Final Project Report: SpotifyShare

Compsci 316

Project Team: Martha Aboagye, Selen Berkman, Elise Brown, Benjamin Nativi, Caroline Wang

I. Project Description, Motivation, and Related Work

The goal of SpotifyShare is to allow users to evaluate their music tastes and compare their tastes to other users based on the user's listening habits on Spotify. Our project was motivated by two ideas. First, we were interested in providing a service for Duke students to connect with their friends and decided that comparing music tastes would be a good way to achieve this goal. Second, as Spotify users we had used Spotify Wrapped, a once-a-year service provided by Spotify at the end of each year. Spotify Wrapped allows users to look back on their music stats for that year. Although Spotify Wrapped is fun to interact with, we found that this application was missing a lot of functionality that we as users would want to have, such as more recent music stats and comparisons with other Spotify users.

There are other sites as well that provide similar services to Spotify Wrapped such as statsforspotify.com and spotify.me. Both of these services are very similar to Spotify Wrapped as they provide stats on the listener and lists of top songs and artists, although it appears that both of these applications use up-to-date listener statistics.

We used these applications as a starting point for our project. As all of these applications provide listener statistics such as number of hours listened, we decided this would not be one of the primary goals of our application. The two things we felt all of these applications lacked were the ability to compare your music taste with your friends and the ability to see a more expansive list of your top songs (these applications only allow you to see your top 5-10 songs), and these are the two things we focused on with our application.

II. User Scenarios

- Scenario 1: New User
    - A user who has never used our website goes to our home page. They see multiple tabs to click on and as a new user decide to click on Login.
    - They are redirected to the login page, where they see a line "Don't have an account? Sign up!" As someone without an account, they click on sign up.

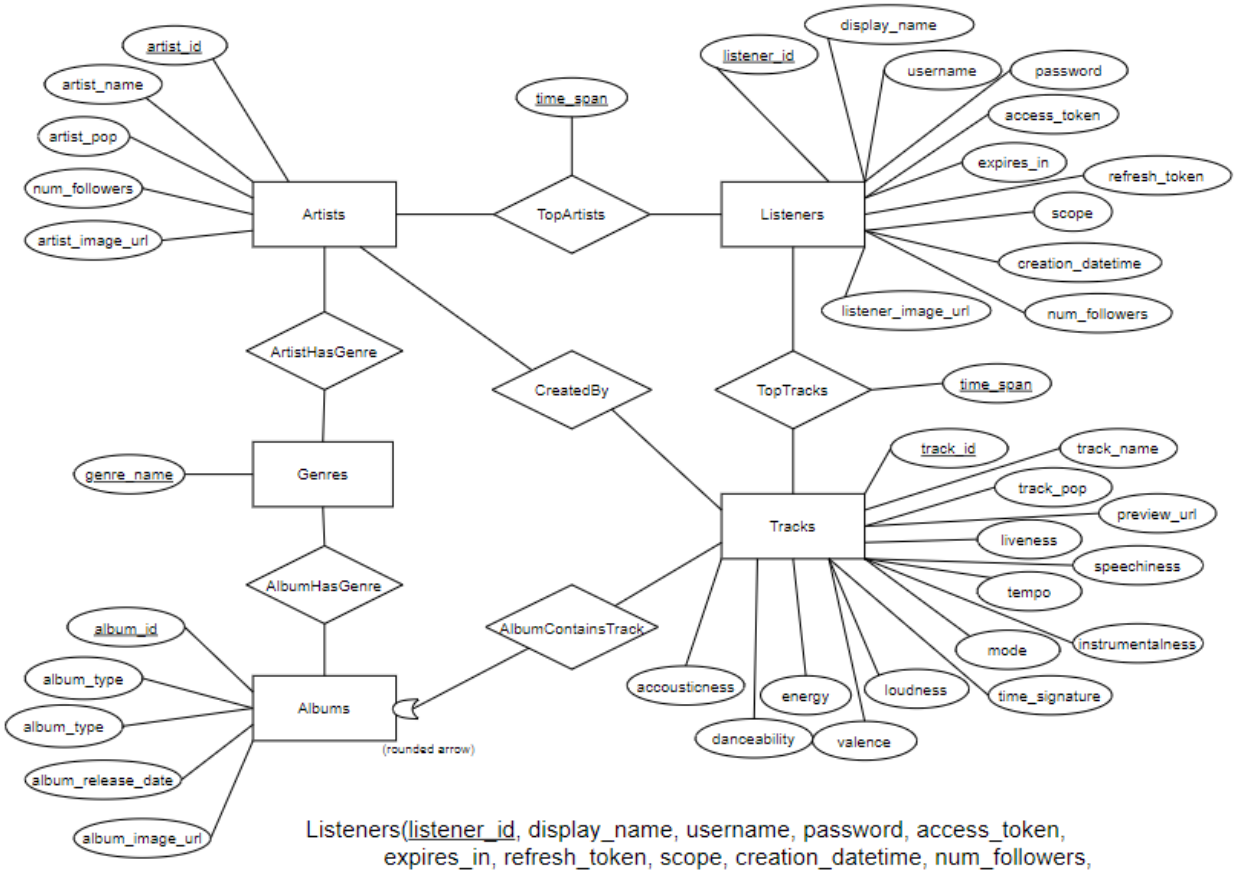
- They are redirected to the new user sign up page. They see boxes for username and password. They type in a username and password and click the button that says “Register”.
- After a couple of seconds of processing (for pulling their data from Spotify and inserting it into the database), they are redirected to the yourdata page. They see their Spotify profile picture and username at the top. Scrolling down they first see a list of their top 20 artists and top 20 songs. Scrolling further they see 2 graphs with information about their most popular music genres and average statistics about the songs they listen to. At the bottom they see a graph with roughly 30 points on it. Hovering their mouse over the graph displays various usernames of users of SpotifyShare. They see their name and notice that their point is close to several other points, indicating that these are the users with the most similar music tastes. They see that one of the users is one of their friends!
- Scrolling to the top of that page, they click the back button to return to the home page. They click the logout button and leave the website.

- Scenario 2: Returning User
  - A user who has already made an account enters our home page. Forgetting that they have not logged in yet, they click on the “Go to your data page” link. As they are not logged in yet, they are redirected to the login page.
  - They type in their username and password and click “Sign In”. They remain at the login page and a message is displayed “Password is incorrect. Try again.” They realize they mistyped their password and try again. This time they enter their username and password correctly and are redirected to the yourdata page.
  - They look at their various Spotify data then click the back button to return to the home page.
  - They see a button on the home page that says “Meet the team”. They click the button and are redirected to the welcome page.
  - On the welcome page, they see information about the developers of the app. They click the back button to return to the home page.
  - On the home page they logout and leave the website.

III. Technical Structure of Project

A. The app is built using Flask
B. The app is deployed using nginx on a Duke VM.
C. The app is backed by a PostgreSQL database

IV. ER Diagram & List of Tables



Listeners(listener_id, display_name, username, password, access_token, expires_in, refresh_token, scope, creation_datetime, num_followers, listener_image_url)
Artists(artist_id, artist_name, artist_pop, num_followers, artist_image_url)
Tracks(track_id, track_name, track_pop, preview_url, accousticness, danceability, energy, valence, loudness, tempo, instrumentalness, speechiness, mode, time_signature, liveness)
Albums(album_id, album_type, album_name, album_release_date, album_image_url)
Genres(genre_name)
TopArtists(listener_id, artist_id, time_span)
TopTracks(listener_id, track_id, time_span)
CreatedBy(track_id, artist_id)
AlbumContainsTrack(album_id, track_id)
ArtistHasGenre(artist_id, genre_name)
AlbumHasGenre(album_id, genre_name)

A. **Listeners**: each tuple corresponds to a user of SpotifyShare who has authorized us to view their Spotify data. Each tuple stores information about the user including their SpotifyShare username and password.
B. **Artists**: each tuple corresponds to an artist that can be found in at least one other table in the database (e.g., as the creator of a track, or as the top artist of a user), and stores information about the artist and their profile picture.
C. **Tracks**: each tuple corresponds to a song on Spotify. We only generate tuples in Tracks for songs that are a top song of some user. Each tuple contains information pertaining to

the song such as the song's name and popularity. Each tuple also contains track features such as tempo, energy level, and speechiness.

D. **Albums**: each tuple corresponds to an album that can be found in at least one other table in the database, and stores basic information about the album, and the cover image.
E. **Genres**: each (single-element) tuple corresponds to a genre that can be found in at least one other table in the database.
F. **TopArtists**: each tuple relates a listener to an artist, indicating that that artist is a top artist of that listener. The tuple also contains an attribute time_span which indicates whether the artist is a short term, medium term, or long term top artist. A listener can have many top artists and an artist can be a top artist of many listeners.
G. **TopTracks**: each tuple relates a listener to a track and a time frame, indicating that a track is the top track for that listener, given that time frame. A listener can have many top tracks and a track can be a top track of many listeners.
H. **CreatedBy**: each tuple relates a track to an artist, indicating that the artist was involved in creating that track. An artist can create many tracks and a track can be created by multiple artists.
I. **AlbumContainsTrack**: each tuple relates a track to an album indicating that the track is contained in that album. An album can contain many tracks but a track is only contained in one album on Spotify.
J. **ArtistHasGenre**: each tuple relates an artist to a genre, indicating that the artists' works fall into that genre. An artist may have many genres.
K. **AlbumHasGenre**: each tuple relates an album and a genre. An album can be classified as having many genres and there can be many albums related to a genre.

V. Assumptions about Data

A. Every track is in exactly one album (meaning that even if a song is featured in more than one album, we have each version of that song as a unique track).
B. The attribute "artist_pop" represents an artist's current popularity compared to other artists (not necessarily the ones cached in our web-app).
C. When there are artists who make music on their own who are also in a group, sometimes music made by the group is attributed to the individuals as well as the group, but other times music made by the group is only attributed to the group. We take in the data that Spotify provides.
D. Each album can have 0 or more genres
E. Each genre can belong to 0 or more albums or 0 or more artists
F. The audio attributes for each track (e.g., liveness, speechiness, instrumentalness, etc.) are not empty for most tracks

VI. New Approaches and Algorithms

Although other apps also display lists of top tracks and artists like we do, our two unique features are our radar graphs and comparison graph.

The first radar graph shows how popular each of 9 music genres is for the user. The relative popularity is computed by counting the number of the users' top songs that fit that genre and weighting each song inversely with its popularity (as less popular songs might give a better indication of a user's music taste). More top songs of a genre produces a higher score. The second radar graph works similarly but instead displays track features. This includes 7 features such as energy level, accousticness, and speechiness which capture certain aspects of a given track. This radar graph allows the user to see whether their songs have a high or low level of accousticness, speechiness, etc.

For our comparison graph, we generated a list of music features. We use the listeners' data to compute values for each of those music features to generate a vector for each listener. Now, with the set of vectors for all the listeners in our database, we use t-SNE to project these data points onto 2 dimensions. t-SNE is a dimension reduction technique which attempts to preserve distances between data points. Thus if two points are close together on our 2D graph, it should indicate that they have more similar music tastes.

VII. Open Issues and Future Directions

Refine user comparison graph: Dimension reduction algorithms often benefit from careful parameter tuning. If we had more time, we would have worked with the parameters to improve the projection for our comparison graph. Given more time we also would have liked to try other dimension reduction techniques so that we could use whichever produced the best results with our data.

Allow users to select which other users to compare to*:* We would add functionality to allow users to compare themselves only with a subset of other users. This would be useful with a high volume of users as the comparison graph would become cluttered and implementing group functionality would allow friends to compare their tastes only with people they know.

Cluster users into various groups: With more users, we could use clustering techniques to group people based on their music interests. It would be interesting for users to see who they are grouped with and it also would be interesting to see how the groups created by a clustering algorithm compare to traditionally defined music genres.

Display additional listener statistics: We would display additional statistics such as the number of hours a song has been played, the top genre for a listener, etc. However, as these are included on other Spotify music-taste sites, these features would not be a unique benefit of our web app.