Final Report for Sean Griffen

Group #AD_8

Sean Griffen

AD_8 Sean Griffen

I learned that you need to speak early and often to people. Attack problems as soon as they arise between people because those problems may just turn into habit and become worse over time. Stay in contact with your team members. Always be asking when their next commit will come, what will be in it, and say the same about you. It is always good to have the latest information about the project so you can plan for changes in your own coding.

Spring was interesting to learn about; it was quite a lot like java in syntax, but it had a few minor differences. The introduction of controllers was an easy one to process: controllers are used to take in http or web socket requests and respond back to the requester. Services were also easy to figure out: services are used to help minimize code in the controller by having methods that the controller can call on. The different types of mappings (get, post, put, and delete were the ones I used in my group) were difficult to figure out; not difficult to place in the code, just understand which types of requests were meant for each mapping. I personally did not work with web sockets much, so I don't really know who difficult those are with spring, but judging from my teammate who did extensive work with them, they were not the easiest to implement. Making entities and mapping them to a database table was easy as well: make a regular java class for the eitity and put the right annotations (@Entity and @Table) above the class name. Making relationships between entities was not hard, I just failed to understand the real benefit of having relationships. This was also true for having element collections with embedded objects in entity classes. I knew the benefit for having embedded objects, just not the benefit for defining a specific relationship type.

The timeline of our project completely fell apart. For this project to work, we needed the final implementation web sockets to work reliably at least 1 week before the web socket demo (demo four I believe). We got a version of web sockets working for demo four, but the implementation of web sockets that we needed did not work until one week before the final demo. This put major pressure on the frontend to get displaying different locations on our game map working, which meant that a lot of the code I worked on was not really getting implemented and tested by the frontend until about one week before the final demo. I did use postman to test my http requests a lot and to provide examples of syntax and responses of the different queries, but there were some cases where the frontend couldn't work with the format I laid out, or just needed more information from the database, so there was a major crunch to get at least a vision of what we were

going for for this project working for the final demo. In the future, I would put a real effort to defining a flexible syntax (by flexible I mean a format which allows the addition or subtraction of response and query data but still has the same base format) with the frontend for communication between us so this problem as a much less chance of happing.

In the end, we got the basic gameplay and design working: we got players to create and join game sessions, and in those sessions, players were correctly assigned targets and were able to eliminate targets on the game map if within a certain radius, and items were randomly generated and places in random locations within the game's play area. The project was a minor success in my mind because there were a lot of features that were left out: a global leaderboard that ranked players by a kill/death ratio, or games won, a store where players could purchase items not generated at the start of a game, and items were not even used on the frontend, they were only ever fully implemented on the backend.