Retrospective summary

This time around, what we did differently was to use planning poker. This was great in many ways. Firstly we were able to discuss in plenary the expected workload on the different requirements that had to be done. The team would have a common expectation of what to expect in terms of the amount of work that was going into each task. Doing planning poker also led us to consciously make big requirements into smaller ones, because we experienced last sprint how overwhelming it could be if it wasn't done. Breaking the larger requirements down to smaller ones also gave us a better understanding of what we had to do, and also what to prioritise, making it quite clear which functions we had to start develop first.

The use of the Trello board improved as well; the team members were consistent in tagging themselves on the tasks that was pinned up. This lead to a better workflow because everyone knew what was being worked on at all times, so that there would never be duplicate work done.

Another thing which went very well, albiet not planned, but inevitable, was branching out. The team was pretty much scared of merge conflicts from the start, but at some point during this sprint, we saw that we had to branch out because we saw that what we was working on could break our current working code, but also the fact that several features were being worked on at the same time. This in general was a safer practice than what we previously had been doing, and turned out to work pretty well.

In general, this sprint has been a very good one for the team; not very much has not been working out for us. The members have been good at reporting what they currently are working one, what they finish, current problems they are facing and what they see need to be done. What could be improved is better documentation and explenation of written code. In the current state, it is not always easy to get into code you have not written yourself if you want to contribute to something. If you wanted to contribute to something that you haven't worked on, you would have to be walked through the code like a baby, which is not ideal. The first step to fix this is to be stricter with comments and formatting of code before pushing any new changes. Javadoc which is a pretty nice way to describe methods has always been on our minds, but we ended up not following it of laziness. But Javadoc is something we atleast will try corporate just because of its simplicity and a convention every individual team members has used before.