Colton Gering

Professor Rene German

Chapman University

23 May 2020

Final Project

For my final project, I wanted to create something that I could use with my extra time during the quarantine. Since I am learning the piano, I figured doing something related to that would be a good idea. I have seen sites dedicated to piano theory and other resources, and I wanted to make a resource like those. When studying piano, I find that whenever I learn or practice a scale that I do not quite remember, I do not have an efficient process for getting familiar with the scale quickly. I can never seem to remember the fingerings for them, and getting the arpeggios correct can be a hassle. There are resources out there which offer interactive pianos, but not with the fingerings I have been given from my teacher. Practicing with poor fingering can be harmful for both your technique and hand/wrist health. Another aspect of practice I found tiresome was losing track of the pieces I have recently completed. This was a big problem for me since I play pieces on both tablet and sheet, so getting them all in one place was ideal. It would also be useful to keep track of the status of each piece I'm working on, because sometimes it can range up to 5 or so pieces concurrently, which makes it harder to remember exactly where I left off.

I found a resource called react-piano, which has an interactive piano, and I thought it would be interesting to build a project on it. This influenced me to learn and use react for the front-end, which was very challenging for me. The major components of my project are: a JavaScript React App that is run in the browser, a Flask App which provides REST endpoints for talking with the

database, and the database itself. Within the React app, there were many different things I wanted to accomplish: the first component is the profile of the user, which is created to keep track of the status of the pieces they're working through and the keys that they have completed studying. The table related to the part of the profile which keeps track of the status of pieces is the Pieces table. The user can keep track of the piece name, composer, and status by entering the information into a table in the UI. This table is filled in when the component mounts, and each time a new button is clicked a POST is made to insert a new record into the database table so that it can be loaded on next mount. I also made a table in DataGrip called KeysCompleted, which had a Boolean for each key, and on button click a POST would be sent to tell flask to make a query to switch the Boolean value. When the component mounts, Flask sends an object containing the values of all of the key completions, and the buttons are colored according to this. This would help the user keep track of the key that they are working on.

I found this process of sending data over the network to be a crux for this project, as building many of my components came down to being able to make these REST calls to flask and then format the data that I send back from flask. Jsonify is a useful tool in flask that I took advantage of because it allows for python objects to be converted to JSON objects. A JavaScript library in React called axios was also useful for this process, as it made protocols like GET and POST much easier to perform. Flask also made it straightforward when I needed to perform queries that were requested from axios, as the python code with MySQL is simple. Knowing Python beforehand was also useful because if I had to learn React and Python at the same time while learning this project, I'm not sure if it would be feasible.

Another component of the project was adding functionality to the piano, which was the most confusing part to me. The react-piano library was created by someone else, and has a lot of

interesting features, but it can be pretty challenging to implement for someone who is new to React. For example, when I imagined how to code a scale being played, I thought a simple for loop with all of the MIDI numbers (each note has a number) would work. The problem with this was that since setState is asynchronous, the notes were not played evenly, and it sounded very messy. A more "react" way to accomplish the playing, and a concept which took me a long time to understand was using a method from the updating section of the React Lifecycle. Whenever a state or prop is changed, the component has to be re-rendered, which calls the five updating lifecycle methods. One of these methods is called componentDidUpdate().

This was useful because it allows to check if the current state matches the state previous state. If the states don't match, this means that the program is either beginning to play, or stopping playing. If the state does not match the previous state and the updated state is a playing state, this means that the notes are beginning to play, and the interval for the whole set of notes that are going to be played is set. This interval is part of what pushes the for-loop like structure of state change. The piano has a prop called activeNotes, which keeps track of the note that is currently being played. This prop checks if the state is playing, and if it is then the active note will be the index of the song. The button to play the scale is what triggers the change of playing state, which causes the chain of updates. The componentDidUpdate() will be called by the setInterval function every 300 seconds, so the chain won't stop until the set of notes is completely played and to finish, the state of playing is checked against the previous playing state to see if the playing should stop (index matches the length of the set of notes).

The last component to the project was the login system. The further I delved into the other parts of the project, the importance of the system became increasingly apparent. This is because I have a lot of REST calls that use queries which query by the current user id. Creating a session

each time a user logs in allows for the current user id to be stored, and that can be moved around by passing it as a prob and making REST calls with it to query that user's information when needed.

Routing was somewhat difficult for me because I have never really done it before. React does a lot of things to make this process easier for beginners, however, including the react-router-dom library. I learned that you must have a router which connects a URL to a component, which allows for clickable events to be connected to the router. Using React Material, which is a UI framework that makes it easier to attain a professional look, the routing became more difficult because a lot of the custom items like MenuItems have to be wrapped by items that have routing capabilities. Another difficult thing for me was css, since I am still a beginner at that. None of my css is complicated, I tried to use it minimally to structure my pages and make it look as organized as I could.

Overall, this project was very enjoyable for me. I found flask to be a straightforward and useful tool, along with the fact that it is in python which is my best language. Since I'm familiar with python, I was allowed a lot of my time to spent on getting React to get the queries that are being made in flask over the network. Although due to inexperience there were a lot of obstacles on the front end, it was still a fun project that I plan to spend more time on in the future. There are still a lot of things left to be implemented. One of these things is theory that would be implemented with the piano. This includes chords, inversions and augmented and diminished chords. I figure that the further I progress in my piano studies, the more I can add to the site with certainty that I understand the material. Another thing that could be added is the resources that I have found useful. As a beginner, technique can be counter-intuitive at times and there is a lot of complexity to attaining relaxed and consistent playing (I am still working on it). I imagine adding this would be

similar to adding the theory, because the more I study piano the more I will be able to grasp these concepts and organize them in a helpful manner. To summarize, I am glad at how this project went, but there is still a lot of room for improvement.