Continuous Integration

Group Number: Cohort 2 Team 7

Group Name: pickNmix

Group Member Names:

David Lun

Sameer Minhas

Harry Muir

Phyo Lin

Alex McBride

Introduction:

Continuous Integration was essential for our project because it enables us to detect issues early, and ensure that our code is consistent and maintains a high quality. Continuous integration also supports automated testing, building and artifact generation.

The Continuous Integration was created using GitHub actions for an automated build with a gradle.yml file. For our project we have developed two continuous integration pipelines: A build pipeline and a dependency-submission pipeline. The build pipeline was appropriate because it compiles the code and provides artifacts for testing. The dependency-submission was appropriate because it creates a dependency graph that can identify all of the project's dependencies (e.g libraries and packages) in one place.

Build Pipeline:

Inputs:
- Source code of the GitHub repository via actions/checkout@v4
- Java Development Kit version 17 (JDK 17) via action/setup-java@v4
- Configuration of Gradle via gradle/actions/setup-gradle

Outputs:
- EngSim-*.jar

Triggers:
- Push events to master branch
- Pull events on master branch

Dependency-Submission Pipeline:

Inputs:
- Source code of the GitHub repository via actions/checkout@v4
- Java Development Kit version 17 (JDK 17) via action/setup-java@v4

Outputs:
- dependency graph via gradle/actions/dependency-submission

Triggers:
- Push events to master branch
- Pull events on master branch

Build Pipeline:

This build job is triggered on every push event to the master branch and every pull event on the master branch. The build pipeline setup requires the latest ubuntu to ensure the linux based environment is consistent. The build job takes in three inputs: the source code of the GitHub repository which was checked out using actions/checkout@v4; Java Development Kit version 17 (JDK 17) that was setup using action/setup-java@v4 and the temurin distribution; the configuration of Gradle which was set up using gradle/actions/setup-gradle. These inputs allow for the setup of Gradle and JDK for the build environment and so that the game can be built using the Gradle wrapper with this command, run: cd game; ./gradlew build. We have reduced workflow execution time by setting up Gradle with caching enabled which reuses downloaded dependencies.

The build pipeline then outputs a compiled JAR file, EngSim-*.jar, The wild card (*) will change based on which version number the JAR file is on e.g EngSim-2.0.jar. The build artifact is subsequently uploaded to GitHub using actions/upload-artifact for testing, a benefit of this is we can reuse it in other workflows. The pipeline also uploads the workflow artifact that is the JaCoCo coverage report which uses actions/upload-artifact@v4.4.3 for quality analysis.

Dependency-Submission Pipeline:

This dependency job is triggered on every push event to the master branch and every pull event on the master branch. The dependency job takes in the same two inputs as the build job: the source code of the GitHub repository which was checked out using actions/checkout@v4 and Java Development Kit version 17 (JDK 17) that was setup using action/setup-java@v4.

The dependency-submission pipeline as an output generates a dependency graph of the project's libraries and dependencies using gradle/actions/dependency-submission which is then uploaded to GitHub. This enables Dependabot Alerts for all project dependencies and monitoring vulnerabilities.