LO5 - Conduct reviews and inspections and design and implement automated testing processes

In summary, this document outlines developing comprehensive test cases, ensuring the automated triggering of the pipeline, verifying the build process, executing, and reporting on automated tests, simulating deployment, and testing rollback procedures. These steps are crucial for ensuring the CI pipeline responds correctly to changes in the codebase, successfully builds, and tests the software.

Processes I used:

IntelliJ IDEA was used during the development and was the main driver of catching issues early. This was done by error handling/warnings. Supplementary to this, I would review by hand the code to also catch issues before moving to the testing stage.

How a Continuous Integration Pipeline could identify issues:

A CI Pipeline would be able to identify a myriad of issues at multiple stages…

At the Build Stage: If the code builds successfully we know the code is functional.

At the Test Stage:

- If all the unit and integration tests are passed successfully we will be able to state that we have total functional correctness.
- If the system can work under an increased and extraordinary load.

Outlining Continuous Integration Pipeline Plan:

When implementing a CI pipeline, it would be proper to introduce a multi-stage approach. These would be the source, build, test stage, and deployment stages. Firstly, as I have been developing using IntelliJ IDEA the code would be managed using Git, and then stored in an open repository such as GitHub. This enables us to automatically track changes and version control. This is the foundation of continuous integration.

Next is the build stage, I am using Maven which is crucial in compiling the source code and linking all the essential libraries, as well as conducting initial tests. This stage is also essential as it determines if the build is stable and therefore can carry on to the next stage.

Following the build stage, the testing stage is where we conduct various and thorough tests as we have extensively outlined both this document and the supporting linked documents. Finally, the deployment stage where we release the build into a controlled environment for alpha and beta testing. Upon success of all this the software can then be rolled out to the end-user.

<u>**How I would automate aspects:**</u>

As mentioned previously, automation of testing would be very useful. If automation were implemented it would affect the following.

Unit Testing: Ensuring the correctness of new features.

Integration Testing: Ensuring newly modified functions continue to function in conjunction with others.

Regression Testing: New functions do not negatively impact completed functions.

Performance Testing: Ensuring that the new build performs under our requirements (specifically non-functional requirements)