Feature Request: Search Result Paging

**May 2nd 2018**

# **OBJECTIVE**

Provide users of the Jammming application to review and select all results that match their search terms when there are more than 20 matching results.

# **BACKGROUND**

The Jammming application search function currently returns only 20 results – the default query limit. This feature request will add the ability for the user to request additional results if they exist on Spotify and “page” through them to review results that are not returned in the first 20. Because Spotify can match up to 10,000 tracks for a given query it is important to allow users to find and select tracks that are not in the 20 most popular matches for their query. The application will still display no more than 20 results at a time as more would potentially have a negative impact on user experience by making the page look crowded and many users will find the track they are looking for in the most popular (top 20) results.

The Search Result Paging feature will:

* Display a page forward and page backward button on the Results panel with up to 20 results at a time.
  + The page forward button will be active if there are more results than those already displayed to the user available
  + The page backward button will be active if the user has paged forward through the results so they can go back to previous result pages
* Hovering the mouse pointer over the page forward or page backward button will provide the user with a “text hint” for the function of the button.
* Cache page backward results to limit the number of queries sent to the Spotify Search API for the same information.

# **TECHNICAL DESIGN**

# **Page Offset**

The existing *this.state.searchResults* array of objects in the **App** component will be used to store the expanded results set. In order to fetch the additional results from the Spotify Web API /search endpoint an *&offset=* value will need to be provided. This same value can be used to track the offset of the currently displayed page in the Jammming results panel. A new state property labelled *this.state.searchResultsOffset* will be added in the **App** component to hold this value. The **maximum value** for *this.state.searchResultsOffset* is **9979** – this ensures that the maximum of 10000 supported by the Spotify Web API /search endpoint is not exceeded when the search *&limit=21* is added.

*this.state.searchResultsOffset* will be string concatenated with *this.state.searchTerm* by the **App** component **search()** method before being passed to **the Spotify.search()** function. The **App** component **search()** method currently uses the Object form of setState() to set *this.state.searchResults* to the array of objects returned from **Spotify.search()**. This will need to be updated to use the Callback form of setState() to concatenate the previous array with the new results. A prototype for the new **App** component **search()** method is:

|  |
| --- |
| search() {  const query = this.state.searchTerm + ‘&offset=’ + this.state.searchResultsOffset;  Spotify.search(query).then( searchResults => {  this.setState((previousState) => {  return { searchResults: previousState.searchResults.concat(searchResults) };  });  });  } |

this.state.searchResultsOffset will also be added as a property passed to the **SearchResults** component by the **App** component and then passed on to the **TrackList** component. The **Tracklist** component **render()** function will use the *.searchResultsOffset* to slice the appropriate elements of the *this.props.tracks* array before mapping the required object properties to the call to the **Track** component.

# **CAVEATS**

This section is used to lay out alternative solutions and their respective drawbacks, as well as potential drawbacks to the proposed solution above. This is used to make it clear why the technical implementation detailed previously was chosen instead of alternatives. It additionally allows stakeholders or other developers to consider those drawbacks and choose one of the alternate solutions if they prefer it. This may occur if they feel the benefits or drawbacks of that solution are more desirable than the current solution, or if they can identify other benefits and drawbacks not currently listed.