Feature Request: Non-duplicate Song Display

December 18th 2017

OBJECTIVE

To prevent tracks that have already been added to the playlist from appearing in the search results.

BACKGROUND

Jammming has the functionality to return a list of tracks, artists and albums that relate to a users search. The user can then select which tracks they'd like to add to their playlist. Once a track is in the users playlist it remains in the search results list. If a new search is made which returns a track already in the playlist it is still displayed in the search results list.

A user is unlikely to want to have a track duplicated in their playlist. By returning tracks in the search results that already exist in their playlist the user is prone to including multiple instances of a track in their playlist. There is also the disadvantage that the number of different tracks the user can see, and may wish to add, is lessened if tracks in the search results are duplicates.

This feature accomplishes the following:

- Displays only the tracks from a search that are not currently in the playlist
- Removes a track from the search results when added to the playlist
- Adds a track to the search results when removed from the playlist (if the search is applicable to that track and returns it in the results).

TECHNICAL DESIGN

When a user makes a search the method Spotify.search() is called which communications with Spotify via the API to return the track ID, name, artist, album and uri in array of tracks that match the search term input by the user.

The returned search result array is then stored as a state in the App component which then re-renders, passing the new searchResults state through the SearchResults component to the Tracklist component where a Track component is rendered for each track in the searchResults array. No further processing of the searchResults array occurs before being displayed to the user.

A new filteredSearchResults state will be required in the App component so that the returned searchResults from Spotify aren't overridden when an array with the duplicate tracks removed is returned. This allows the original searchResults to be consulted when a track is removed from the playlist to check whether that track should reappear in the search results.

This feature will require an intermediary step between retrieving the search result array and displaying it to the user. This step will:

1. Loop through all the tracks in the searchResults array
2. Match tracks that share the same ID with any of the tracks in the playlist
3. Remove a track from the searchResults array when a match is made
4. Set the state of searchResults to the new, filtered searchResults array once all the tracks in the have been checked
5. Render the updates searchResults array to the user

Step 1 can be achieved with a filter method called on the searchResults array using the parameter searchResult.
Steps 2 and 3 will work by firstly creating an array of the track IDs in the playlistTracks array which can then have the includes() method called to see if the searchResult ID is present. The value returned is either *true* or *false* of which the inverse will be used in order to return tracks to the new array only if they don't appear in the playlistTracks array.
Steps 4 and 5 can be achieved by using the setState() method which not only updates the value of the state specified but also calls for the corresponding component to be re-rendered.

The filterSearch() method will need to be called when:

- a new search is made
- a track is added to the playlist
- a track is removed from the playlist

This can be done by adding this function call to the end of the addTrack() and removeTrack() methods. However, these two methods themselves already call setState() in order to update the playlistTracks state. This is required since the updated playlistTracks array is needed in the filterSearch() method, however calling setState() in these methods followed by calling setState() in the filterSearch() method causes problems in the rendering. To avoid this setState() will only be called in filterSearch() since this method will always be called for removeTrack(), addTrack() and search(). The parameters that therefore need to be passed to the filterSearch() method are searchResults and playlistTracks.

Arguments to be passed to filterSearch() for each method:

addTrack(): filterSearch(this.state.searchResults, updatePlaylistTracks)

removeTrack(): filterSearch(this.state.searchResults, updatePlaylistTracks)

search(): filterSearch(results, this.state.playlistTracks)

CAVEATS

The proposed solution updates the state of playlistTracks or searchResults immediately prior to the filteredSearchResults. A better solution could be for the playlistTracks/searchResults to be updated at the end of the addTrack(), removeTrack() and search() methods before the filterSearch() method is called to then use the updated information. The problem with this is that it renders the DOM twice as mentioned in the technical details. An alternative solution that avoids the render when setState() is called in addTrack(), removeTrack() and search() could be a good option. The method shouldComponentUpdate(object nextProps, object nextState) always return true by default which could be overwritten in the component where necessary however this is advised against.