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. To add on to this project, as suggested, I made the database display images via URL and image file uploads.

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.

Action Plan

The first pathway was the "MyAnimeList (MAL) API pathway". MAL is an existing anime database that has almost all Japanese anime. This did not go so well so, I promptly switch to the second pathway, which had its own complications.

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 live filter out the result Sin the search bar.

First Issue

The first issue I encountered 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 the SQL query was not specific enough for the search function on the back-end side. The query was only joining all the user’s id and anime into one result, which was needed but it created a duplication of the results in the list. I did some further research on SQL joins and figured out that left joins could be applied twice to the same table. In the end the double left joins finalised the search query.

One sub-issue was the database was not normalised or even in third normal form. But I still managed the redundancy in the results by left joining twice to the same table.

Future Implementation

In future I would make further developments in the search page. For instance, when a search result is not found it would display an option to add the searched item to the database, by carrying 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 improved the search result page by generating a new list of the search result instead of live java-scripting search. Furthermore, I would have added the feature of the searched result page being paginate.

Another implementation I could have added in this project would be a table in the database similar to an archive. This would serve the purpose if a user deletes an anime from their list it can be stored in the archive for other users to bookmark (assuming when the user deletes the anime no other user had bookmarked it)