International Journal of Science and Engineering Applications
Volume 12-Issue 05, 41– 49, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1205.1013

A Case Study of Test Automation in Agile Software Development

Dr. Dilshan De Silva
Department of Computer Science and Software Engineering
Institute of Information Technology
Malabe, Sri Lanka

M. P. Gunathilake
Department of Computer Science and Software Engineering
Institute of Information Technology
Malabe, Sri Lanka

Sooriyabandara H.M.T.S
Department of Computer Science and Software Engineering
Institute of Information Technology
Malabe, Sri Lanka

Chandrarathna M.G.D.P.M.B
Department of Computer Science and Software Engineering
Institute of Information Technology
Malabe, Sri Lanka

Bandara S.M.D.S
Department of Computer Science and Software Engineering
Institute of Information Technology
Malabe, Sri Lanka

Thilakarathna S.A.Y.R
Department of Computer Science and Software Engineering
Institute of Information Technology
Malabe, Sri Lanka

Abstract: Agile software development has revolutionized the industry by allowing developers to produce high-quality software in shorter timeframes. However, testing continues to be an important part of software development to ensure that the product fulfills its necessary quality requirements. Manual testing can be time-consuming and error-prone, resulting in software delivery delays. Test automation, on the other hand, may significantly reduce testing time while enhancing the accuracy and consistency of results. The paper examines the relationship between agile software development and testing, with a specific focus on web-based testing. It covers the issues of web testing in an agile environment and examines how test automation can assist to overcome these challenges. Furthermore, it compares manual testing with test automation and analyses the benefits and drawbacks of each approach, provides an overview of popular test automation tools, such as Selenium and JMeter, and explains why they are the best options for automated testing, and the paper highlights the benefits of web testing, including as increased software quality, reduced costs, and improved user satisfaction.

Keywords: Agile software development; Test automation; Web-based testing; Selenium and JMeter; Case study

1. INTRODUCTION

Testing is an important part of software development because it ensures that the product fulfills the appropriate quality requirements. Agile software development and testing are inexorably linked, and testing is a necessary step in the agile development process.

As shown by [1], manual testing is utilized in traditional software development to ensure that software satisfies the necessary quality requirements. Manual testing can be time-consuming and error-prone, resulting in software delivery delays. Test automation is a method of testing that involves the use of tools and scripts to automate the testing process. Test automation reduces testing time while increasing the accuracy and consistency of test results.

Web testing, which consists of testing web-based applications, is a vital part of software testing. Because software is built in shorter sprints, agile software development created new challenges to web testing. Web testing must be performed concurrently with software development in an agile environment, and testing must be integrated into the development process.

The use of tools and scripts to automate the testing of web-based applications is known as web test automation. Selenium [2] and JMeter [3] are two technologies that can be used to automate web tests. Selenium is a popular open-source technology for automating web browsers, and JMeter is a tool for web application load and performance testing.

In Java development, creating a Maven [4] project in Eclipse is usual practice. In Eclipse, a Maven project provides a framework for automated testing, including dependency management and build automation. To simplify dependency management and streamline the testing process, develop a Maven project in Eclipse for automated testing.

According to [5], web testing has various advantages, including increased software quality, lower costs, and more user satisfaction. Testing detects and removes errors early in

the development process, lowering the cost of later defect correction. Testing also guarantees that software fulfills quality requirements and increases satisfaction for users.

The various locators are available in Selenium WebDriver for identifying web page elements. Class name, CSS selector, ID, name, link text, partial link text, element name, and XPath are among these locators. Each of these locators has its advantages and disadvantages, and the selection of the most suitable locator is dependent on the specific characteristics of the web page and the element being identified. The difficulties that come with selecting the right locator. Dynamic web pages, multiple elements with similar properties, hidden elements, frames, browser compatibility, timeouts, and maintenance are among the challenges. To overcome these challenges, it is essential to have a solid comprehension of web page structure.

The purpose of this paper is to study the relationship between agile software development and testing, to discuss the role of test automation in ensuring software quality; to compare traditional testing with test automation and analyze the benefits and drawbacks of both approaches; and to investigate how web testing occurs in an agile environment, including the successes and failures. The article will review popular test automation tools and explain why best Selenium and JMeter are for automated testing, in addition to the benefits and features of web testing. As part of an investigation into how web testing occurs in an agile environment and the benefits of test automation, it demonstrates how to test a login page using test automation tools such as Selenium and JMeter.

2. RELATED WORK

Automated software testing has been a popular area of study for many years, and there is an important corpus of literature on the subject. This section describes a few significant works related to the current work on web testing automation in agile software development.

The paper "Benefits and limitations of automated Software Testing: Systematic literature review and practitioner survey" by T. Tretmans et al. (2013) provides an exhaustive review of the advantages and disadvantages of automated software testing. To determine the current state of automated testing, the authors analyzed a large number of research articles and surveyed practitioners. According to [5], it provides an overview of the benefits and drawbacks of automated testing.

"Agile Testing: A Practical Guide for Testers and Agile Teams" by Lisa Crispin and Janet Gregory explores the principles and practices of agile testing and as demonstrated in [6], explains how testing can be integrated into agile development processes. The book provides valuable information on how agile testing functions and the various practices that can be implemented to ensure successful testing in an agile environment. It describes how agile teams and testers can collaborate to ensure that quality is built into the product at every stage of agile development.

According to [7], the book "Maven: The Definitive Guide" explains the importance and benefit of using the Maven build automation tool in software development projects. Maven is a popular tool for managing project dependencies, as well as creating, testing, and packaging Java-based applications. This book is a beneficial resource for gaining an understanding of the features and capabilities of Maven, as well as how it can be utilized effectively in software development projects.

Testim.io's article "Test Automation vs. Manual Testing" compares the benefits and drawbacks of test automation and manual testing. As shown by [8], it discusses the advantages and cons of both test automation and manual testing, which will be essential factors when determining how to approach testing. It also provides information regarding the limitations of test automation, which can be used to determine which categories of testing to automate and which types to perform manually.

As demonstrated in [9], "A Study of Automated Software Testing Automation Tools and Frameworks" by P. Bansal et al. (2019), the authors examine a variety of automated software testing tools and frameworks. They have compared the capabilities and features of various tools and provided suggestions for selecting the most suitable tool for a given task.

The article "Analysis and Design of Selenium WebDriver Automation Testing Framework" by N. Singh and V. Kumar (2015) provides a comprehensive analysis of Selenium WebDriver, a widely used tool for automating web testing. The authors discuss the features of Selenium WebDriver and offer guidelines for designing an effective framework for testing.[10] and also highlights the importance of test automation in ensuring software quality and the advantages of using Selenium WebDriver in agile software development.

In "A Study on Functioning of Selenium Automation Testing Structure" by M. S. Mohammed and R. G. Rasool (2017), the authors of [11] discuss Selenium WebDriver's operation in depth. In addition, they have compared Selenium WebDriver to various testing tools and frameworks. This article offers a comprehensive understanding of Selenium WebDriver.

As demonstrated by previous research (A case study of test automation in agile software development" by E. Knauss et al, 2014) provides a case study of test automation in agile software development. The authors examine the difficulties and advantages of test automation in an agile development environment.

These works provide insightful details about the advantages and disadvantages of automated software testing, how to test effectively in an agile environment, what testing tools and frameworks are available, and how to use test automation effectively in an agile development environment.

International Journal of Science and Engineering Applications
Volume 12-Issue 05, 41– 49, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1205.1013

3. METHODOLOGY

3.1 Selenium Architecture

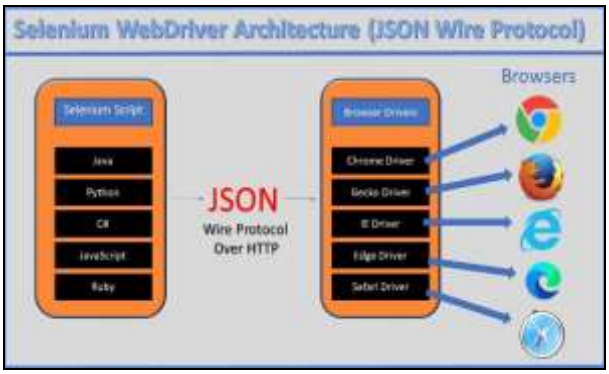

Figure 1. Selenium Architecture

As shown in Figure 1, Selenium supports multiple programming languages like Java, Python, Ruby, C#, and JavaScript. This client library provided by Selenium helps to write the code in any of these. Then any of the languages convert to JSON format and transferred to the Selenium web driver [15] through the JSON wire protocol over HTTP. That browser driver in turn launches the script actions to the real browser.

3.2 Setup Maven and Selenium WebDriver in Eclipse IDE

3.2.1 Install Java and configure the Java path

3.2.2 Install Eclipse IDE

3.2.3 Install Maven in Eclipse

3.2.4 *Create a new Maven project*: When creating a new Maven project in Eclipse IDE [13], select "Maven Project" from the list of project types. Then, select an archetype, which is a project template that defines the basic structure and contents of the project. Once selected an archetype, enter a Group Id and Artifact Id for the project. The Group Id is typically a unique identifier for the organization or group that is creating the project, while the Artifact Id is the name of the project.

3.2.5 Define the dependencies in the pom.xml

```
<dependencies>

    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.8.3</version>
    </dependency>

    <dependency>
        <groupId>io.github.bonigarcia</groupId>
        <artifactId>webdrivermanager</artifactId>
        <version>5.3.2</version>
    </dependency>

</dependencies>
```

Figure 2. Code snippet of the pom.xml file

In Figure 2, the selenium-java dependency contains the Java bindings for Selenium WebDriver, while the webdrivermanager dependency simplifies the management of WebDriver binaries for different browsers.

3.2.6 Create a new Java class and run the Selenium WebDriver test

```
1 package com.rcv.SeleniumTest;
2 import org.openqa.selenium.chrome.ChromeDriver;
6
7
8 public class DemoAutomation {
9
10     public static void main(String[] args) {
11
12         WebDriverManager.chromedriver().setup();
13         ChromeDriver driver = new ChromeDriver();
14
15         // write test cases here
16
17     }
18
19 }
20
```

Figure 3. Creating a Java class for the Selenium WebDriver test

According to Figure 3, run the Selenium WebDriver test After creating a new Java class. Import the relevant Selenium WebDriver API classes into the Java class. In the class's main method, write the Selenium WebDriver code.

International Journal of Science and Engineering Applications
Volume 12-Issue 05, 41– 49, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1205.1013

3.3 Testing a Login Page

To automate testing on the login page, the test case plan was prepared as shown in Table 1.

Table 1. Test cases for automation test plan

Test case ID	Positive/ Negative criteria	Test case	Test step	Test data	Expected result
TC-001	Positive	Verify whether login is successful by entering the valid credentials	1. Go to the URL 2. Enter the valid credentials in the username and password field 3. Click the 'LOGIN' button	Username: "standard_user" Password: "secret_sauce"	The user should be able to log in to the website
TC-002	Negative	Verify whether the form can be submitted without filling in any field	1. Go to the URL 2. Click the 'LOGIN' button	None	Display an error message
TC-003	Negative	Verify whether the form can be submitted when entering the incorrect password	1. Go to the URL 2. Enter values for the mandatory fields with an incorrect password 3. Click the 'LOGIN' button	Username: "standard_user" Password: "sec%386@e"	Display an error message

3.3.1 TC-002: Test automation script to check whether the user can log in to the website by providing valid credentials into the login form

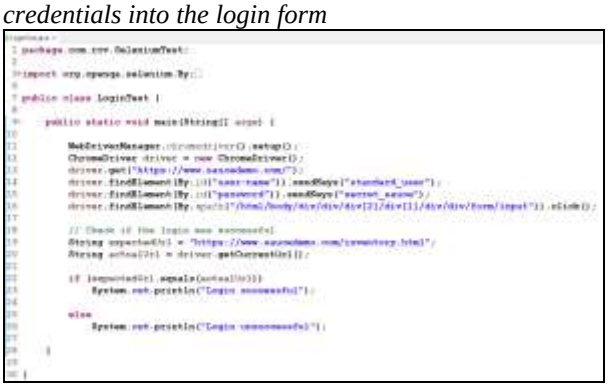

Figure 4. Test automation script to test login form with valid credentials

According to Figure 4, this script logs into the Sauce Demo web application using valid credentials and checks if the login was successful. Using the WebDriver's findElement and sendKeys methods, an XPath expression [7] to find the login button, and an if statement to compare the expected and actual URLs. This is an example of how Selenium can be used for automated functional testing of web applications.

3.3.2 TC-002:Test automation script for testing whether the login form's null field validation is functioning properly

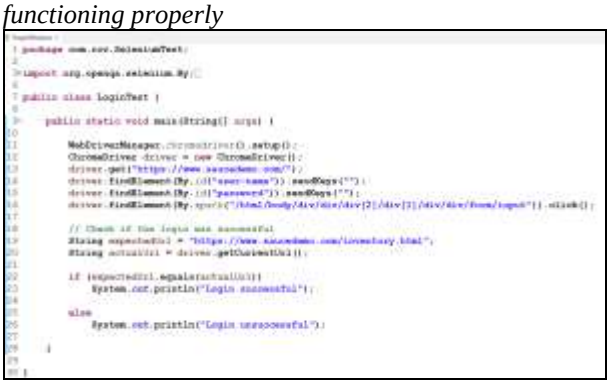

Figure 5. Test automation script to test login form with empty data

Figure 5, test automation script that is designed to test whether the login form's null field validation is functioning properly. By sending an empty string to the username and password fields, checking if the validation mechanism of the login form is working as expected and if the appropriate error message is displayed when the user tries to submit an empty form.

International Journal of Science and Engineering Applications
Volume 12-Issue 05, 41– 49, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1205.1013

3.3.3 TC-003:Test automation script to check whether the user can log in to the website by providing invalid credentials into the login form

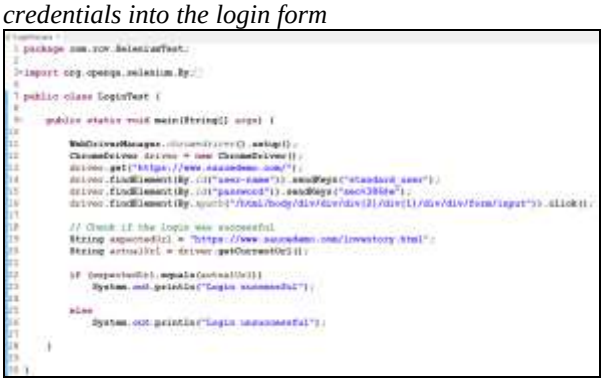

Figure 6. Test automation script to test login form with an invalid password

According to Figure 6, the code segment tests whether the login form on a website is rejecting invalid credentials as expected. By providing invalid credentials into the password field using the sendKeys method, the script is likely checking if the login form is working as expected and rejecting the login attempt, and if the appropriate error message is displayed when the user tries to log in with incorrect credentials.

3.4 Non-functional testing using JMeter

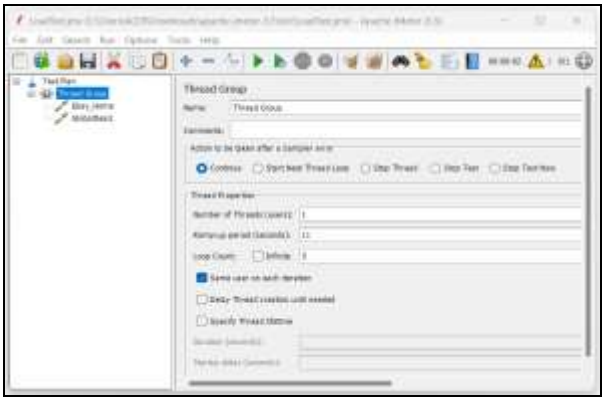

Figure 7. Create a thread group for a new test plan

In JMeter [14], create a new test plan and add the necessary elements, such as Thread Group, HTTP Request, and Listener. As shown in Figure 7, in the Thread Group element, specify the number of threads (virtual users), ramp-up time, and loop count.

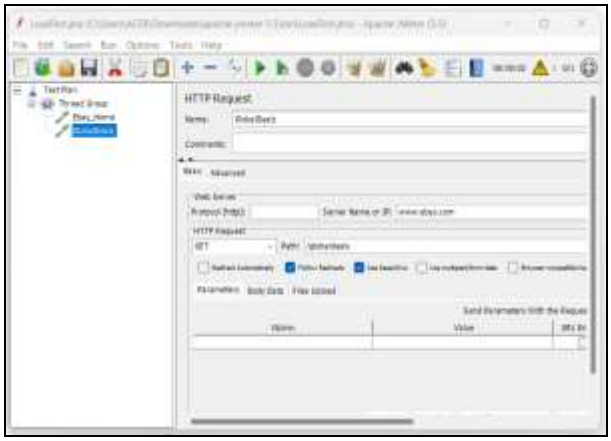

Figure 8. Create an HTTP Request

According to Figure 8, in the HTTP Request element, enter the URL of the website or application under test, and select the appropriate method, such as GET or POST.

Figure 9. Execute a JMeter test using the command-line interface

As shown in Figure 9, here is what each of the options in the command does:

- -n: This option tells JMeter to run the test in non-GUI mode.
- -t: This option is used to specify the path to the JMX file that contains the test plan.
- -l: This option is used to specify the path where you want to save the results of the test execution.

4. RESULTS

The testing process was carried out using both functional and non-functional testing techniques with test automation. Functional testing was performed using Selenium, while JMeter was used for non-functional testing. The results of the testing process are presented in this section, demonstrating that the application meets the functional and non-functional requirements.

4.1.1 Result of the functional testing

Under the TC-001 test case id plan shown in Figure 4 of the test automation plan, Selenium WebDriver will conduct a login test on the website https://www.saucedemo.com/. The program launches the Chrome browser, navigates to the login page, inputs a valid username and password, and clicks the login button. The program then compares the current URL to the expected URL of the inventory page. As the actual result, the user successfully login to the website and at the end of the test terminal printed "Login successful" as shown in Figure 10.

International Journal of Science and Engineering Applications
Volume 12-Issue 05, 41– 49, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1205.1013

Figure 10. Eclipse terminal output for TC-001 test case

After the test script in Figure 5, of TC-002 execution made all mandatory fields null and clicked on the login button, then as the actual result, the form displayed an error message "Username is required" according to Figure 11, at the end of the test terminal printed "Login unsuccessful".

Figure 11. Eclipse terminal output for TC-002 test case

According to the TC-003 test script in Figure 6, inputs a valid username with an invalid password, and click the login button. As the actual result, the terminal print “Login unsuccessful” and Figure 12 is the screenshot of the terminal output.

Figure 12. Eclipse terminal output for TC-003 test case

4.1.2 Results of the non-functional Testing

timeStamp	elapsed	label	responseC	responseM	threadNar	dataType	success	failure	bytes	sentBytes	allThreads	Latency	IdleTim	Connect
1.68E+12	1429	Ebay_Hon	200	OK	Thread Gr	text	TRUE		233054	230	1	237	0	35
1.68E+12	247	Ebay_Hon	301	Moved Pe	Thread Group 1-1		TRUE		208	115	1	237	0	35
1.68E+12	1177	Ebay_Hon	200	OK	Thread Gr	text	TRUE		232846	115	1	927	0	387
1.68E+12	1065	GlobalDea	200	OK	Thread Gr	text	TRUE		344537	297	1	251	0	0
1.68E+12	251	GlobalDea	301	Moved Pe	Thread Group 1-1		TRUE		317	171	1	251	0	0
1.68E+12	813	GlobalDea	200	OK	Thread Gr	text	TRUE		344220	126	1	674	0	0

Figure 13. Performance test results in a table using JMeter

With the CSV file open, start analyzing the data to identify any performance issues. Some common metrics to look for include response time, throughput, error rate, and transaction rate. Based on the analysis of the CSV file, can identify any performance issues, such as slow response times or high error rates. This information helps to make changes to the application or system to improve performance.

According to the information provided in Figure 13, the website appears to be performing properly. There are differences between the response times of various requests. For example, the first request for Ebay_Home required 1429ms, while the second request for the same page required only 247ms. There was also some variation in the GlobalDealz requests, with the first request taking 1065ms and the second request taking 251ms.

There may be a few minor performance issues that could be addressed, but the website appears to be functioning properly overall. To gain a deeper comprehension of the website's performance, it would be useful to collect more data over a longer period.

Label	# Sample	Average	Min	Max	Std. Dev.	Error %	Throughp	Received	Sent	Avg. Bytes
Ebay_Home	5	1206	856	1510	248.26	0.00%	29.0/min	112.04	0.11	237184
GlobalDealz	5	1053	927	1254	116.49	0.00%	30.6/min	204.17	0.15	409601
TOTAL	10	1129	856	1510	208.38	0.00%	53.1/min	279.23	0.23	323392

Figure 14. Load testing results using JMeter

International Journal of Science and Engineering Applications
Volume 12-Issue 05, 41– 49, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1205.1013

Based on the provided load testing results in Figure 14, it can be determined that the website performs well under the test conditions. The error percentage for both pages is zero, indicating that no errors or failures occurred during the testing period. This website can manage 53.1 requests per minute, which is a respectable performance.

The website's received data rate is also not particularly high, but its sent data rate is quite low, showing that the website is not transferring a lot of data to the client side. Finally, the average bytes per request is roughly 323392, indicating that the website does not transfer a large amount of data per request.

5. DISCUSSION

Testing is an essential component of the software development lifecycle in Agile software development. Manual testing and automated testing are both essential to ensuring the integrity of the software product. In many instances, automated testing is preferred to manual testing. Table 2 illustrates some explanations for this.

Table 2. Comparison between manual testing Vs automated testing

Manual Testing	Automation Testing
Test cases are executed manually.	Test cases are executed with the help of tools.
Reliability is less.	Reliability is more.
Humans can make mistakes and hence accuracy is less.	The machine hardly makes mistakes
Using manual testing, it could be difficult to test the application on different Operating systems.	With the help of Automation testing, we can easily test the application on different Operating systems.
Sometimes it becomes difficult to execute all the test cases and it impacts test coverage.	In Automation testing, we can achieve the test coverage target.
In this testing, have to make reports on your own.	Here tool will generate a test case execution report. TestNG is the framework that will automatically generate a report.

Web testing presents some challenges and limitations, including the need for cross-browser compatibility testing, the complexity of testing dynamic web pages, and the difficulty of simulating authentic user behavior. In addition, web applications are susceptible to security flaws, which necessitate specialized testing strategies. In addition, web testing often includes testing for multiple layers, such as the front-end, back-end, and middleware, which can be resource- and time-intensive.

Table 3 provides a summary of the challenges and suggested solutions in writing test cases covering all website functionalities according to Table 1

Table 3. Challenges and suggestions in writing test cases

Challenges / Limitations	Solution
Time-consuming, delaying the development process	Separate test cases into smaller groups and write them in iterations. Prioritize and complete first the most important software test cases.
Difficulty in maintaining test cases as the software evolves	Review and update the test cases regularly, preferably after each sprint. Utilise testing automation tools that can update test cases automatically in response to software updates.
Lack of test coverage, overlooking some scenarios	Prioritize the test cases using a risk-based approach. Focus on the scenarios that are most important for the software and ensure that each sprint's acceptance criteria are covered by the test cases.
Insufficient feedback on software quality	insufficient feedback regarding software quality Adopt an approach to testing in which test cases are frequently executed throughout the development cycle. This can provide immediate feedback on the quality of the software and enable early detection of defects during the development cycle.

Due to browser compatibility issues, it can be difficult to run Selenium WebDriver tests on different browsers. Each browser has its peculiarities, which can result in unsuccessful tests or unexpected results. In addition, different browsers offer different levels of WebDriver support, which can impact the functionality of tests. To overcome these challenges, Maven is a build automation tool used for Java projects, including Selenium WebDriver tests. Maven provides several advantages over performing tests directly, such as simplified dependency management, enhanced project structure, and faster build processes. Overall, using Maven for Selenium WebDriver tests improves the testing process's efficiency and reliability.

If running Selenium WebDriver tests on different browsers, need to install the corresponding browser drivers for each browser as shown in Figures 15 and 16.

```
System.setProperty("webdriver.edge.driver", "C:\\browserdrivers\\msedgedriver.exe");
EdgeDriver driver = new EdgeDriver();
```

Figure 15. Run tests on the Microsoft Edge browser

```
System.setProperty("webdriver.gecko.driver", "C:\\browserdrivers\\geckodriver.exe");
FirefoxDriver driver = new FirefoxDriver();
```

Figure 16. Run tests on the Firefox browser

International Journal of Science and Engineering Applications
Volume 12-Issue 05, 41– 49, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1205.1013

Table 4. Web automation testing tools

Name of tool	Languages	Operating system	Type	Language supported	Browser supports
Selenium	Java	Cross-platform	Software testing framework for web application	Domain-specific language	All major browser
Apache JMeter	Java	Windows, Linux, Mac OS X, and other Unix-based systems	Open-source, Java-based, performance testing tool	Java	JMeter can be used to test web applications running on any web browser
Test Complete	Java	Microsoft Windows	Test Automation Tool	VBScript, Jscript, C++, Delphi Script, c#Script.	IE, Firefox, Google Chrome
FitNesse	Java	Cross-platform	Test Automation Tool	C++, Python, Ruby, Delphi, c#, etc	Platform independent
HP Load Runner	C	Microsoft Windows & Linux (load generator only)	Load Testing Tool	VB, VBScript, Java, JavaScript, c#	Any browser
TestNG	Java	Windows, Linux, MAC	Testing Framework	Java also includes more object-oriented feature	IE, Firefox, chrome
TOSCA	C#, java, VB6	Microsoft Windows	Test Automation	Delphi, .NET including WPF, java swing/SWT/ AWT Visual Basic	IE, Firefox
Silktest	4Test Scripting Language	Microsoft Windows	Test Automation	Java, 4Test, VB, c#, VB.net	IE, Firefox
HP-QTP	VB Script	Microsoft Windows	Test Automation	VBScript	IE 6,7,8,10, Firefox 3.0, and later

Among the automation tools shown in Table 4, the Selenium and Apache JMeter tools have reasons to choose;

Selenium is a well-known open-source automation tool used to evaluate the performance of web applications. It supports multiple programming languages, including Java, Python, and C#, and provides a set of tools for automating web browsers across multiple platforms as shown in Figure 1. Selenium enables testers to develop reliable automated scripts and run them across multiple browsers and operating systems, making it an ideal tool for functional testing [17].

On the other hand, JMeter is a powerful open-source tool used for non-functional web application testing. It is primarily used for web application load, stress, and performance testing. JMeter can simulate a large number of concurrent users, network bandwidth, and other parameters to assess an application's performance under various conditions. In addition, it can generate various reports and graphs that help identify performance bottlenecks and optimization opportunities.

Table 5. Locators in selenium web driver

Locator	Description
class name	Locates elements whose class name contains the search value (compound class names are not permitted)
CSS selector	Locates elements matching a CSS selector
id	Locates elements whose ID attribute matches the search value
name	Locates elements whose NAME attribute matches the search value
link text	Locates anchor elements whose visible text matches the search value
partial link text	Locates anchor elements whose visible text contains the search value. If multiple elements are matching, only the first one will be selected.
tag name	Locates elements whose tag name matches the search value
xpath	Locates elements matching an XPath expression

Table 5 shows that in Selenium WebDriver, locators are used to identify web page elements. Locators are essentially a method for the WebDriver to "locate" an element that can be

interacted with, such as a button or text field. Not every locator will work in all situations. Use a locator that uniquely identifies the element to interact with and test the code to ensure that locators [16] are accurate and reliable.

There are some challenges to be faced while choosing a locator, dynamic web pages, multiple elements, hidden elements, frames, browser compatibility, timeouts, and maintenance can make it difficult to choose the appropriate locators in Selenium WebDriver. Each locator has its limitations and difficulties, including unreliability, complexity, and inefficiency. To overcome these difficulties, it is essential to have an in-depth knowledge of the web page's structure, to implement appropriate locator strategies, and to review and update locators as necessary. A combination of locators may be required for efficient and reliable test automation.

6. ACKNOWLEDGMENT

We would like to express our gratitude to all those who have contributed to the successful completion of this research paper.

Firstly, we would like to thank Dr. Dilshan De Silva and M. P. Gunathilake from the Department of Computer Science and Software Engineering at the Institute of Information Technology in Malabe, Sri Lanka, for their guidance and support throughout the research process.

We would also like to extend our appreciation to our fellow group members for their valuable contributions, cooperation, and teamwork.

Lastly, we would like to acknowledge the support provided by our respective families and friends during this research endeavor.

Thank you all for your valuable contributions and support.

7. CONCLUSION

Software testing is essential for ensuring the quality of software, particularly in agile software development. Test automation is an efficient method that reduces testing time and enhances test results' accuracy and consistency. Web testing is a crucial component of software testing that necessitates concurrent testing throughout software development in an agile environment. Selenium and JMeter are popular web test automation tools. Developing a Maven project for automated testing in Eclipse simplifies dependency management and speeds up the testing process. Testing has many advantages, including enhanced software quality, lower costs, and greater user satisfaction. Future research could investigate the use of other test automation tools and their advantages and disadvantages in agile web testing.

8. REFERENCES

[1] Softwaretestinghelp. (2021, August 14). Manual Testing Vs Automation Testing: Which One Should You Use? [Blog post].

[2] Author/Source: Selenium. (n.d.). Selenium documentation. Selenium.

[3] Reference: Apache JMeter. (n.d.). Test Plan.

[4] Apache Maven. (n.d.). Creating a Project.

[5] T. Tretmans et al. (2013). "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey." Information and Software Technology, 55(1), 125-141.

[6] Lisa Crispin and Janet Gregory. (2009). "Agile Testing: A Practical Guide for Testers and Agile Teams." Addison-Wesley Professional.

[7] Sonatype, Inc. (2015). "Maven: The Definitive Guide." O'Reilly Media.

[8] Testim.io. (2021). "Test Automation vs. Manual Testing." Retrieved.

[9] P. Bansal et al. (2019). "A Study of Automated Software Testing Automation Tools and Frameworks." International Journal of Advanced Research in Computer Science, 10(4), 218-223.

[10] J. Balaji and S. Arumugam, "Analysis and Design of Selenium WebDriver Automation Testing Framework," in International Journal of Computer Applications, vol. 114, no. 1, pp. 25-31, March 2015, doi: 10.5120/20014-2072.

[11] M. S. Mohammed and R. G. Rasool. (2017). "A Study on Functioning of Selenium Automation Testing Structure." International Journal of Advanced Research in Computer Science and Software Engineering, 7(5), 537-542.

[12] E. Knauss et al. (2014). "A case study of test automation in agile software development." Empirical Software Engineering, 19(6), 1834-1864.

[13] Oracle. (n.d.). Eclipse and Java for Total Beginners.

[14] Apache JMeter. (n.d.). Getting Started - Running Apache JMeter.

[15] Selenium. (n.d.). WebDriver. Retrieved April 20, 2023.

[16] Selenium. (n.d.). Locating elements. Retrieved April 20, 2023.

[17] Singh, A., & Singh, B. (2014). Comparison of Automated Testing Tools: Selenium, Quick Test Professional, and Test Complete. International Journal of Computer Science and Information Technologies, 5(1), 119-123.