**Rhythmic Tunes: Your Melodic Companion**

**1. Introduction**

**Team Members**

**SANDHIYA G (TEAM LEADER)**

**NAKSHATRAA S**

**NIVETHA M**

**PAVITHRA D**

**2. Project Overview**

**Purpose:**

**Rhythmic Tunes** is a web application designed to offer music enthusiasts a personalized and interactive experience in discovering, listening to, and sharing music. The platform aggregates music from various genres, provides search and filtering functionalities, and offers recommendations based on user preferences and listening history.

**Features:**

* **Music Aggregation:** Access to a wide variety of music from different artists and genres.
* **Genre-based Filtering:** Users can filter music based on preferred genres, moods, or themes.
* **User Customization:** Personalized playlists and favorite tracks.
* **Interactive Dashboards:** Visual analytics of listening history, favorite tracks, and more.
* **Cross-platform Support:** Mobile-responsive and desktop-ready design.

**3. Architecture**

**Component Structure**

* **App.js:** Main entry point that manages routing and layout components.
* **Header:** Navigation bar with links to home, search, playlists, and user profile.
* **Music Feed:** Displays aggregated music recommendations with filtering options.
* **Song Card:** Individual music track component with metadata like title, artist, and album art.
* **Search Bar:** Allows users to search tracks by song title, artist, or genre.
* **User Preferences:** Manages user settings, favorite genres, and customized playlists.

**State Management**

* The application uses **Context API** for global state management. User preferences, theme settings, and API data are shared across components using context providers.

**Routing**

* Implemented using **React Router v6** with routes defined for:
  + **/ –** Home page with music feed
  + **/search –** Music search page
  + **/playlists –** User playlists page
  + **/profile –** User profile and preferences

**4. Setup Instructions**

**Prerequisites:**

* **Node.js** version 18 or above
* **npm** version 9 or above

**Installation:**

https://github.com/sknakshatraa-cyber/rhythmic-tune.git

1. Clone the repository:
2. git clone https://github.com/yourusername/rhythmtunes.git
3. Navigate to the client directory:
4. cd rhythm tunes/client
5. Install dependencies:
6. npm install
7. Create a **.env** file with necessary API keys (e.g., for music data provider):
8. REACT\_APP\_MUSIC\_API\_KEY=your\_api\_key\_here
9. Start the development server:
10. npm start

**5. Folder Structure**

client/

├── public/

├── src/

│ ├── components/

│ │ ├── SongCard.js

│ │ ├── Header.js

│ │ ├── MusicFeed.js

│ │ └── SearchBar.js

│ ├── context/

│ │ └── AppContext.js

│ ├── pages/

│ │ ├── Home.js

│ │ ├── Playlists.js

│ │ ├── Search.js

│ │ └── Profile.js

│ ├── utils/

│ │ └── api.js

│ ├── App.js

│ ├── index.js

├── .env

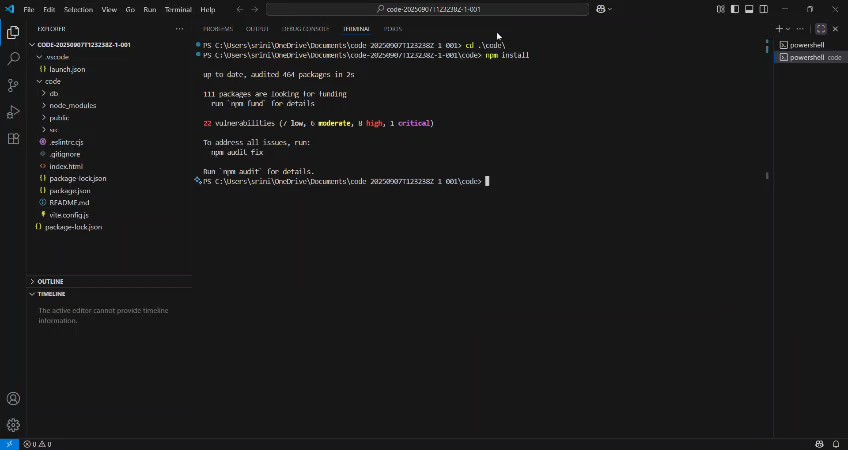
├── package.json

└── README.md

**Utilities**

* **api.js:** Handles API requests to music service providers.
* **Custom Hooks:**
  + Use Fetch Music: Fetches music tracks and metadata.
  + Use Theme: Manages the user interface theme (light/dark mode).

**6. Running the Application**

1. Navigate to the client directory and run the following command:
2. npm start
3. ****

This will launch the application locally at http://localhost:3000/.

**7. Component Documentation**

**Key Components:**

* **Music Feed:** Displays music recommendations based on user preferences and browsing history.
  + **Props:** filter, search Query
* **Song Card:** Displays an individual song's title, artist, album art, and play controls.
  + **Props:** song (object with song metadata)
* **User Preferences:** Allows users to manage their music preferences, like favorite genres or artists.
  + **Props:** None

**Reusable Components:**

* **Search Bar:** Handles keyword search input for music tracks.
  + **Props:** on Search callback function.
* **Header:** Navigation bar used across all pages.
  + **Props:** None

**8. State Management**

**Global State:**

* Managed using **Context API**.
  + Stores: User preferences, theme data, and music metadata from API responses.

**Local State:**

* Managed using **use State** inside components like:
  + **Search Bar:** Stores the search query input.
  + **Song Card:** Stores the number of likes or user interactions with a song.

**9. User Interface**

**Screenshots:**

