*Jammming* Feature Request: Improved Authentication Flow

**September 18th 2021**

# **OBJECTIVE**

The overall goal of this feature request is to enhance user experience by improving *Jammming*'s authentication flow.

# **BACKGROUND**

In its current state, Jammming does not inform the user that authentication is required in order to use the app.

When a user performs a search for the first time they are redirected to the *Spotify* authorization page. They give the app the relevant permissions and are then redirected to the app home page. The search bar is empty, so they have to re-enter their query and click the 'SEARCH' button.

On subsequent page loads, the user performs a search, *Spotify*’s authorization service recognises that the user has previously given the relevant permissions, and the user is swiftly redirected to the app home page. Again, the search bar has emptied, so they have to repeat the query process.

This feature improves the authentication flow by:

* Informing the user that authentication is required in order for the app to become functional
* Including an ‘AUTHENTICATE’ button so the user can initiate the authentication process
* Disabling the ‘SEARCH’ and ‘SAVE TO SPOTIFY’ buttons until the authentication process is successful
* Disabling the ‘SEARCH’ and ‘SAVE TO SPOTIFY’ buttons once the current access token expires

# **TECHNICAL DESIGN**

**Inform the User of the Need for Authentication**

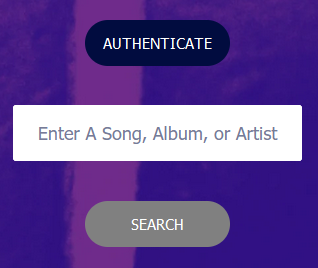
Two paragraphs should be included that inform the user of the app’s requirement for initial authentication and periodic re-authentication. A black font on a background colour that complements purple should be used in order to differentiate this informative text from the descriptive text of the app.



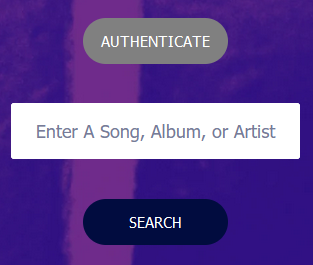
**Informative text that explains the app’s authentication requirements**

**Include an ‘AUTHENTICATE’ Button**

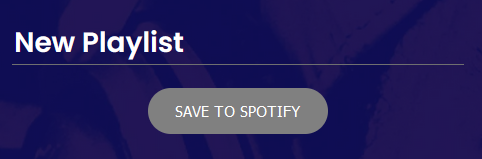
A new ‘AUTHENTICATE’ button should be added in searchbar.js. This button should be enabled when the web app is loaded, allowing the user to initiate the authentication process. It should become disabled when the authentication process is successful. Conversely, the ‘SEARCH’ and ‘SAVE TO SPOTIFY’ buttons should be disabled when the web app is loaded; these buttons should become enabled when the authentication process is successful.



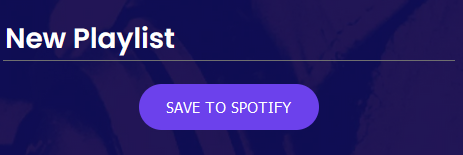
**Initial state of the ‘AUTHENTICATE’ and ‘SEARCH’ buttons**



**Authenticated state of the ‘AUTHENTICATE’ and ‘SEARCH’ buttons**



**Initial state of the ‘SAVE TO SPOTIFY’ button**

****

**Authenticated state of the ‘SAVE TO SPOTIFY’ button**

A function called checkAccessToken() that is invoked when the **onload** *event* fires should be created and attached to the <body> of index.html. This checkAccessToken() function will handle the authentication logic, freeing the Spotify.getAccessToken() and Spotify.search(term) functions to simply return an already-defined access token and perform a search. checkAccessToken() is invoked when:

1. The web app is navigated to
2. After the authentication process completes.

checkAccessToken() performs the following operations:

**If there is an access\_token parameter in the URL:**

* Extracts the access\_token and expires\_in parameters, storing them in accessToken and expiresIn variables respectively
* Sets accessTokenin local storage, using the key “accessToken”
* Removes the returned parameters from from the URL
* Converts the expiresIn variable to a Number
* Sets a function to run in (expiresIn \* 1000) milliseconds that: removes accessTokenfrom local storage; enables the ‘AUTHENTICATE’ button; and disables the ‘SEARCH’ and ‘SAVE TO SPOTIFY’ buttons
* Sets the ‘AUTHENTICATE’ button to disabled and enables the ‘SEARCH’ and ‘SAVE TO SPOTIFY’ buttons

**If there is no access\_token parameter in the URL:**

* Disables the ‘SEARCH’ and ‘SAVE TO SPOTIFY’ buttons
* Removes accessToken from local storage, if it exists

**Allowing the User to Initiate Authentication**

As the ‘AUTHENTICATE’ button is visibly enabled after the page is navigated to, the user is able to initiate the authentication process by clicking the button. This button invokes an authenticate()function that redirects the user to the Spotify authorization page, and the authentication process unfolds in the manner outlined in the ‘Background’ section above.

In order to implement the authenticate() function, the clientID and redirectURI constants should be moved to App.js, and an authenticationURLshould be created in the form `[https://accounts.spotify.com/authorize?client\_id=${clientID}&response\_type=token&scope=playlist-modify-public&redirect\_uri=${redirectURI](https://accounts.spotify.com/authorize?client_id=$%7BclientID%7D&response_type=token&scope=playlist-modify-public&redirect_uri=$%7BredirectURI)}`. This authenticationURL should be passed to the <SearchBar /> element as an attribute called ‘authenticationURL’. An authenticate() function should then be created in - and bound to - SearchBar.js that is invoked when the user clicks the ‘AUTHENTICATE’ button. This function sets the window location to ‘this.props.authenticationURL’, thus facilitating the authentication process.

Once authentication is successful, an accessToken is set in local storage until such time that it expires. Now when the user performs a search, Spotify.getAccessToken()simply returns the accessToken that is stored in local storage, and the Spotify.search()function completes the authenticated user’s search operation. The ‘SEARCH’ and ‘SAVE TO SPOTIFY’ button functionality is enabled until such time that the accessToken expires, when it becomes necessary to repeat the authentication process.

# **CAVEATS**

The above solution is preferable to the existing implementation as it separates authentication logic from search logic, in line with the ‘separation of concerns’ programming maxim, and results in an improved user experience.

Further recommendations include moving the search-term-checking logic from Spotify.search(term) to the search(term) function defined in App.js. Before Spotify.search(term) is invoked from within the App.js search(term), a check should be made to ensure that the user has entered a valid search term using the following code:

if(term.trim().length === 0) { return; }

This check avoids an error being generated if Spotify.search(term) is invoked without a valid search term. Similarly, a check should be made that a valid playlist name and a valid playlist exists before the Spotify.savePlaylist(this.state.playlistName, trackURIs) is invoked by the App.js savePlaylist() function -

if(this.state.playlistName.trim().length === 0 || !trackURIs.length) { return; }

**An implementation of this feature request can be viewed at:**

[**https://codecademy-spotify-app.surge.sh**](https://codecademy-spotify-app.surge.sh)

**https://codecademy-spotify-app.netlify.app**