Status Report

Project Name: SpotiSort

Team Members: Kieran Stone, Lukas Voemel

In the last few weeks, we have gotten about 50-60% of the project done. Lukas has done most of the backend and Kieran has done most of the frontend. So far the backend can access the Spotify API, let the user log in, and get their liked song data. As for the frontend, we have a login page that will direct you to the Spotify login page, a home page that explains the site, and a sort page where the playlist sorting will take place.

So far we have not made any changes or noticed any problems that will cause us to make significant changes from our project summary in project 5.

The Singleton pattern has been implemented which handles the app management, and Lukas is working on implementing the Factory pattern which will display the data.

Class Diagram

Plan for next iteration

We only have a few things left to implement and I would estimate we are about 60% done. We still have to implement the three different song sorting methods and then display the resulting playlists for user approval, let the user accept or reject them, and then create the playlists in their Spotify account if they accept them. We also have to finish the code to display the users liked songs on the Sort page. We haven't really changed anything from our Project 5 Project Summary and are on track to complete the project by the Project 7 due date.

UML

Singelton Pattern

SpotifyAPI

- instance: None
+ api_key: str

+ get_instance: SpotifyAPI
+ get_api_key: str
+make_api_request: dict

Factory Pattern

SongInfo<<interface>>

- playlist_id: int
- playlist_type: string
- number_of_songs: int
- songs[]: dict

+ create_playlist(type): void

Concrete Info

getArtist():

Product
<<interface>>

get_info()

GetArtist

GetArttist()