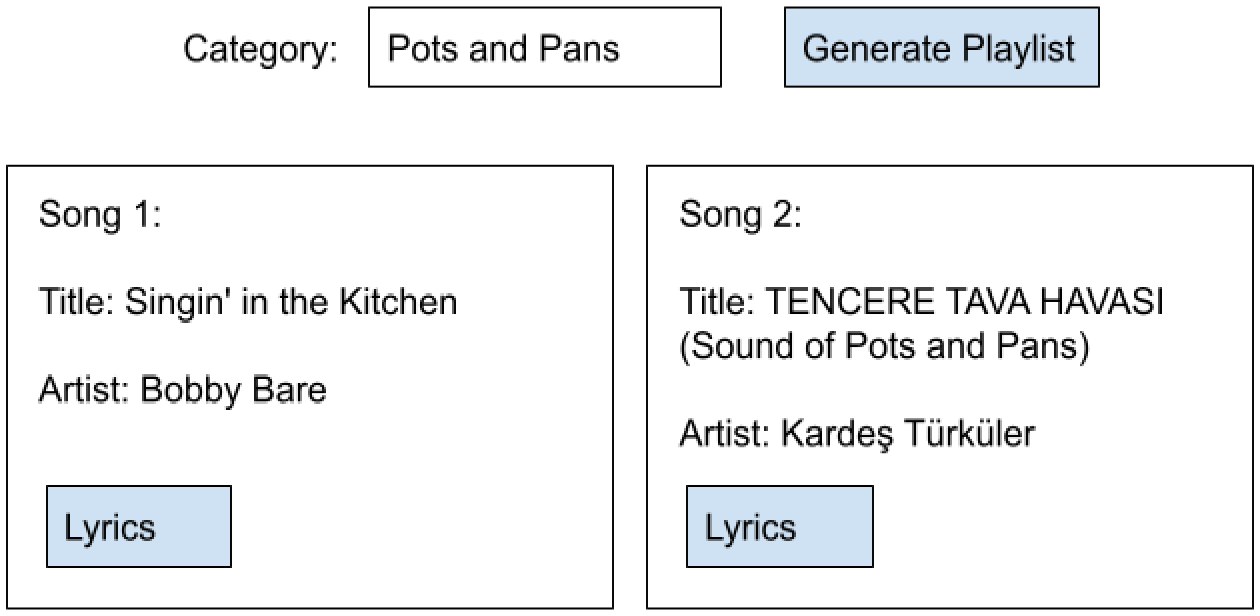
Our users find our app to be very engaging, but there is always one key piece missing. There's no sound! In a digital world filled with distractions like Netflix and Youtube we've realized that we need to add a soundtrack to accompany users as they determine their digital marketing strategy.

Therefore, we want to generate a "smart" playlist that is based off of the product category that an advertiser sells in.

The goal of this task is to **build an API** that is able to serve data for the following view:


Category: Pots and Pans

Song 1:

Title: Singin' in the Kitchen

Artist: Bobby Bare

Song 2:

Title: TENCERE TAVA HAVASI (Sound of Pots and Pans)

Artist: Kardeş Türküler

API Requirements:

Users will enter the category text and the API will return 2 songs immediately, including the title, artist and lyrics. Note that the actual music file doesn't need to be returned as the frontend will handle that.

Once the frontend has played the first song it will be removed from the list, the second song will start playing and a 3rd song will be fetched with a subsequent request to the backend. This process will repeat with the frontend needing to fetch the 4th, 5th, etc song in subsequent requests.

How To Generate the Playlist:

Because this is a "smart" playlist we want to pick songs using the category name as the starting point, but have each song after the first be as similar as possible in lyrics to the song before.

To do this we use the MusixMatch API (described further below). We first use the **track.search** API to fetch the first song using the “Any word in the lyrics” or `q_lyrics` query parameter. We choose the first song returned.

Next, for every subsequent song we want to take **5** random words from the lyrics in the previous song, and combine those to use as the new `q_lyrics` query parameter. This process is repeated for every further song in the playlist.

The playlist should technically be infinite until the chain is broken by either no search results or a song with no lyrics.

Note that no song should ever repeat in a playlist.

API

This project will need to use the MusixMatch API which is documented here: https://developer.musixmatch.com/documentation

It is much easier to understand how to use the API by using the API playground as the documentation is somewhat dense: https://playground.musixmatch.com/

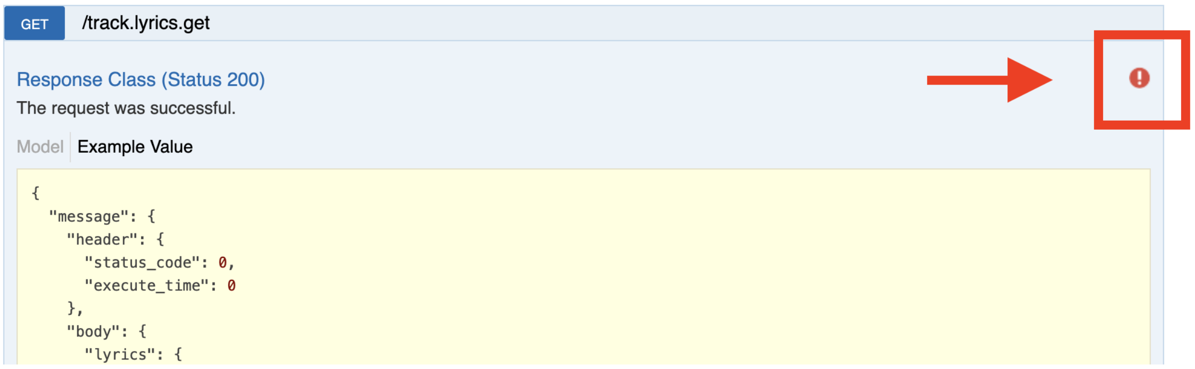
In particular look at:
- https://playground.musixmatch.com/#!/Track/get_track_search
- https://playground.musixmatch.com/#!/Lyrics/get_track_lyrics_get

API Keys to Use
- 12832752fc79e94b205e24b1a5bf7a4d
- c84b0eb027afd7ec974d2bd2073da031

Each API key allows for up to 2000 API calls per day which should be enough for this task .The second key is there in case you accidentally run out of daily requests using the first one.

Hints for using the API

Make sure to authorize the requests in the playground with the keys using the red icon at the top right of the request:


GET /track.lyrics.get

Response Class (Status 200)

The request was successful.

Model | Example Value

```
{
  "message": {
    "header": {
      "status_code": 0,
      "execute_time": 0
    },
    "body": {
      "lyrics": {
```

If the playground doesn't work for some reason still here is an example curl request to get you started:

```
curl -X GET --header 'Accept: text/plain'
'https://api.musixmatch.com/ws/1.1/track.search?format=jsonp&callback=callback&
q_lyrics=driving&quorum_factor=1&apikey=12832752fc79e94b205e24b1a5bf7a4d'
```

Bonus (Extra Points):

If you manage to finish the above task before time runs out and want to add some bonus points, consider either:

- Using a more intelligent approach to picking the 5 random words
- Hosting your server somewhere so it can be accessed publicly