5. Continuous Integration

Group 11 - 11 Musketeers
Osama Azaz
Adam Dawtry
Tom Jackson
Brendan Liew
Holly Reed
Harry Ryan

5.a

- Our continuous integration methods and approaches were based on research we carried out and reading of Ian Sommerville's Software Engineering book. [1]
- The most important aspect of the approach was to ensure that whenever a commit was pushed to Github, testing would be carried out that was completely automated and required no human input.
- Another key aspect of our methods and approach was the idea of continuously updating code. From the Spring Week 2 Continuous Integration Lecture, it was made clear that developers should integrate their code back into the code base as soon as changes are made. [2]
- It was possible for us to implement this approach as the system is not very large and the development platform is the same as the target platform, meaning it is possible to run system tests in a private workspace.
- H. H. Rodriguez explains four key aspects that we based our methods and approaches around to ensure our continuous integration was well formed and successful.
- A version control system is used to store the code and additional files needed for a build. The build must be automated and run without human intervention, it must run unit and integration tests, and it must run on a separate integration machine.
- We decided we would use GitHub Actions to implement continuous integration as the platform offered good functionality and we had already been using GitHub in the current and previous project. GitHub Actions also provides a nice summary of the continuous integration process after each push to the repository. It details who triggered it, its status, the total duration, and the number of artifacts produced. We found this view very useful as it provided a quick overview of our implementation therefore we continued to use GitHub Actions throughout the whole project.
- In summary, the contiguous integration method builds a jar file and runs and outputs the unit tests results every time a new push is made to the Github repository.

5.b

Key points

- Platform Used: GitHub Actions.
- Workflow: Java Continuous Integration with Gradle
    - Java CI with Gradle uses a gradle.yml file.
- Artifacts: unit-tests-results, york-pirates-jar

Brief report

- Using the platform GitHub Actions, we decided to use the workflow Java Continuous Integration with Gradle.
- The accompanying gradle.yml file was set up to run all of our unit tests and to build a new executable jar file every time a new implementation was pushed to the GitHub repository.
- The unit-tests-results artifact details a summary of all the tests carried out. This includes any tests that failed or were ignored, and how long it took to carry out the tests.
- A number of different workflow runs were carried out as we tested and tweaked the gradle.yml file. We first implemented the automatic running of the unit tests, then updated it to also produce a jar file.

References:

[1] I. Somerville, *Software Engineering*, Global Edition. Pearson Education, 2016. [E-book] Available: https://ebookcentral.proquest.com [Accessed: 22 April 2022].

[2] H. H. Rodriguez, Engineering 1, “Continuous Integration.” ENG1, Computer Science, University of York, York, 2021.