Aliisa Roe

CSCI 1310

Project 2 Explanation

My project is a very small study music library that outputs a 3 song playlist according to user’s responses to the time of day, type of homework and how they are feeling about the homework. The music library is loaded from a textfile provided in the command line with the executable; the file contains several attributes about the song. This file is loaded within the actual Library class upon its initialization by the project driver. It parses the textfile and keeps the songs in vectors of their own Song class. Song classes have song titles, artists and URIs. (The URI is the special Spotify specific code used in Spotify’s API. So if you search that URI in Spotify, even though it looks cryptic, the code is the easiest way to be taken directly to a specific song, by bypassing the navigation steps returned in a regular search.) The Song class also has getter and setter functions, most of which are not used, but coded flexibly that they should still remain for future alterations to/reuse of the project.

With this Library class of Song class objects, the project driver will prompt the user 3 questions. Each prompt is nested in a loop, so that the user must type a valid response in order to continue. Each of these responses are saved and will later correspond to individual songs that seem relevant to the user’s conditions. 3 responses correspond to the 3 songs. Once the user responds to these prompts, the project driver will call the Library class’ recommendPlaylist() method according to the logged user responses which were concatenated into an overall “vibe” of the playlist to be recommended. In this function, each user response adds that specific song to the playlist by associating an artist with a response. The song’s are grabbed from the Library instance from the Library class’ getSongForArtist() method which call’s each song’s getArtist() method comparing it to the provided artist. Once the song is found, the song’s toString() method prints the song into the terminal, and the song’s URI’s are exported into a text file titled the user “vibe”. Finally, the user has the option to launch Spotify where they can login themselves and search these URI’s in the search bar and check out the playlist for themselves.

In order to run the project, all cpp files (Library.cpp, Song.cpp, ProjectDriver.cpp must be run together with g++ and –std=c++11 to form a single executable. Then, with this executable the given textfile developer.txt must be in the command line to run the project.) The original idea was that any user could exportify (a web API I found on github which exports playlists into CSV’s –just change it to a .txt file) their own Spotify playlist to make their own user specific library. The problem with getting vibes for large unknown music styles, is that exportify does not provide a song “vibe”, “mood”, or even “genre” or “similar artists”, so I was forced to scale down my project into a significantly less flexible project by hardcoding at least a few artist vibes that I am willing to log. In order for this specific project to work, the developer.txt file must be used, unless another playlist with the exact same artists is provided.