CSC326 - Reflection Paper

**Team Forming**

Team forming activities in the beginning of the project helped with setting common goal, ground rules. We had a better understanding about responsibility, accountability, and consequence for not participating. The part I like the most is that we should let the team know if someone is stuck so that others can help her out or ask for help from instructor/piazza. People will ask for help if they are really stuck, but most of the time, people will be able to figure out themselves if they spend much time on it. It’s good to mention it in ground rules because we feel more comfortable about asking help even we can figure something out with more time. However, we missed the consequence for not fulfilling the task. Even we demonstrated that we will have a to-do list before each meeting, we don’t have an action to take if anyone doesn’t meet the goal. Next time, I will add the consequences for not meeting the goal to the accountability session. Another thing I want to change is how we do the work. This time, each person has a task to finish. I think it’s a good idea for each person to have a major task and a minor task. People are responsible to finish the major task and help others with the minor task. The major task can be the something individuals want to try and minor task can be the one they are good at. This way, we can learn more from team members and get exposed to the different tasks with the help of an “expert”.

**Requirements**:

Writing the use cases did help with better clarification and understanding about the requirements, but details are usually hard to get through the statements. I needed to ask Kai for more specification, but I think it’s good. This is how we will deal with the real-life project in the future. We will need to translate the user stories to the use cases that is more understandable to developers. There will be many back-and-forth discussions about the specific details along the implementation.

We divided the work. Each person wrote one use case and it saves a lot of time. By translating long paragraph into use cases, we can find the information more easily. However, it can be the case that one person misunderstood the requirements and others didn’t go back to check the initial statements. Then, it takes more time to fix things. Next time, I will have at least two people to write the same use case separately. By comparing their use cases, they can find out the difference and ask the third person, like another team member or TA, to review and decide which to use.

**Configuration management**

We managed our branches differently in iteration 1 and 2. During iteration 1, we created different branches for different use cases. We were divided into 2 groups working on the different branches separately. It caused conflicts when we tried to merge to “development” branch because we edited the same file. We couldn’t resolve the conflict and ended up copying all the files to the “development” branch. During iteration 2, we decided to work on the same branch “UC16”. However, it really slowed our process down and actually didn’t solve the problem. We still got a conflict when merging to the “development”. We realized that we should have created a new branch instead of working on the old one at the beginning since the histories of commits are different. We copied the files to the “development” branch again.

We didn’t really do well in branch management, but I know what to change next time. We will not push the changes that both subgroups will edit to the GitHub until all the safe changes are merged to the “master”. For example, there is a group A working on UC1 and group B working on UC2. They both need to edit the existing file “conflict.java” and add other new files. Group A and B will push all the changes except the “conflict.java” to their branches. After these two branches are merged to “master” successfully, we can create a new branch to edit the “conflict.java” file. In addition, instead of creating all branches at the beginning, it is a good idea to delete the old branches after each iteration and create new ones for next iteration. The new “master” branch will have all the updated merged files and is less likely to cause conflicts.

**Jenkins Executors**

We didn’t do well in managing Jenkins executor because we didn’t merge to “development” as often as we should. We will change to do small pull requests instead of huge ones next time. It keeps “master” and “development” branches updated and allows us to fix small problems along implementation.

**Iterations**

We spent the first two weeks to implement new users, UC15(emergency record), and UC16(personal representatives). Then, we spent the rest of the time to finish UC17(lab procedure) and UC18(immunization for extra points). It works well but next time I will suggest to start the hardest task(UC17) as soon as possible so that we have more flexibility to change our plan accordingly.

**Processes**:

We used the wireframe process to reflect on the changes of our design and used issues on Github to associate commits with issues. I don’t benefit from updating the wireframe because we usually change the wireframe according to the website. It’s an extra work to do. In addition to issues, we also used milestone on GitHub, which is not required. I found it helpful especially when the tasks are divided according to iterations. We actually created most of the issues during the first week so it really helps me to quickly find the information. We also used labels, like enhancement, extra credit, and checkstyle, so that we can distinguish and prioritize the issues. Moreover, we used bug report to describe the bug so that other team members can regenerate it. Next time, I will use more labels with different colors.

**Communications**

In addition to in-person meetings to share information, we set up the ground rules at the beginning and agreed to use GroupMe to send message. We communicated effectively during this process. If we are stuck or have questions, we will describe the issues and ask for help in the group. If anyone knows how to do it, they will send message to each other instead of to the group.

**Team Collaboration Reflections**

There are 2 different reflections in the lab. One is reviewing other team’s work and giving feedback. It gives me a new perspective to view the issue and to approach the same problem. The other one is reviewing the work done by your team and asking questions. It makes sure everyone in the team is on the same page. Both ways are helpful but I prefer the reflection in the team because team members usually know what we are doing and can give more valuable feedback. Sometimes, reviewing other team’s work can be tedious especially if they want us to give feedback on specific code. The most common thing we can give is that more comments are needed because it is hard to figure out the code in a limited amount of time.

**Division of team roles**

We took turns to play the roles, but in the fact, we didn’t really focus on the roles. We usually know each other’s strength after the first iteration. There are implicit technical leaders. I know who to find if I have different types of problems. I think it is a good thing and a bad thing at the same time. It can save my time to try to find someone who knows the answer, but also omits the possibility that someone may know the answer better.

**Leadership**:

Perfect team leader should be calm, have good communication skill, and know how to find resource. In my exercise, I don’t think I did a good job in communicating, but I was calm enough and knew how to redirect team members to the resource. Calm team leaders knows the schedule and keeps things on the right track. Good team leaders should be able to communicate effectively with team members, which is also essential during team building. Since team members usually ask team leader for help and resource, it is necessary that either they know the answer to the question or they know someone who knows the answer the the question.

**Enjoyment**

I really enjoyed this project because I really like all the team members and enjoyed so much to work with them. It is so great to have the chance to know each person in the team and work with them.

**Advice**

I think it is good to talk about the OneToMany and ManyToMany during lab or lecture. I was so confused by it and couldn’t figure it out.