RhythmicTunes: Your Melodic Companion

1. **Introduction**

**Project Title:** **RhythmicTunes: Your Melodic Companion**

**Team ID: NM2025TMID41763**

**Team Leader: RETHANYA S**

**Mail id:** [**rethanyaa02@gmail.com**](mailto:rethanyaa02@gmail.com)

**Team Members:**

1. **RETHANYA S**– Lead Developer (Frontend & State Management)

**2. PAVITHRA K**– UI/UX Designer & Frontend Developer

**3. RITHIKA SREE S**– Testing & Quality Assurance

**4. ROHITHA P**– Documentation & Deployment Support

1. **Project Overview**

**Purpose:**  
The RhythmicTunes: Your Melodic Companion application is designed to enrich users’ musical experience by providing an intuitive and personalized platform for music discovery, playlist creation, and seamless playback. It empowers users to explore new genres, enjoy curated recommendations, and manage their favorite tracks effortlessly. With features like smart search, offline listening, cross-device synchronization, and AI-driven personalization, RhythmicTunes makes music more accessible, enjoyable, and tailored to individual preferences for casual listeners and music enthusiasts alike.

**Features:**

* **Music Library Management – Browse, add, and organize songs, albums, and playlists.**
* **Smart Search & Filters – Find tracks by artist, genre, mood, release year, or language.**
* **Personalized Playlists – AI-generated playlists based on listening habits and preferences.**
* **Offline Listening – Download and play music without an internet connection.**
* **Cross-Device Sync – Seamless access to playlists and favorites across multiple devices.**
* **Favorites & Recommendations – Save favorite songs and receive curated suggestions.**
* **Social Sharing – Share playlists and songs with friends or collaborate on playlists.**
* **Lyrics Integration – View synced lyrics and enjoy a karaoke-style experience.**
* **Voice Controls – Hands-free playback and navigation through voice commands.**
* **Responsive Design – Works smoothly on mobile, tablet, and desktop.**

1. **Architecture**

**Component Structure:**  
The application is structured with modular React components such as **MusicLibrary, Search & Filters, PlaylistManager, NowPlaying, Recommendations, UserProfile, and NavBar**. Each module handles a specific responsibility and interacts via props and context.

**State Management:**

* **Global State: Managed using Context API (MusicLibraryContext, PlaylistContext, NowPlayingContext, UserContext).**
* **Persistence: Implemented with a custom hook (useLocalStorage) to save playlists, favorites, playback settings, and user preferences.**
* **Local State: Controlled with React hooks (useState, useEffect) for handling search filters, playback progress, volume control, and UI interactions.**

**Routing:**  
Basic routing is implemented using **React Router**, including pages such as **MusicLibrary, Playlists, NowPlaying, Recommendations, and UserProfile**, along with a **NotFound** page for invalid routes.

1. **Setup Instructions**

**Prerequisites:**

* Node.js
* npm

**Installation Steps:**

1. Clone the repository:
2. git clone https://github.com/Rethanya-S-bot/RythmicTunes.git
3. Navigate to the project folder:
4. cd Rethanya-S-bot/RythmicTunes
5. Install dependencies:
6. npm install
7. Start the development server:
8. npm run dev
9. **Folder Structure**

**Client (React App):**

* **src/components – Reusable UI components (SongCard, PlaylistCard, NavBar, PlayerControls, etc.)**
* **src/pages – Pages such as MusicLibrary, PlaylistDetail, NowPlaying, Recommendations, UserProfile**
* **src/context – Global state providers (MusicLibraryContext, PlaylistContext, NowPlayingContext, UserContext)**
* **src/hooks – Custom hooks (for local storage, media playback, etc.)**
* **public/ – Static assets and base HTML**
* **tailwind.config.js – TailwindCSS configuration (or alternative styling system)**

1. **Running the Application**

To run the frontend locally:

npm start

The app will be available at [**http://localhost:3000**](http://localhost:3000) in your browser.

1. **Component Documentation**

**Key Components:**

* **MusicLibrary – Displays all available songs, albums, and genres.**
* **PlaylistDetail – Shows playlist contents, track order, and playback options.**
* **NowPlaying – Displays the current track, lyrics, progress bar, and player controls.**
* **Recommendations – Provides AI-driven song and playlist suggestions.**
* **UserProfile – Manages preferences, listening history, and saved playlists.**
* **NavBar – Enables navigation between library, playlists, now playing, recommendations, and profile.**

**Reusable Components:  
SongCard, PlaylistCard, PlayerControls, and SearchBar are reusable across different contexts.**

1. **State Management**

* **Global State:** Managed using **Context API** (MusicLibraryContext, PlaylistContext, NowPlayingContext, UserContext).
* **Local State:** Controlled with **React hooks** (useState, useEffect) for handling search filters, playback progress, volume, and UI interactions.
* **Persistence:** Implemented with a custom hook (**useLocalStorage**) to save playlists, favorites, playback settings, and user preferences.

1. **User Interface**

The UI is designed to be intuitive, responsive, and visually engaging, built with **TailwindCSS** for consistency across devices. Key screens include:

* **Home Dashboard** – Quick access to featured playlists, favorites, and recommendations.
* **Music Library** – Displays songs, albums, and genres with filtering and search options.
* **Playlist Detail** – View playlist contents, manage tracks, and start playback.
* **Now Playing** – Full-screen playback view with track info, lyrics, progress bar, and controls.
* **Recommendations** – Personalized suggestions and AI-driven playlists.
* **User Profile** – Manages preferences, listening history, and saved playlists.

1. **Styling**

 **CSS Framework**: TailwindCSS is used for rapid UI development and responsive design.

 **Theming**: Custom Tailwind configurations define colors, typography, and spacing to create a clean, modern, and food-inspired aesthetic.

1. **Testing**

 **Testing Strategy:** Unit and integration testing are implemented using **Jest** and **React Testing Library** to ensure component reliability. Critical features such as music playback, playlist management, search, and recommendations are thoroughly tested.

 **Code Coverage:** The default CRA testing setup provides baseline coverage, while additional custom test cases ensure that playback controls, user preferences, and playlist handling achieve high coverage levels.

1. **Screenshots or Demo**

(Screenshots of the UI — home dashboard, music library, playlist detail, now playing screen with lyrics, and recommendations.)

A screenshot of a computer

AI-generated content may be incorrect.

1. **Known Issues**

* Data persistence is limited to local storage; no cloud sync is available.
* No backend integration for multi-device access or real-time updates.
* Limited authentication and user role management.
* Music recommendations are basic and not fully personalized.
* Dependency on third-party APIs for content and licensing.

1. **Future Enhancements**

* Add backend API integration for cloud storage, real-time sync, and multi-device access.
* Implement advanced personalization with AI-powered music recommendations.
* Introduce social features such as collaborative playlists and sharing options.
* Enable lyrics integration with karaoke and synchronized visualizations.
* Support user roles, profiles, and community-driven playlists.
* Add voice-assisted controls for hands-free navigation and playback.
* Provide offline playlist export/import functionality for portability.