**GR Tunes Documentation**

**1. Introduction**

**Project Title: GR Tunes**

**Team Members: Name & Email Id**

|  |  |
| --- | --- |
| **Name** | **Email Id** |
| **S. Ganesh Raja** | **selvarajganesh2010@gmail.com** |
| **T.Mohammed Ameer Ali Khan** | **mohammedameer2778@gmail.com** |
| **R. Dinesh** | **dineshraj009841@gmail.com** |
| **A.Yuvaraj** | **yuvarajmass640@gmail.com** |
| **M. Naveen** | **naveen1112004@gmail.com** |

**2. Project Overview**

**Purpose**

RhythmicTunes is a modern web-based music player application that allows users to discover, organize, and enjoy their favorite music. The application provides a seamless experience for managing playlists and favorites while offering an intuitive user interface.

**Key Features**

- Music playback with audio controls

- Playlist management

- Favorites collection

- Search functionality

- Responsive design

- Real-time updates

**3. Architecture**

Component Structure

App (Root Component)

├── Sidebar

│ └── Navigation Links

├── Songs (Main Content)

│ ├── Search

│ ├── Song Cards

│ └── Audio Player

├── Favorites

│ └── Favorite Songs List

└── Playlist

└── Playlist Songs

```

**State Management**

- Uses React's built-in useState and useEffect hooks

- Implements axios for API calls

- Local storage for user preferences

- Real-time state updates for music playback

Routing

Uses React Router v6 with the following routes:

- `/`Home/Songs page

- `/favorites` - Favorites collection

- `/playlist` - User's playlist

**4. Setup Instructions**

**Prerequisites**

- Node.js (v14 or higher)

- npm (v6 or higher)

- JSON Server

**Installation**

**Install dependencies**

cd rhythmic-tunes

npm install

npm install react

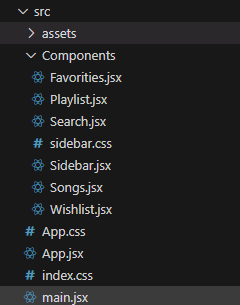
**Start JSON Server**

json-server --watch db.json --port 3000

**Start the application**

npm run dev

**5. Folder Structure**



**6. Running the Application**

**Development Server**

**Start the development server**

npm run dev

**Start the JSON server**

npx json-server --watch db.json

**7. Component Documentation**

**Key Components**

Sidebar (Sidebar.jsx)

.jsx

<Sidebar />

- Navigation component

- Fixed position

- Contains links to main sections

Songs (Songs.jsx)

jsx

<Songs />

- Displays song grid

- Handles music playback

- Manages favorites and playlist operations

Wishlist (Wishlist.jsx)

.jsx

<Wishlist />

- Displays favorite songs

- Handles removal of favorites

**8. State Management**

**Global State**

- User preferences

- Currently playing song

- Favorites collection

- Playlist items

**Local State**

- Search queries

- Audio playback status

- UI interactions

**9. User Interface**

Main Features

- Dark theme design

- Responsive grid layout

- Interactive audio controls

- Search functionality

- Favorites and playlist management

**10. Styling**

CSS Framework

- Bootstrap 5.3.2

- Custom CSS modules

- Responsive design principles

**Theme Colors**

.css

:root {

--primary-bg: 363c42;

--sidebar-bg: 9192bb;

--card-bg: 9192bb;

--text-primary: ffffff;

--accent-color: ff6b00;

}

**11. Testing**

Testing Strategy

- Component testing using React Testing Library

- Integration tests for API calls

- User interaction testing

Test Coverage

- Component rendering

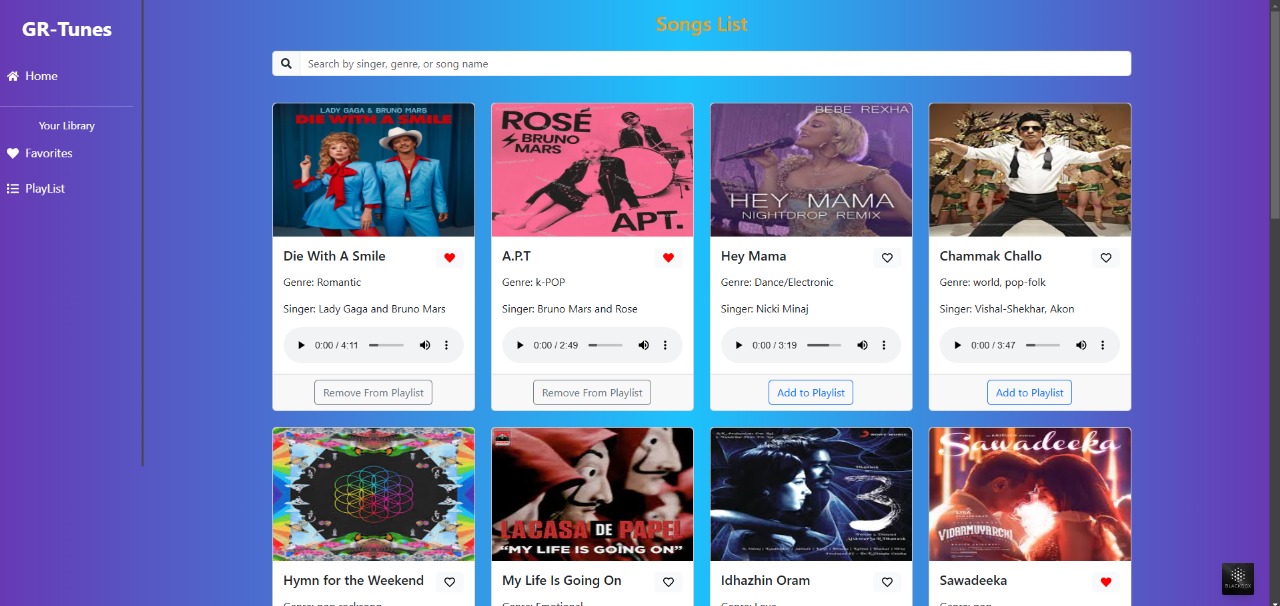
- User interactions

- State management

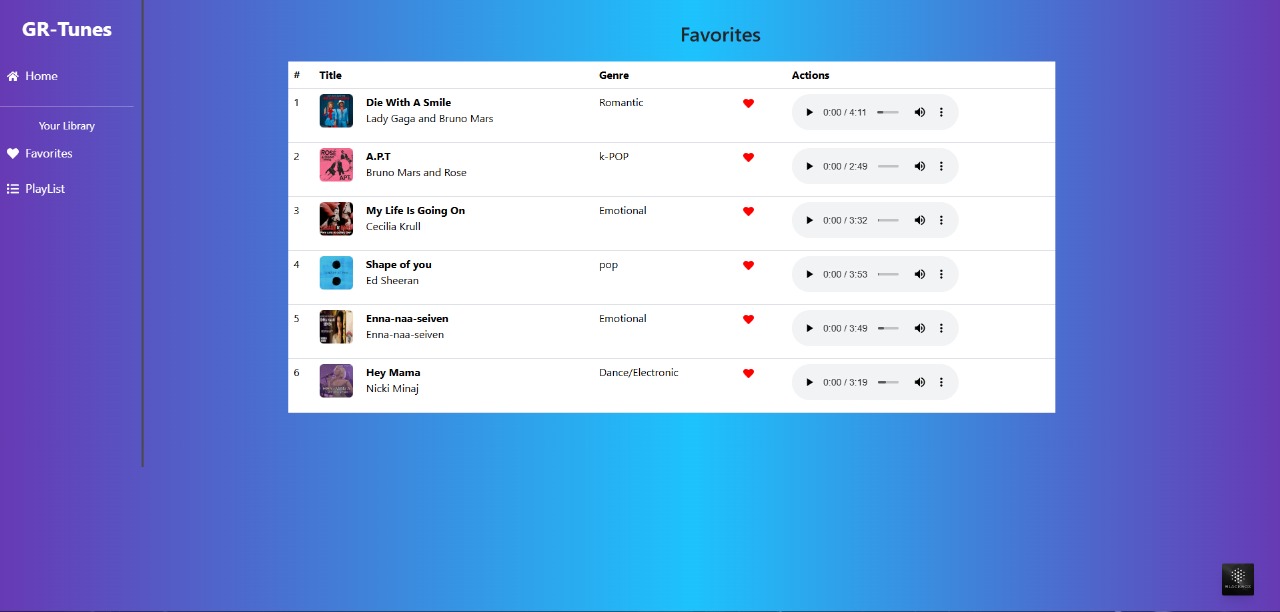
- API integration

**12. Screenshots**

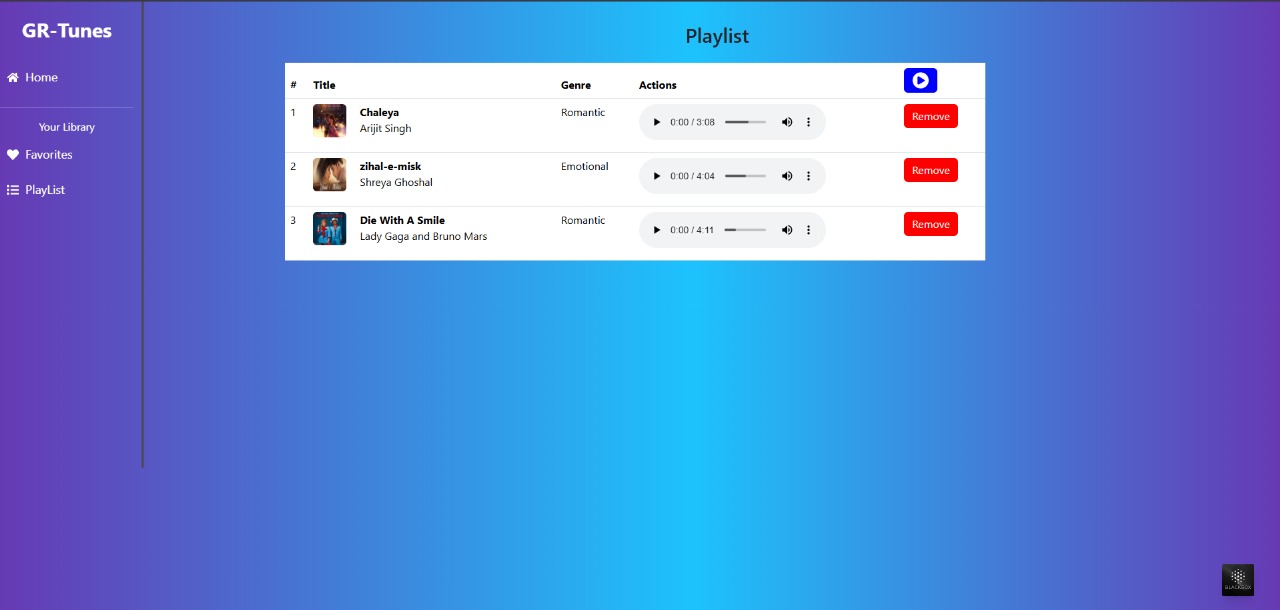
Home page:



2. Favorites:



3.Playlist:



**13. Known Issues**

1. Multiple audio playback conflict

2. Mobile responsive issues in some views

3. Search performance optimization needed

**14. Future Enhancements**

1. User Authentication

- Login/Signup functionality

- User profiles

2. Advanced Features

- Equalizer

- Crossfade

- Offline mode

3. Social Features

- Share playlists

- Follow other users

- Social media integration

4. Performance

- Lazy loading

- Progressive image loading

- Caching strategies

5. UI/UX Improvements

- Dark/Light theme toggle

- Custom audio visualizer

- Drag and drop playlist management

**API Integration**

Endpoints

.javascript

// Songs

GET /items // Fetch all songs

POST /items // Add new song

// Favorites

GET /favorities // Get favorites

POST /favorities // Add to favorites

DELETE /favorities/:id // Remove from favorites

// Playlist

GET /playlist // Get playlist

POST /playlist // Add to playlist

DELETE /playlist/:id // Remove from playlist

**Performance Considerations**

1. Single audio playback

2. Optimized image loading

3. Efficient state updates

4. Responsive design optimization

**Security Measures**

1. Input validation

2. API error handling

3. Secure routing

4. Data sanitization