Capstone I Project Proposal – Andrew Kenfield

For my Capstone I project I will design and create a “playlist manager” site using the Spotify API. It will allow users to either create a new playlist or import a currently existing one. In addition to the standard features of adding/deleting songs, the website will let a user arrange the playlist order using a number of characteristics not visible/accessible through the standard Spotify UI. Thus someone will be able to order their playlist in terms of danceability, loudness, even musical key.

The demographic for this website will be similar to that of Spotify in general; that is, skewed towards the younger side. I would envision the site having broad applicability across that demographic, as the basic concept of the site is already familiar, [only] adding additional functionality without dramatically altering the way in which a user interacts with a playlist.

I plan on using the Spotify API, which is widely-used, well-documented, and should provide a facile source for this educational project. Within the greater landscape of the Spotify API, the site will particularly concern itself with the Playlist, Track, and AudioFeatures objects/endpoints. For the sake of project scope I will limit myself to these, but the site could certainly be built out to also include Artist, Album, and others as sortable objects.

As described above, the initial primary models used will be Playlist, Track, and AudioFeatures. I am a slight bit unsure as to why the Spotify developers decided to separate out AudioFeatures as its own object distinct from a Track; perhaps they could be joined in the site’s separate/’shadow’ database. There will certainly be users and authentication/authorization, at the very least through Spotify proper.

I don’t foresee any major issues in using the Spotify API; the biggest decision still unmade is whether to solely copy over Spotify’s data into the project’s proprietary database, or to use the data received in creating a distinct database of songs, playlists, etc. The abovementioned question of users is one example of this.

The only sensitive information processed through the website will be passwords. This will likely be both a user’s website password, as well as their Spotify password. However, if the site is feasible without having to log in to Spotify, it may only require its own password.

A site user will first be asked to sign in to their Spotify account; this is a necessary step, as Spotify does not allow the playing of music without registration. At this point, I am still debating adding the additional step of site-specific registration and login. This would allow for the capture and preservation of searches, ordering, etc. specific to each site user. Alternatively, this data-saving could be done strictly within the bounds of Spotify, though it would likely be more complex and less elegant.

At this point, the logged-in user will be able to load their previously-extant playlists, search for playlists created by others, or start a new playlist fresh. This is where the user will be able to order/manipulate the playlist using the above-mentioned characteristics.

Given the scope and time restraints, I do not envision this project extending much beyond what I’ve described. However, this idea has significant potential for expansion. One possibility would be to extend the AudioFeatures analysis to a playlist, album, or artist as a whole. In this way, separate playlists, albums, artists, etc. could be compared based on their overall “danceability”/etc. rating.