**Quick links**

<https://reactjs.org/docs/create-a-new-react-app.html>

**Architecture overview**

* Front-end technologies– React JS
* Third party applications used – Spotify for developers
* API – Fetch API
* ES-6 methods

Front-end -> Fetch API -> Spotify developer functions

**Prerequisites**

* Node.js with NPM
* Spotify developer account

**Getting started**

* Clone the repository from the <https://github.com/shawntharoo/Spooty_Music.git> URL.
* Open command prompt (Windows) / Terminal (MacOS) and ‘cd’ into the cloned folder (cd <project-directory>).
* Type ‘npm install’ and wait until the node modules are installed.
* Type ‘npm start’ and hit enter.
* The application will be open in a web browser with the following URL <http://localhost:8888/home>

**Application walkthrough**

* In order to use Spotify searching APIs’ the user should have been authenticated using one of Spotify authentication methods. To get authenticate token click on the “Authenticate with Spotify” button in the home page.
* A new window will appear which Spotify authentication page is, click on the “verify” button. If the user verification is succeeded, the user will be redirected to the search page of the application.
* User can search artist, playlist or tracks by entering a search query in the provided field. The limit inbox is used to get the input from user how many results should show from each category. Then after press search icon.

**Spotify APIs’ and Access control**

The application uses ‘Fetch’ APIs’ to retrieve data from the Spotify data stores.

The ‘Fetch’ method is a great step of getting http requests native in ES6; fetch will help us with doing low-level operations easily. Since the application is using few APIs’ to connect with Spotify, fetch is more suitable. (Compared to ‘axios’)

**Application Routing**

The routing of the application has been achieved with “react-router-dom” module.

The routing has been used because there are two state of the application.

One is user is not authenticated with the Spotify. Then user will not be able to go the search page since without authenticated from Spotify the search functionalities cannot be performed. So the application is developed to retain users in the Home page who are not authenticated.

The second state is user is authenticated with the Spotify. Then the user should be able to search from the Spotify and therefore only those users will be redirected to the search page.

With routing added to application the two states can be achieved easily and conveniently.

**Front-End Styling**

‘Reactstrap’ is a library of reusable front-end components. This library contains React Bootstrap 4 components. The library does not depend on jQuery or Bootstrap JavaScript.

For small applications like the one we developed, the ‘Reactstrap’ is great since it is lightweight and will not effect for the performance. It also consist with easy to use pre-configured components and rich documentation.

**Spotify authentication methods**

For the application, the implicit grand authentication method of Spotify authentication methods is used.

Implicit grand flow authentication method is for clients that are implemented entirely using JavaScript and running in the resource owner’s browser.

Reason for not using client credentials flow method and Authorization code flow method.

* Authorization code flow method is suitable for long-running applications in which the user grants permission only once and our application is short-term application.
* The Client Credentials flow is used in server-to-server authentication.

**Folder Architecture**

The component folder contains all the individual components.

* Components are created in separate folders for reusability. E.g.: “HeaderComponent” and “FooterComponent” should display in each page of the application and creating them as separate components will helps to just import and use it. Or the header and footer can be applied by adding them in each page which is not a good practice.
* “MainComponent” act as the parent of all the individual components. All the components renders inside the “MainComponent”.
* “App” component internally called to the “MainComponent” and render the application with routes.

Utils folder contains the app utility files such as configurations and constants.

* “Constants” file is used to store the values that are constant to the application. In addition, those values will be used in several places through the application. Changing constants file values will affect the whole project where that value has been used. Therefore, maintain a separate file for constants will make it easy to change a value, which has been used in several places in the application.

Service folder contains the files which are related to back end connections.

* In our case, we have to call the Spotify APIs’ in order to retrieve the data from the Spotify. The connections to the Spotify has maintained inside the server folder. (Here we have used “SpotifyAPI.js” file)

Assets folder contains the static images.

* Assets folder contains all the static images that the application is used.

**To continue developing…**

* Just download the code, create new components in the component folder, and add them to the routing module in MainComponent.js
* Services can be implemented in server folder using the same “spotfy.js” file or create your own files and import them in relevant components.
* Constants can be added to the constant file.
* Use “Reactstrap” styles for styling components as it already have been installed.