Continuous Integration

Group 18

Team B

Olivia Betts
Zac Bhumgara
Nursyarmila Ahmad Shukri
Cameron Duncan-Johal
Muaz Waqas
Oliver Northwood
Teddy Seddon

Methods and approaches:

Continuous integration, CI, is a software development practice where members of a team integrate their work frequently, usually daily, with each integration being verified by an automated build to detect integration errors as quickly as possible. This allows for issues to be detected much sooner and means they are smaller thus easier to solve.

One benefit of CI is development velocity. Ongoing feedback enables developers to make minor changes more frequently, instead of waiting for one release. Another is in stability and reliability since automated, continued testing ensures that codebases stay stable and are release ready at any time.

The first step in CI is to ensure that the code runs and shows errors as we planned. It also has to be robust such that any code pushed to the version control will not break the CI. We used Github Actions as our CI server since everyone in our team is already familiar with Github and it is also widely used. Github Action offers workflows that can build the code in a repository and run tests. CI begins in shared repositories, where teams collaborate on code.

Firstly, we set up a workflow file and wrote unit-testing code so that our code could be tested regularly after pushing code to version control and any errors could be detected immediately. The code also should be able to automatically build java files (.jar file) after code is pushed to version control.

Secondly, we wanted to make sure every integration is tested. It is triggered by changes in the existing code. Integration testing will run every time there is a change in the code so that the integrated modules work as expected.

Thirdly, we wanted to ensure that every delivery is automated to reduce any error that could have occurred if it is done manually. This can be done by documenting the deployment process so that everyone can have access to it and understand what has been done.

Report on the actual continuous integration infrastructure:

Firstly, we have created a workflow file on Github Actions YAML. This allows us to automate our build and testing process, by specifying some actions which are executed every time someone commits code to our repository. These steps include 1. Building the project, 2. Running automated tests and 3. Generating a JaCoCo test coverage report.

We have also set up an automated build. This helps us as developers to commits code changes more frequently by running automated tests on every commit. This helps to ensure that new code is thoroughly tested. This will also mean that any errors are caught relatively earlier in the development cycle (as opposed to right at the end). This not only saves time, but also reduces the overall risk of issues being introduced into our codebase.

Thirdly, we have integrated a JaCoCo test coverage tool into our workflow. JaCoCo generates a report that provides details about any areas of code that are not being tested. We use it with automated tests to ensure that our tests are comprehensive and complete. This helped us to improve our quality of code and reduce the overall risk of bugs. It also helped to make debugging easier.

Lastly, we have added a JAR upload to the workflow in YAML. This will ensure that a JAR file is available when needed for use. This is important because it means that our group can easily distribute our software to our client.

Unfortunately, due to time constraints, we were unable to make the workflow pass successfully. The workflow kept failing due to the fact that the gradle wrapper could not be located.