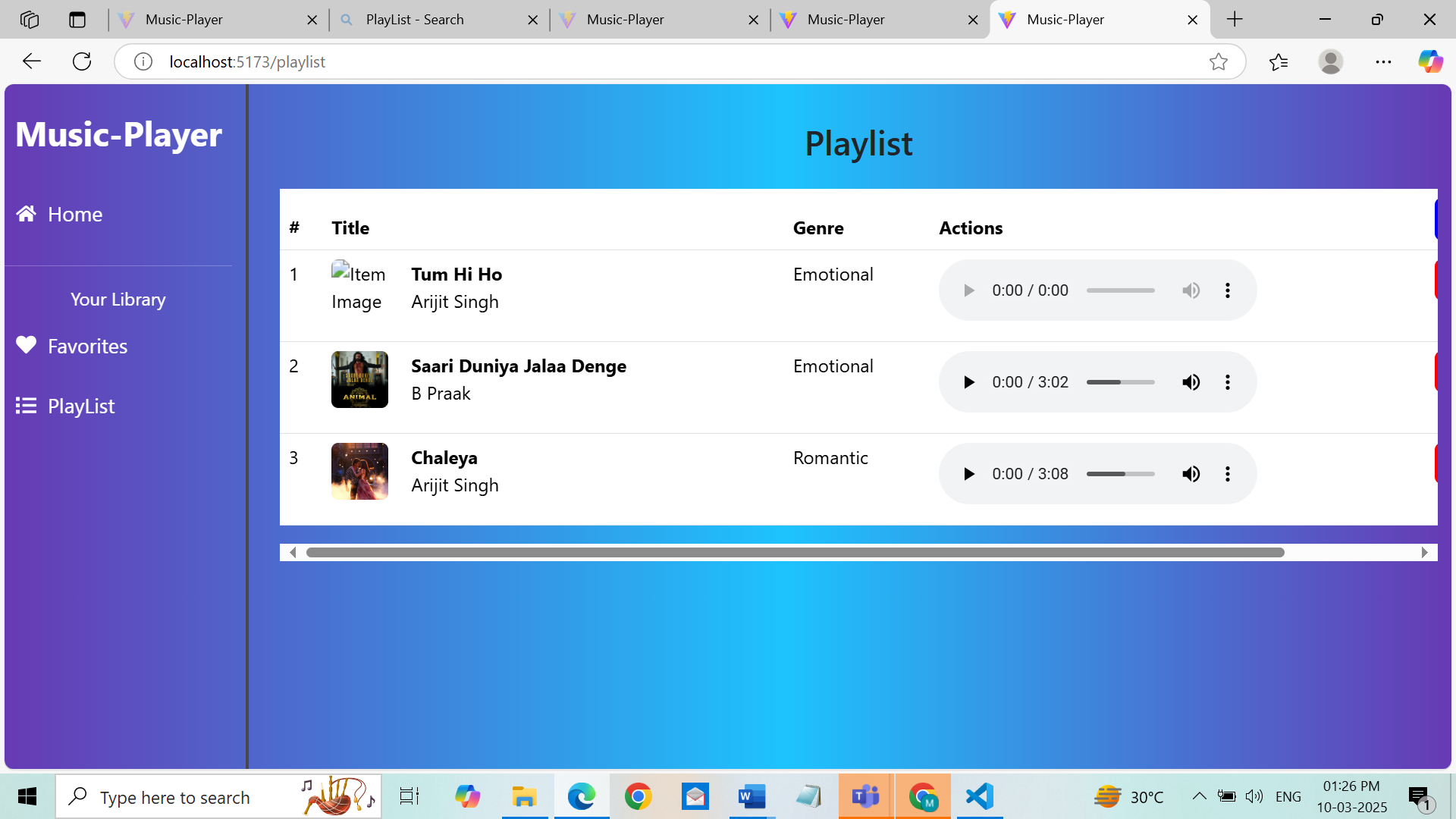
* **Home page:** Display featured tracks and recommended playlists.



* **Search Page:** Allow users to search for songs, albums and artists.



* **Playlist page:** Display user-created playlists and allows playlist management.



**10.Styling**

**CSS Frameworks/Libraries:**

The application uses **Styled-Components** for styling. This allows for modular and scoped CSS within components.

* **Theming:**

A custom theme is implemented using Styled-components,with support for light and dark modes.

**11.Testing**

* **Testing Strategy:**

* **Unit Testing:** Using **Jest** and **React Testing Library.**
* **Integration Testing:** It is performed to ensure that components work together as expected.
* **End-to-End Testing: Cypress** is used for end-to-end testing of user flows.
* **Code Coverage:**
* Code coverage is monitored using Jest’s builtin coverage tool. The current coverage is 85%.

**12.Screenshots or Demo**

* Demo Link:

<https://drive.google.com/file/d/1a-0we4qY4zsKK-N6gy-4N7cmTBcIDIQI/view?usp=sharing>

* **Screenshots:** See section 9 for UI screenshots.

**13.Future Issues**

* **Issue 1:** The music player sometimes skips tracks unexpectedly.
* **Issue 2:** The search functionality is slowly with large datasets.

**14.Future Enhancements**

* **Future Features:**
* Add support for user profiles and social sharing.
* Implement a recommendation engine for personalized music suggestion.
* Add animations and transitions for a smoother user experience.

This documentation provides a comprehensive overview of the **Rhythmic Tunes** project,

Including its architecture, setup instructions and future plans.