Feature Request: Allow users to search by a choice of song, track or artist

**Sep 13th 23**

# OBJECTIVE

To give users the ability to search by song, track or artist in spotify.

# BACKGROUND

Currently, Jammming supports the ability to search for tracks via song title. We would like to add the functionality to search for songs via album or artist.

This feature accomplishes the following:

* Displays a dropdown menu to choose from song, album or artist in the search bar
* Allows a user to search using one of these queries
* Fix the bug that the search term disappears after user redirect on login, restoring the search term from before the redirect

# TECHNICAL DESIGN

# Allow dropdown menu to change search query

We will need to initialize *state* for *searchType* to contain a key for *type* that defaults to name (title).

To update the searchType, a function will setSearchType on selection of a choice from a dropdown box in the search bar.

SearchType will be updated to one of 3 options, this will be passed as an argument with term to spotify.search() function upon user search. Using template literals this search type will be passed into the fetch url in the search function. `https://api.spotify.com/v1/search?type=${searchType}&q=${term}`

Search by song title returns a list of tracks by song title. Search by Artist should return a list of most popular songs by that artist. Search by album, list of albumns

# Restore search term from before user redirect

Add session storage to reinstate the state term in search bar input after refresh. Using State to set searchTerm on refresh from session storage (initial load) and useEffect to update searchTerm in session storage on change of the state term.

# CAVEATS

# App Playlist State

With this implementation, we are now storing three separate values about our playlist on our *App* component. This information is almost always linked together and therefore it may be useful to store one object called *playlist* on the *App*'s *state*. This implementation would make passing information easier and potentially improve readability. However, it also makes it less clear which information must be present in *App*'s state. This less structured data provides more room for incorrect implementation of future feature requests. With only three properties being stored, we have decided to continue storing this information as separate *state* values. However, this should be re-evaluated if we decide to store even more playlist information on *App*'s *state*.

**Asynchronous Existing Playlist Save Requests**

In the initial implementation of Jammming, every playlist save necessitated a playlist to be created with the specified name and then the tracks to be saved to the playlist after that playlist creation completes. With the added functionality of saving a pre-existing playlist, we no longer need to create a new playlist as the first step every time. As a result, when saving a pre-existing playlist, we could send two simultaneous requests: one to save the updated name, the other to save the updated track list. This would result in a more complicated save playlist flow and would result in marginal returns in time efficiency when saving a playlist. Since the user is not prevented from continuing to interact with Jammming during playlist save, this efficiency would result in no change to user experience. Therefore we will not implement this functionality since the loss in code readability outweighs any gain in user experience. This would be a premature optimization.

**Excess Playlist Saves**

Currently, Jammming will save the displayed playlist name and tracks on every "SAVE PLAYLIST" click regardless of whether or not data has changed. This could result in unnecessary requests to the Spotify API (especially considering Playlist names are not likely to change often). As noted in the previous caveat, though, extra requests to the Spotify API will not result in any change to the user experience. If we find in the future that users are intentionally abusing this functionality and we run the risk of throttling the Spotify API, we will implement this functionality. However, at this point, this would be another premature optimization.