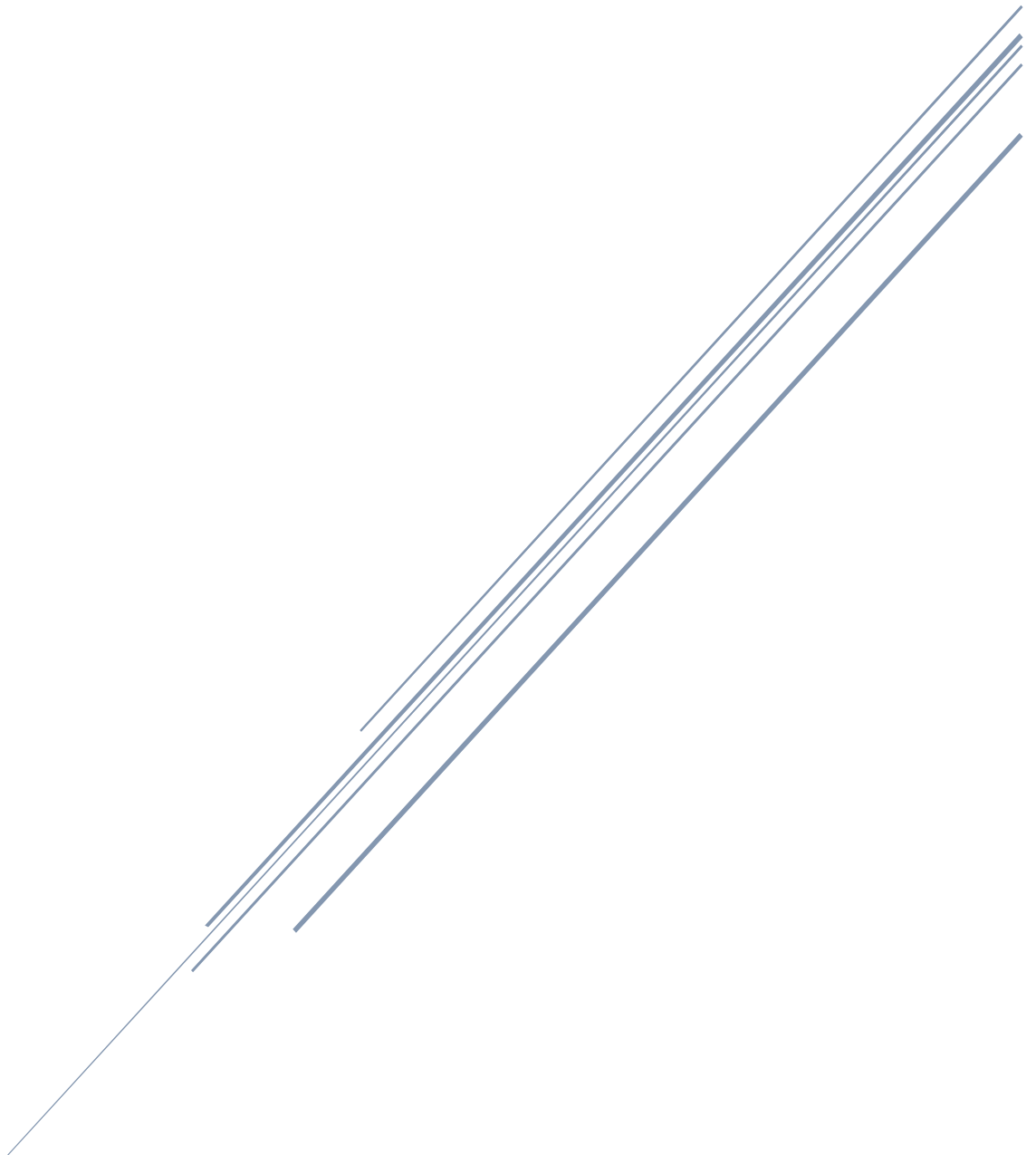


NETWORK PROGRAMABILITY

ASSIGNMENT 2 – CODE IMPLEMENTATION



Cybersecurity and Networks
Che Morris - 2203458

Table of Contents

Introduction of Project.....	2
Implementation	2
Function (lastfm.py): callAPI	2
Function (lastfm.py): getTrackInfo	3
Function (lastfm.py): downloadCoverArt.....	3
Function (lastfm.py): getAlbumSongs	4
Function (lastfm.py): findSimilarArtists	4
YouTube.py.....	5
Downloading the songs.....	5
MP3 Conversion	5
Tagging the MP3 using LastFM data	6
Main.py	7

Introduction of Project

The project that I've decided to create something that would be useful to me in the form of creating a song downloader that uses an API to automatically add metadata to the songs as I listen to all my music locally and if I add it to the music manager I use and it isn't tagged, it won't show correctly. I aimed for multifunctionality here with being able to download single songs and whole albums as this would help me hugely in obtaining songs since instead of it being a huge job where I'd have to do it all myself, this automates the entire process.

After looking at and trying out multiple APIs, I ended up deciding on using the Last.FM API as it's something I'm quite familiar with and have used previously to create scripts. This allowed me to use multiple features such as, getting the track information (track name, album name, track number and album cover art), getting information about a whole album (used for getting track names) and getting similar artists to the one the user chose to download.

To create what I wanted to create I had to use a few libraries that you'll have to install, these can be found in requirements.txt on the GitHub repo and installed via pip (pip install -r requirements.txt), such as pytube for downloading the content, moviepy for converting to mp3 from mp4 since pytube doesn't support this and eyed3 for adding the metadata to the files.

If you wish to not use my API key, you can if you would like to, where to change it can be found on line 3 of the lastfm.py file. I've also provided example data that can be used within the readme on the GitHub project.

Implementation

To implement the API into what I want to do, I decided to use multiple files to do this as it's just a habit I have for things to be organized while I'm coding so there's three files 'main.py' which is what you would run to actually use the program, 'lastfm.py' which contains all of the code for communicating with the Last.FM api and 'youtube.py' which has all the code for downloading the songs from YouTube. I won't touch massively on the YouTube side of things here as most of that is just using the pytube library and I don't talk to any APIs throughout that, but I'll mention it later.

Within the lastfm file, I created multiple functions for collecting various data from the API to be used later.

Function (lastfm.py): callAPI

```
def callAPI(artist, track):
    url = "http://ws.audioscrobbler.com/2.0/?method=track.getInfo&api_key=" + lastFmApiKey + "&artist=" + artist + "&track=" + track + "&format=json"
    response = requests.get(url)

    if response.status_code != 200:
        return {"error": "Error " + str(response.status_code) + ": " + response.reason}
    else:
        return response.json()
```

You, 2 hours ago • error handling for api

Starting off, I created the callAPI function. This is just so that I didn't have to write out those few lines of code for the functions 'getTrackInfo' and 'downloadCoverArt' as they both get information from the same place and I'm just a bit lazy. This includes error handling and will return an error if the status code isn't 200 (OK) and later in the other two functions they stop if 'error' is included in the response.

Function (lastfm.py): getTrackInfo

```
def getTrackInfo(artist, track):
    response = callAPI(artist, track)
    if "error" in response:
        return None
    else:
        trackInfo = response["track"]
        artistName = trackInfo["artist"]["name"]
        trackName = trackInfo["name"]
        albumName = trackInfo["album"]["title"]
        try:
            trackNumber = trackInfo["album"]["@attr"]["position"]
        except KeyError:
            trackNumber = 0 # sometimes the track number isn't set on LastFM so set it to 0

    return artistName, trackName, albumName, trackNumber
```

The 'getTrackInfo' function is used to gather the information about the specific track, to search through the LastFM API and get accurate information, you need to specify both the artist and the track so this is what this function will look for. It gathers the artist's name, track name and album name to begin with and then I ran into a small hurdle where the program would stop completely if the track's position in the album was missing, which thankfully isn't on many tracks but just happened to be on one I tried (Liam Gallagher – Wall of Glass if you want to try it) so I added a try statement there to catch if it isn't there and will just replace it with 0 if so.

Function (lastfm.py): downloadCoverArt

```
def downloadCoverArt(artist, track):
    response = callAPI(artist, track)
    if "error" in response:
        return None
    else:
        trackInfo = response["track"]
        coverArtUrl = trackInfo["album"]["image"][3]["#text"]
        coverArtImage = requests.get(coverArtUrl).content

        with open("cover.png", "wb") as file:
            file.write(coverArtImage)
```

The 'downloadCoverArt' function is used solely to gather the album cover art for the track so that it can later be applied on to the mp3 itself. This uses the same error handling as the previous function and all it does is gets the link for the art from the API and then gets the content of the image which is image/png and then writes it to a file which we'll see later gets deleted. I tried just writing the image content to the file but that didn't work 100% of the time so I had to resort to downloading it instead.

Function (lastfm.py): getAlbumSongs

```
def getAlbumSongs(artist, album):
    url = "http://ws.audioscrobbler.com/2.0/?method=album.getInfo&api_key=" + lastFmApiKey + "&artist=" + artist + "&album=" + album + "&format=json"
    response = requests.get(url)

    if response.status_code != 200:
        return {"error": "Error " + str(response.status_code) + ": " + response.reason}

    response = response.json()
    tracks = response["album"]["tracks"]["track"]
    songs = []
    for track in tracks:
        songs.append(track["name"])
```

The function 'getAlbumSongs' is used for gathering a list of songs for an album as the user has an option to download a full album also, what this will do is use album.getInfo on the LastFM API to find the names of tracks on that album and then will add them to the 'songs' list which is later used within youtube.py.

Function (lastfm.py): findSimilarArtists

```
def findSimilarArtists(artist):
    url = "http://ws.audioscrobbler.com/2.0/?method=artist.getsimilar&artist=" + artist + "&api_key=" + lastFmApiKey + "&format=json"
    response = requests.get(url)

    if response.status_code != 200:
        return {"error": "Error " + str(response.status_code) + ": " + response.reason}

    response = response.json()
    similarArtists = []
    for artist in response["similarartists"]["artist"]:
        similarArtists.append(artist["name"])
    similarArtists = similarArtists[:10]
    return similarArtists
```

The function 'findSimilarArtists' is used for finding similar artists to the one the user chose to download. All this does is uses artist.getsimilar to grab the list LastFM provides then adds it to the similarArtists list and limits it to 10.

YouTube.py

This whole file is where all the downloading from YT happens.

Downloading the songs

```
def searchForSong(songName):
    artist = songName.split(" - ")[0]
    track = songName.split(" - ")[1]

    search = Search(songName + " official audio")
    video = search.results[0]
    youtubeUrl = "https://www.youtube.com/watch?v=" + video.video_id
    downloadSong(youtubeUrl, artist, track)

def downloadSong(url, artist, track):
    yt = YouTube(url)
    fileName = artist + " - " + track
    print("Downloading song: " + fileName)

    # because windows doesnt allow these
    fileName = fileName.replace('/', '').replace(':', '').replace('|', '').replace('"', '').replace('?', '').replace('<', '').replace('>', '')

    yt.streams.filter(only_audio=True).first().download(filename=fileName + ".mp4")
    convertToMP3(fileName)
```

These two functions are mainly where the magic happens, searchForSong will search for the artist and track the user chose along with official audio and then will grab the url and send it to downloadSong which will then download the song as a .mp4.

MP3 Conversion

```
def convertToMP3(fileName):
    clip = moviepy.AudioFileClip(fileName + ".mp4")

    clip.write_audiofile(fileName + ".mp3")
    os.remove(fileName + ".mp4")
    print("Song saved as: " + fileName + ".mp3")
```

This is later converted to a mp3 using moviepy and saved as a mp3.

Tagging the MP3 using LastFM data

```
def tagMP3(artist, track, fileName):
    tags = lastfm.getTrackInfo(artist, track)

    if tags == None:
        print("Could not find tags for: " + artist + " - " + track)
        return

    artistName, trackName, albumName, trackNumber = tags

    audioFile = eyed3.load(fileName + ".mp3")

    audioFile.tag.artist = artistName
    audioFile.tag.title = trackName
    audioFile.tag.album = albumName
    audioFile.tag.track_num = trackNumber

    lastfm.downloadCoverArt(artistName, trackName)
    with open("cover.png", "rb") as file:
        imageData = file.read()
        audioFile.tag.images.set(3, imageData, "image/png", u"Cover")

    audioFile.tag.save()
    os.remove("cover.png")
    print("Tags saved for: " + artistName + " - " + trackName)
```

This is where most of the LastFM data is used, this uses eyed3 to tag the artist, track and album names along with the track number and the cover art.

Main.py

```
import youtube
import lastfm

def downloader():
    choice = input("\nWould you like to download a song or an album? (song/album): ")
    while choice != "song" and choice != "album":
        choice = input("Please enter either song or album: ")

    if choice == "song":
        song = input("\nPlease enter the song title (Format: Artist - Song): ")
        artist = song.split(" - ")[0]
        song = song.split(" - ")[1]
        youtube.downloadSongFromTitle(artist, song)

    elif choice == "album":
        title = input("Please enter the album title (Format: Artist - Album): ")
        youtube.downloadAlbumFromTitle(title)

    print(f"\nSince you searched for {artist}, here are some similar artists that you may enjoy:")
    similarArtists = lastfm.findSimilarArtists(artist)
    for artist in similarArtists:
        print(artist)

print("Welcome to the YouTube to MP3 downloader!")
downloader()

# once done, ask the user if they want to download another song or album
while True:
    choice = input("\nWould you like to download another song or album? (y/n): ")
    while choice != "y" and choice != "n":
        choice = input("Please enter either y or n: ")

    if choice == "y":
        downloader()
    elif choice == "n":
        break
```

This is where the inputs from the user are actually taken in, I use a function for actually prompting the user as I can then easily reuse that later when I ask if they want to download again within the infinite loop and this is also where findSimilarArtists is actually used.