**RHYTHMIC TUNES – YOUR MELODIC COMPANION PROJECT DOCUMENTATION**

This documentation provides a comprehensive guide to understanding, setting up, and using the Rhythmic Tunes project. It includes details about the project structure, architecture, installation steps, running instructions, and future improvements.

**1. Introduction:**

**Project Title:**

Rhythmic Tunes: Your Melodic Companion

Team Leader: S.Abirami (abiramisiva1408@gmail.com)

Team Member: N.K.Bhavani ( nkmagesh05@gmail.com)

Team Member: A.Dharani (dhara.a9962@gmail.com)

Team Member: K.Dhivyasree (kdhivyasree26@gmail.com)

Team Member: J.Sharmila (kalyanijeeva31@gmail.com)

**2. Project Overview:**

**Purpose:**

The Rhythmic Tunes is a web-based music player application designed using React.js. It provides an intuitive interface for users to play, pause, speed, and manage playlists efficiently.

**Features:**

* User-friendly interface: A simple and attractive UI for playing music.
* Playlist management: Users can add or remove songs from their playlists.
* Playback controls: Play, pause, speed, and adjust volume.
* Responsive design: Works on different screen sizes.
* State management: Uses React’s built-in state or Context API to handle song states.

**3. Architecture:**

**Component Structure:**

The application follows a component-based architecture. The major components are:

* App.jsx: The main component that includes the overall structure.
* PlayerControls.jsx: Manages play, pause, next, previous, and volume controls.
* Playlist.jsx: Displays and manages playlists.
* SongItem.jsx: Represents an individual song in the list.

**State Management:**

The application handles state management in two ways:

* Local State: Manages UI elements like buttons, current song, and volume settings.
* Global State: If required, React Context API or Redux can be used to manage the overall application state.

**Routing:**

If the application has multiple pages (e.g., Home, Playlist, Settings), React Router can be used for navigation.

**4. Setup Instructions:**

**Pre-requisites:**

* Ensure the following software is installed on your system:
* Node.js (for running JavaScript on the backend)
* npm (Node Package Manager) or yarn (for managing dependencies)

**Installation Steps:**

**Step1**: Clone the repository:

git clone: <https://github.com/asunm13332213331026004/music-streaming-app>

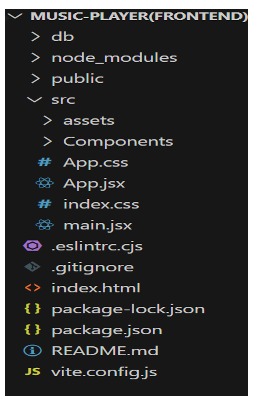
**Step2:** Navigate into the project directory:

cd music-player-frontend.

**Step3:** Install required dependencies:

npm install.

**5. PROJECT STRUCTURE:**



**6. Running the Application:**

To start the development server, run the following command:

* npm start
* This will launch the application on http://localhost:3000

**7. Component Documentation:**

**Key Components:**

App.jsx: The main component that controls the application flow.

**Components:**

* PlayerControls.jsx: Handles play, pause, next, and volume adjustments.
* Playlist.jsx: Displays and manages user playlists.
* SongItem.jsx: Represents a single song entry.

**Reusable Components:**

Any component that can be used multiple times across different parts of the app (e.g., buttons, song cards).

**8. State Management:**

**Global State Management:**

Context API or Redux can be used to store and share state across components, such as the current playing song.

**Local State Management**:

Each component has its own useState() hooks to manage UI behaviors (e.g., play/pause, volume).

**9. User Interface:**

**Screens and UI Features**:

Home Page: Displays the list of available songs.

Playlist Page: Allows users to create and manage playlists.

Player Controls: Floating music player with play/pause, speed, and volume controls.

**10. Styling:**

**CSS Frameworks/Libraries**:

Tailwind CSS or Bootstrap (if used) for styling.

Sass or Styled Components for advanced theming.

**Theming:**

If the application supports light mode/dark mode, mention how it’s implemented.

**11. Testing:**

**Testing Strategy:**

**Unit Testing:**

Test individual components using Jest or React Testing Library.

**Integration Testing:**

Test how components interact.

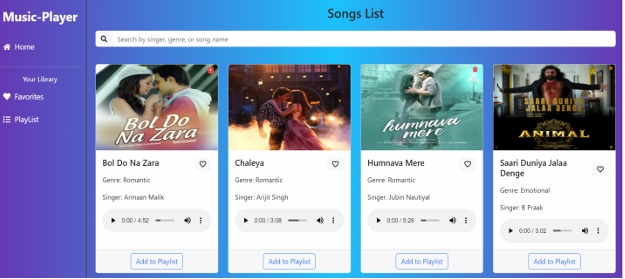
**End-to-End Testing**:

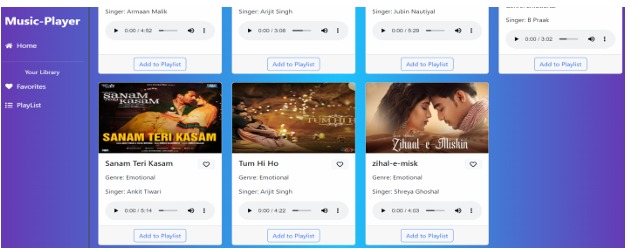
Use Cypress to test full user flows.

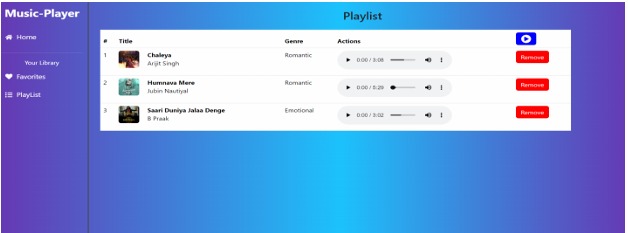
**Code Coverage**:

Measure code quality using tools like Jest coverage reports.

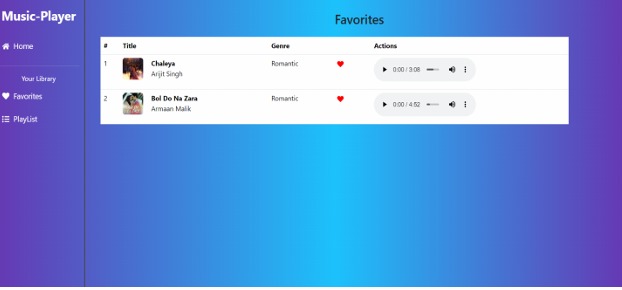
**12. Screenshots:**

**Songs Lists: **

****

Playlist: 

Favourites:



**13. Known Issues:**

List of existing bugs or known issues users should be aware of.

**Example:**

"Volume slider does not work on mobile devices."

"Playback controls may be slow when switching songs quickly."

**14. Future Enhancements:**

**Planned Features:**

* Offline playback support (allow users to save songs for offline listening).
* Lyrics display (fetch and display song lyrics).
* Animations and improved UI for better user experience.
* Social sharing (share favorite songs/playlists with others).
* More themes and color customization options.

**15. Project Demo Link:**

**<https://drive.google.com/file/d/1XJqOj4iHo4wlMduHTMkZLLkKJQhlmmzl/view?usp=drivesdk>**

**16. Conclusion:**

This documentation provides a structured overview of the Rhythmic Tunes project. It includes setup instructions, component descriptions, state management details, and future improvements.

\*\*\*Happy Coding\*\*\*