

CA670 Software Process Quality

Name	Archana Kalapgar
Student Number	19210184
Programme	MCM (Cloud)
Module Code	CA650
Assignment Title	Assignment Two – Test Tool Evaluation using Selenium and JMeter
Submission date	12-April-2020
Module coordinator	Reenat Verbruggen

I declare that this material, which I now submit for assessment, is entirely my work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged, and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

I have read and understood the referencing guidelines found recommended in the assignment guidelines.

Name : Archana Kalapgar

Date : 12-April-2020

Test Tool Evaluation using Selenium and JMeter

Abstract:

In today's world, the web is the need of the hour. The vast majority of product applications are rendered online that continue to run in a web-based system. Physically testing these applications requires a lot of man-hours. Besides, it's boring and prone to mistakes. Test automation is a method in software testing that makes use of special software, in our case SELENIUM to control execution and run repeatable tests and then compares actual test results with predicted or expected results. Selenium is an open-source automated testing framework which supports multiple programming languages to validate web applications across different browsers and platforms. Test automation gives engineers feedback about the application. It helps find mistakes and imperfections which one could be lacking in manual testing. It also helps to perform boundless testing of the same code examples on an ongoing basis.

Introduction:

Web testing is an important classification of programming testing, it is mainly focused on web-based applications. Testing is a key component of programming advancement, as it gives you a chance to measure programming quality. Without testing, there is no real way to verify that the product is functioning properly without any bugs. Testing is often carried out physically, by performing operations in an orderly manner, and by showing whether a specific step was competent. In the recent events of the outbreak of coronavirus, people need to take proper precaution by covering their face or by using the mask once they get out of public serenity. So, I decided to perform automation testing on the Amazon website and search for an N99 mask. Compare the obtained results and actual result. Generate reports with testing and find the coverage of the testing.

Problem Statement:

To search and check out a mask on the Amazon website.

Automation Steps:

1. Open browser with Amazon website "<https://www.amazon.co.uk/>".
2. Search for an N99 Mask.
3. Add to the cart.
4. Delete from the cart.
5. Generate emailable reports using TestNG.

Technologies Used:

1. Java for Testing
2. Selenium
3. Maven
4. TestNG
5. JMeter

A brief explanation of the setup and usage of the tool:

I. What is testing:

Testing is evaluating software by observing its execution. It is the most expensive and time-consuming part of application development. Not testing is even more expensive as it risks life as well as capital. Testing results in improvement of quality, reduction of losses and preservation of customer satisfaction. Test automation is the use of software to control the execution of tests, compare the actual outcome to predicted outcome. It reduces cost and human error.

II. Why use selenium for automation testing:

To perform automation testing of the web product, one has to choose the most suitable tools for monitoring the program. It was difficult to choose a cost-effective automation tool to fit the requirements. Selenium web-driver is used for functional testing. There is a whole range of various tools like JMeter, QTP, Test Complete but Selenium has found more advantages over other tools for testing the website.

- Selenium is an incredibly popular open-source tool.
- Testing in selenium does not require prior programming skills.
- It can be launched on various OS and browsers.
- It supports various programming languages.
- It is possible to deploy the tests in a diverse environment.
- Selenium comes with its web framework like selenium WebDriver.

III. Information of tools used:

1. Java for Testing:

Java is a popular programming language developed by James Gosling. As per StackOverflow, it is the third most popular back-end technology. Writing test cases using java has many advantages such as programs written in Java are faster than other popular languages such as Python. Java supports selenium hence integrating test is easier.

2. Selenium:

To create various test suits for web application with java, a better understanding of selenium WebDriver is required. To kickstart selenium with google chrome,

ChromeDriver needs to be downloaded from the web. Chrome driver helps to connect with the web.

3. Maven:

Apache Maven provides support for managing the full lifecycle of a test project. Maven is used to adding dependencies, define project structure in IntelliJ, build and test applications. To satisfy and configure dependencies needed for building testing and running code.

4. JUnit:

JUnit is an open-source unit testing tool that helps to test units of code. It is mainly used for unit testing Java projects. However, it can be used with Selenium WebDriver to automate the testing of web applications. Incorporating JUnit annotations and methods into first Selenium test suits.

5. TestNG:

TestNG can execute multiple test cases on multiple browsers. The testing framework can be easily integrated with Maven. Is a testing framework inspired by JUnit. It has extended new functionalities, which made it more powerful and easier than the other testing frameworks. It can be implemented to generate an emailable TestNG report. Advantage of using TestNG with Selenium is of running multiple test cases from multiple classes using the XML configuration file.

6. Jmeter:

Apache JMeter is an open-source tool used as a load testing for analysing and measuring the performance of a variety of services, with a focus on web applications. It is a pure Java software, which was developed by Stefano Mazzocchi. Test results of JMeter can be displayed in a different format such as chart, table, tree and log file.

IV. Installation and configuration:

1. The IntelliJ IDEA community edition is a java Integrated Development Environment developed by JetBrains. To download IntelliJ visit the JetBrains site. The advantage of using IntelliJ is that.
 - It is easy to use can work on different operating systems supporting various languages.
 - Different database servers can be accessed easily.
 - Getter and setter methods for the object can be accessed quickly.
 - Various short cuts are available to write code and find mistakes in the code.
 - Comes with an inbuilt run with coverage to find the coverage of the code.
2. Selenium jar files 3.9.1 and apache maven 3.6.3 can be downloaded from Google Chrome plugin from Seleniumhq.org official site and maven.apache.org. Extract the jar files to the desired location. Launch IntelliJ IDE and create a new maven project.

3. Click build project and now add the download selenium jar files in IntelliJ as external libraries.
4. Open the generated POM file to add maven plugins as well as required dependencies of various tools like SeleniumHQ 3.1.0, Junit, TestNG 6.9.8. Specify a groupId, artifactId and version. Hit install to load all the plugins and IntelliJ IDEA generates appropriate information in the target folder with an executable JAR.
5. Once you have successfully added the .jar files into IntelliJ. Create a class in your java project. Now IntelliJ is ready to run test cases.
6. Download Apache JMeter 5.2.1 from the official website. It has its executable jar file in the bin folder. JMeter is used for performance testing mainly for the web applications.
 - To Create a test for amazon website in JMeter first create a new test plan.
 - Then create a thread group with a certain number of thread, ramp-up period and loop count.
 - Add three samplers HTTP request for searching a mask, adding it to cart and deleting from the cart.
 - The three samplers contain the path of the website as shown in Image 1.1 below.
 - Add the required Listeners to samplers to view results.

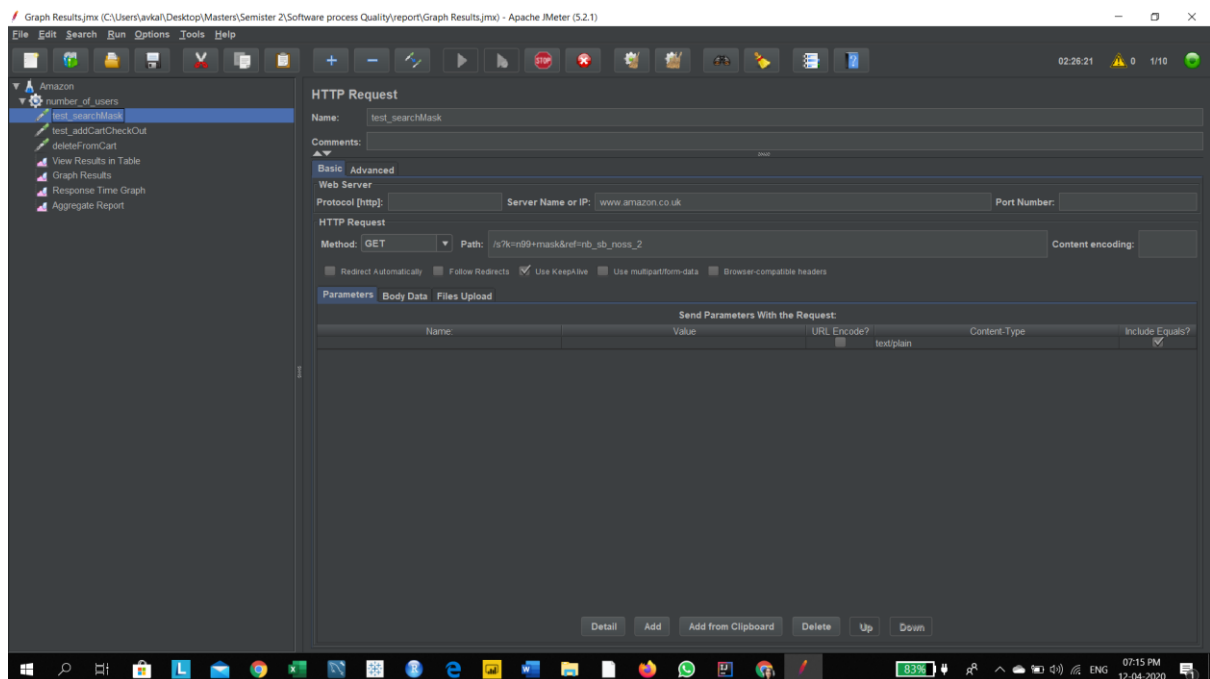


Image 1.1

V. Test Design:

It is a process of designing input values that will effectively test the application. Selenium web driver connects to the web and maximizes chrome window. Sends search request for amazon website. Each method searches the element from the website using find element by XPath and id, which we get from inspecting the website. Test automation consists of three test suits first for searching an N99 Mask, then adds the desired mask in the cart and deletes the mask from cart assigning each task certain priority. Each method is depended on one another as shown in the test case one below, i.e dependsOnMethods. Each test is enabled as true so that it performs the test case. After test execution, the driver quits the chrome browser and provide results with TestNG report. A sample test case is shown below in Image 1.2.

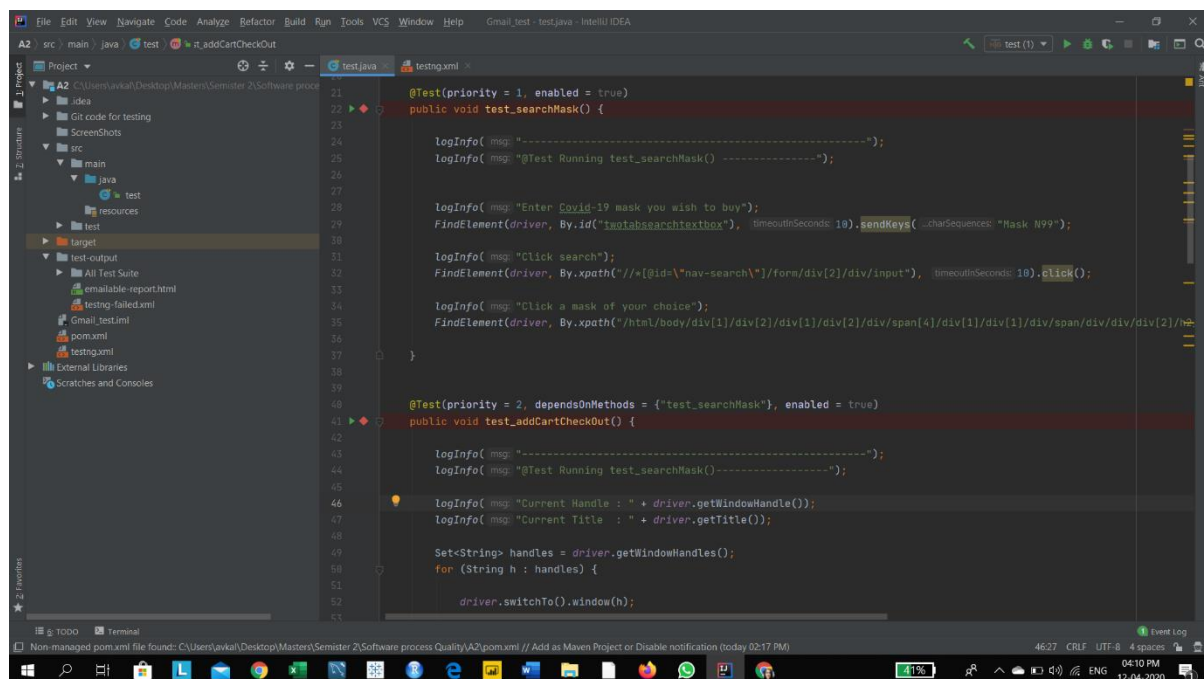


Image 1.2

An analysis of the coverage methods available using the tool:

In Code Coverage there are mainly three types of testing methods for developers, they are mainly Black-box testing, White-box testing and Model-based testing, we will mainly look white-box testing which derives tests from the source code internals of the software, specifically including branches, conditions, and statements. Lets us look at various coverage methods covered using selenium and testing. The Functional Coverage is the functionality of the design covered by the test bench which is performed by the testers.

I. Statement Coverage:

Statement coverage is a white box test design technique which involves the execution of all the executable statements in the source code at least once.

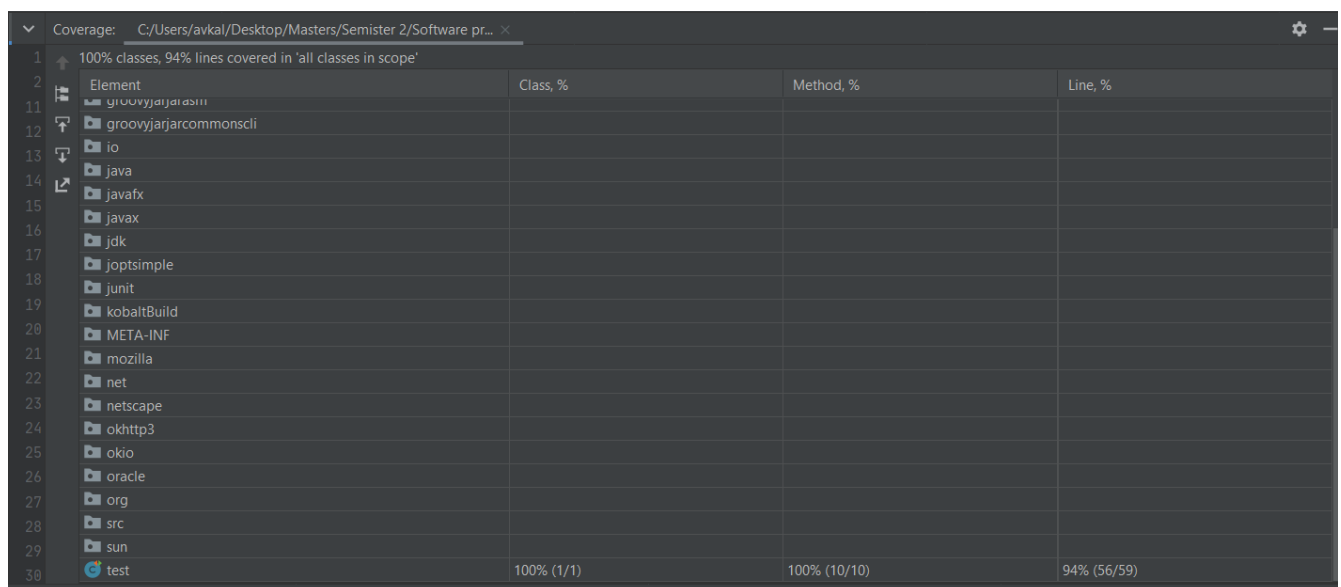
Statement coverage is line coverage which we get in IntelliJ when we run test cases with coverage. It is used to calculate and measure the number of statements in the source code which can be executed given the requirements.

Statement coverage is used to derive a scenario-based upon the structure of the code under test. Statement coverage is shown in image 1.3 below.

The formula to calculate Statement Coverage:

$$\text{Statement Coverage} = \frac{(\text{Number of the executed statement})}{(\text{Total number of statement})} * 100$$

Statement Coverage = 94%



Coverage: C:/Users/avkal/Desktop/Masters/Semister 2/Software pr... X

100% classes, 94% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
groovyjarjarasm			
groovyjarjarcommonscli			
io			
java			
javafx			
javax			
jdk			
joptsimple			
junit			
kobaltBuild			
META-INF			
mozilla			
net			
netscape			
okhttp3			
okio			
oracle			
org			
src			
sun			
test	100% (1/1)	100% (10/10)	94% (56/59)

Image 1.3

II. Condition Coverage:

Conditional coverage or expression coverage is similar to logic coverage. In this coverage, expressions with logical operands are only considered. It reveals how the variables in the conditional statement are evaluated. The conditional coverage offers better sensitivity to the control flow than decision coverage. Condition coverage does not give a guarantee of full decision coverage. In our case, we have two try-catch blocks which will be evaluated further. The formula to calculate Condition Coverage:

$$\text{Statement Coverage} = \frac{(\text{Number of executed operand})}{(\text{Total number of operand})} * 100$$

Condition Coverage = 50%

III. Branch Coverage:

Each result from the code module is checked in the branch coverage. For instance, for a binary result, to check both true and false outcomes. Every possible branch from every condition is executed at least once. In our case, we have 2 try-catch blocks to catch the exception. Either of one is executed once a time. The formula to calculate Branch Coverage:

$$\text{Branch Coverage} = \frac{(\text{Number of executed branches})}{(\text{Total number of branches})} * 100$$

Branch Coverage = 50%

IV. FSM Coverage:

Finite State Machine covers the behaviour of the test design to look for how many time-specific states are visited, transited. It also checks how many sequences are included in a finite state machine. Here is the list of all the FSM Coverage using selenium. FSM coverage is shown in Image 1.4 below.

S1: The state in which the chrome driver is connected to the Amazon website using selenium WebDriver.

L1: In this state, we can log-in to the website. But while performing automation testing, amazon website asks for authentication. So we skipped this step.

S2: In this state, an N99 Mask is searched on the website.

S3: In this state, we select a desired N99 Mask.

S4: In this state, we add the selected mask in the cart.

S5: In this state, we delete the mask from the cart.

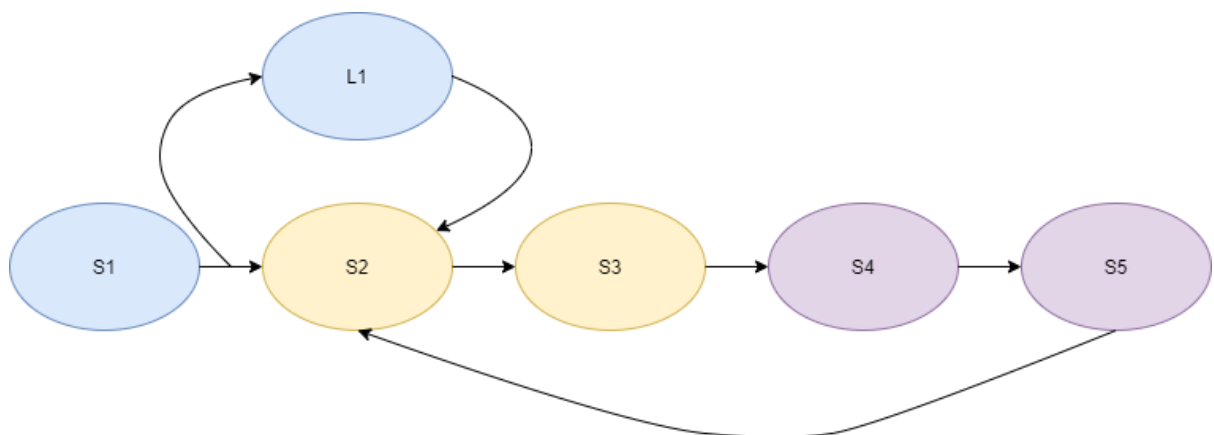


Image 1.4

V. Functional Coverage

We used JMeter for performance testing of amazon website sending HTTP packet request for searching N99 Mask, adding it to cart and deleting from the cart. We created the thread group consisting of 10 Number of threads, Ramp up period of 1000 and Loop count of 811. It means that 10 users will loop 811 times in ramp-up interval of 1000 seconds to perform the following tasks on the website. Generated reports as shown in Image 1.5 below which shows the overall error of 1.60%.

	Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
	test_searchMa...	271	42	38	58	65	136	18	260	0.37%	2.6/sec	1.31	0.48
	test_addCartC...	270	27	25	31	35	60	13	315	0.00%	2.6/sec	1.40	0.50
	deleteFromCart	270	319	293	469	539	765	21	929	4.44%	2.6/sec	90.46	0.75
	TOTAL	811	129	39	352	422	628	13	929	1.60%	7.9/sec	93.11	1.74

Image 1.5

A sample run of the tool using its main functionality to set up and run three tests.

1. Open browser with Amazon website - **PASS**.
2. Search for an N99 mask - **PASS**.
3. Search for n99 mask and add selected one in the cart - **PASS**.
4. Edit the basket and delete the mask from the cart - **PASS**.
5. Quit the browser, shown in the Image 1.6- **PASS**.

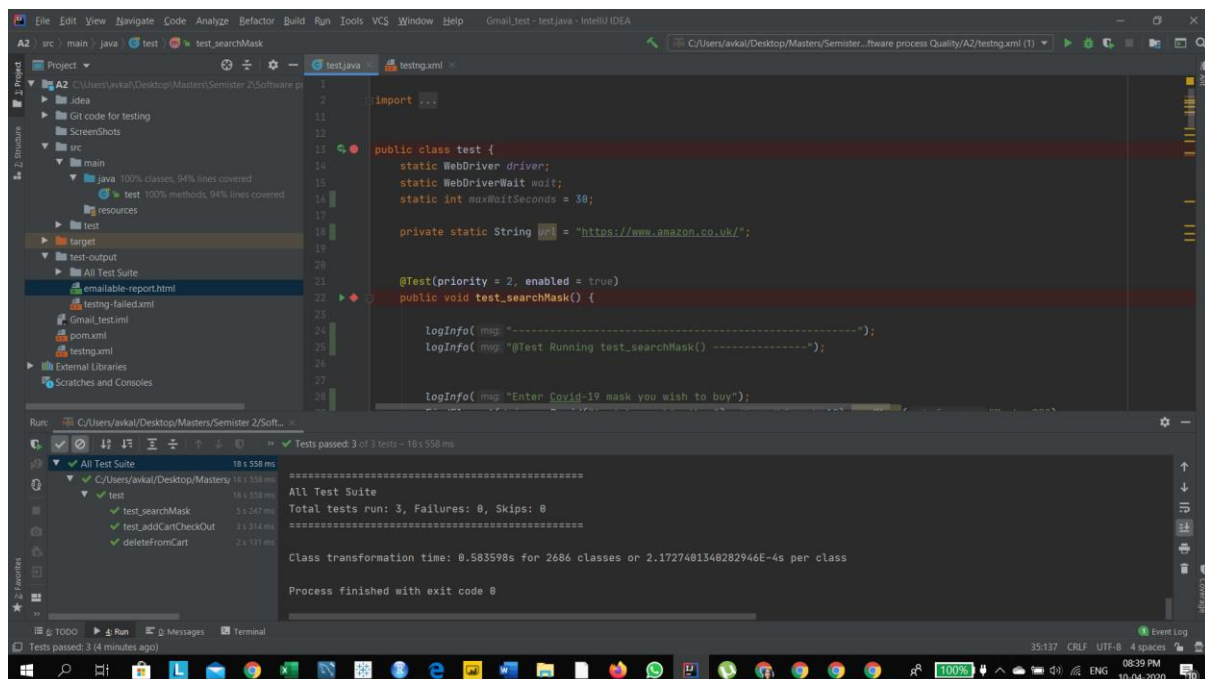
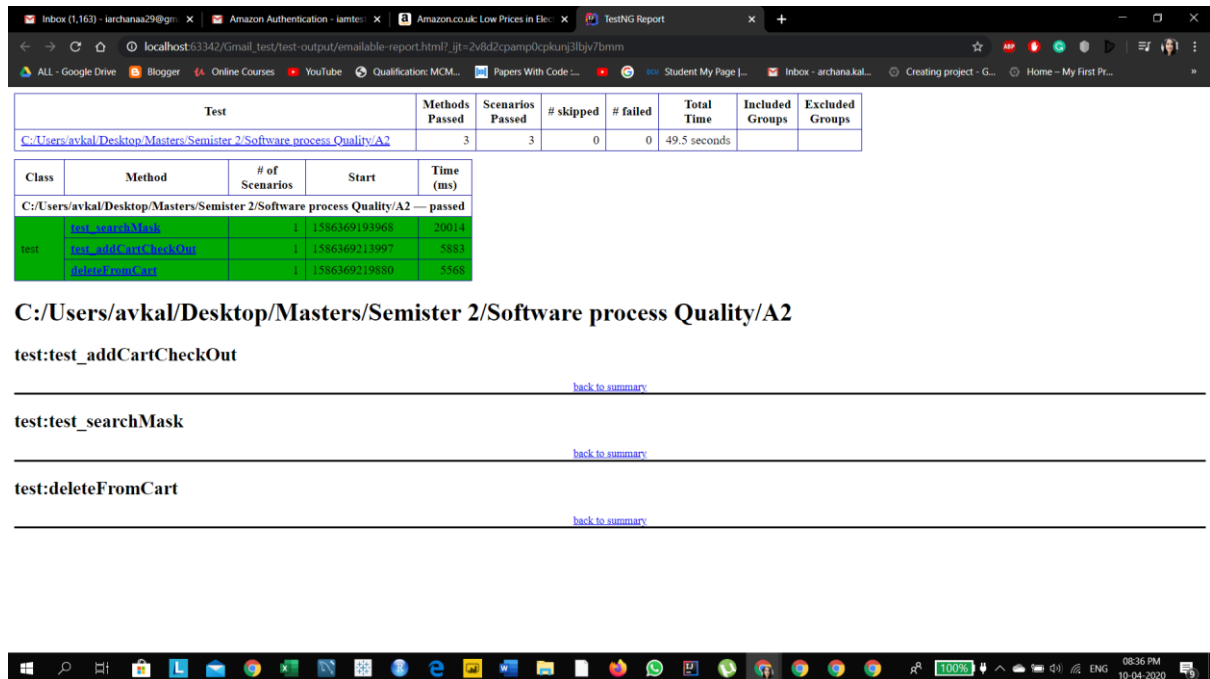


Image 1.6

Reports generated with Selenium and JMeter after the sample run:

TestNG: After Running the test suits using selenium in IntelliJ, TestNG generates emailable reports of the test cases. Green colour indicates the passes tests and red indicated failed tests. Total time run the test suit was 49.5 seconds. It can be performed even faster but we had to put the timeout of 10 seconds after each test. This shows that the speed can be improved. Image 1.7 shows the testing report.



Test	Methods Passed	Scenarios Passed	# skipped	# failed	Total Time	Included Groups	Excluded Groups
C:/Users/avkal/Desktop/Masters/Semister 2/Software process Quality/A2	3	3	0	0	49.5 seconds		

Class	Method	# of Scenarios	Start	Time (ms)
C:/Users/avkal/Desktop/Masters/Semister 2/Software process Quality/A2	passed			
test	test_addCartMask	1	1586369193968	20014
test	test_addCartCheckOut	1	1586369213997	5883
test	deleteFromCart	1	1586369219880	5568

C:/Users/avkal/Desktop/Masters/Semister 2/Software process Quality/A2

test:test_addCartCheckOut

[back to summary](#)

test:test_searchMask

[back to summary](#)

test:deleteFromCart

[back to summary](#)

Image 1.7

JMeter: After Running the performance test in JMeter, it generated Response Time Graph and performance testing graph. Blue colour indicates test_searchMask, Red Colour is the test of adding mask in the cart, and Green colour indicates the deleteFromCart test. The graph shows that the delete from cart took the longest time. Image 1.8 shows the JMeter report.

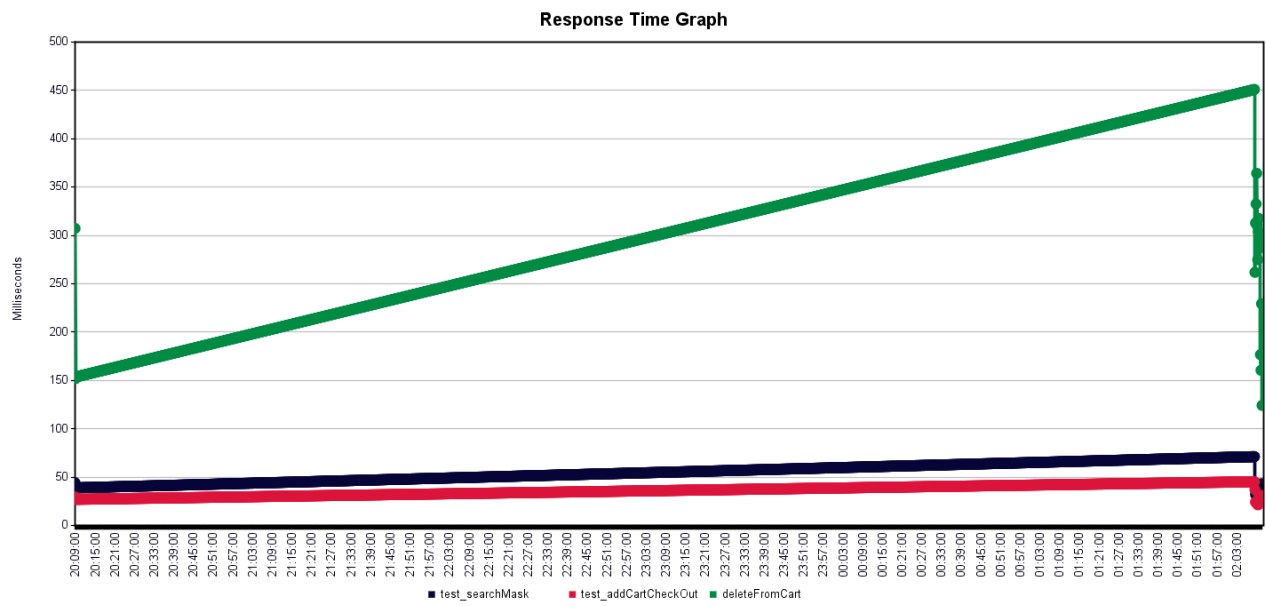


Image 1.8

Image 1.9 shows the performance of the website with 2028 users. It shows Deviation, Throughput, average and median.

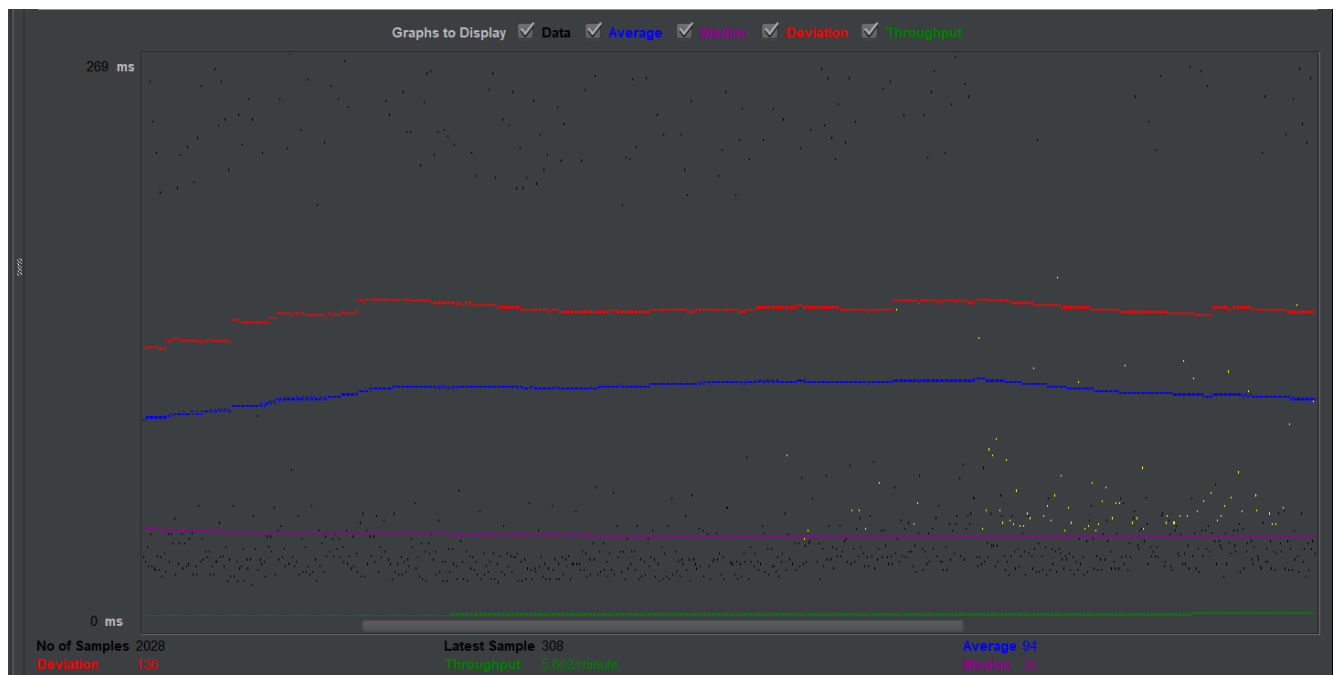


Image 1.9

An overall assessment of the usability, coverage and robustness of the tool.

With the growing acceptance of testing and test automation in projects, selection of the right testing tool would have adverse effect on entire project. JMeter and selenium both simplifies the testing process. Both are specifically designed for web-based applications.

1. JMeter

- As JMeter focuses mainly on functioning, specific target request per second requires less resources for JMeter.
- JMeter does a better job on load testing with n number of users.
- JMeter focuses on functional coverage so Automation testing is difficult.
- It collects responses from the server and visualizes the details in a chart or graph which is easy to understand.

2. Selenium

- Though selenium can be used for performance testing it cannot be relied on to load check the performance with n number of users.
- Selenium is best suited for automation testing or using logic to execute the tests in a certain order and aggregating multiple test scripts.
- It cannot be used to generate a controlled number of requests of users for an equivalent interval of time.
- Selenium does a better job if the test is focused on how the page is presented to a user.
- Selenium focuses on code coverage.
- Selenium is not just a tool but a suite of software which is more robust than any application. JMeter can collaborate with selenium if that serves the purpose.

REFERENCES

1. R. Verbruggen, "CA650-Software Process Quality," [Online]. Available: <https://loop.dcu.ie/course/view.php?id=35087>. [Accessed 01 03 2020].
2. "Java," [Online]. Available: <https://www.java.com/en/download/>. [Accessed 12 01 2020].
3. "Jetbrains," [Online]. Available: <https://www.jetbrains.com/>. [Accessed 01 03 2020].
4. "Selenium," [Online]. Available: <https://www.selenium.dev/downloads/>. [Accessed 12 01 2020].
5. "Maven," [Online]. Available: <https://maven.apache.org/download.cgi?Preferred=ftp://ftp.osuosl.org/pub/apache/>. [Accessed 12 03 2020].
6. "JMeter," [Online]. Available: <https://jmeter.apache.org/>. [Accessed 12 03 2020].
7. "ChromeDriver," [Online]. Available: <https://chromedriver.chromium.org/downloads>. [Accessed 12 01 2020].
8. "Guru99," [Online]. Available: <https://www.guru99.com/install-testng-in-eclipse.html>. [Accessed 12 03 2020].
9. "Guru99," [Online]. Available: <https://www.guru99.com/code-coverage.html>. [Accessed 12 04 2020].
10. Chandrasekhar, K., 2018. Testing Web Application using Selenium Testing Tool with Respect to Test'ng. International Journal for Research in Applied Science and Engineering Technology, 6(4), pp.4766-4770.
11. Permatasari, D., 2020. Pengujian Aplikasi menggunakan metode Load Testing dengan Apache JMeter pada Sistem Informasi Pertanian. Jurnal Sistem dan Teknologi Informasi (JUSTIN), 8(1), p.135.
12. Gojare, S., Joshi, R. and Gaigaware, D., 2015. Analysis and Design of Selenium WebDriver Automation Testing Framework. Procedia Computer Science, 50, pp.341-346.
13. Day, P., 2014. n-Tiered Test Automation Architecture for Agile Software Systems. Procedia Computer Science, 28, pp.332-339.