**Frontend Development with React.js**

**Project Documentation**

**Introduction**

Project Title: React News App -Rhythmic Tunes

**Team Members**:

Project Title: Rhythmic Tunes

Team members:

1. Kirankumar M (Team leader) Email id : Kk1442004@gmail.com

2. Udhayakumaran S Email id: sudhayakumaran167@gmail.com

3. Lakshmanan M Email id: mlakshmanan369@gmail.com

4. Santhosh K Email id: ksanthosh3355@gmail.com

### **2. Project Overview**

* **Purpose:** Rhythmic Tunes is a web application designed to offer users an engaging and intuitive music streaming experience. It allows users to explore music libraries, play tracks, manage playlists, and enjoy a personalized listening journey.

**Features:**

* Music player with play, pause, next/previous track, and volume controls.
* Search feature for discovering songs, artists, and albums.
* User authentication system (login/signup).
* Custom playlist creation and editing.
* Responsive design compatible with desktop and mobile devices.

### **3. Architecture**

* **Component Structure:** The application is built using React.js with a modular, component-based architecture. Key components include: </CONTENT>

</PAGE1> <PAGE2> <CONTENT> - \*\*Header:\*\* Navigation menu and search input. - \*\*Player:\*\* Controls for music playback (play, pause, etc.). - \*\*Sidebar:\*\* Navigation links and playlist overview. - \*\*HomePage:\*\* Showcases trending tracks, featured playlists, and recent releases. - \*\*SearchPage:\*\* Interface for searching music content. - \*\*PlaylistPage:\*\* Displays and manages user playlists.

* **State Management:** The app uses [Redux/Context API] for global state management, handling user data, playback state, playlists, and search results.
* **Routing:** Navigation is managed with React Router. Defined routes include:
  + /: Home page
  + /search: Search page
  + /playlist/:id: Playlist details page

**Setup Instructions**

**Prerequisites:**

- Node.js (v16 or higher)

- npm or Yarn

- A News API key (e.g., from [NewsAPI.org](https://newsapi.org/))

**Installation:**

1. Clone the repository:

```bash

git clone https://github.com/[your-username]/react-news-app.git

```

2. Navigate to the project directory:

```bash

cd react-news-app

```

3. Install dependencies:

```bash

npm install

```

4. Create a `.env` file in the root directory and add your API key:

```

REACT\_APP\_NEWS\_API\_KEY=your\_api\_key\_here

```

5. Start the development server:

```bash

npm start

**Folder Structure**

**Client:**

src/

├── components/ # Reusable UI components

│ ├── Navbar.js

│ ├── NewsCard.js

│ ├── NewsList.js

│ └── SearchBar.js

├── pages/ # Page-level components

│ ├── Home.js

│ ├── Category.js

│ └── NewsDetail.js

├── assets/ # Images, fonts, etc.

│ └── logo.png

├── context/ # Context API for global state

│ └── NewsContext.js

├── utils/ # Utility functions

│ └── api.js # API fetch logic

├── App.js # Root component

└── index.js # Entry point

**Utilities:**

- `api.js`: Contains helper functions to fetch data from the News API, including error handling and caching logic using `localStorage`.

- Custom Hook: `useNewsFetch` - A hook to manage API calls and loading states.

### 

### **6. Running the Application**

* **Frontend:**
  + Start the frontend server: npm start (in the client directory).
  + Additional setup (if applicable):
    - npm install
    - npx json-server [your-db-file] (if using a mock backend).
    - npm run dev
  + Access the app at: http://localhost:3000

### **7. Component Documentation**

* **Key Components:**
  + **Header:** Renders navigation and search bar.
    - **Props:** onSearch (function for search input handling).
  + **Player:** Manages music playback controls.
    - **Props:** currentTrack (track object), onPlay, onPause, onSkip. </CONTENT>

</PAGE3> <PAGE4> <CONTENT> - \*\*PlaylistCard:\*\* Displays playlist details like name and image. - \*\*Props:\*\* `playlist` (playlist object), `onClick` (selection handler).

**Reusable Components:**

* + **Button:** Customizable button component.
    - **Props:** text, onClick, disabled.
  + **Input:** Reusable input field for forms/search.
    - **Props:** type, placeholder, value, onChange.

### **State Management**

* **Global State:** The [Redux/Context] store manages:
  + user: Authenticated user data.
  + player: Current track, playback status, and volume.
  + playlists: User playlist data.
  + searchResults: Results from search queries.
* **Local State:** Local state is handled with React’s useState hook within components, e.g., search input state in SearchPage.

### **9. User Interface**

* **Screenshots:**
  + **Home Page:** Displays trending tracks and curated playlists. </CONTENT>

</PAGE4> <PAGE5> <CONTENT> - \*\*Search Page:\*\* Interface for searching songs, artists, and albums. - \*\*Playlist Page:\*\* View and manage playlists. </CONTENT> </PAGE5> <PAGE6> <CONTENT> ### 10. Styling

---

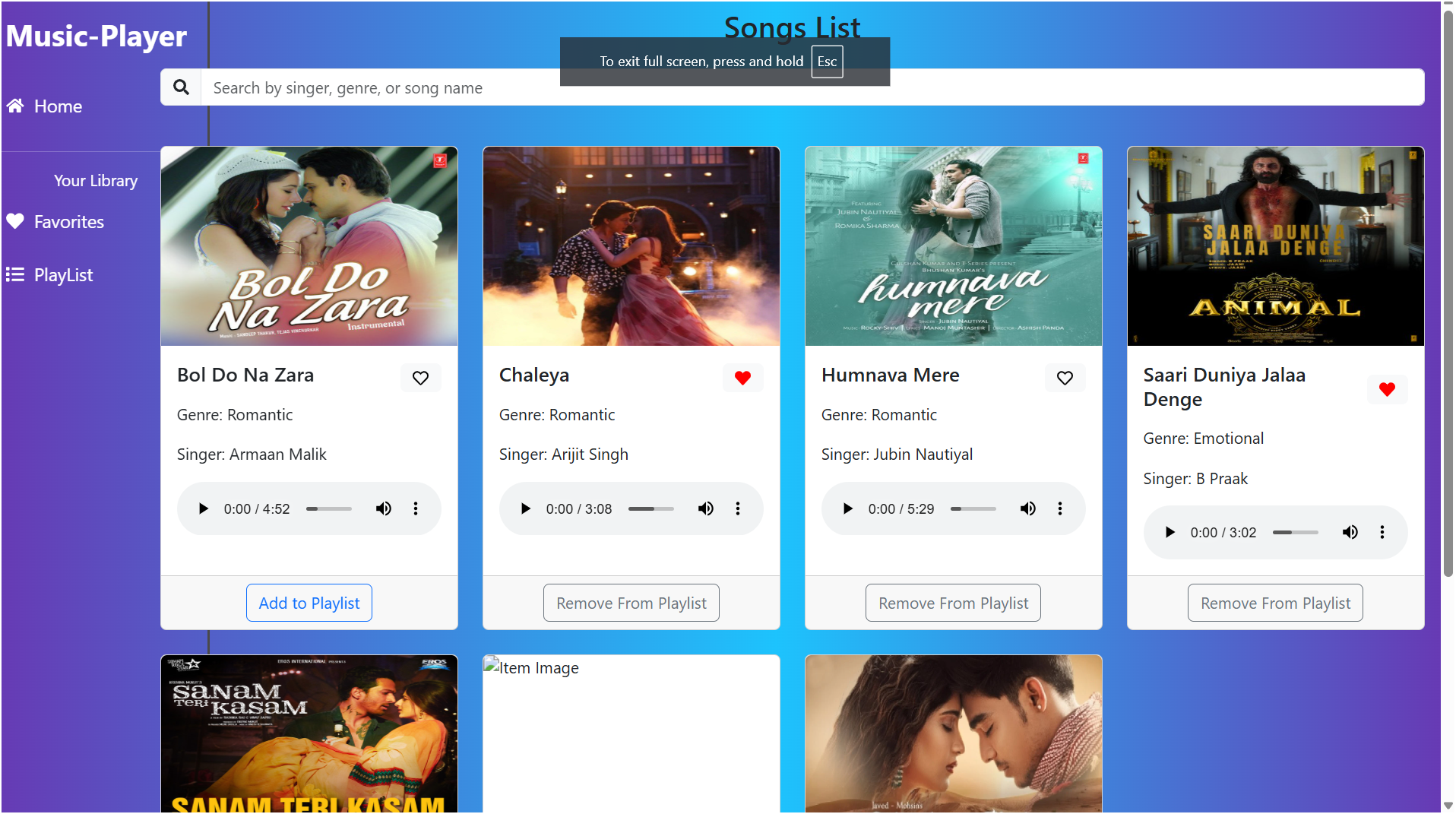
* **CSS Frameworks/Libraries:** The app uses [Styled-Components/Tailwind CSS/Material-UI] for styling, ensuring modular and maintainable CSS.
* **Theming:** A custom theme is implemented with support for [light/dark modes or specific styles].

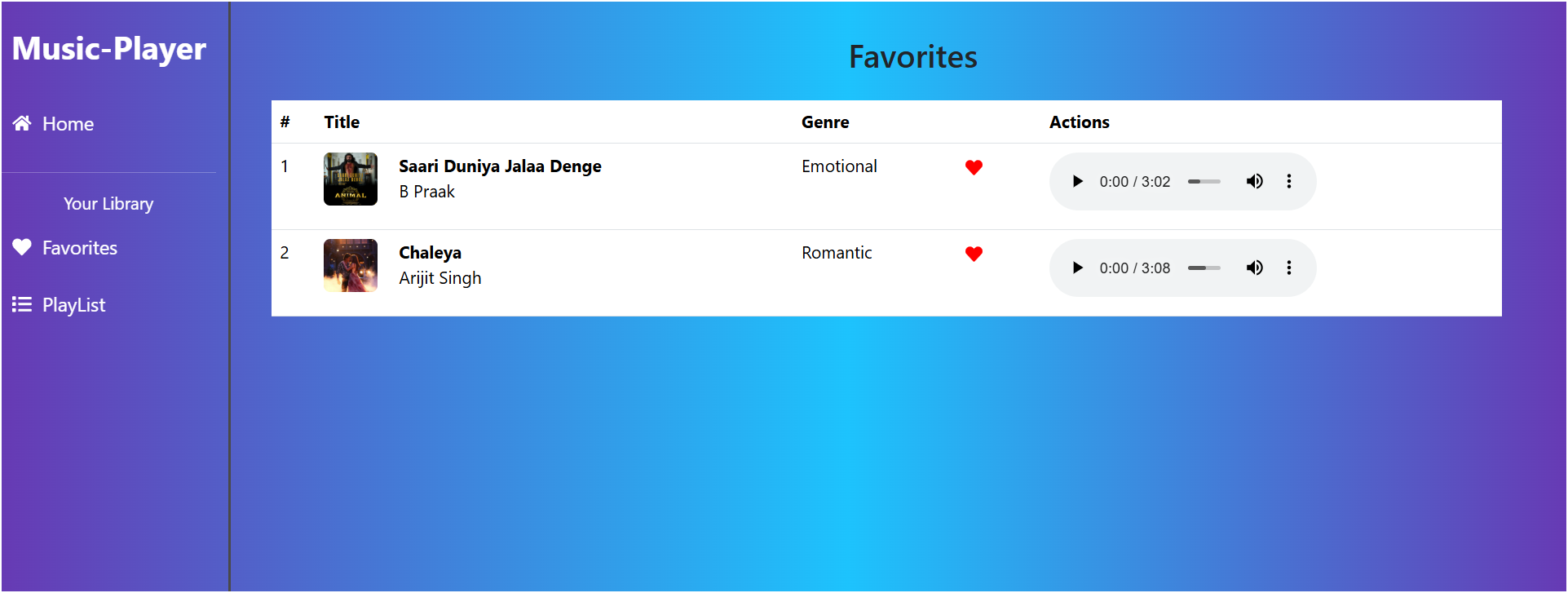
### **11. Testing**

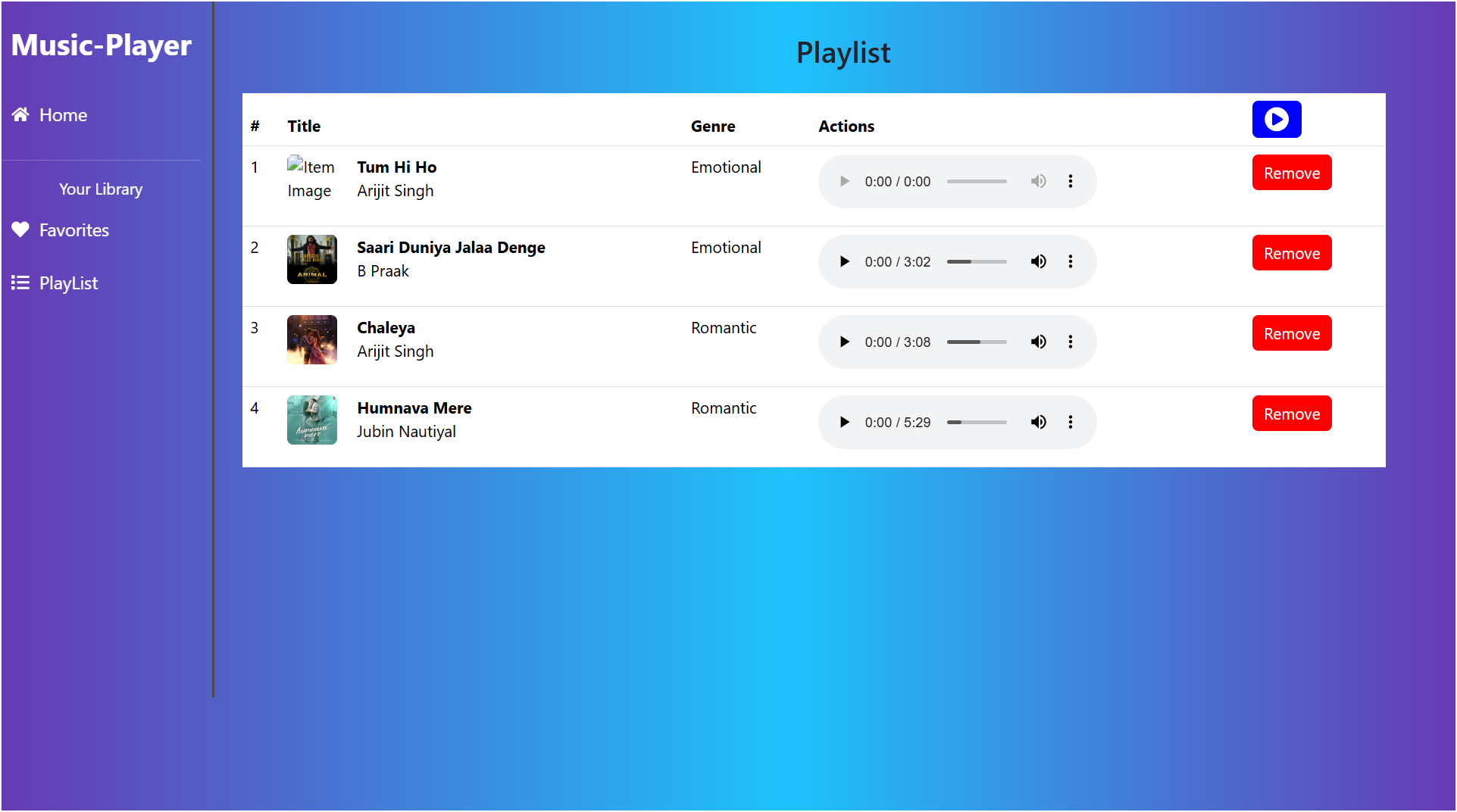
* **Testing Strategy:**
  + **Unit Testing:** Conducted with Jest and React Testing Library.
  + **Integration Testing:** Ensures component interoperability.
  + **End-to-End Testing:** Performed with [Cypress/another tool] for user workflows.
* **Code Coverage:** Monitored using [Jest/tool name], targeting [your coverage percentage, e.g., 80%]. </CONTENT>

</PAGE6> <PAGE7> <CONTENT> ### 12. Screenshots or Demo

**Screenshots**







### **13. Known Issues**

* **Issue 1:** [e.g., Playback lags on low-bandwidth connections.]
* **Issue 2:** [e.g., Search results occasionally fail to update.]

### **14. Future Enhancements**

* **Future Features:**
  + Add offline playback support.
  + Integrate a recommendation system for personalized content.
  + Enhance UI with animations and transitions.