Reflection note

By Daniel Lysak

Personal contributions

I would say my personal contribution to this exam has been mainly setup, which include:

- Setting up GitHub, branches, rules and training/getting team up to speed on collaboration using GitHub.
- Made sure everyone understood the implementation of the SUT and the discussions while programming.
- Maven
- Continuous integration
- Requirement 3
- Predesign of SUT (more on all these points under).

Learning experience

In my opinion the hard part is the collaboration, in this exam we chose to do as much as possible designing/setup before we start programming the SUT itself. I'm not used to this form of collaboration as usually in this early stage (first year grad) it's much easier to just jump into coding from the get-go and take the design/errors as they come.

Designing the whole SUT from the start was challenging and took a lot of foreseeing, but by doing it this way we managed to not have any problems when coding individually. More on this in the following section (What I wish I did better).

I learned the importance of testing and why we do it, I also learned how to implement unit/integration/acceptance in practice.

I used this exam to experiment with lots of new and unknown subjects. Selling the use of new and not exam expected (subjects we still have not learned) tools was hard. As I believe some of the team members wanted to stick to the required/learned tools and methods of working with the project.

Some of the examples of the tools in mind are:

- Setting up and using Maven and pom.xml to load dependencies. We have still to learn this, but I wanted to include use of Maven in this project for learning experience and ease of use for both my team members and the examiner.
- Individually I experimented with using java functional style of programming for learning experience, I believe this style of programming is widely used in the job market.
- Using DevOps practice called Continues integration (CI), as we were trying to design everything before code was written, I was sure we would encounter lots of merge conflicts. I wanted to learn more about this widely used practice.

What I wish I did better

As we wrote the Test plan, we divided the requirements to individual team members, this was the plan from the start and big reason for why we chose to design the whole SUT from the get-go.

I had visualized and written down my own perception of how the SUT would look like at the end and so had some other team members for their individual requirements. The hard part was visualizing their design and then also visualizing how this will impact my design of the overall SUT.

I wish I focused on designing my own requirements, then let the rest of the team design theirs. I probably tried to play Tech-lead too much 😊

Not to say that the design did not improve/help other members to understand the overall SUT and how it MAY be implemented.

I wish I understood Java better and how everything is connected, as this would save lot of discussion time that were used to understand each other's intentions.

Collaboration improvement under SUT

Overall, our collaboration was good. Everyone did their part, some excessive time discussing and/or disagreeing shows our team is not scared of sharing ideas and discussing them. Even if this took some time from the project. I believe this can be contributed to us being new to collaboration when coding and are standard growing pains of the learning process.