Jadon Lam PUI Final Project Writeup - Dec 13 2023

(1) **Part 1:** Description of the project

My website, the MuSel Playlist Generator, is a page that allows users to generate a playlist based on what they would like to hear based on some of the Spotify API's get recommendations variables. Using it allows people to either find songs similar to their current interests, or find music outside of what they would usually find, while giving them a high level of control over what they are searching for. Using the music characteristics/ parameters, I help users think more about what specifically makes a piece of music closer to what they are looking for. This can be interesting for users of the tool, as it allows them to experience music they haven't heard before while increasing the probability that it matches their tastes. The site provides an alternative to Spotify's own machine-learning algorithm that suggests music to users, and allows more people that may not use Spotify often to have a way of generating music suggestions. The target audience of this app is, realistically, either music-enthusiasts that already know the characteristics that make music stand out to them, or people that don't know much about what makes pieces of music sound distinct from each other, but as a baseline the target is people that would like to explore more about what pieces of music exist.

(2) **Part 2:** Interactions

- First, I have a homepage link on the logo that takes the user to the index.html page when clicked on. Currently, this just reloads the page, as there are no other pages to link from.
- I have implemented a series of sliders that, when adjusted, modify which songs may be included in the suggestions playlist based on the characteristic related to them. For all sliders, to see how the value changes, open the console, and click and drag the slider. On release, the console will be logged with the new value of the slider.
- For the two user inputs, Genre and Playlist Length, the user is able to input a string that will likewise modify which songs may be included. Genre is necessary for the playlist to be generated, as there must be at least one seed variable to create the playlist, while playlist length will default to 20 pieces. Both will fill with any string that the user inputs, but should not be:
  - Genre should take a single input from the dropdown of suggestions that appears, fills, and adjusts based on the string the user inputs. To use, the user must click on the input box, and when the text input cursor appears, type letters- and, when a desired genre appears, click on the genre, which will replace the text string. Clicking off won't give the user an alert, though will try to use whatever contents as a string if the generate button is clicked. Clicking on a suggestion or filling it in and hitting enter will give an alert and console.log the new value.
  - Length is limited to numbers at or between 1 and 50. Hitting enter after inputting a number will save and console.log the value, and alert the user of the value. If outside of the range and the generate button is pressed, it will reset the value to 20 for the request and console.log that while generating the new playlist.
- Mouse clicking on the generate button will parse the page for all the values of the sliders and text entry, send an API request for the suggestions, and return an array with the suggestions, which is then used to show a popup of the list of songs with their main credited artists.
  - This popup can be closed by mouse clicking on the close button at the bottom of the list.
- The page will adjust visual format at two widths: mobile and desktop views. Mobile is coded at 375px minimum and maximum, while desktop view is coded at 720px minimum and maximum- please use those two page widths to test responsiveness.

(3) **Part 3:** Description of the external tool

- <u>Name of tool</u>
- I decided to use the Spotify Web API.
- <u>Why you chose to use it?</u>
- I chose this as one of the only methods of interacting with the Spotify framework in order to access and receive the content I needed. I chose this because, firstly, as a fairly consistent user of Spotify, I had been wanting to understand more about how it worked after seeing it embedded in many websites and tools Secondly, I chose it because I wanted to learn more about using APIs after practicing them in class, and seeing that I didn't really understand how they worked.
- <u>How you used it?</u>
- Initially, during practice I used it to get a list of genres, which I then saved to a .json as a resource to make sure I had a consistent list of genres to test and use, even if it prevented me from accessing other genres. After accessing the tool and getting my key and secret, I use the API to generate session access tokens, and subsequently generate a list of suggestions based off of a request the user creates.
- <u>What does it add to your website?</u>
- The API essentially makes up the most important part of the website- which is generating the content the user requests from it- otherwise, it would only be code that makes the site move and change values with the user's input. It allows the user to, after having played with the variables on the site, get the content value from the website.

(4) **Part 4:** Iteration and design process/changes

When building the website for responsiveness, I decided to implement a mobile view that I hadn't considered in my mockups, which allowed me to design more for people with smaller screen spaces before moving up to desktop or tablet views. In terms of scope, I had to cut back a lot on my design when implementing my website on nice-to-haves, or even base functionalities. This is because, when initially designing, I hadn't yet known about or considered some limitations of the API or the way it works, such as how I would have to generate the playlists. Initially, I wanted to embed the playlist into the site itself, but I decided that didn't work because 1) that isn't how the API necessarily works, to my understanding, and 2) it wouldn't work as well on mobile view and devices. I then moved to a design where the page would redirect the user to the spotify website to log in, where they would return and generate the playlist onto their account- though I likewise decided to cut down from that idea, both due to my own limitations in understanding of the API and for reasons of scope. I landed on a design that is slightly less streamlined for the user for the sake of maintaining the experience on the site, especially since I wouldn't know how redirecting would work on a mobile device.

(5) **Part 5:** Implementation challenges

Implementing the Spotify API was especially difficult for me, since even with the documentation, I didn't know where to start for quite a while. I also had an issue with scope, and designing for features I didn't even know were possible for certain, especially with my level of coding experience. This is especially the case, since a good chunk of tutorials and documentation for the API online is either outdated or utilizes a component that I have no experience with, such as Vite or Node.js.