**Assignment:**

1. Write a script that receives an argument and queries the Musixmatch api with the argument as a query string. Create a list of songs with the following attributes: - Song lyrics contains the argument supplied - Song language is English - Included in an album, with release date prior to 01-01-2010 (Album release date, not the song as a single) If no argument is passed, the default argument is “car” Save the results into CSV file in the following format: song name, performer name, album name, song share URL.

2. Create a docker file exposing a service that receives a keyword and uses it as an input for the same query then prints the list of songs (you can use the script from the previous step as a base for the service). You should start with the minimal alpine image, then add any needed software in order to run the script and print the results.

3. Send the script, Dockerfile and any other file if needed, compressed.

**Assignment Response:**

**Examples – playing with API**

Look for lyrics with car in it:

curl "https://api.musixmatch.com/ws/1.1/track.search?q\_lyrics=car&page\_size=100&page=1&apikey=3621259dea6ab9eabdeb5dca97728d1b" | jq .

Output CSV list of data

curl "https://api.musixmatch.com/ws/1.1/track.search?q\_lyrics=car&f\_lyrics\_language=en&f\_track\_release\_group\_first\_release\_date\_max=2010101&apikey=3621259dea6ab9eabdeb5dca97728d1b&page=1&page\_size=100" | jq '.message.body.track\_list[].track | [.track\_name, .artist\_name, .album\_name, .track\_share\_url] | @csv'

**Script for Part 1:**

#!/usr/bin/python3

import requests

import csv

import sys

# Set default query argument if none is passed

if len(sys.argv) < 2:

query = "car"

else:

query = sys.argv[1]

# Create an empty list to store the song data

song\_data = []

for page\_number in range(1,2):

# Musixmatch API endpoint and parameters

endpoint = "https://api.musixmatch.com/ws/1.1/track.search"

params = {

"q\_lyrics": query,

"f\_has\_lyrics": 1,

"f\_lang\_code": "en", ### Hardcoded english language

"f\_track\_release\_group\_first\_release\_date\_max":"2010101", ## Hardcoded release date of song

"page\_size": 20, ## Hardcode page size - so as not to overload API

"page": page\_number,

"apikey": "3621259dea6ab9eabdeb5dca97728d1b" ## Hardcoded API key

}

# Send the API request and get the JSON response

response = requests.get(endpoint, params=params).json()

# Iterate through the tracks in the response

for track in response["message"]["body"]["track\_list"]:

# Get the song data

track\_name = track["track"]["track\_name"]

artist\_name = track["track"]["artist\_name"]

album\_name = track["track"]["album\_name"]

track\_share\_url = track["track"]["track\_share\_url"]

try:

album\_id = track["track"]["album\_id"]

# Get the album release date

endpoint2 = "https://api.musixmatch.com/ws/1.1/album.get"

params2 = {

"album\_id": album\_id,

"apikey": "3621259dea6ab9eabdeb5dca97728d1b"

}

response2 = requests.get(endpoint2, params=params2).json()

album\_details=response2["message"]["body"]["album"]

album\_release\_date = album\_details["album\_release\_date"]

if album\_release\_date < "2010-01-01":

# Add the song data to the list

song\_data.append([track\_name, artist\_name, album\_name, track\_share\_url])

except:

pass

# Write the song data to a CSV file

filename="songs\_with\_"+query+".csv"

with open(filename, "w", newline="") as csvfile:

writer = csv.writer(csvfile)

writer.writerow(["Track name", "Artist Name", "Album name","Track Share URL"])

writer.writerows(song\_data)

print(f"{len(song\_data)} song(s) found and saved to",filename)

**Running the Script:**

﻿***With no parameters:***

root@ubuntu2004:~# python3 ./getPythonSongs.py

18 song(s) found and saved to songs\_with\_car.csv

root@ubuntu2004:~# cat songs\_with\_car.csv

Track name,Artist Name,Album name,Track Share URL

Metal Killed My Cat,30footFALL,"Ever Revolving, Never Evolving",https://www.musixmatch.com/lyrics/30footfall/Metal-Killed-My-Cat?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494

My Car,Atlas Sound,Things I&apos;ll Miss EP,https://www.musixmatch.com/lyrics/Atlas-Sound/My-Car?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494

302 Cubic Inch V-8 Powered Blues / Ride With Zeke / Lawson (live),Zeke,Live and Uncensored,https://www.musixmatch.com/lyrics/Zeke/302-Cubic-Inch-V-8-Powered-Blues-Ride-With-Zeke-Lawson-live?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494

….

***With a parameter:***

﻿

root@ubuntu2004:~# python3 ./getPythonSongs.py rabbit

16 song(s) found and saved to songs\_with\_rabbit.csv

root@ubuntu2004:~# cat songs\_with\_rabbit.csv

Track name,Artist Name,Album name,Track Share URL

I Hate Rabbits,Eddie Cochran,Rock and Roll Blues,https://www.musixmatch.com/lyrics/Eddie-Cochran/I-Hate-Rabbits?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494

Spirit of Dunkirk,Camberwell Now,All&apos;s Well,https://www.musixmatch.com/lyrics/Camberwell-Now/Spirit-of-Dunkirk?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494

One.Wolverine,Sub.Bionic,How.to.Build.A.Casket,https://www.musixmatch.com/lyrics/Sub-Bionic/One-Wolverine?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494**Part 2:**

Update python script to use flask to enable to be put into the docker service.

﻿

import requests

import csv

import sys

import os

from flask import Flask, jsonify, request, Response

def get\_lyric\_data(query):

# Create an empty list to store the song data

song\_data = []

for page\_number in range(1,2):

# Musixmatch API endpoint and parameters

endpoint = "https://api.musixmatch.com/ws/1.1/track.search"

params = {

"q\_lyrics": query,

"f\_has\_lyrics": 1,

"f\_lang\_code": "en", ### Hardcoded english language

"f\_track\_release\_group\_first\_release\_date\_max":"2010101", ## Hardcoded release date of song

"page\_size": 20, ## Hardcode page size - so as not to overload API

"page": page\_number,

"apikey": "3621259dea6ab9eabdeb5dca97728d1b" ## Hardcoded API key

}

# Send the API request and get the JSON response

response = requests.get(endpoint, params=params).json()

# Iterate through the tracks in the response

for track in response["message"]["body"]["track\_list"]:

# Get the song data

track\_name = track["track"]["track\_name"]

artist\_name = track["track"]["artist\_name"]

album\_name = track["track"]["album\_name"]

track\_share\_url = track["track"]["track\_share\_url"]

try:

album\_id = track["track"]["album\_id"]

# Get the album release date

endpoint2 = "https://api.musixmatch.com/ws/1.1/album.get"

params2 = {

"album\_id": album\_id,

"apikey": "3621259dea6ab9eabdeb5dca97728d1b"

}

response2 = requests.get(endpoint2, params=params2).json()

album\_details=response2["message"]["body"]["album"]

album\_release\_date = album\_details["album\_release\_date"]

if album\_release\_date < "2010-01-01":

# Add the song data to the list

song\_data.append([track\_name, artist\_name, album\_name, track\_share\_url])

except:

pass

filename="songs\_with\_"+query+".csv"

with open(filename, "w", newline="") as csvfile:

writer = csv.writer(csvfile)

writer.writerow(["Track name", "Artist Name", "Album name","Track Share URL"])

writer.writerows(song\_data)

return filename

# Create a flask app for a python app

app = Flask(\_\_name\_\_)

@app.route(‘/’ defaults={‘query’; : ‘car’})

@app.route(‘string:query>’)

def lyric(query):

my\_filename = get\_lyric\_data(query)

file\_contents = open (my\_filename,"r").read()

os.remove(my\_filename)

return (file\_contents.split(&apos;\n&apos;))

# main function

if \_\_name\_\_ == ‘\_\_main\_\_’:

app.run(host="0.0.0.0", port=5000, debug = True)

Note, the highlighted information has been added on, and we have made the working part of the script into a function.

Now when we run the application – this is what we see:

﻿

root@ubuntu2004:~/getLyrics# python3 docker\_getPythonSongs.py

\* Serving Flask app &apos;docker\_getPythonSongs&apos;

\* Debug mode: on

**WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.**

\* Running on all addresses (0.0.0.0)

\* Running on http://127.0.0.1:5000

\* Running on http://10.0.2.15:5000

Press CTRL+C to quit

\* Restarting with stat

\* Debugger is active!

\* Debugger PIN: 473-880-095

If we send a request from another terminal, we get the following output:

Curl with no input – defaults to ‘car’ as before:

﻿root@ubuntu2004:~# curl http://127.0.0.1:5000

[

"Track name,Artist Name,Album name,Track Share URL",

"Metal Killed My Cat,30footFALL,\"Ever Revolving, Never Evolving\",https://www.musixmatch.com/lyrics/30footfall/Metal-Killed-My-Cat?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

"My Car,Atlas Sound,Things I&apos;ll Miss EP,https://www.musixmatch.com/lyrics/Atlas-Sound/My-Car?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

"302 Cubic Inch V-8 Powered Blues / Ride With Zeke / Lawson (live),Zeke,Live and Uncensored,https://www.musixmatch.com/lyrics/Zeke/302-Cubic-Inch-V-8-Powered-Blues-Ride-With-Zeke-Lawson-live?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

…

]

If we send with a parameter – we get the output for the parameter input:

root@ubuntu2004:~# curl http://127.0.0.1:5000/rabbit

[

"Track name,Artist Name,Album name,Track Share URL",

"I Hate Rabbits,Eddie Cochran,Rock and Roll Blues,https://www.musixmatch.com/lyrics/Eddie-Cochran/I-Hate-Rabbits?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

"Spirit of Dunkirk,Camberwell Now,All&apos;s Well,https://www.musixmatch.com/lyrics/Camberwell-Now/Spirit-of-Dunkirk?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

"One.Wolverine,Sub.Bionic,How.to.Build.A.Casket,https://www.musixmatch.com/lyrics/Sub-Bionic/One-Wolverine?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

"Hound Dog (Minus All Guitars) [In the Style of &apos;Elvis Presley&apos;],Backing Tracks For Guitarists,Backing Tracks for Guitarists - Volume 4,https://www.musixmatch.com/lyrics/Backing-Tracks-For-Guitarists/Hound-Dog-Minus-All-Guitars-In-the-Style-of-Elvis-Presley?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

…

]

To docker this – create the docker file as follows:

FROM alpine:latest

# Install python/pip and other requirements

ENV PYTHONUNBUFFERED=1

RUN apk add --update --no-cache python3 && ln -sf python3 /usr/bin/python

RUN python3 -m ensurepip

RUN pip3 install --no-cache --upgrade pip setuptools

RUN pip3 install requests flask

# Copy the script file into the container

COPY docker\_getPythonSongs.py .

# Make the script executable

RUN chmod +x docker\_getPythonSongs.py

# Docker service should run on port 80

EXPOSE 80

CMD ["python3","docker\_getPythonSongs.py"]

Build the docker file as follows:

docker build -t check-lyrics .

Then run the docker image as follows:

﻿

root@ubuntu2004:~/getLyrics# docker run -d -p 80:5000 check-lyrics

37828133c409c2eecf1d973477ab73452b3efec6a1e8688cd55bbed9879bbed3

root@ubuntu2004:~/getLyrics#

If we run a docker ps – we see that it is running:

﻿

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

37828133c409 check-lyrics "python3 docker\_getP…" 53 seconds ago Up 52 seconds 80/tcp, 0.0.0.0:80->5000/tcp, :::80->5000/tcp pedantic\_poitras

root@ubuntu2004:~/getLyrics#

From another second terminal – we now see that a post to port 5000 fails:

﻿

root@ubuntu2004:~# curl http://localhost:5000

curl: (7) Failed to connect to localhost port 5000 after 0 ms: Couldn&apos;t connect to server

But a post to port 80 (the exposed port) – works as before:

﻿***Without a parameter: ( defaults to ‘car’)***

root@ubuntu2004:~# curl http://127.0.0.1

[

"Track name,Artist Name,Album name,Track Share URL",

"Metal Killed My Cat,30footFALL,\"Ever Revolving, Never Evolving\",https://www.musixmatch.com/lyrics/30footfall/Metal-Killed-My-Cat?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

"My Car,Atlas Sound,Things I&apos;ll Miss EP,https://www.musixmatch.com/lyrics/Atlas-Sound/My-Car?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

"302 Cubic Inch V-8 Powered Blues / Ride With Zeke / Lawson (live),Zeke,Live and Uncensored,https://www.musixmatch.com/lyrics/Zeke/302-Cubic-Inch-V-8-Powered-Blues-Ride-With-Zeke-Lawson-live?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

"The Moth,Reaching Quiet,In the Shadow of the Living Room,https://www.musixmatch.com/lyrics/Reaching-Quiet/The-Moth?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

…

]

***With a parameter (rabbit):***

root@ubuntu2004:~# curl http://127.0.0.1/rabbit

[

"Track name,Artist Name,Album name,Track Share URL",

"I Hate Rabbits,Eddie Cochran,Rock and Roll Blues,https://www.musixmatch.com/lyrics/Eddie-Cochran/I-Hate-Rabbits?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

"Spirit of Dunkirk,Camberwell Now,All&apos;s Well,https://www.musixmatch.com/lyrics/Camberwell-Now/Spirit-of-Dunkirk?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

"One.Wolverine,Sub.Bionic,How.to.Build.A.Casket,https://www.musixmatch.com/lyrics/Sub-Bionic/One-Wolverine?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

"Hound Dog (Minus All Guitars) [In the Style of &apos;Elvis Presley&apos;],Backing Tracks For Guitarists,Backing Tracks for Guitarists - Volume 4,https://www.musixmatch.com/lyrics/Backing-Tracks-For-Guitarists/Hound-Dog-Minus-All-Guitars-In-the-Style-of-Elvis-Presley?utm\_source=application&utm\_campaign=api&utm\_medium=Private%3A1409623098494",

…

]

**Points for improvements:**

1. We could make an argument, so that the output from the REST can come out in different formats – e.g., CSV, Json or whatever we need.
2. We could make the album lookup – a little more efficient – by using a dictionary, we could store existing album information in memory, so that if the same album id is returned, we could make use of the cached information, instead of looking it up again.
3. Now, we have limited the output of the API to only one or two pages of information – we could add in more arguments that would specify how many pages of output we want – we were limited by using a non-commercial (home) license.
4. We could add in some more error checking – for example, if the lyric does not exist.

***Mark Prager, January 2023***