Project Report

Colin Legge, Jacob Ericson, Suqian Wang

Introduction

The problem we decided to solve was a recommendation problem. We decided to provide a recommendation to users of songs or playlists that they may be interested in based on our library of songs. We will be using the data gathered from Spotify as our source for providing our recommendations. The program will work with the user choosing to search for various keywords regarding artists or songs. Our program will then return a list of most relevant songs of the artist or a list of most relevant songs including the keywork. For each returned results we give our recommendations to the user on what that user would be interested in based on his/her search and our library of data. Our program will solve our problem because it will provide a recommendation to our users for songs, playlists, videos, and related songs based on their searches.

Prior Work

We didn't research any paper. We did use a lot of websites to solve obstacles along the way but no direct cite or use sample codes.

Method

Data Crawling

We used Spotipy and JSON library as our helper tools, logged in to spotify, created a spotify object and got the credentials manager to access other user's playlists and store playlists' info to the output file. This output file contains all the playlists information, usernames, playlist IDs, playlist names. Then we implemented a parser to parse the output file and extracted all playlists' ID for future use.

Finally we used Spotipy library again to iterated across playlist IDs and store songs' data to output files.
However, one output file have too many songs' information and we want individual song's information for out database, so we used JSON library to split the files so that each song's

information was in a separate output file. In this cases, we were able to get 131k documents easily for our database.

Building a Search Engine

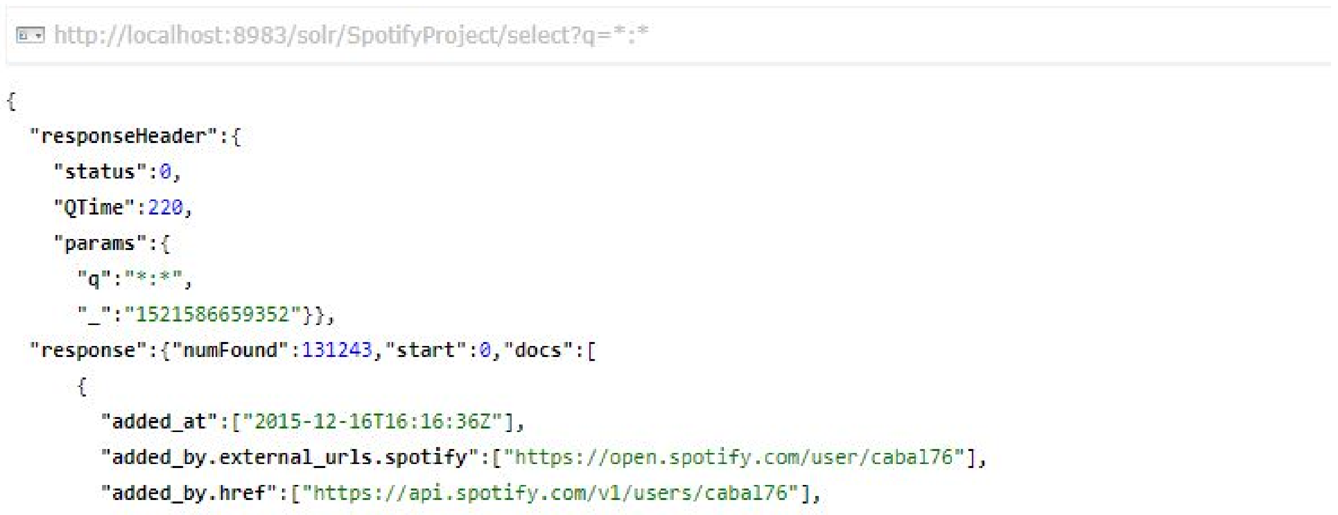
Our goal for this milestone is to build a search engine for the 131k documents we crawled in the previous task. We used Solr, a full-text search server based on Lucene, the library because Solr has JSON Interfaces which is perfect for our documents.

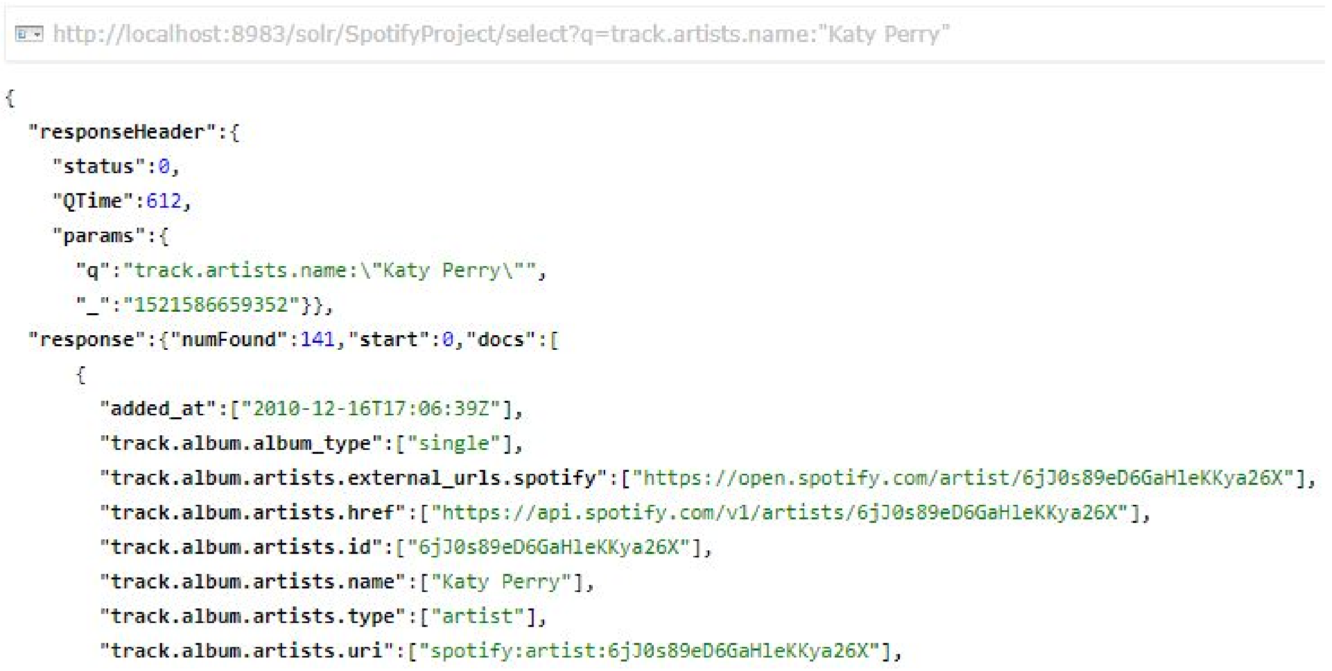
First, we worked on setting up the Solr server:

- download Solr from http://apache.mesi.com.ar/lucene/solr/7.2.1/
- launch Solr from bin folder: ./solr start
- create a core in order to be index and search and assigned a port number for it
- use bin/post tool and indexed our documents on the server

Then, we tested our Solr server by performing searching operation on it using wild-card queries or specific keywords:

- searching all documents: q = *:*
- searching certain artist: q = track.artists.name:“Katy Perry”
- searching a certain song: q = track.name:“I knew you were trouble”
- searching using wild-card query: q = track.artists.name:ade*

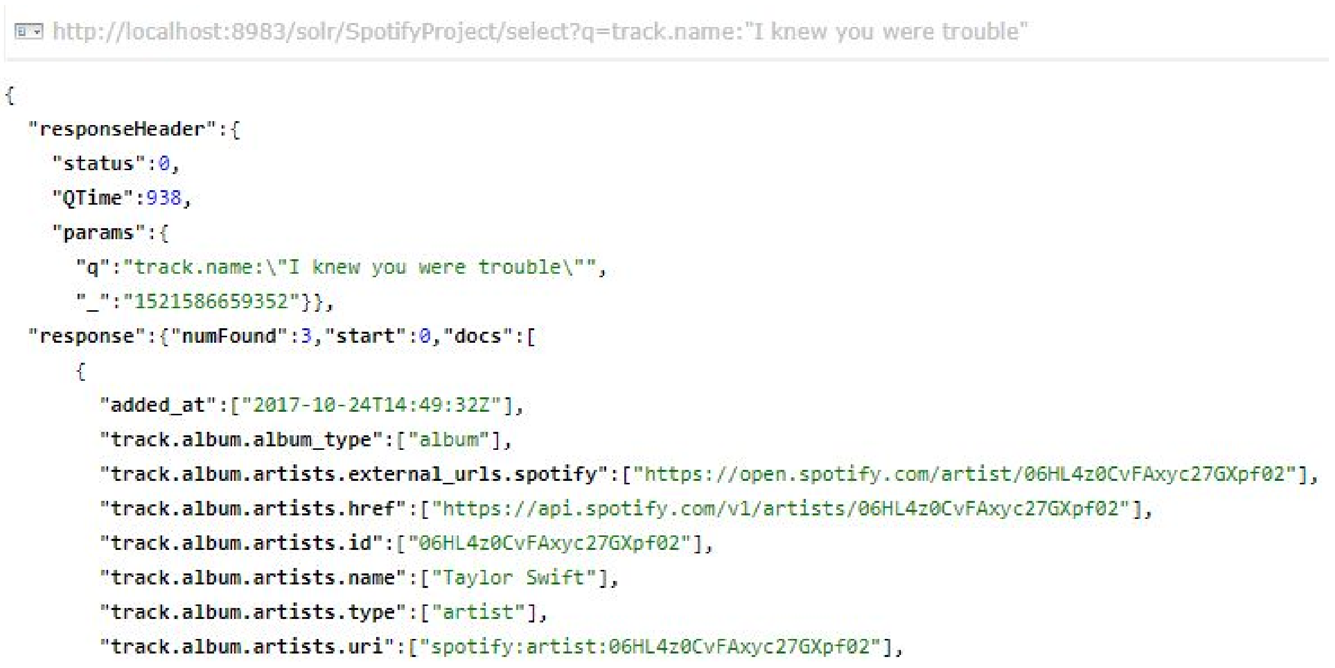

```
http://localhost:8983/solr/SpotifyProject/select?q=*:*

{
  "responseHeader":{
    "status":0,
    "QTime":220,
    "params":{
      "q":"*:*",
      "_":"1521586659352"}},
  "response":{"numFound":131243,"start":0,"docs":[
      {
        "added_at":["2015-12-16T16:16:36Z"],
        "added_by.external_urls.spotify":["https://open.spotify.com/user/cabal76"],
        "added_by.href":["https://api.spotify.com/v1/users/cabal76"],
```
Figure 1: searching all document using wild-card query



```
http://localhost:8983/solr/SpotifyProject/select?q=track.artists.name:"Katy Perry"
```

```
{
  "responseHeader":{
    "status":0,
    "QTime":612,
    "params":{
      "q":"track.artists.name:\"Katy Perry\"",
      "_":"1521586659352"}},
  "response":{"numFound":141,"start":0,"docs":[
      {
        "added_at":["2010-12-16T17:06:39Z"],
        "track.album.album_type":["single"],
        "track.album.artists.external_urls.spotify":["https://open.spotify.com/artist/6jJ0s89eD6GaHleKKya26X"],
        "track.album.artists.href":["https://api.spotify.com/v1/artists/6jJ0s89eD6GaHleKKya26X"],
        "track.album.artists.id":["6jJ0s89eD6GaHleKKya26X"],
        "track.album.artists.name":["Katy Perry"],
        "track.album.artists.type":["artist"],
        "track.album.artists.uri":["spotify:artist:6jJ0s89eD6GaHleKKya26X"],
```

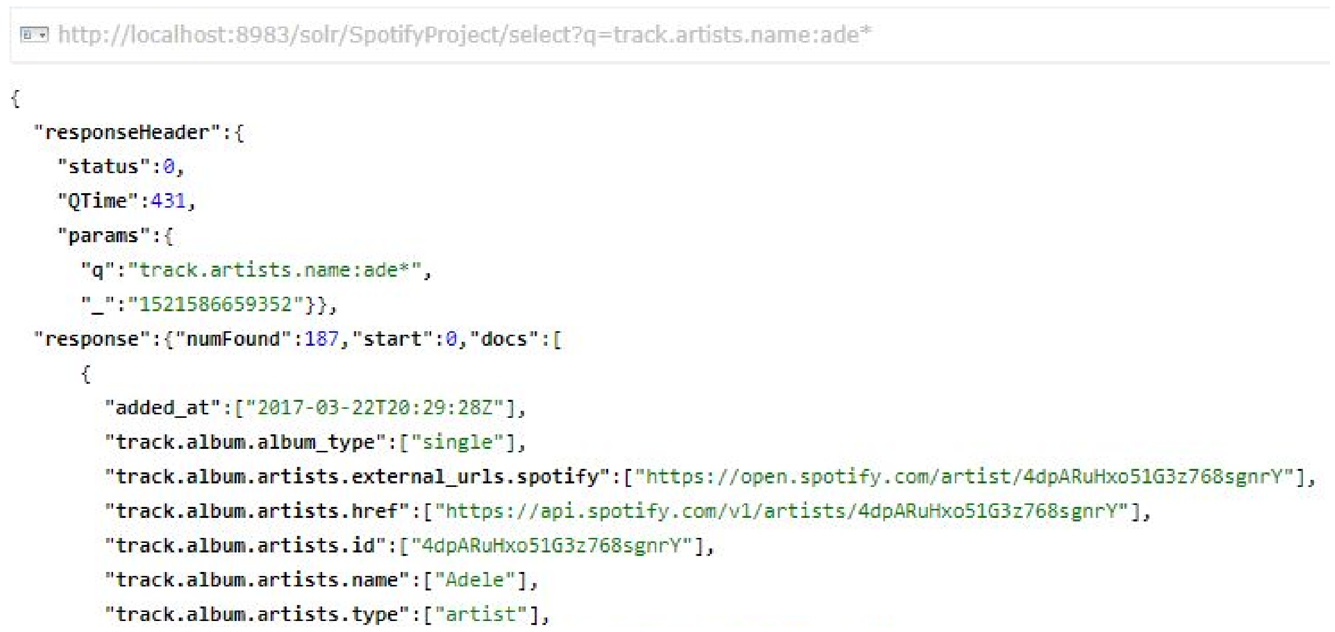
Figure 2: searching artist using keyword



```
http://localhost:8983/solr/SpotifyProject/select?q=track.name:"I knew you were trouble"
```

```
{
  "responseHeader":{
    "status":0,
    "QTime":938,
    "params":{
      "q":"track.name:\"I knew you were trouble\"",
      "_":"1521586659352"}},
  "response":{"numFound":3,"start":0,"docs":[
      {
        "added_at":["2017-10-24T14:49:32Z"],
        "track.album.album_type":["album"],
        "track.album.artists.external_urls.spotify":["https://open.spotify.com/artist/06HL4z0CvFAxyc27GXpf02"],
        "track.album.artists.href":["https://api.spotify.com/v1/artists/06HL4z0CvFAxyc27GXpf02"],
        "track.album.artists.id":["06HL4z0CvFAxyc27GXpf02"],
        "track.album.artists.name":["Taylor Swift"],
        "track.album.artists.type":["artist"],
        "track.album.artists.uri":["spotify:artist:06HL4z0CvFAxyc27GXpf02"],
```

Figure 3: searching song track using keyword



```
http://localhost:8983/solr/SpotifyProject/select?q=track.artists.name:ade*
```

```
{
  "responseHeader":{
    "status":0,
    "QTime":431,
    "params":{
      "q":"track.artists.name:ade*",
      "_":"1521586659352"}},
  "response":{"numFound":187,"start":0,"docs":[
      {
        "added_at":["2017-03-22T20:29:28Z"],
        "track.album.album_type":["single"],
        "track.album.artists.external_urls.spotify":["https://open.spotify.com/artist/4dpARuHxo51G3z768sgnrY"],
        "track.album.artists.href":["https://api.spotify.com/v1/artists/4dpARuHxo51G3z768sgnrY"],
        "track.album.artists.id":["4dpARuHxo51G3z768sgnrY"],
        "track.album.artists.name":["Adele"],
        "track.album.artists.type":["artist"],
```

Figure 4: searching artist using wild-card query

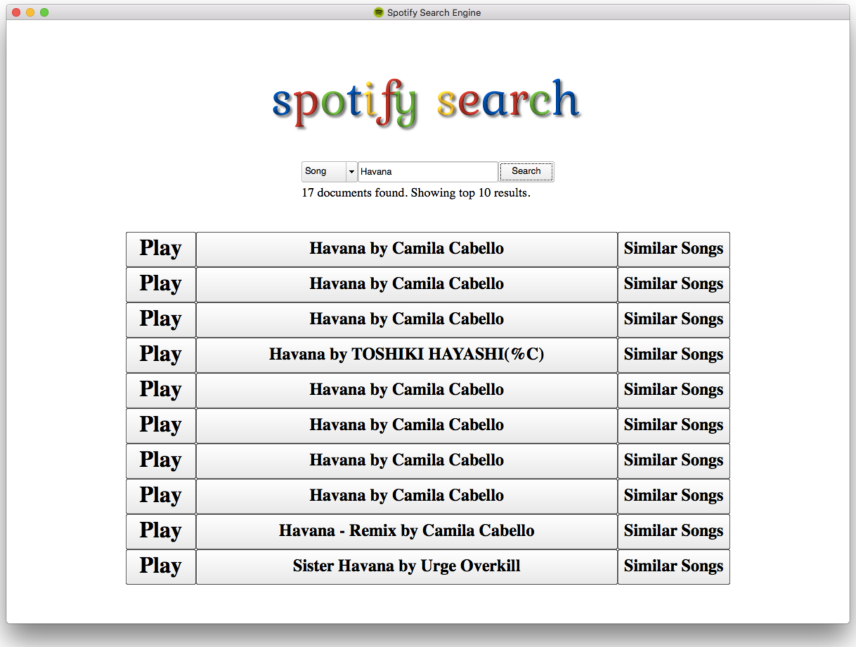
During the process, we ran into a few problems. In the first task, we got 131k documents each represent a different song's data. When trying to indexing all of them, the server gives error saying argument list is too long. Therefore, we have to divide all 131k documents into 10 subsets and repeat the indexing command for each subset. But this works on some computer but not all of them so we have to future divide all 131k documents into 1000 subsets for it to works on all of us computers.

However, indexing 131k documents is not very time efficient, especially the more subsets we divide, the slower the indexing time will be. Therefore we tried to use one big JSON file to save the Solr indexing time, but we failed due to Solr's file size limit. To solve this problem, we get a smaller amount of documents which group the information by playlists which inspired us to create two nodes, one for all documents with different songs and the other for different playlists that contain the same song information.

Spotify Search Application

We used a python API, simplejson library, that can be used to establish a connection with Solr database and perform search queries on it. Also, we used the PyQt library to make a window with Python. The window class has functions that set up the configurations including connections with Solr, window size, title, icon, style, search type, query input, etc.

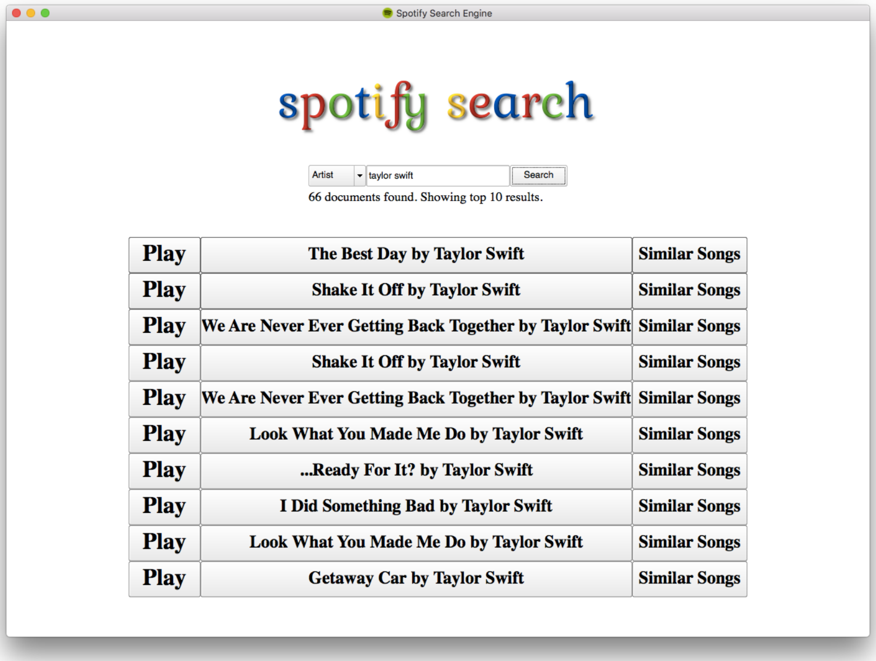
We designed our search logo by the inspiration from Google's icon and created a drop down box for users to choose either song or artist regarding what they want to search for. We chose to display ten most relevant results and display the result with one button that allows users to see our recommendation based on their search and the other button that allows users to listen to the song in Spotify.



17 documents found. Showing top 10 results.

Play	Havana by Camila Cabello	Similar Songs
Play	Havana by Camila Cabello	Similar Songs
Play	Havana by Camila Cabello	Similar Songs
Play	Havana by TOSHIKI HAYASHI(%C)	Similar Songs
Play	Havana by Camila Cabello	Similar Songs
Play	Havana by Camila Cabello	Similar Songs
Play	Havana by Camila Cabello	Similar Songs
Play	Havana by Camila Cabello	Similar Songs
Play	Havana - Remix by Camila Cabello	Similar Songs
Play	Sister Havana by Urge Overkill	Similar Songs

Figure 5: search song keyword



66 documents found. Showing top 10 results.

Play	The Best Day by Taylor Swift	Similar Songs
Play	Shake It Off by Taylor Swift	Similar Songs
Play	We Are Never Ever Getting Back Together by Taylor Swift	Similar Songs
Play	Shake It Off by Taylor Swift	Similar Songs
Play	We Are Never Ever Getting Back Together by Taylor Swift	Similar Songs
Play	Look What You Made Me Do by Taylor Swift	Similar Songs
Play	...Ready For It? by Taylor Swift	Similar Songs
Play	I Did Something Bad by Taylor Swift	Similar Songs
Play	Look What You Made Me Do by Taylor Swift	Similar Songs
Play	Getaway Car by Taylor Swift	Similar Songs

Figure 6: search artist keyword

Results and Findings

In addition to the evaluation of the correctness of our programs and their functionalities we performed by ourselves, which was shown in Figure 1 to Figure 6 above, more details will be shown in the video. Our application is able to perform its designed functions as we expected. We also gave out a survey to gather our "users" opinion about our application. The results of the survey were shown in the chart below.

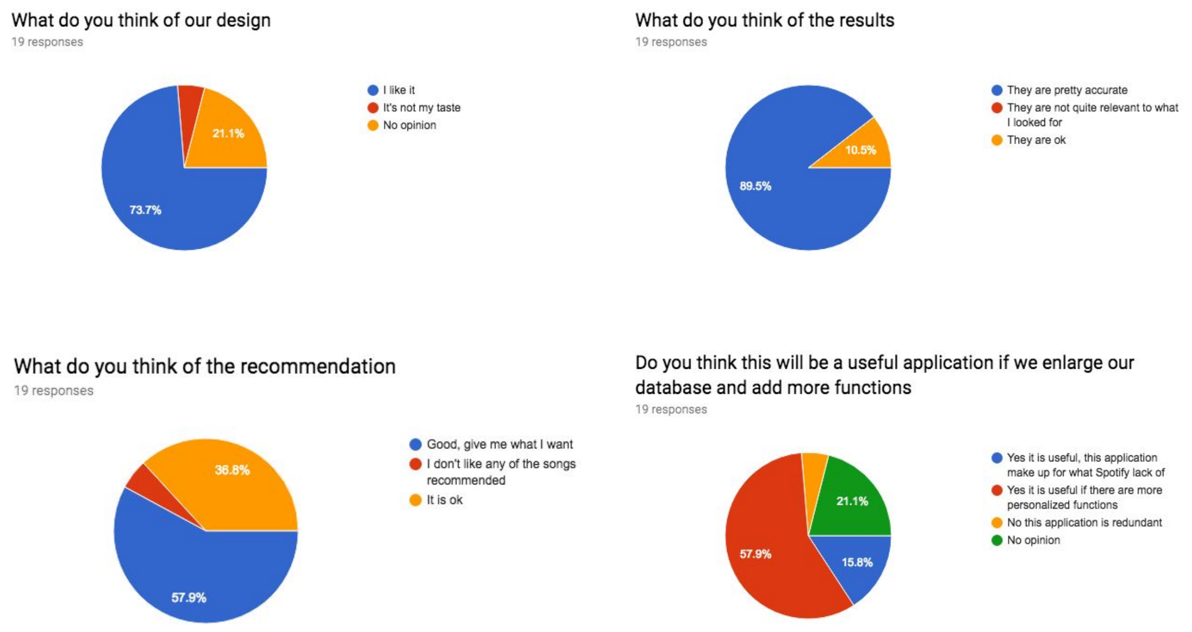


Table 1: Spotify Search Survey Result

Above all, 73.7% participants like our design, 89.5% participants think our application gives out accurate results, 94.7% participants think our recommendation is ok or good, and 73.7% participants think our application is useful. Therefore, the functionalities of our application are proved to be working pretty well. The application is well designed and the purpose of this application is proved to be effective. If we were given more time, we can add more functionalities to our application and improve the recommendation more.