

CS 122 Final Project Report
Project: Spotify Song Recommender
Team Chrapple

Christopher Oh (christopher.oh@sjsu.edu)

Apple Ko (hnin.ko@sjsu.edu)

Introduction

We believe that the music we listen to is not just entertainment, but also a reflection of our personalities. With that in mind, we built My Spotify Dashboard, a customizable music dashboard that lets users see a snapshot of their music identity. Users can explore their top artists, tracks and genres and see how their tastes and personalities have evolved over time. To take it a step further, we also implemented a personal recommendation system that helps users discover new artists and albums to keep evolving their music taste and maybe even find their next favorite artist!

Methods and Materials

Creating Spotify Web App

First we created an app on the Spotify for Developers platform, which let us interact with the Spotify ecosystem. Using this app's credentials, Client ID and Client Secret, we authenticate our app and make requests to the Spotify Web API to get access to the user's Spotify data. Users do not need to create their own app again to be able to run the program. However, if they wish to recreate the project from scratch, please refer to the Spotify Web API Documentation referenced below to see detailed steps on how to create this app.

Libraries Imported

For API calls:

- **Spotipy:** We use this Python library as it supports all features of the Spotify Web API such as accessing endpoints and user authorization.
- **Requests:** We use this library to make GET requests to retrieve user's data.
- **musicbrainzngs:** We use this library to access MusicBrainz database

For Data Manipulation and Visualization:

- **Pandas:** We use pandas to organize the retrieved data into Data Frames for easy manipulation.
- **Matplotlib and Seaborn:** We use Matplotlib and Seaborn to visualize the user's top genres as a chart.

For Profile Dashboard:

- From **IPython.display**
 - **Image :** This library is needed to support retrieval of images such as user's profile picture, artist photos and album art.
 - **HTML :** This allows us to style the template of dashboard layout and add clickable links to redirect users to.

For Web Scraping

- **Playwright:** End-to-end web access. Used for accessing information from ListenBrainz

API

We used Spotify's Web APIs to make get requests and gain access to the user's profile info and listening history. We defined the SCOPE that gives permission to our app to get access to the user's top tracks, library and recently played songs. Some of the crucial API calls that were used in this project are mentioned below.

```
SCOPE = 'user-top-read user-library-read user-read-recently-played  
playlist-read-collaborative playlist-read-private'
```

sp.current_user() : to get current user's information

sp.current_user_top_artists() : to get top artists

sp.current_user_top_tracks() : to get top tracks

sp.search() : to search for artists in the same genre

Important Parameters for Customization

Users have the ability to customize their music profile dashboard by changing some parameters in the defined functions.

```
def get_top_artists(sp, limit=10, time_range='short_term'):
```

For example, this function `get_top_artists` takes in the object `sp`, `limit` and `time_range` as parameters and returns the user's top artists. We can customize the number of artists to be returned by changing the `limit` parameter. The example above would return top 10 artists.

The `time_range` parameter controls the time period for determining top artists and the scope of the data being fetched from Spotify. By changing the value of this parameter, we can either show the user's current preferences, or their tastes over the last six months or their all time favorites. The table below shows three options we can customize our dashboard.

'short_term'	data from last 4 weeks	shows current's favorites
'medium_term'	data from last 6 months	shows a broader view of recent tastes
'long_term'	entire listening history	shows all time favorites

How to Run the Program

This project was built in Python on Google Colab notebook, which allows users to run this program without needing to download any extra software on their devices. Users require an existing Spotify account to be able to access their user profile info. For detailed steps on how to run the program, please refer to the README.md text file in the github repo.

Results



Welcome, apple!

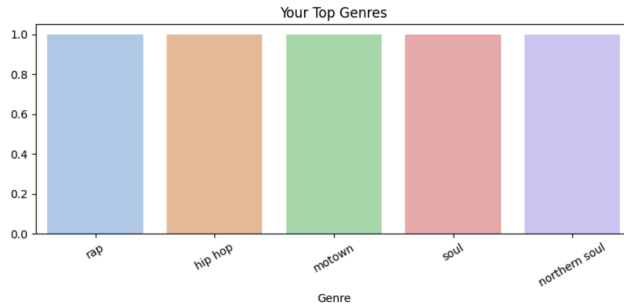


Figure 1: User's music profile dashboard showing user profile name, profile picture and top genres.



Your Top Artists



Drake



Marvin Gaye



Billy Joel



Oasis



Sabrina Carpenter



Your Top Tracks



A Storm on a Summers Day
A Storm on a Summers Day



Get It Together
More Life



Superman
Home Brewed



NOKIA
Some Sexy Songs 4 U



DIE TRYING
Some Sexy Songs 4 U

Figure 2: User's top 5 artists and top 5 tracks



Custom Artist Recommendations



Kendrick Lamar



SZA



Travis Scott



Kanye West



Playboi Carti



Lil Wayne

Figure 3: Custom artist recommendations based on user's preferences

Song Recommendation System

The song recommendation system utilizes two databases, MusicBrainz and Listenbrainz. MusicBrainz is an open source database for Music including data about record and album releases, artists, instruments, release dates, producers, and so on. ListenBrainz is a site operated by MusicBrainz which lets users record their listening habits.

MusicBrainz has a built-in API for Python called `musicbrainzngs` which allows the caller to receive information about a song, album, or artists from its MusicBrainzID. ListenBrainz also has an API, but it was inadequate for the goals of this project. Instead, a third party web-scraper had to be utilized.

Playwright is a Python library which automates web interactions. It acts like a regular user, clicking on links and navigating through pages. The key aspects of Playwright are *async* and *await*, allowing for asynchronous access to web pages. Playwright also allows for headless browsing, or browsing without GUI, which is much faster and ideal for web scraping.

The recommendation system works as follows. The user will have to find the listenbrainz link for an album(s) they enjoy and copy and paste it into the colab page. The page will then use Playwright to extract the usernames for the top listeners of that album (no such page exists on listenbrainz for individual songs, hence why the system is based on albums instead). Based on the usernames, the script will then visit the user profile and find what songs listenbrainz has recommended for them. The links for those songs on the listenbrainz site will lead to the songs' musicbrainz pages (and they also have the same IDs). Using the musicbrainz API, we can extract the song title and artist. With the song title and artist we can use the Spotify API to find the spotify urls for those songs. Next, to make sure the music recommendation doesn't just show popular songs that are recommended to everyone, the songs are ordered in ascending order based on the popularity score from Spotify API. Also, any songs made by the same artists as the albums the user inputted will automatically be removed.

Discussion

During this project, we were able to explore and learn more about using different Python libraries, setting up user authentication systems and making API calls to create a simple program on Google Colab notebooks.

However there were some challenges that we encountered during this project. Due to Spotify discontinuing many of their API services such as recommendations and audio features, it could not be used as a way to recommend new songs. Instead, we had to find help from a different database: MusicBrainz and its subsidiary site ListenBrainz that would allow us full access to the music data, and the Playwright library for accessing data for the latter site. We implemented our own recommendation logic by iterating through the user's top albums and finding what songs were recommended to the other fans of those albums. We then compiled those songs and ordered them based on popularity to give bias towards lesser known tracks and artists.

For future improvements, we would like to deploy this as a Flask application to build a smoother user authentication process that does not require the user to manually copy and paste the redirect url link, while also building a more dynamic dashboard that users can interact with.

This would also make it easier for users to share their profile dashboard among their friends. We would also like to integrate the recommendation system into the dashboard to make it a more seamless experience for the user.

Appendix

Set up and authenticate the Spotify app

```
# Credentials from Spotify Developer Dashboard to identify my app when making API requests.
CLIENT_ID = 'f455399ce6b14ecd9feba8b404ce9579'
CLIENT_SECRET = '2cb4fe209b3747ffb823906ed0b5cfe'
REDIRECT_URI = 'https://example.com/callback' # user gets redirected here after authorizing the app

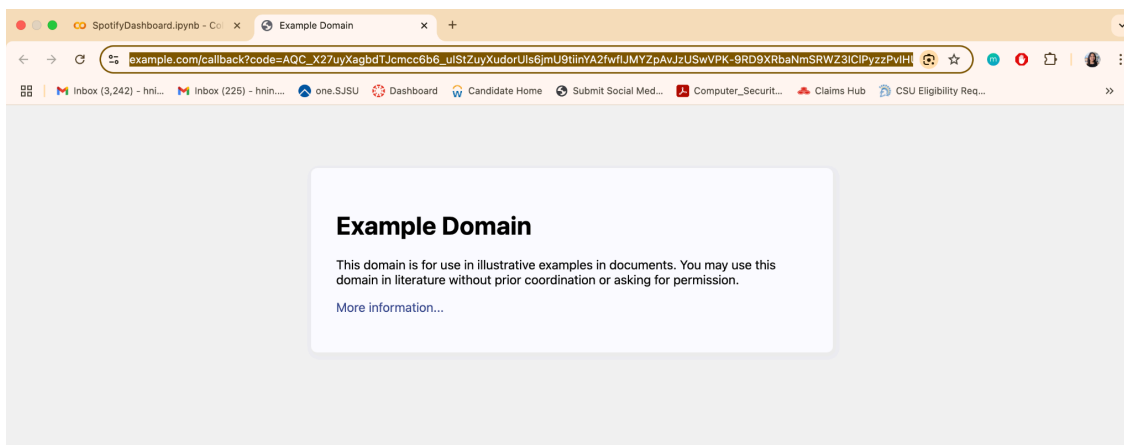
# the permissions that the app will request from the user
# top tracks, library, recently played songs
SCOPE = 'user-top-read user-library-read user-read-recently-played playlist-read-collaborative playlist-read-private'

# Create object to handle authentication flow
sp_oauth = SpotifyOAuth(client_id=CLIENT_ID,
                        client_secret=CLIENT_SECRET,
                        redirect_uri=REDIRECT_URI,
                        scope=SCOPE)

# Generate authorization URL
# User has to click on this link to open new browser and authorize access
auth_url = sp_oauth.get_authorize_url()
print("Click this link to authorize access:")
print(auth_url)
```

Click this link to authorize access:
https://accounts.spotify.com/authorize?client_id=f455399ce6b14ecd9feba8b404ce9579&response_type=code&redirect_uri=https%3A%2F%2Fexample.com/callback&scope=user-top-read+user-library-read+user-read-recently-played+playlist-read-collaborative+playlist-read-private

Appendix A: To set up and authenticate the Spotify app, click on the url link generated in the output after running the cell to be redirected to a new tab.



Appendix B: Copy the full url of the redirected page.

```
# paste redirected URL from the browser here
redirected_url = input("Paste the full redirect URL here: ")

# Parse the code and get access token
code = sp_oauth.parse_response_code(redirected_url)
token_info = sp_oauth.get_access_token(code)


# Check if token_info is returned as dictionary or string
if isinstance(token_info, dict):
    access_token = token_info['access_token'] # dictionary
else:
    access_token = token_info # string

# Use the token
# Create sp object to make API calls
sp = spotipy.Spotify(auth=access_token)

# Confirm login
# API call to get user's profile info
me = sp.current_user()
print("Logged in as:", me['display_name'])
```

... Paste the full redirect URL here:

Appendix C: Paste the full url in the input box generated by the fourth cell.

 Paste the full redirect URL here: <https://developer.spotify.com/documentation/web-api>
Logged in as: apple

Appendix D: Check for this message to confirm successful authentication.

```
{'title': 'Incontro Con Marlene',
  'artist': 'Stelvio Cipriani',
  'spotify_url': 'https://open.spotify.com/track/6J8F5QqrcGuo7kadcG03VD',
  'popularity': 0},
{'title': 'Starlight',
  'artist': 'Trúbrot',
  'spotify_url': 'https://open.spotify.com/track/1RAQwP40QkgN7vvbrR2NTE',
  'popularity': 0},
{'title': 'What Are the Parts of a Tree',
  'artist': 'Marais & Miranda',
  'spotify_url': 'https://open.spotify.com/track/4GyFJmK48qi8Ysp0j9Sj8c',
  'popularity': 0},
{'title': 'Oasis',
  'artist': 'Tony Carey',
  'spotify_url': 'https://open.spotify.com/track/47xQp01N7fNuZiXabrYHWN',
  'popularity': 0},
{'title': 'Just One of Those Things',
  'artist': 'Dudley Moore',
  'spotify_url': 'https://open.spotify.com/track/5W9uVaQx4sWdKcntHpKHrT',
  'popularity': 1},
{'title': 'City Streets',
  'artist': '鈴木茂',
  'spotify_url': 'https://open.spotify.com/track/44I3y0I5lLc3xk7x10T6aV',
  'popularity': 2},
{'title': 'que los eunucos bufen',
  'artist': 'Cóclea',
  'spotify_url': 'https://open.spotify.com/track/2tz1DYaY0nyJVlkWabKLT3',
  'popularity': 2},
{'title': 'Zinni',
  'artist': 'The Durutti Column',
  'spotify_url': 'https://open.spotify.com/track/5IRcytpVDWubsD5U24Uszf',
  'popularity': 4},
```

Appendix E: Snippet of the results for the recommender system

References

Documentation for Spotify Web API

<https://developer.spotify.com/documentation/web-api>

Documentation for MusicBrainz API

<https://python-musicbrainzngs.readthedocs.io/en/v0.7.1/api/>


Documentation for Playwright

<https://playwright.dev/docs/intro>


Github Repo

https://github.com/Hninyeeko/CS122_SpotifyRecommender

SpotifyDashboard.ipynb Colab Notebook

 SpotifyDashboard.ipynb

Listenbrainz Recommendation.ipynb Colab Notebook

 Listenbrainz Recommendation.ipynb