Week 5

During week 5, I worked on implementing the wireframe of the project together with our group. I met with the group on Thursday for two hours and during this meeting we discussed the design and layout of the features that we wrote in our project proposal on the website. We created a Google Slides to do this, where each slide detailed an important feature. After brainstorming many ways that our website could be structured, we ultimately decided to have most of our features centered on our home screen, and have multiple separate pop-up menus where users can upload files, search for audio files, and add to their playlist. I added images and shapes to the slides, cementing the design of features on the homescreen, etc. because I knew that having a solid wireframe would be a useful front-end visualization for our group to continuously reference in the coming weeks.

Towards the end of the meeting, it was decided that I would work on implementing the search bar feature. For this, I knew I had to familiarize myself with React basics like the filter() and map() methods and hooks. I began watching YouTube videos to familiarize myself with these important React concepts.

Week 6

During week 6, I met with our group during our usual meeting time on Thursday. We had fleshed out our website with the wireframe, so now it was mainly independent work based on our assigned features of the website.

During the meeting, for Git management, I suggested the idea of all of us working on our separate branches, and having all of us pull from main and make a pull request before merging changes on our branch with main, as this would prevent conflicts in main. The group agreed. Thus, I created my own branch named ‘Angela’ and a new file called ‘SearchBar.js’. This would be where the component searchBar() would be defined and exported, and then imported and rendered in App.js as </ SearchBar>.

I drafted up the basic functionality of how the searchBar component would work. I created a sample array of objects with song titles and tags, songs = [{title: ‘Fireworks’, tag: ‘Pop}). I used useState to add a state variable searchQuery for searching by title. Every time the searchQuery was changed, then the list of audios would re-render and update to match the corresponding titles. For filtering by tags, I added another state variable, selectedTags. Every time the user clicked on a button, the filter() function would be called and update selectedTags. I also added another state variable named playlist. For the playlist, I defined functions including addToPlaylist and removeFromPlaylist, which updated the playlist when a user wanted to add a song to it. It was initially challenging to wrap my idea around the concept of useState, and I encountered many syntax errors that displayed on the website after running npm start, and other errors involving the use of the spread operator to copy arrays when implementing addToPlaylist, and the value of state variables being undefined (I had to initialize useState to an empty string). Next week, I would need to learn how to style my component.

Week 7

In the previous week, I had decided that this week would be styling, but I realized that it might be a better idea to try to get all the functionality of the components done before styling. There were two reasons for this: 1) I was more proficient in Javascript at the moment than CSS and 2) Styling could be done largely independent of the other member's work, while it was important now to start linking the functionality of the frontend components like searching with the backend database of audio files. Currently, Cyrus was working on creating back-end endpoints with all the data we would need to fetch, and he requested that the uploadForm component be started. Thus, I decided to take up this task. Cyrus let me know the required input fields and their associated keys: 'audio', 'title, 'tag', and 'owner_id'. I coded each of these fields as a state variable. After doing research on the form tag in React, the whole process of coding the basic functionality of uploadForm went a lot smoother due to the experience I gained from building the searchBar. For the value of the 'owner_id', I used a getCookie() function to grab the cookie value of the visitor of the page (undefined if the user was not logged into their spotify account).

Cyrus coded the endpoint '_upload_audio', where I would need to fetch the data. I read up on the documentation of how React makes requests to the back-end using the Axios library. When the component launches, useEffect() would employ Axios to fetch all the uploaded audios in the endpoint. Unfortunately, an error 404 appeared, saying that the server could not be found. I spent many hours trying to debug, but I could not resolve it with my limited knowledge of the back-end. I decided to meet with Cyrus the next day and after two hours, the solution (from Stack Overflow) turned out to be adding a route in web.py, as well as a debugging an issue of unmatching keys in the front-end that was named 'user_id' and the back-end was named 'owner_id'.

I realized that for the past weeks I had been forgetting to commit and push my changes to the repository, which I realize is a bad idea because if my changes were somehow lost locally, then there would be no way to retrieve them.

Next week, I will start working on styling the search bar and upload form.

Week 8

At the beginning of this week, I learned the basics of css. I noted down two major ways that css could be added, one being through the addition of a separate file such as ‘searchBar.css’, or downloading styled-components using npm. I decided to go with the latter because I did not want to have to reference two files at the same time, and thought styled-components could offer better organization and whose unique names make the style of the html tags that are being rendered clear.

I decided I did not want the pop-up hamburger menu of the searchBar to be a window. I used css properties to style the hamburger menu and the three bold black lines on the bottom left of the screen which would open the hamburger menu when the user clicks on it. Then hamburgerMenu would call searchBar so that the searchBar renders inside of the hamburgerMenu. One challenge I faced was having the list of audio files disappear when the user clicked outside of the searchBar. Having the list of audios displayed when the user clicked on the search bar was relatively simple with the use of a state boolean isMenuOpen, but I had to do my research on event listeners and add an event listener for when the user clicked outside the search bar. Another challenge was having the audios that the user adds to the playlist stay the same, even after closing the hamburger menu. For that, I had to pass in parameters of the playlist and the addToPlaylist and removeFromPlaylist functions into the hamburgerMenu function and move the playlist state variable into the overarching hamburgerMenu.

In the same way as the search bar component, I used an Axios request to populate the database of user uploaded audio files, and I tested this with uploading an audio file in my uploadForm component and seeing it appear in the search bar.

For the following weeks, I would have to complete the final functionality and styling of uploadForm and searchBar.

Week 9 + 10

I decided I would work on the home screen because I had already coded the most components that would be shown on the home screen (uploadForm and searchBar), and coding the home screen would be a simple matter of bringing together the components. I styled the home screen according to our wireframe, and made sure to give credit to the artist of the main image. I styled the blue audio playlist on the homescreen, and one neat CSS property I learned from this was gradient, which allowed a beautiful transition from light to dark blue. For the home screen, I also had to add navigation to the other pages like ‘My Stats’ using React Router. This part was a little tricky because I had to learn that all the pages were called inside a Router tag inside of App.js.

During this week I also styled uploadForm. I added another hamburger menu similar to the search bar, but smaller. I made slight modifications to my original unstyled version of uploadForm. I made the input field for the owner_id hidden, so that the page would automatically retrieve the user’s cookie value and input it in the form. I also added a dropdown menu for the specified tag (instead of a text input field for tags). I also styled these buttons to match with the overall theme of our website.

It was brought to my attention that various feedback messages in searchBar and uploadForm also needed to be coded. Thus, for searchBar, when a user adds an audio file to the blue audio playlist on the home screen, a small message called “added to playlist!” would render and then disappear after 3 seconds (using useEffect). That way, the user could tell when an audio file was added to their playlist even with the pop-up hamburger menu blocking their view of the playlist. Additionally, I had to modify the Axios error handling including the back-end status error codes. If ‘response.status === 502’, I set the state variable errorMessage to match the back-end’s ‘duplicate title already uploaded’, if the user’s cookie id was invalid, then the message was ‘please log in to Spotify first’, if one of the input fields was blank, then the message was ‘Please fill on all fields’, otherwise the message would be ‘success!’.