Reflection

Development process and outcome

The overarching theme of my project was to make my anime tracking web app, AniDEX. The first thing I did was start designing on the front-end side of the AniDEX as I was most familiar with this. Next, I thought about the various stages and pathways to complete this project.

Stages

The first stage was to finalise the front-end side of AniDEX and some of the back-end side such as user's login, logout and signup. The next stage was to either add an API of an existing database or create my anime database based on user's input. The final stage was to combine the anime database either via API or own users' creation to the search page and users home page. Within the search page, users should be able to add anime to their list. On the home page of the users should also be able to remove anime from their list and edit the episodes they've watched.

First Pathway

The first pathway was the "MyAnimeList (MAL) API pathway". MAL is an existing anime database that has almost all Japanese anime in its database. The API pathway did not as smoothly since it turned out to be very hard to retrieve the authorisation token from the API. Therefore, I promptly switch to the second pathway; own users create the database pathway, which then has its complications.

Second Pathway

In the second pathway, I made a basic database with phpMyAdmin. In that database, users can add, remove and update specific attributes such as how many episodes watched in their home page and remove anime from their list. I changed the way searching was handled. Initial it was a form post request which would then display the results of what was posted/searched. But for the purpose of this small-scaled assignment, I changed it to show all the anime in the database on load and then filter out the result in the search bar.

First Issue

The first issue I faced was a repetition of the results on the search page. The function of the search page was to display all anime in the database then filter the results based on the user’s input. What happened was with the search function on the back-end side of things, the SQL query was not specific enough. It was just joining all the user’s id and anime into one result. Which was needed but I didn't need the ones that the user already

had. I did more research on SQL joins and figured out that left joins could be applied twice to the same table, that was what I needed.

One sub-issue was the databases data was not normalised or even in third normal form. Thus, the results had redundancy I managed this time by doing a hackie way of left join twice to the same table.

Second Issue

The second issue was I was too fixated on this crud web app assignment when 2-3 weeks in I had the basic requirements of the assignment. I could have just submitted what the assignment required. But instead, I made everything functional and interlacing such as, fixing the duplicates in the search results, displaying the correct watched episodes for each user and whether the user has the anime in their list or not.

Future Implementation

In the result page, when filtering if there is result show something such as "the result could not be found would you like to add it to the database". Then carry over the search result to the anime name field on the add anime page.

On a Larger-scale project of this web app, I would have, for more efficiency, made the search bar submit first then search the database for results, then make the searched result page paginate.