Frontend Development with React.js

Project Documentation

1. **Introduction**
   - **Project Title**   : Rhythmic Tunes: your melodic companion
   - **TEAM ID**   : **NM2025TMID48325**
   - **TEAM LEADER**   : **GNANASHREE C**
   - **ROLE**   : CODING AND DEVELOPMENT
   - **TEAM MEMBER**   : **GOPIKA S, GOWTHAMI D**
   - **ROLE :** DEMO VIDEO
   - **TEAM MEMBER : LOGUDHARSHINI M,LINGESWARI L**
   - **ROLE :** DOCUMENT CREATER

2. **Project Overview**
   - **Purpose**: The purpose of the Rhythmic Tunes Music Player is to provide a user-friendly application for playing music..
   - **Features**: Key features include: music playback, playlist creation, and search functionality.
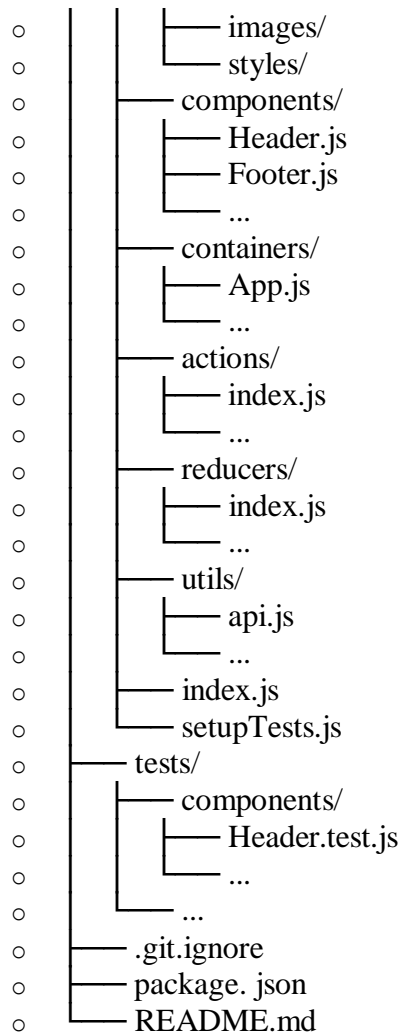
3. **Architecture**
   - **Component Structure**: Key features include: music playback, playlist creation, and search functionality.
   - **State Management**: The application uses the Context API for global state management to handle the current song, playback status, and user-created playlists.
   - **Routing**: React Router is used to manage navigation between different views, such as the home page and the playlist page

4. **Setup Instructions**
   - **Prerequisites**: The required software dependencies are Node.js and npm ,React.js.
   - **Installation**: Provide a step-by-step guide to Clone the repository from GitHub. Navigate to the client directory.
   - * Run npm install to install dependencies.
   - * Configure environment variables for any API keys

5. **Folder Structure**
   - **Client**: The main React application is organized into folders such as components, pages, and assets for images and audio files.
   - **Utilities**: Helper functions for music playback and custom hooks for managing audio state are located in the utils folder
   - **Code**:
   - project-name/
   - ├── public/
   - │   ├── index.html
   - │   └── favicon.ico
   - ├── src/
   - │   ├── assets/

```
o  │      │      ├── images/
o  │      │      └── styles/
o  │      ├── components/
o  │      │      ├── Header.js
o  │      │      ├── Footer.js
o  │      │      └── ...
o  │      ├── containers/
o  │      │      ├── App.js
o  │      │      └── ...
o  │      ├── actions/
o  │      │      ├── index.js
o  │      │      └── ...
o  │      ├── reducers/
o  │      │      ├── index.js
o  │      │      └── ...
o  │      ├── utils/
o  │      │      ├── api.js
o  │      │      └── ...
o  │      ├── index.js
o  │      └── setupTests.js
o  ├── tests/
o  │      ├── components/
o  │      │      ├── Header.test.js
o  │      │      └── ...
o  │      └── ...
o  ├── .git.ignore
o  ├── package. json
o  └── README.md
```

6. **Running the Application**
   - o Provide commands to start the frontend server locally.
     - ▪ **Frontend**: npm start in the client directory.
7. **Component Documentation**
   - o **Key Components**: Manages music playback controls. It receives a song prop.
   - o * Playlist Component: Displays the list of songs and manages playlist creation.
   - o * Search Component: Handles searching for songs
   - o **Reusable Components**: Button Component: A configurable button component used throughout the application.
   - o * Song Card Component: A component for displaying individual songs in lists.
8. **State Management**
   - o **Global State**: The Context API is used to manage the global state, ensuring that the current song and playback status are accessible to all components..
   - o **Local State**: se use State hooks to manage their own local state, such as form inputs or UI toggles
9. **User Interface**
   - o Provide screenshots or GIFs showcasing different UI features, such as pages, forms, or interactions.

10. **Styling**

- **CSS Frameworks/Libraries**: Describe any CSS frameworks, libraries, or pre-processors (e.g., Sass, Styled-Components) used.
- **Theming**: A custom design system with light and dark themes is implemented
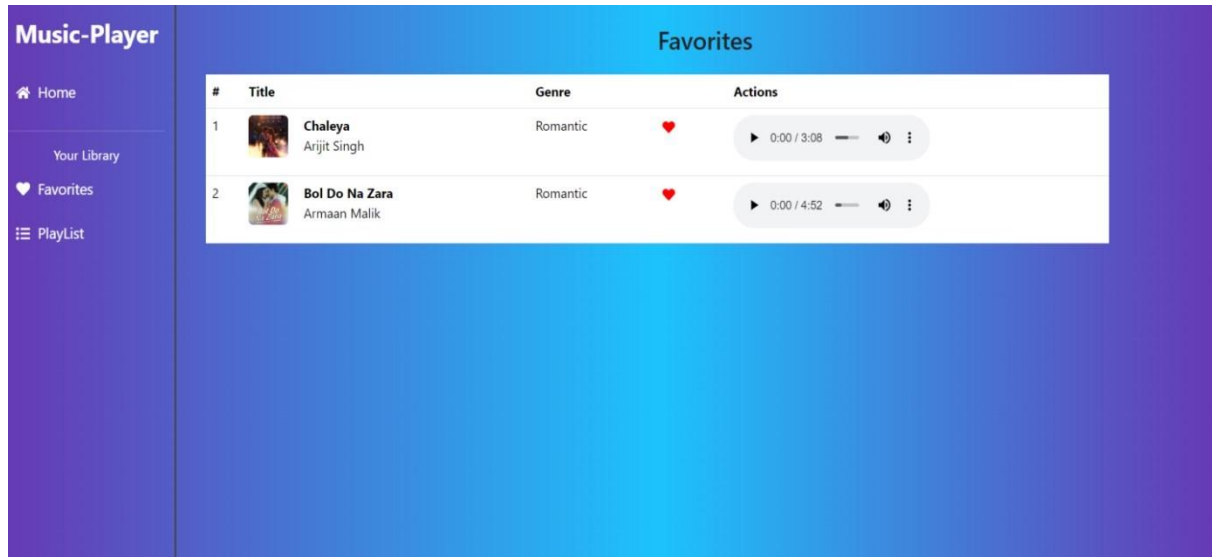
11. **Testing**

- **Testing Strategy**: The application uses Jest and React Testing Library for unit and integration testing of components.

12. **Code Coverage**: Code coverage is tracked using a tool to ensure all key components are adequately tested.

13. **Screenshots or Demo:**
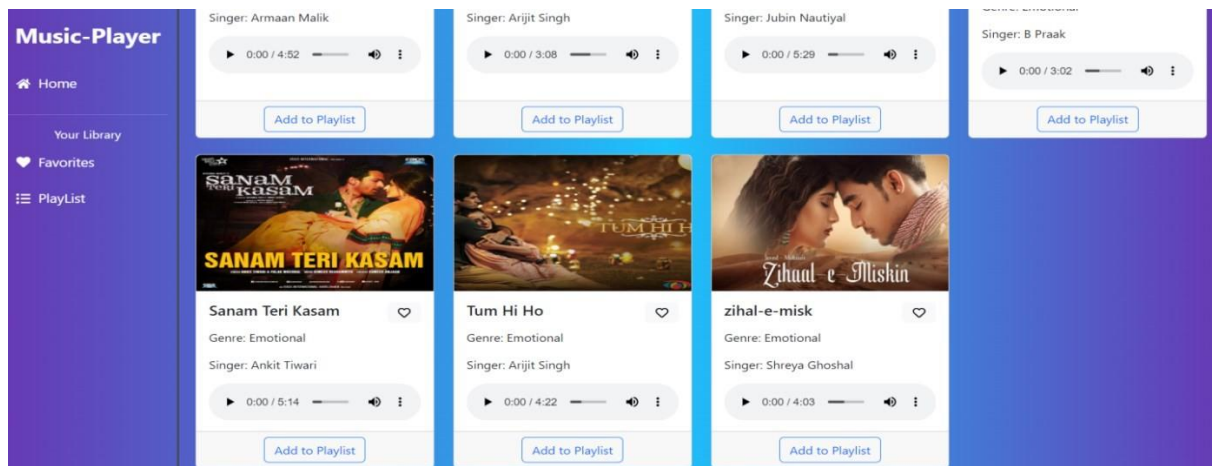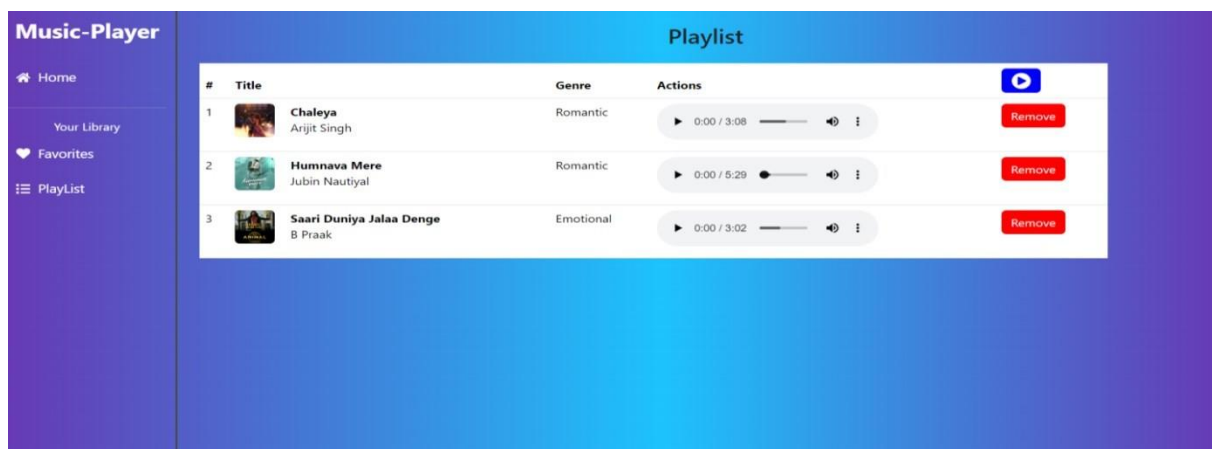
.favorties

13. **Known Issues**

- Any known bugs or issues, such as occasional playback glitches on certain browsers, should be documented here

14. **Future Enhancements**

- Potential future features could include a user authentication system, enhanced search functionality, and adding more animations to the user interface.

15. **DEMOLINK:**
- https://drive.google.com/file/d/1DozFUIo3wyvhVFyyd-E3alVq5Dy_BK54/view?usp=sharing