Feature Request: Search Restoration after Login Redirect

**January 13, 2018**

**OBJECTIVE**

To allow the user’s search query to continue after the Spotify authentication process.

**BACKGROUND**

Currently, when a user performs a search in Jammming, if they are not yet signed into Spotify, they are redirected to the Spotify authentication portal to log in or set up an account. Upon redirect back to Jammming, the search query is not retained, and the user must perform their search again, which is a source of frustration for many users. The ability to retain the search term and complete the search would improve functionality by providing a smoother login process for the user.

This feature has the following functionality:

* Saves the search term to the to the browser storage.
* Checks if there is a saved search term in the browser storage and performs the search on the saved term if one is found.
* Removes the saved search term from the browser storage after performing the search.

**TECHNICAL DESIGN**

In order to implement this functionality, the search term entered by the user needs to be saved in the browser storage so that it can be retrieved later. In the **search(**) method of the **Spotify** component, the provided search term needs to be saved to **window.localStorage** by calling **window.localStorage.setItem('searchTerm', term)**.

In the SearchBar component, we should initialize the state value of input to an empty string by adding **this.state = {term: ''}** to the **constructor()** method. We then need to add a **value** attribute to the **input** tag and set it to **this.state.term**. This will allow the search term to repopulate in the input field when the page is reloaded after Spotify authentication. With no saved search term, the input field will be blank when it is rendered.

The **SearchBar** component should check to see if there is a saved search term before it is loaded so that the page can load with the results of the user’s search query. This can be done with a mounting lifecycle method. In the **SearchBar** component, a **componentWillMount()** method should be created. Within this method, we will check for a saved search term and perform the search if one is found.

We will need to declare a variable, **savedTerm**, which will hold the result of calling **window.localStorage.getItem(‘searchTerm’)**.

We then need to evaluate **savedTerm** to check if a value has been set. We can use an if statement to check the truthy value of **savedTerm**. (An evaluation of false indicates that no search term has been saved, and the following code will not run. The page will load with no search query.) If **savedTerm** evaluates to true, we set **savedTerm** as the value for **term** in the **state** as is done in the **handleTermChange()** method.

We will then call the **onSearch()** method in the **props** with **savedTerm** as the argument in order to perform the search. When the component mounts, the page will load with the results of the user’s search query.

After the search is performed, we no longer need the search term to remain in the user’s browser storage. We will then clear **searchTerm** from the browser storage by calling **window.localStorage.removeItem('searchTerm')**.

**CAVEATS**

This implementation requires saving information to the user’s browser storage. If the user has disabled this feature on their browser, or if their browser does not support window.localStorage, then this functionality will not work. Another option for storing the search term is window.sessionStorage; however, it has the same drawback that it will not work if it is disabled in the user’s browser or if the browser is incompatible. Upon testing with sessionStorage, the app would not accept additional search queries following the initial search. This is something that can be tested further and debugged if we decide to use seissionStorage since sessionStorage does not require a removeItem call. It will clear when the user closes the browser window or tab. With local storage, we can create an additional feature that will retain the search term and results if the user refreshes their page or opens the page in a different window. This would safeguard their search in the event of a browser crash. We can also look into applying this to restore the playlist name and selected songs in the case of a browser refresh. These options are not available with sessionStorage as it is cleared when the session ends.

Cookies can also be used for storage of the search term, but this also loses functionality if users have cookies disabled, and cookies are sent to the server while localStorage and sessionStorage remain local to the user’s browser where it will be used. Since the server does not need this information to restore the search term and complete the search, storing the information as a cookie is unnecessary.