***Frontend Development with React.js Project Documentation for Rhythmic Tunes***

* **Introduction**
* **Project Title**: 
* **Team Members**:

 **Shreeja M** (**Team Leader**) [Email Id: [shreejam036@gmail.com](mailto:shreejam036@gmail.com)]

 **Keerthika N** [Email Id: [keerthikanagalingam28@gmail.com](mailto:keerthikanagalingam28@gmail.com)]

 **Devi Sri S** [Email Id: [devisri020920031@gmail.com](mailto:devisri020920031@gmail.com)]

 **Harini VB** [Email Id: [vbharini03@gmail.com](mailto:vbharini03@gmail.com)]

* **Project Overview**
* **Purpose**:

Rhythmic Tunes is a web application that offers users a smooth and enjoyable music listening experience. It enables users to explore, search for, and play music tracks, create custom playlists, and discover new music tailored to their tastes.

* **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.
* **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.
* **HomePage**: Displays featured tracks, recommended playlists, and new releases.
* **SearchPage**: Allows users to search for songs, albums, and artists.
* **PlaylistPage**: Displays user-created playlists and allows playlist management.
* **State Management**:

The application utilizes Redux for global state management. The Redux store handles user authentication, the currently playing track, playlist information, and search results.

* **Routing**:

The application uses **React Router** for navigation. Routes include:

* /: Home page
* /search: Search page
* /playlist/:id: Playlist details page
* /login: User login page
* **Setup Instructions**
* **Prerequisites**:
* Node.js (v16 or higher)
* npm (v8 or higher)
* Git
* **Installation**:
* Clone the repository: git clone <https://github.com/shree1320/RHYTHMIC-TUNES>
* Navigate to the client directory: cd code
* Install dependencies: npm install
* Configure environment variables: Create a .env file in the client directory and add the necessary variables (e.g., API keys).
* Start the development server: npm start
* **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**: Handles API requests to the backend.
* **auth.js**: Manages user authentication and token storage.
* **hooks/usePlayer.js**: Custom hook for managing the music player state.
* **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:3000](http://localhost:3000/)
* **Component Documentation**
* **Key Components**:
* **Header**: Displays the navigation bar and search bar.
* Props: onSearch (function to handle search queries).
* **Player**: Controls the music playback.
* Props: currentTrack (object containing track details), onPlay, onPause, onSkip.
* **PlaylistCard**: Displays a playlist with its name and cover image.
* Props: playlist (object containing playlist details), onClick (function to handle playlist 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.
* **State Management**
* **Global State**:

The Redux store manages the following global states:

* **user:** Current authenticated user.
* **player:** Current playing track, playback status (playing/paused), and volume.
* **playlists:** User-created playlists.
* **searchResults:** Results from the search functionality.
* **Local State**:

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

* **User Interface**
* **Screenshots**
* **Home Page:** Display featured tracks and recommended playlists.



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



* **Playlist Page:** Displays user-created playlists and allows playlist management.



* **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.

* **Testing**
* **Testing Strategy**:
* **Unit Testing:** Using **Jest** and **React Testing Library**.
* **Integration Testing**: 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 built in coverage tool. The current coverage is 85%.
* **Screenshots or Demo**
* **Demo Link:**

[**https://drive.google.com/file/d/1hM3SDe85IIxsvh84Vd3\_RDBIiD\_aoJzy/view?usp=sharing**](https://drive.google.com/file/d/1hM3SDe85IIxsvh84Vd3_RDBIiD_aoJzy/view?usp=sharing)

* **Screenshots:** See section 9 for UI screenshots.
* **Known Issues**
* **Issue 1**: The music player sometimes skips tracks unexpectedly.
* **Issue 2**: The search functionality is slow with large datasets.
* **Future Enhancements**
* **Future Features**:
* Add support for user profiles and social sharing.
* Implement a recommendation engine for personalized music suggestions.
* 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.