Matthew Fong

Rich Media 2 Project 1 Documentation

**Site’s Purpose:**

The idea behind my app is reverse Shazam. Instead of feeding in lyrics from a song and getting back the song’s info, such as artist, genre, song name, release date, etc., my app allows users to enter in the name of a song and artist. This will then add that song to their local or client-side “playlist”. Afterwards, the user can get the lyrics of a certain song that they have added to in their playlist.

**API Handling:**

In terms of POSTing, users posts songs to two playlists. One is on the client-side and the other is on the server. The playlist on the client-side is solely for viewing purposes and GETting data with query parameters, while the playlist on the server-side is used in checking, adding, modifying, and deleting songs. My app will return a 201 status code from the server if a new song has been added to the user’s playlist and a 204 status code if the user tries to enter the same song by the same artist for whatever reason. This will eliminate the possibility of duplicating a song by accident (or on purpose, for your testing pleasures, Cody). When the user GETs the lyrics back, they can either type in the song name and artist in a form or click on the song <div> that appears in the playlist section. Users can send both HEAD and GET requests to the server, in which the server may respond with either a 200 status code on the first search or 304 status code if no content has been modified. Furthermore, if the user’s body (POST) and/or query parameters (GET) are missing, the server will return a 400 ‘bad request’ status code informing them that they are either missing the song name and/or artist name parameters. If the user tries to go to a page that has not been implemented yet, the server will return a 404 ‘not found’ status code. Finally, since I am using musixmatch’s API for lyrics finding, if the song name and/or artist match cannot be found in their API database, my server will return a 404 ‘not found’ status code as well.

**What went right:**

Overall, I am pleased with how my app came out in the end. The colorway and look and feel of the app really gave my app a very simple style and musical theme. After going through musixmatch’s API documentation and looking through the JSON object that the API methods return, it was fairly easy to extract the necessary information (lyrics) I needed for my app. In terms of user-friendliness, I added several small implementations to ensure that both visual and basic functionality expectations are met. For example, the playlist and lyrics section will not overflow with content if content goes beyond the bounds of the sections. Additionally, when the user hovers over a song in their playlist, the cursor will change to indicate that the user can in fact, click on the song <div> to get the lyrics of that song. User feedback is also very clear in terms of GETting and POSTing data, as well as, from other user interactions. I have also trimmed user’s input for song name and artist name in the front and back of their query and body parameters to minimize issues/typos. As mentioned, I believe that my app is a very simple and intuitive app for users to use; I wanted to make sure the app had a crisp and clear design and was easy to use.

**What went wrong:**

During development, I think the biggest thing that went wrong, and I would say it was more of an unfortunate part than anything, is that musixmatch’s API only returns 30% of the lyrics to a song to developers who are using their API for Non-Commercial Use. If one would like to view the full lyrics, they would have to buy the licensing data package first. However, 30% is still better than 0% 😊. I also spent a bit of time trying to figure out how to call Javascript code for dynamically created HTML elements. Specifically, I wanted to add more versatility to my app by allowing the user to click on the song <div> in their playlist to get the lyrics to that song, rather than having to type in the song name and artist every time in a form. Unfortunately, I am using JQuery to access the document’s .on(‘click’, ‘classname’, function) event to hook up the click event for the song <div>s. I am also using the same technique to hook up click event for the remove song buttons.

**Future Improvements:**

If I had more time and continued working on this in the future, I would like to continue implementing several things. For one, I would like to include the full lyrics to songs when the user requests them. This might mean finding another lyrics API since musixmatch’s API only returns 30% of the lyrics to a song if one does not have the licensing data package. Furthermore, I would like to include the audio file to the song lyrics as well. This will also mean I would have to include another external API, one that fetches the audio for a song, perhaps youtube’s API. Finally, CSS and page layout can always be further improved upon I feel. Making my app responsive in different screen sizes is also another feature I would like to work on.

**Above and Beyond:**

One of my above and beyond features is I am using an external API, specifically musixmatch’s API (<https://developer.musixmatch.com/documentation>) to retrieve lyrics to a song that the user has added to their playlist. Once all the query parameters are met, my API will tap into musixmatch’s API and search for the lyrics to a song by artist name and song name. Another above and beyond feature I implemented is, rather than having the user type the song name and artist every time to get the lyrics, the user can alternatively click on the song <div> in their playlist to get the lyrics. I also implemented a feature for the user to delete songs from their playlist, if they misspelled a song/artist name or simply do not want the song in their playlist anymore. Deleting a song will delete it from both playlists, on the server-side and client-side. Doing so, will ensure that the user cannot GET the lyrics of a song that is no longer in their playlist. They would have to re-add the song first. In terms of CSS and page layout, I felt the overall design of my app is very appropriate and musically thematic. I wanted to stick with simple colors, rather than vibrant colorful ones. I also am using a borrowed musical notes wallpaper to enhance the look and feel of the app.