Continuous Integration Environment

March 14, 2022

Group 20

Table 1. Name and social security number of each group member.

Simon Andreasson	19981011-4919
Nohman Hussain Tahir	19910601-5614
Florin Adrian Vasiu	19980527-T393
Suvoj Reddy Kovvuri	20010306-T032

Link to GitHub repository: https://github.com/naganannan54/Software-Testing-Project

1. Introduction

The report consists of information on CI (continuous integration) and information regarding the environment setup and the tools used for the configuration. This report also consists of the tools and softwares used for the integration. The report also consists of the challenges we faced during the project. At the end of the report we included experiences of all the contributors of the project.

2. Environment setup

At the start of the project there were some issues getting in touch with all group members, and thus the setup was started by Simon and Suvoj who decided to create a basic calculator in Java. Simon then created a GitHub page and a suggested workflow using a Maven project in IntelliJ with its built-in GitHub integration. While Suvoj started writing some JUnit tests, experimentation with a remote test server continued, finally settling on GitHub Actions. Once all the group members were contacted and connected, development continued where the remaining members focused on improving the test suite by adding more and better unit and integration tests.

3. The environment

A model of the continuous integration environment in its current state can be seen in figure 1. Since Java was the chosen programming language, IntelliJ was a natural choice for a modern and popular IDE that supports a lot of other plugins and integrations. The

same reasoning was used when choosing a Maven project with JUnit tests. They are popular tools that provide good compatibility while also being useful to learn for the future. GitHub was chosen for version control, partly due to previous experience, but also because of the excellent integration with IntelliJ, as well as being common in the industry. It also provided access to GitHub Actions which was eventually chosen over Jenkins as the CI tool to build and run the tests. It is conveniently integrated with GitHub, allowing it to update with every commit, and since it runs completely remotely everyone can easily access the results.

The basic workflow of using the CI environment can be seen in figure 1. If for example a new test should be added, it is written in the IntelliJ IDE using the JUnit framework. Once the code is written, it can be committed and pushed to the GitHub repository through the built-in tools in IntelliJ. After a commit is pushed to GitHub, the GitHub Actions component reacts and starts building the project and running the tests automatically. The results can be viewed on the GitHub page, either through the checkmark on each commit, or by looking at the GitHub Actions tab where each test is described in detail. The CI environment could also be expanded by adding more actions to GitHub, potentially testing more aspects of the project.

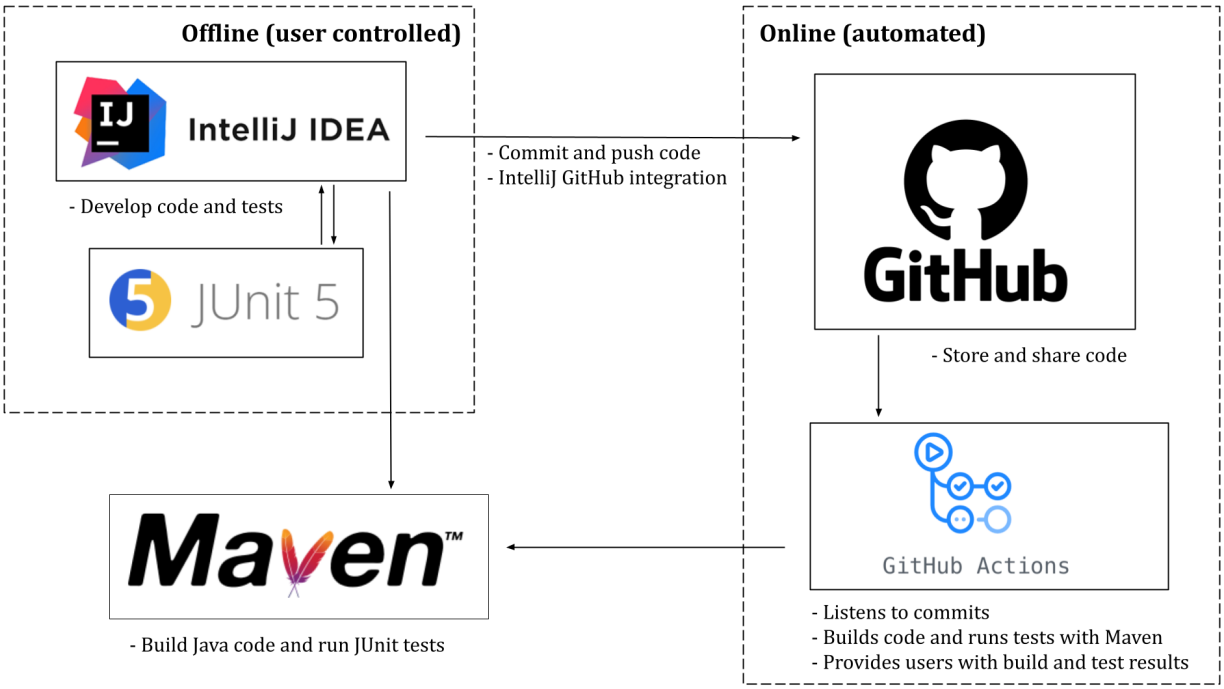

Figure 1. Architecture model for the CI-environment

4. Challenges

Due to choosing a common set of tools that are supposed to work well together, few technical challenges or issues occurred. Setting up the Maven project with JUnit tests worked immediately by following the tutorial in the IntelliJ documentation [1]. Uploading the project to GitHub and sharing it with everyone required a bit of work due to no previous experience with IntelliJ and its GitHub integration, but in the end everything worked well and everyone could connect and contribute to the repository.

The most challenging part was setting up the test server, and especially getting it to work remotely. At first, Jenkins was used and after some trial and error with the settings in both Jenkins and the project files it worked locally, checking for new commits every minute and then building and running the tests. It was rather unclear how to give everyone access to it though, even with the theoretical resources required through a spare computer that could act as a server. Thankfully, a different solution using GitHub Actions was discovered. Adding these through the GitHub website was very easy using the default Maven template and setting it up to build and run the tests on every new commit to the master branch.

5. Experiences

Table 2 shows an estimation of how much time each group member has spent on the project. In addition to the time spent setting up and developing the test environment, the reported times include research, meetings and report writing. In the following sections, each group member will describe their experiences using the continuous integration environment.

Table 2. Approximate time spent by each group member.

Simon Andreasson	36 hours
Nohman Hussain Tahir	28 hours
Florin Adrian Vasiu	31 hours
Suvoj Reddy Kovvuri	34 hours

5.1 Simon Andreasson

Setting up the environment was surprisingly easy and most of the tools seemed to work well together, requiring little manual configuration. I had some previous experience working with game projects in C++ where GitHub was used extensively, but I had never used any of the other tools in this project making it a good learning opportunity. GitHub Actions is something I will certainly use in the future, perhaps even with a full suite of unit tests, but at the very least to verify that new commits can be built without errors. Overall,

working with the CI-environment was a good experience that will help me attempt similar things in the future.

5.2 Nohman Hussain Tahir

Due to the fact that most of the work had never been done before, it was a challenging project. Many mistakes were made throughout the process and it took some time to fix them. Setup of the environment and connection of Jenkins to Git was particularly problematic. However, once that was accomplished, everything else was straightforward. There were different team members handling each step of the project. This was my first experience with such an environment, and it was difficult for me to manage the tasks. Learning new technologies like IntelliJ and Jenkins was a lot of fun. There were lots of challenges and we enjoyed learning how to deal with them. The project was a very good way to get a grasp of how integrated environments are set up for testing. I learned it with interest.

5.3 Florin Adrian Vasiu

Learning the CI environment has been a great academic experience. At the beginning of the project I had some time problems due to personal reasons, but my team helped me to continue with the project in the best possible way. Once the basics of the project were explained to me, I was able to experiment creating test cases, learn how to update the project with Github and work with the CI environment. This project has helped me a lot to understand and improve, so I am very excited to do a similar project in the future.

5.4 Suvoj Reddy Kovvuri

Although continuous integration is pretty new for me I enjoyed the process thoroughly. In spite of having initial hiccups I managed to follow up with help from my teammates. After the initial setup of the required tools I managed to add test cases and push into our repository through GitHub Actions which was cumbersome in the beginning but easy once I got the hang of it. Altogether I had a very great experience with the CI environment and found it to be interesting and I got to learn many new things out of it.

6. References

[1] JUnit 5 | IntelliJ IDEA (jetbrains.com) [Accessed: 2022-02-26]