Continuous Integration Report

Group Name: Team siKz
Group Number: 6

Ryan Bulman
Frederick Clarke
Jack Ellis
Yuhao Hu
Thomas Nicholson
James Pursglove

5a)

We decided to use Github Actions[1] as the foundation for our Continuous Integration system. This is because it is already highly integrated with our existing codebase on Github, meaning that there would be minimal setup time as well as reducing the risk of complications from bringing a new system into use. Similarly, as a group, we are familiar with Githubs' UI and workflow. This means that even though it is a new system to learn, elements will feel comfortable to use, which should also help to speed up the process of setting up a CI system.

As per Ian Sommerville's description of continuous integration in an agile development environment like ours, the system should automatically integrate any work on a finished task into the codebase, and provided all present unit tests are passed, the main system is automatically updated to include the new features.[2] There are a number of reasons why our environment is suitable for this method:

- No network/live system connectivity - (NFR.NETWORK)

If our system were connected to a network or had live users, this may not be suitable, as tasks could be pushed which may not be feature complete, or may introduce bugs and exploits.Which would place great stress on the risk of incomplete unit tests. Instead we are working in a private development environment and can have a slightly more relaxed CI system.

- People working on distinct branches

We will divide the development of the code so that people are working on distinct features with a lot of variance in each branch. Manually merging, resolving the conflicts from these mergers, running all the unit tests and building a .jar file could combine to require a prohibitive amount of effort. With this style of reactive CI system this barrier is lowered, and people are more encouraged to work together and contribute to others' work, further enabling our agile development style.

- Limited scope of the system

The system, as it exists currently and in design, is relatively simple and lightweight. This means that we can run multiple automated build cycles in a short period of time - if necessary. Were we working on a larger system, we might have to choose a stricter system that holds changes until there is a backlog to be performed at once. Instead we can build when we need to, letting us make changes rapidly and respond to feedback from group members quickly.

5b)
The actual CI infrastructure we ended up using was relatively similar to the one described in principle. However, it was a limited version in comparison to the ideal scenario mentioned in part **5a)**.

This is primarily due to the limitations of our development environment; the master branch was protected and therefore required manual approval for all the changes. This meant that it was not possible to automatically push updates as soon as they were ready. However, this did not affect workflow too greatly due to the small scale nature of the project and that whenever a change was ready to be pushed to master somebody was around to approve the change quickly.

Secondly, merge conflicts required human intervention to determine what the correct decision was in each scenario. The CI system was not able to help with this issue, but again this did not limit the output of the project too much, as merge conflicts are an expected aspect of working collaboratively in this form.

With these limitations in mind, the final CI system implemented through Git Actions is relatively lightweight. When a branch has passed the human review, which is not infallible and is often poor at catching potential errors, the build must first check that there are no outstanding conflicts between the files, it then automatically runs every unit test are complete, failing the build if any unit test fails. Finally, an executable .jar file is automatically built.

Another workflow added through Git actions was one that automatically built and published the website on each push. While not as many changes were being made to this aspect, this automation helped greatly with working on the website when changes were needed.

While it is not the full scale, seamless, system as envisioned by Sommerville and the original section, It is still a viable system that helps streamline the workflow and reduce the need for human intervention at each step.

Bibliography

[1] https://docs.github.com/en/actions
[2] I. Sommerville, *Software Engineering*, 10th ed, Harlow, Pearson Education, 2016