Report on Continuous Integration Environment

Group 14	
Names	**Personal number**
Sree Lakshmi Hiranmayee Kadali	20010920-T244
Umair Saeed Cheema	19940731-2470
Murtaza Naqvi	19900509-7432
Irene Tejedor Herranz	19990909-T549

Introduction

During the course, we have studied different software testing techniques: automated and manual. As we know manual tests require more time and effort than automated testing, therefore it is important to understand the efficiency and effectiveness of automated testing to improve software quality. Unlike manual testing, where the test cases are required to be run once or twice, automated testing is more rapid in terms of repetitive testing.
Performing automated tests is very important for continuous testing and feedback. We need a continuous integration (CI) environment where automated testing is implied. In this environment, the developers write the codes in their local IDE and commit them to a shared repository, where CI monitors and examines the changes that are taking place in the code.

Continuous Integration (CI)

Continuous integration(CI) is a software development technique in which developers integrate their code in a central repository regularly, and then automated tests are created and run [1].It has many advantages :

- Detect errors and correct them
- Decrease testing time
- Faster software development
- Make new updates
- Divide the work equally

Our environment is composed of:

1. **Integrated Development Environment**: An integrated development environment (IDE) is a software package that provides the basic tools needed to write and test software. In our case, we have used IntelliJ as IDE.

2. **Version Control Software:** It is a system that helps us to search for the modifications made in the code, it allows us to control the flow of versions and if we find any bug in the software, we can roll back the changes already made.

3. **Version Control Server:** This host application tool performs actions like pulling and pushing code from this tool to do continuous integration.
   In our case we have used GitHub as a version control server.

4. **CI tool:** CI is used to record and analyze the changes and updates that are being made in the project by multiple developers/testers.We used Jenkins as CI for our project.

5. **Automated build tool:** Maven is used as the automated build tool. This tool connects to the CI tool and provides support to run automated test framework(JUnit).

Choice of Tools to develop the Continuous Integration Environment

- **Intellij:**

  Intellij is an open source development environment that has comparatively more satisfaction in terms of user experience.The reasons we chose Intellij are as follows:

  1. Includes most of the plugins to extend the core functionality.
  2. The pull and push to the shared repository is easy.
  3. Intellij has framework integrations such as Junit and Maven for continuous integration.

- **Github:**

  Github is an online platform, where developers can store their projects in a cloud based Git repository.All the changes made and the versions updated are being recorded in the version control system(Git). Below are some features of GitHub that we experienced in our CI project:

  1. We used it for creating the online repository as a team.
  2. It was easy to use as a group, where we required regular updating and maintaining the logs for all the changes made in the project.
  3. We used the git plugin to link gitHub to the intellij, where the changes made in the project were easily visible to all the group members.
  4. Pulling and pushing the code from the GitHub repository was fairly easy.

- **Junit:**

  When testing Java code, Junit is one of the best frameworks that is easily used with Intellij. Below are a few reasons to use JUnit.

  1. Adding dependencies to the Maven project was easy.

2. The results were easily displayed by running the Junit test by a single click on a play icon (User friendly GUI).
3. Success and failures were promptly displayed on the changes that were made.
4. And we observed Junit to be highly compatible with intellij for developing and testing purposes.

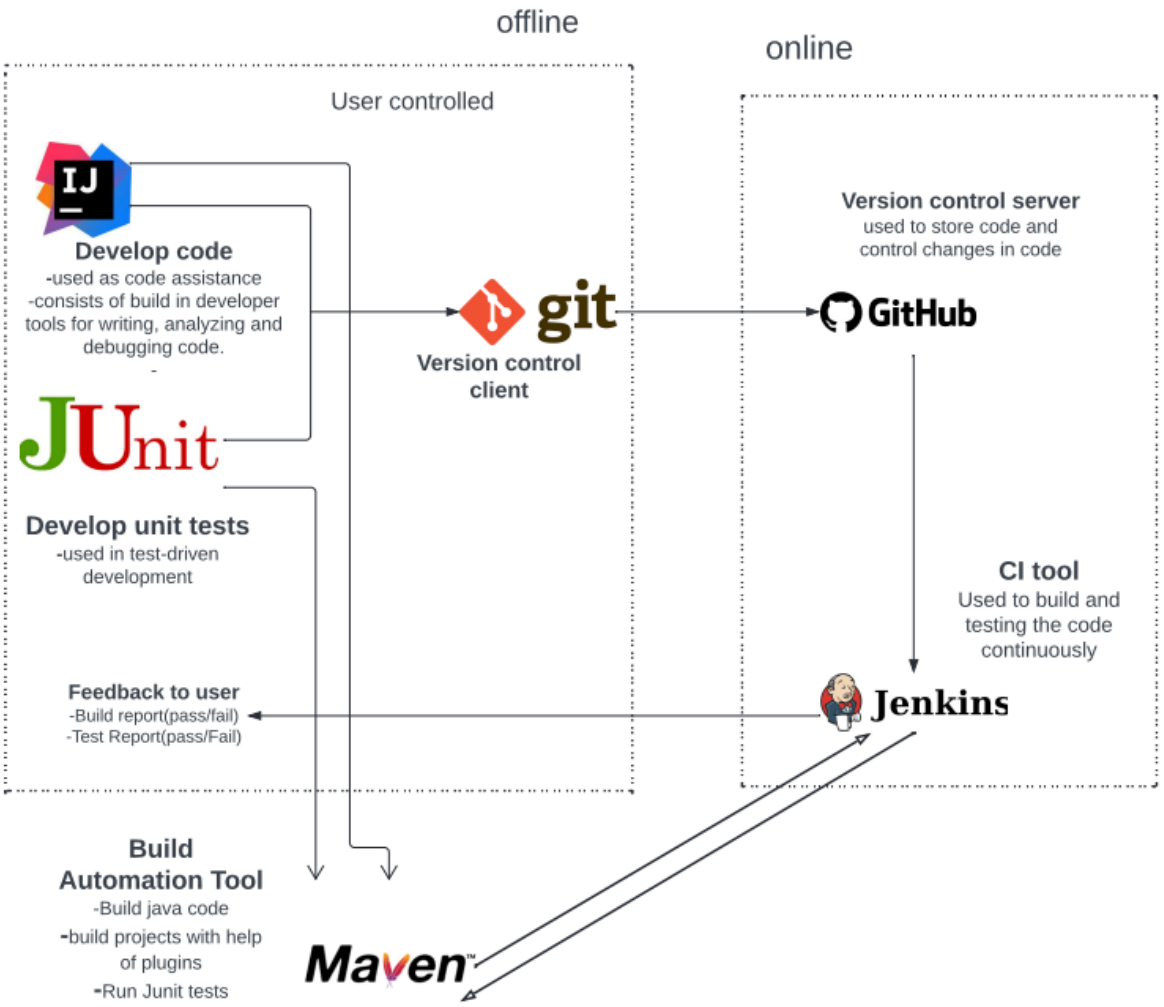
- **Maven:**

Maven manages the dependency of a project and the whole project lifecycle using a build automation tool. Maven is an Apache Software Foundation project that is primarily used for Java projects [2].
In order to automate tests, we would need Selenium dependencies associated with it. This is usually done manually throughout the lifecycle of the project, but as the project scales, managing dependencies becomes increasingly difficult. Therefore, Maven helps to automate these tasks.
In more specific terms, Maven is a tool for managing software projects using the concept of a project object model (POM). The user can create an initial folder structure, compile and test the final product before deploying it.

- **Jenkins:**

Jenkins is a well-known CI/CD system. It provided several plugins for integration with multiple tools and frameworks in the testing pipeline. Jenkins' plugins for test automation allow you to run test suites, gather and display results, and provide error information [3].

1. Jenkins offers plugins for multiple testing frameworks, such as Selenium, Cucumber, Appium, Robot Framework, etc. Automation tests can be run with these in continuous integration pipelines.
2. After being summarized, the test results are usually displayed in an HTML page by most plugins.
3. A trend graph is displayed by Jenkins for each set of results. This provides a better picture of how past tests have done.
4. A summary is compiled with the test results, and errors are logged.



Repository Link:

https://github.com/kadalihiranmayee/ST__project

Challenges and Solutions:

Challenge 1:
None of the group members were familiar with each other, therefore setting up a meeting and contacting each other was another challenge.

Solution:
Later the setting up of the meetings and one of the group members take care of interaction sessions and the work was divided accordingly.

Challenge 2:
The first challenge was the selection of the right IDE, that is easily integrateable with the version control server, Maven and JUnit.

Solution: Every group member does their research work and comes up with a tool that is easily compatible with the rest of the integrations. We shortlisted Eclipse and Intellij. Since most of us could not integrate Junit in Eclipse, therefore we started with Intellij.

Challenge 3:
Understanding the project requirements was another challenge.

Solution: All the group members read and present their version of understanding about the project requirements and that provided us further clarity to understand more.

Time logs

Names	Work Hours
Sree Lakshmi Hiranmayee Kadali	70 hours
Umair Saeed Cheema	52 hours
Murtaza Naqvi	61 hours
Irene Tejedor Herranz	49 hours

Experiences

Sree Lakshmi Hiranmayee Kadali: We all initially decided on working with eclipse, we tried to setup the environment with eclipse for few days but couldn't because of few errors. All the team again decided on moving to new environment i.e IntellJ. Working with Intellij was very easy compared to eclipse. The environment was set very easily with Inellij.Initially, we had different ideas for code but because of few errors, we finally decided on calculator error. screen recording was a little tiresome process while working on the assignment.Using GitHub was easy and pushing the code was simple in Intellij rather than Ecllipse.Regarding unit test and integration tests it was bit difficult but by watching some videos, made me to get to know about it. Finally, it was a good experience to gain knowledge about CI environment and all other tests.

Murtaza Naqvi: For me setting up the environment was challenging. I started with Eclipse to be used as an IDE but integration of Junit was having trouble initially, which was later resolved by the help of one of my group members. Integrating GitHub was also a challenge, and this was the point where we were recommended by one of the group members, who was experienced in using Intellij, to move to Intellij. Since, I had never worked with Intellij before, therefore it was something I took as a learning for myself to get to know the new IDE and it turned out that all the issues I found with integrating version control server and automated test framework was no more an issue, as they were easily installable within Intellij. Therefore, overall experience of working in Intellij, writing the code, committing the code and pushing the code to Github and later to Jenkins was very easy and my group members were very cooperative in all the process for helping me out where I needed them. Finally, the overall learning process of automated testing was superb.

Irene Tejedor: I had some experience with Intellij as I had used it in previous projects.

However, I had no experience with either Jenkins or GitHub.
Thanks to my team members, committing code to Github and Jenkins didn't become too difficult. Therefore, the biggest challenges I had were the integration of GitHub and Jenkins. With this project I have learned to work as a team, and the most important thing has been the knowledge acquired about automated testing.

Umair Saeed Cheema: As it was my first experience working on CI environment, it was difficult for me to understand how to set up the whole environment. Even though most of the part of setting the environment was done with the help of videos, I have watched of how to set up the CI environment; I was unable to install Junit5 properly in Eclipse. My other team members also faced this same issue. So we moved to the Intellij. Dining work on intellij was easier than Eclipse as it provides most of the plugins built-in. Jenkins was a new thing to use, and it was easy to integrate with GitHub. Overall, I have enjoyed working on this project.

References

[1] "Benefits of Continuous Integration," *GitLab*. https://about.gitlab.com/topics/ci-cd/benefits-continuous-integration/#:~:text=Continuous%20integration%20(CI)%20makes%20software.

[2] "Maven Tutorial – Getting Started With Maven for Selenium," *LambdaTest*, May 15, 2020. https://www.lambdatest.com/blog/getting-started-with-maven-for-selenium-testing/#:~:text=Maven%20can%20be%20more%20specifically (accessed Mar. 13, 2022).

[3] "Jenkins for Test Automation : Tutorial," *BrowserStack*. https://www.browserstack.com/guide/jenkins-for-test-automation.