**RHYTHMIC TUNES**

YOUR MELODIC COMPANION

**(MUSIC STREAMING APPLICATION)**

**TEAM MEMBERS**

J.MEGAVARSHENE

NANDHINI N

LAVANYA B

MADHU K

**Frontend Development with React.js**

**Project Documentation**

**Introduction**

Rhythmic Tunes is a feature-rich, React-based **music streaming application** designed to enhance the way users discover, listen to, and manage their favourite tracks. The application provides a **seamless** and **personalized** music experience with features such as playlist creation, offline listening, real-time lyrics, and AI-powered recommendations. Built with **React.js, Node.js, and modern web technologies**, it ensures a **smooth and interactive user experience** across all devices. Rhythmic Tunes aims to redefine how users **engage with music**, offering both **casual listeners** and **music enthusiasts** an all-in-one platform for an immersive auditory experience

**Project Title:** Rhythmic Tunes-Your Melodic Companion

**Team Leader: J.Megavarshene**

**Team Member: Nandhini , Lavanya , Madhu**

**Project Overview**

**Purpose:** Rhythmic Tunes is a cutting-edge music streaming application built using React.js. Designed for modern music enthusiasts, this platform offers seamless navigation, an intuitive interface, and immersive music experiences.

**Features:**

* Song Listings with detailed track information
* Playlist Creation with personalized customization
* Playback Control for enhanced music enjoyment
* Offline Listening for improved accessibility
* Advanced Search Functionality to find desired tracks with ease

**Architecture**

**Component Structure:**

* **App Component:** Root of the application, handles layout and routing.
* **Sidebar Component:** Contains navigation links for quick access.
* **Songs Component:** Displays a list of available songs with search options.
* **Playlist Component:** Manages user-generated playlists.
* **Favorites Component:** Tracks and manages favorite songs.

**State Management:**

* Utilizes React's **useState** and **useEffect** hooks for state control.
* Includes context management to handle global states like playlists and favorites.

**Routing:**

* Implemented using **React Router Dom** for seamless navigation between:
  + / (Home)
  + /favorites (Favorites)
  + /playlist (Playlist)

**Setup Instructions**

**Prerequisites:**

* Node.js and npm
* React.js
* JSON Server for mock data

**Installation:**

1. Clone the repository
2. Run npm install to install dependencies
3. Start the development server with npm run dev
4. Launch JSON Server with json-server --watch ./db/db.json

**Folder Structure**

* **/src**
  + **/components**: Contains reusable components
  + **/pages**: Includes major pages like Songs, Favorites, and Playlist
  + **/assets**: Houses static files (images, audio)
  + **/utils**: Includes custom hooks and utility functions

**Running the Application**

* Run the frontend using: npm run dev
* Run the JSON Server for mock data: json-server --watch ./db/db.json

**Component Documentation**

**Key Components:**

* **Songs Component:** Displays song listings with search functionality and player controls.
* **Playlist Component:** Manages adding/removing songs in the playlist.
* **Favorites Component:** Tracks favorite songs with visual icons for easy interaction.

### User Interface and Styling

**UI Design:**

* The user interface follows a clean, modern aesthetic with minimal clutter.
* Song cards are displayed in a grid layout for easy browsing.
* The color palette blends soothing gradients with vibrant highlights for dynamic visuals.

**Styling**

* Uses **Bootstrap** and **Custom CSS** for a responsive and visually appealing UI.
* Custom class names ensure flexible styling and improved readability.

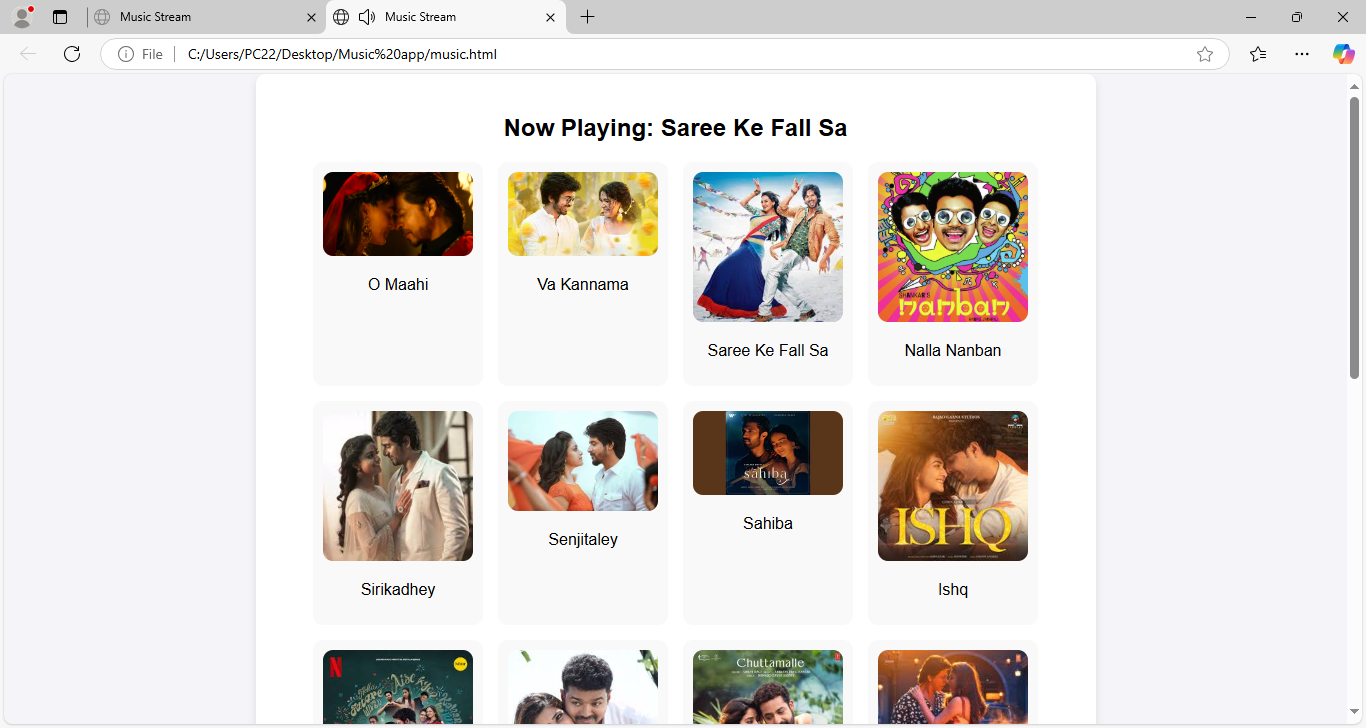
**Testing**

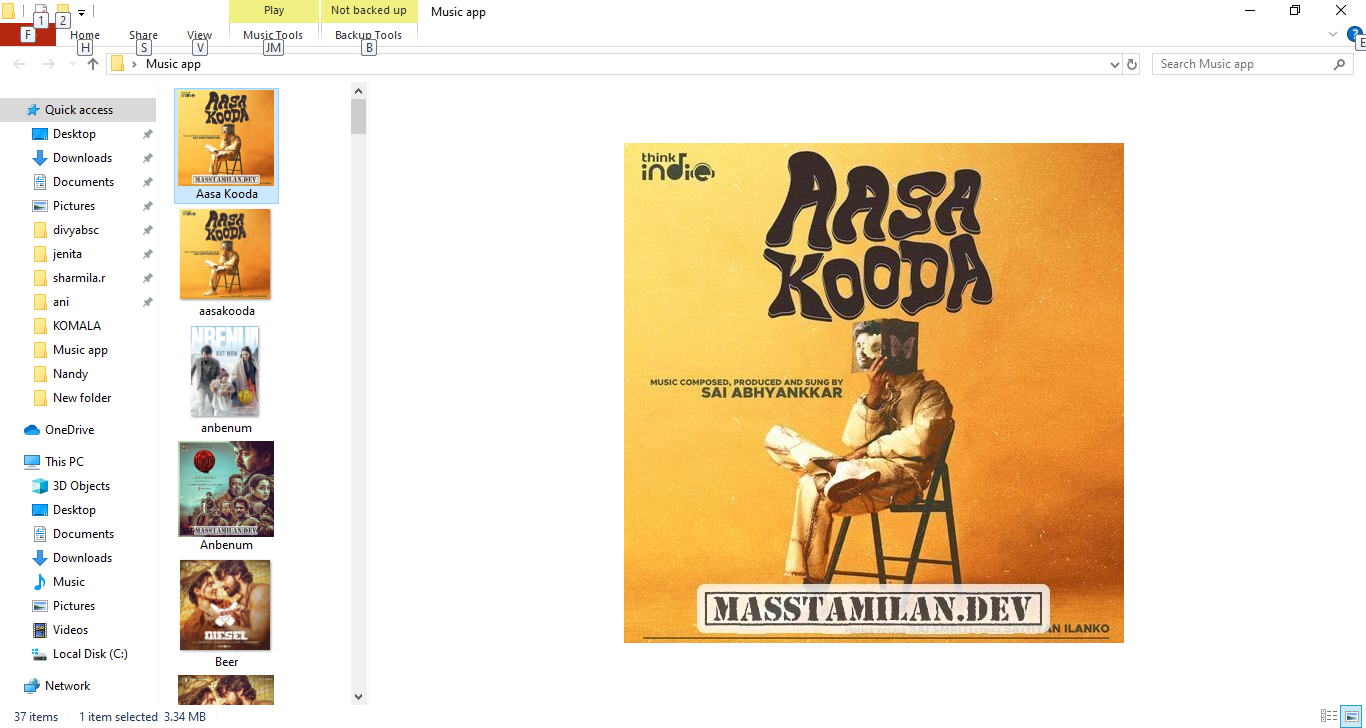
**Testing Strategy:**

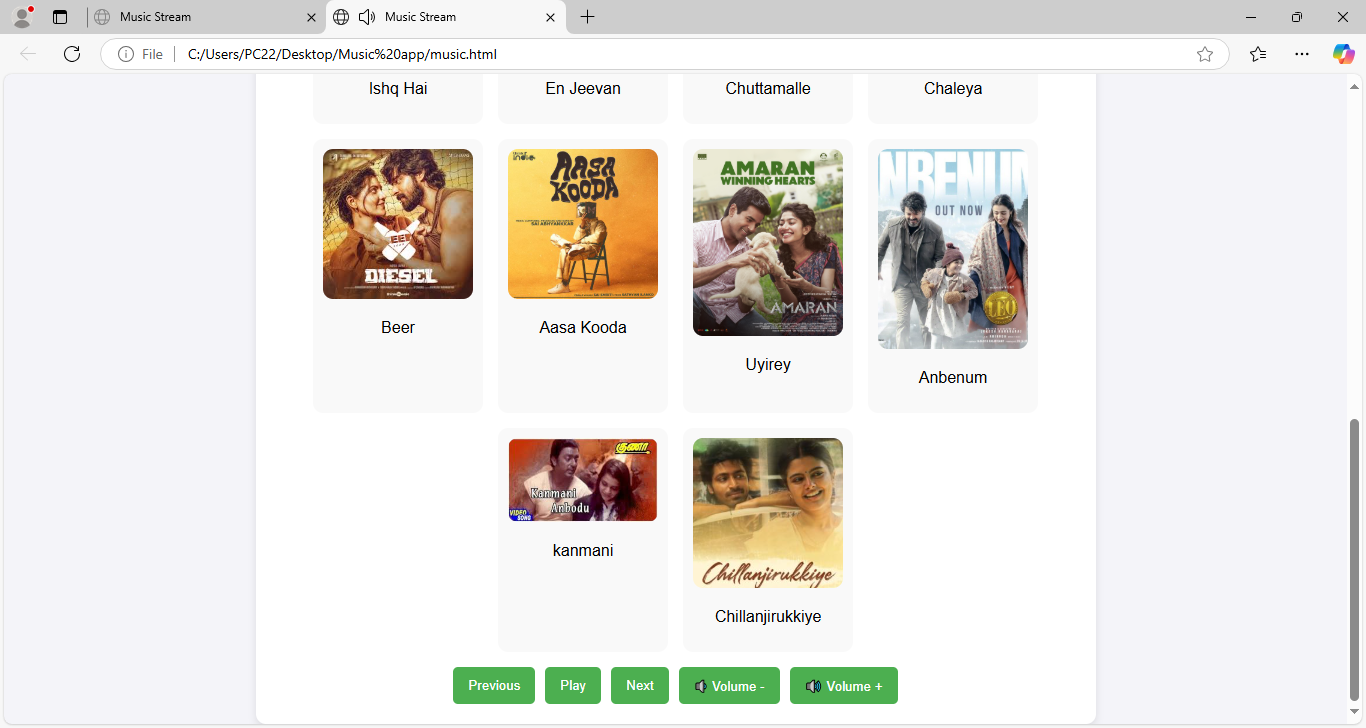
* Uses **Jest** and **React Testing Library** for unit and integration tests.
* Ensures major features like search, playlist management, and playback functionality are thoroughly tested.

**Screenshots or Demo**

1. **Dashboard:**







**Known Issues**

* Occasional playback interruption when switching songs rapidly.
* Limited customization options for playlist themes (planned enhancement).

**Future Enhancements**

* Integration with Spotify API for a broader song database.
* Improved UI animations for smoother transitions.
* Enhanced social sharing features for playlists.