Project Documentation

Late Submission

I want to apologize for submitting the project late. I understand that deadlines are there for a reason and I haven't been keeping up with them perfectly for the past couple weeks for personal reasons. I wanted this project to be perfect before turning it in because I wanted all features working as they should. Overall, I am very proud of the project I am submitting however I am a little nervous about the late deduction. Though I am submitting a couple hours late, please let me know if there is anything I could do to make up potential points I am losing. I am willing to put any extra work in!

Client-Side Architecture

Within the client folder, I have four main JavaScript files for functionality. The main.js file contains all the UI functionality, which is mainly composed of event listeners for the multiple buttons on the page and makes use of the exported functions of the other JavaScript files. The spotify.js file contains all the necessary Spotify utilities, which includes an authorization token function and getTrackInfo() function that both require fetch requests. It also includes a function to search for a track and a function to get a random track, since these require different requires. The rest of the functions are async get function to get specific information necessary for the display which is the song title, artist, and art. The two functions exported here are getting the track information for a searched song and getting the track information for a random song, which both return JSON objects in the format of {name, artists, art}. The youtube.js file contains a function with a fetch requisition that returns a video with the provided query. The isPianoComp() function checks if the video is a piano composition (simply by checking if any of the words in the title of the video, ignoring special characters, are equal to piano). Then another function grabs the videoID which is necessary for providing the embedded YouTube video and link. These two aspects are returned in a JSON object in the exported function. The crud.js file contains the four CRUD operations and their fetch requests.

Then there is the index.html file and styles.css file to handle design of the interface. I used BootStrap to handle certain design elements to make the overall website look cleaner.

Server-Side Architecture

There are two files within the server folder which are database.js and index.js. The database file handles connecting to MongoDB, and makes use of the CRUD operations. I also made a local json file within my project folder so that I could easily display database collections through main.js since I wasn't able to find a clean way to do this through MongoDB. The index file handles all routes which are the two routes from the spotify.js file, the route from the youtube.js file, and the four routes necessary for CRUD operations.

Instructions

1. To run the project, first run "npm install" for "express", "morgan", and "mongodb"

2. Check that “type”:“module” is in the package.json folder.
3. Run “node server/index.js” in the terminal of the root folder(“keyboardWarrior”).
4. There should be a “Hello we are on port 3000” message. Then, navigate to the application by heading to “localhost:3000” in a web browser ☺

DISCLAIMER: I thought I had run into another error with my project until I discovered that Google API has a quota limit for the number of requests per a day. Unfortunately, as of July 13 – 2:55 AM I have reached the quota limit. Please email me if you are unable to find my project successfully working 24 hours after this time!

<u>User-Interface</u>

For this application, the user would search up a song they like, for example “see you again tyler” and then the display of the current song will change along with the embedded Youtube video into a tutorial for the song. I did this by first using the search query to grab a Spotify track using an offset of 0 so that the first, most relevant, result was grabbed from the Spotify API’s search request. If a user chooses to pick a random song, then I call my randomTrack function which works by picking a random letter from ‘a’ to ‘z’ and then calling a random offset from 0 to 1000. Since I’m using my own Spotify, the results would be heavily biased if I didn’t choose a random offset. I displayed the results of the track object on “The Song” side of the screen. Then, I used the query from the search bar plus the artists’s name from the track object, since the artist name in the Spotify track object would be the most specific, and added the string “piano tutorial” to create my Youtube API search query. In the case of a randomized song, I used the song title from the track object rather than the search bar. I found it better to use the search bar as part of the Youtube query if possible since some of the Spotify song titles include all the featured artists which ends up being very long. I also made sure that the video was embeddable, a feature that was part of the Youtube API’s search request. The response is an object with a specific video id which just needs to be concatenated to the end of the string “https://www.youtube.com/embed”. This is how I rendered the composition part of my application.

The rest of the application is simple CRUD operations, specifically for a user to keep track of what songs they’re currently learning or may want to learn in the feature. The user needs to enter their username, and optionally their song and/or status of the song depending on which CRUD operation they are using. Then, the display will render a list of all their songs and the status of their songs.

<u>Database</u>

My chosen database system was MongoDB, and the connection to this can be shown at the top of the database.js file. Since I also have a local database, logs.json, that is more heavily relied on for the display of this application, this can easily be accessed to observe how my CRUD operations work!

<u>Future Features</u>

This project was super fun to code and there were many features that I’m excited to implement in the very near future. First, I would want to implement a user Spotify authentication process so that when the random button is clicked, I will change the offset to range from 0 to 10 so that users get songs they’re more likely to know since the search request’s response would learn towards their taste. Since user data is currently public, I want to add in a login page for user authentication. I also wanted to implement a drop-down menu for the status so that it would be cleaner to look at since there would be limited options. I would also want to have the user’s log on a separate page so that they can write in any notes about the composition.