Frontend Development with React.js

Project Documentation format.

RhythmicTunes: Your Melodic Companion

# 1. Introduction

**Project Title**: RhythmicTunes: Your Melodic Companion   
**Team Members:**  
Team Leader Name: G SUBASREE [ [subasreesubasree671@gmail.com](mailto:subasreesubasree671@gmail.com) ]

Team Member Name: T KIRUTHIK VEL [ [tkiruthikvel@gmail.com](mailto:tkiruthikvel@gmail.com) ]

Team Member Name: S ABDUL RAHMAN [ [abdulrahmanrahman2497@gmail.com](mailto:abdulrahmanrahman2497@gmail.com) ]

Team Member Name: MADESH K [ [kmadesh109@gmail.com](mailto:kmadesh109@gmail.com) ]

# 2. Project Overview

**Purpose:**  
RhythmicTunes is a modern music streaming application designed to offer users an immersive listening experience. It provides a vast collection of songs, playlists, and personalized recommendations. The app ensures a seamless interface for browsing, discovering, and enjoying music.  
  
**Features:**  
- Browse Songs by Genre, Artist, or Album   
- Personalized Playlist Creation   
- Search Functionality with Real-Time Suggestions   
- Media Player with Play/Pause, Next/Previous, and Volume Control   
- Bookmark Feature for Saving Favorite Tracks   
- Light/Dark Mode for Improved User Experience   
- Responsive Design for Desktop and Mobile Devices

# 3. Architecture

**ComponentStructure**:  
**App.js** — Root component managing layout and routing   
**Header.js** — Navigation bar with search functionality   
**SongList.js**— Displays a list of available songs   
**SongCard.js**— Individual card showing song details and controls   
**Playlist.js**— Manages user-created playlists   
**MediaPlayer.js**— Provides audio playback controls   
**Footer.js**— Displays copyright and app links  
  
**State Management**  
**Context API** — Used for managing global states such as theme preferences, playlist data, and bookmarked songs   
 **Routing:**  
**React Router**— Implemented for seamless navigation between music categories, playlist views, and song details

# 4. Setup Instructions

**Prerequisites:**  
- Node.js (v18 or higher)   
- npm or yarn  
  
**Installation:**  
1. Clone the Repository:   
 ```bash  
 git clone https://github.com/darunraj0071/rhythmic-tunes.git  
 cd rhythmic-tunes  
 ```  
  
2. Install Dependencies:   
 ```bash  
 npm install  
 ```  
  
3. Create `.env` file (for API keys or environment variables):   
 ```env  
 REACT\_APP\_MUSIC\_API\_KEY=your\_api\_key\_here  
 ```  
  
4. Run the Application:   
 ```bash  
 npm start  
 ```

# 5. Folder Structure

/src   
 ├── /components   
 │ ├── Header.js   
 │ ├── SongList.js   
 │ ├── SongCard.js   
 │ ├── Playlist.js   
 │ ├── MediaPlayer.js   
 │ └── Footer.js   
 ├── /pages   
 │ ├── Home.js   
 │ ├── Discover.js   
 │ ├── Playlists.js   
 │ └── Favorites.js   
 ├── /assets   
 │ ├── images   
 │ ├── icons   
 │ └── styles   
 ├── /utils   
 │ ├── api.js   
 │ ├── helpers.js   
 │ └── constants.js   
 ├── App.js   
 ├── index.js

# 6. Running the Application

**Frontend Server Command:**  
```bash  
npm start  
```

# 7. Component Documentation

**Key Components:**  
**Header:** Provides navigation, search functionality, and theme toggling   
**SongList:** Displays categorized or searched songs in a structured format   
**SongCard:** Displays individual song details with play/pause control   
**MediaPlayer:** Controls audio playback, volume, and track navigation   
  
**Reusable Components:**  
**Button Component:** Used for actions like "Play," "Pause," or "Add to Playlist"   
**Modal Component:**Displays song details or playlist creation forms

# 8. State Management

**Global State:** Managed via Context API for bookmarks, playlists, and theme control   
**Local State:** Managed using React's `useState()` for track progress, user interactions, and dynamic updates

# 9. User Interface

- Clean, minimalistic UI designed for seamless navigation   
- Fully responsive layout with smooth transitions   
- Light/Dark mode toggle for personalized visual comfort

# 10. Styling

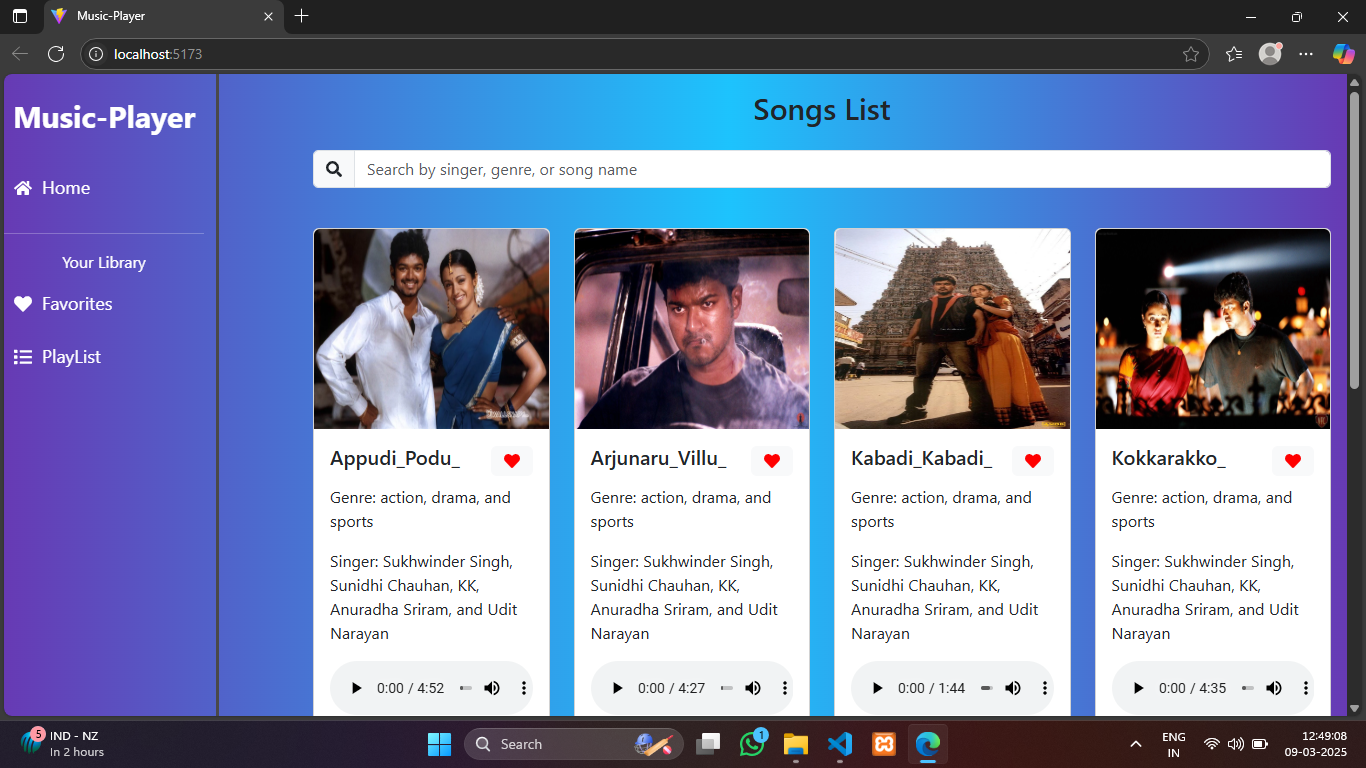
**CSS Frameworks/Libraries:**  
**Tailwind CSS** — Used for fast and modern UI design   
  
**Theming:**  
- Integrated theme switcher for enhanced user experience

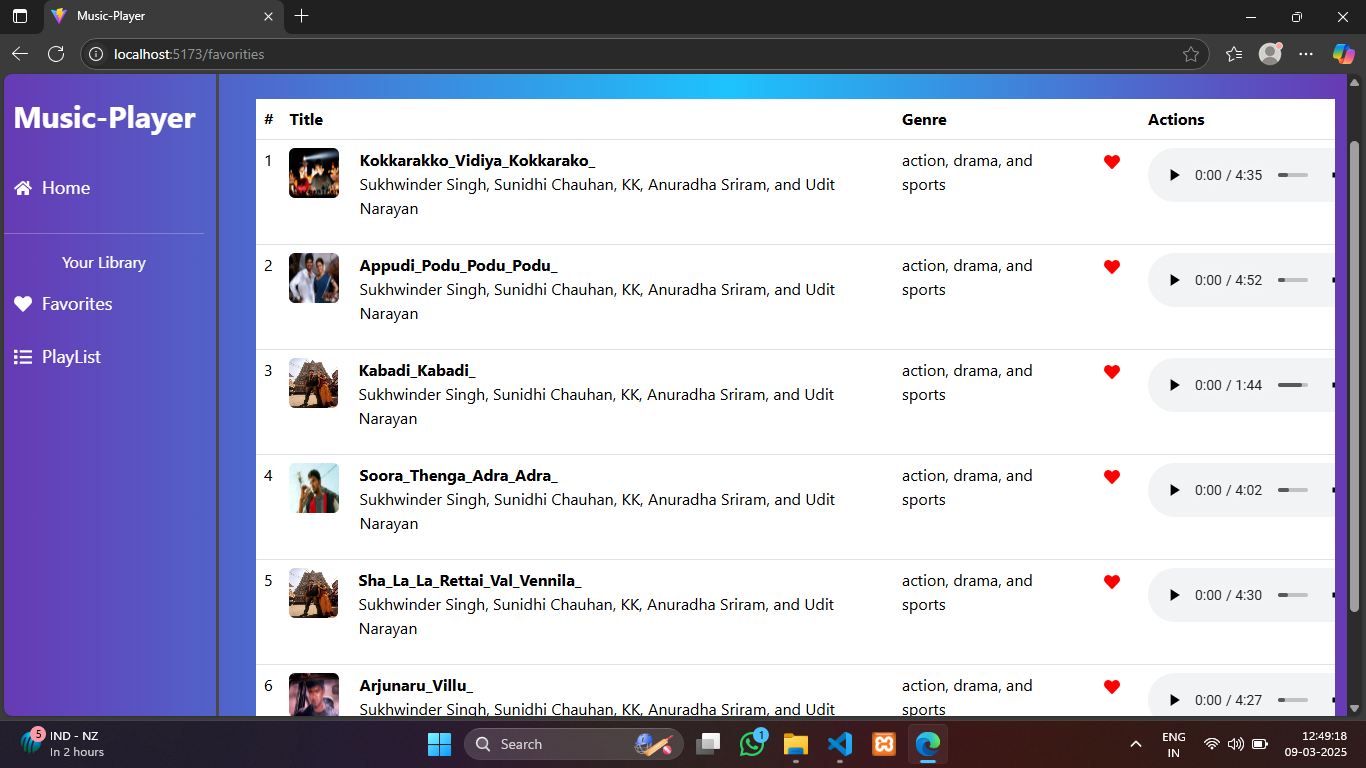
# 11. Testing

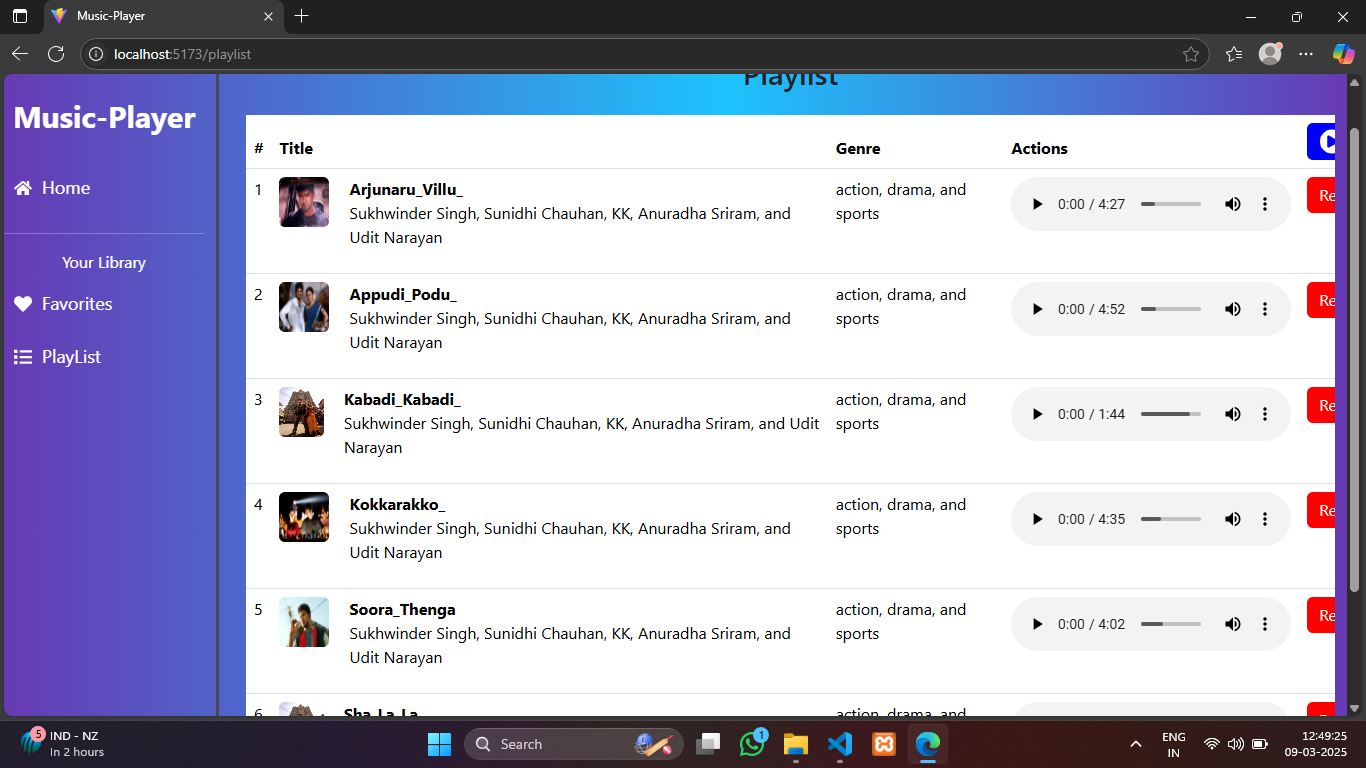
**Testing Strategy:**  
- Jest for unit testing   
- React Testing Library for comprehensive component testing   
  
**Code Coverage:**  
- Ensured high code coverage with Jest reports

# 12. Screenshots or Demo

**Screenshots:**







**Live Demo Link:**

**https://drive.google.com/drive/folders/1IcY7wJUsYrAyfpYE6QiM7fAJE2zfnsfN**

# 13. Known Issues

- Occasional delays in song loading due to API limits   
- Playlist feature may require additional optimization for large data sets

# 14. Future Enhancements

- Implement User Authentication for personalized playlists   
- Introduce Podcast and Radio Streaming Integration   
- Social Sharing Feature for tracks and playlists   
- Offline Mode for saved playlists without an internet connection