SOFT252RCW reflection document

Design Process

I started off by creating a rough version of a UML diagram on paper that I would use as my bare bones structure for the project following the MVC design pattern. The rough version had three classes, Users as a superclass, Admins that extends the users and resources which would be their own superclass. From there I broke down every requirement into its own method that could be called from a JFrame or by a button press and started working on the project. I tried not to get too focused on the initial design because I knew it would change throughout development and the initial UML was so bare and different, I decided not to include it with the submission but still submitted the final up to date one.

Significant design choices

I stuck to the Model-View-Controller design when creating the system so that what the user saw looked clean and simple while the code behind it worked to keep it updated. During the process of programming everything I used lists to store all the users, admins and resources that were created so that the tables I had on the JFrames could easily display two to five details about the object while keeping the rest hidden and then could easily compare a selected item from the table with a part of the object that it came from to relate them and make sure the user gets the correct resource.

After a significant amount of development time passed, I found myself working backwards by creating the JFrame first, designing its layout and then adding methods and code to the buttons that connected back to my classes. I believe this comes from me spending a lot of time developing in unity where I create the game objects first and then apply the scripts to them.

I’m sure there were better ways to display information and lists on JFrames without using the tables, but I still find Java quite difficult and that was the method I managed to get a good understanding of quick enough to develop this project in time for the deadline.

The degree to which the design meets good design criteria

I believe my project meets good design criteria because it meets all of the user requirements and manages to keep them all modular so if there was an issue with one feature of the project it wouldn’t completely crash the entire program. This was excellent for me because it allowed for quick and easy troubleshooting when something went wrong or threw an error, also if this was a project for a client and handed off to their development team, they'd easily be able to see which part of the project does what and easily fix issues if one got thrown at them.

Shortcomings of the project

The project has a few short comings such as the code can look quite messy behind some frames since I used very basic techniques to make sure the application behaved as intended.

There are a couple of methods that throw errors when removing objects from lists because the lists become empty and I couldn't figure out how to remove these without massively changing the project before the submission, but the project still works even though it is throwing these errors out sometimes. If I had time I would dive deep into the error and make bigger changes to the code to see if I could permanently remove these errors.

I also got very caught up in making sure the project works and has serialization of the users and resources that I didn't have time to go back and change the project so there is only one log in page for both users and admins. Currently there are separate log in pages for admins and users and I would have changed this if I had time.