VIVA-TECH IJRI
ANNUAL RESEARCH JOURNAL

VIVA-TECH INTERNATIONAL JOURNAL FOR RESEARCH AND INNOVATION

ANNUAL RESEARCH JOURNAL

ISSN(ONLINE): 2581-7280

AUTOMATE THE WORLD AUTOMATION WITH : SELENIUM

Jyoti Gupta[1], Arif Shaikh[2]

[1](Department Master of Computer Applications, VIVA Institute of Technology, India)

[2](Departmen Master of Computer Applications, VIVA Institute of Technology, India)

Abstract: Automation testing plays a crucial role in ensuring the quality and reliability of modern web applications. Manual testing is time-consuming, error-prone, and inefficient for large-scale systems. This paper presents an experimental study on automated functional testing of web applications using Selenium WebDriver.

A finance-based web application was selected, and critical modules such as login validation, navigation, and form submission were automated using Selenium WebDriver with Java and TestNG framework. The same test cases were also executed manually to compare performance.

Experimental results demonstrate that automation testing significantly reduces execution time, improves accuracy, and enables efficient regression testing. Manual execution of 50 test cases required approximately 4 hours, whereas automation completed the same within 30 minutes with consistent results.

The study highlights the effectiveness of Selenium in real-world software development environments, particularly within Agile and DevOps practices. The findings confirm that automation testing enhances productivity, reduces human error, and improves overall software quality.

Keywords: Selenium WebDriver, Automation Testing, Web Application Testing, Manual Testing, Software Quality

1. INTRODUCTION

1.1. SOFTWARE TESTING

The fundamental and crucial step in the software development process is software testing. Testing is the process of assessing a system or module by giving specified inputs and contrasting them with the intended outputs in order to find and fix any differences between the two. Software testing can be divided into two main areas. Software testing can be done two ways: manually and automatically. Several advantages can be obtained by including automated testing into the software development program. Using a model to automate the generation of automated test scripts and manual test cases reduces labor costs while also increasing coverage and also significantly reduces the time-to-market.

Despite the availability of various testing tools, manual testing remains widely used in many organizations,

especially for web-based finance applications where reliability and accuracy are critical. Manual testing is time-consuming, prone to human error, and inefficient for frequent updates. Therefore, there is a strong need for automated testing solutions that can ensure faster execution, higher accuracy, and improved software quality. This study investigates the effectiveness of Selenium WebDriver in automating functional testing of web applications through experimental evaluation.

1.2.Testing Approaches

Software or applications are tested manually during the process of "manual testing," meaningthat neither test scripts nor software automation tools are used. The process of evaluating software or applications by hand in order to identify errors and confirm that they meet requirements is known as manual testing. To ensure proper behavior, a tester must assume the position of an associate user and make use of nearly all of the application's choices. Before being automatically assessed, a new application must first undergo manual testing. While manual testing is more work, it is essential to determine whether automation is feasible. Every computer code one in all Testing in primary school is complete. Manual testing is crucial and imperative since automation testing is not feasible.
Testing by automation is faster, more dependable, and requires fewer resources per task than manual labor. It can execute more tests faster and reuse them across versions of an application. There are numerous factors to take into account when choosing the testing instrument. It is compatible with the application's design and implementation, test execution, and maintenance, and it integrates easily. Continuous Delivery and Testing Require Test Automation. The majority of manual testing's problems are covered by automation testing. Changing as much of the testing effort as possible with the fewest possible scripts is the aim of automated testing. Tests, reportage results, and scrutiny results with previous test runs can all be carried out by automated testing systems. The automation is not a full replacement for manual testing It is a continuation of manual testing with the goal of increasing testing efforts' accuracy and speed, rather than a full replacement for it.

1.3. Comparison study of Manual and Automation Testing

Factors	Manual Testing	Automated Testing
Time	Testers have to write each and every step hence it is very time consuming and tedious work.	As the tests are performed by software tool the execution is fast
Efficiency	Testers tend to reproduce human errors.	Software tool does not cause human errors and ensure accuracy.
Cost	Huge investment needs to be done in human resources.	Initial set up cost is more but once done can be reused without human intervention reducing cost.
Training	It does not require training for generating scripts	Testers need to be trained to use software tool to generate error free test scripts

Table1.1: Comparison between manual and automation testing

1.4. Best Practices in Software Testing:

There are several best practices in software testing that software developers can follow to ensure that the software product is of high quality. These practices include test planning, test case design, test execution, and bug reporting

Test planning involves defining the testing objectives, scope, and approach for the software product. It includes identifying the testing environment, resources, and timelines for testing. Test case design involves creating test cases that cover all possible scenarios and edge cases of the software product. Test execution involves executing the test cases and recording the results. Bug reporting involves identifying and reporting the defects or issues found during testing.

1.5. Merits of Automation

There are various benefits associated with automation testing that enhance the software testing procedure. Among the main advantages of automated testing are the following:Faster Execution: Automation testing allows for the execution of test cases much faster compared to manual testing. Automation tools can perform repetitive tasks, execute test scripts, and generate test reports swiftly, significantly reducing the overall testing time.

1. Increased Test Coverage: Automation testing enables broader test coverage by executing a large number of test cases and scenarios. It helps to validate different combinations of inputs, perform regression testing, and cover various functionalities that may be impractical to test manually.
2. Regression Testing: Automation testing is particularly beneficial for regression testing, where existing functionality needs to be verified after modifications or enhancements. Automated test suites can quickly execute regression tests, ensuring that changes do not introduce unexpected issues.
3. Reusability: Test automation allows for the reusability of test scripts and test data. Once created, automated test cases can be reused for multiple iterations, releases, and versions of the software,Saving time and efforts in test script development.

2. ANALYSIS

2.1. Software Quality

A quality factor serves as the behavioral indication for a system. Correctness, dependability, efficacy, testability, portability, and reusability are examples of high-level quality qualities. A software system's external features are known as quality factors. Because testability makes it simpler to validate other system properties through testing, such as accuracy, dependability, and efficiency, it is more important to the software quality assurance team.
However, software testing is usually used to enforce and measure functional quality. It is impossible to attain quality by evaluating the finished product. Thus, assessing and grading the quality of a software product requires testing. On the one hand, repeating the test-find-defects-fix cycle during development results in higher-quality goods.
The following are some examples of quality assurance measures: methods, techniques, and tools that support the standard software development process and its structure. Testing is done at different stages according to the requirements of the program that is being developed. Testing can be done either manually or automatically, depending on what best fits the requirements. Testing is a systematic process.

2.2.Process life cycle diagram for Test Automation

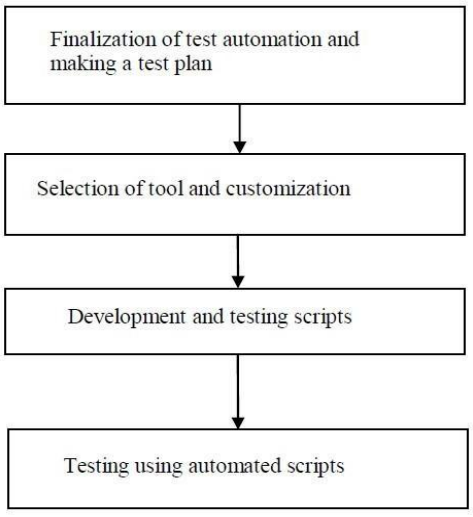


Figure 2.1: Automation lifecycle

3. INTRODUCTION TO SELENIUM

Selenium Framework is an open-source test automation technology that is necessary for automated testing. Selenium is a collection of software tools with a unique approach to test automation. Operating systems and completely independent browsers can both be controlled with it. Many different programming languages, such as Groovy, Ruby, Perl, Python, Java, C, and PHP, are compatible with it. The various parts that make up selenium are divided into three primary instruments. Every individual has a certain role to perform in order to enable the automation of the web application test. Support mobile testing on Android and iOS.

3.1. Components of selenium

Selenium IDE: Selenium test cases are created using the Selenium integrated development environment (IDE). Testers can record their actions as they go through the required workflows for testing with the use of the Selenium IDE Firefox plug-in.

Selenium RC:
The system is a Client-Server architecture that receives the Selenium commands from the editor, and testing is done through a browser. It creates increasingly complex tests by utilizing every capability found in programming languages, including Java, C, PHP, Groovy, Python, Ruby, and more.

parallel simultaneously on multiple computers and browsers, resulting in shorter execution times.

3.2. Selenium 1.0 vs Selenium 2.0 vs Selenium 3.0

With several modifications and improvements throughout time, selenium has undergone changes and been made available to users. Initially, let us look at the key differences between Selenium 3.0, 2.0, and 1.

Selenium 1.0

Selenium 1.0 was the initial version of the Selenium program that was extensively used.

3.2.1. It required the Selenium RC server to be operational and employed automation based on JavaScript to control browsers. Selenium RC's dependence on JavaScript, difficult setup, and incompatibility with a number of contemporary browser capabilities were among its drawbacks.

3.2.2. Programming languages that users can develop test scripts in include Java, C#, Python, Ruby, and Perl.

Selenium 2.0

WebDriver provided a more reliable and efficient way to automate web browsers by direct contact with the browser's native support.

3.2.3. It eliminated the need for Selenium RC server and JavaScript-based automation.

3.2.4. Unlike Selenium 1.0, WebDriver offered a more consistent and cohesive API for users of different programming languages.

3.2.5. WebDriver made it possible to automate tasks more effectively and dependably while also improving browser compatibility and support for modern web technologies.

3.2.6. Selenium WebDriver has become the de facto standard for web automation, replacing Selenium RC as the recommended approach.

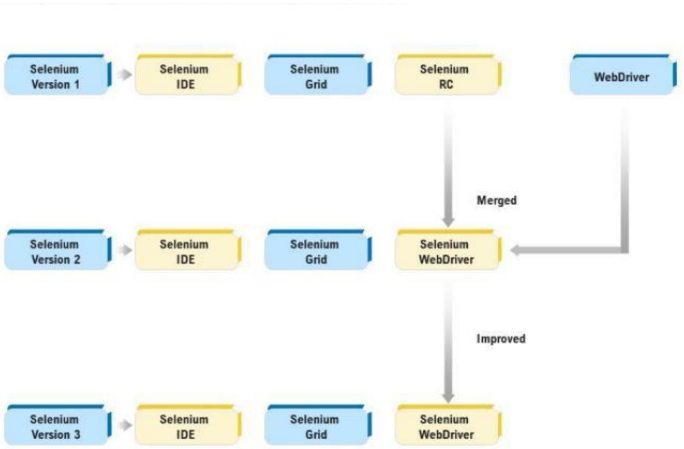
Selenium 3.0: Several improvements and modifications were brought

One of the main changes was the direct integration of the WebDriver API into the Selenium project. There is no longer a need to download Selenium WebDriver separately because it is now a necessary component of Selenium.

The "W3C WebDriver" or "WebDriver standard," which Selenium 3.0 also established, aimed to provide a consistent standard for browser automation across different providers.

3.2.7. It brought improved support for more contemporary browser features, increased stability, and improved compatibility with older browsers.

3.2.8. Selenium 3.0 continued to support a wide range of programming languages, just like its predecessors.



3.3.Frameworks:

Frameworks is a combination of classes, Methods, API's, Libraries these all combined together to work on one application effectively

Different Frameworks :

- **Modular Frameworks**
- **Data Driven Frameworks**
- **Keyword Driver Frameworks**
  - **Junit**
  - **TestNG**
- **Hybrid Frameworks**

- Modular frameworks: They come in both basic and linear forms.
  Modules and sub-modules will be created by the automation engineers within the application.
  The scripts for each module will be created by them.
  The scripts are reliable and will combine all of the scripts to run in a hierarchical manner.
- Data Driven Framework: External files, such as databases, XML, Excel, and TXT, were presented.
  acquiring the input data as well as the resulting result
  It is responsible for executing the provided driver script.
  There are no test cases or data in the driver script; all of this is provided in an external file. Because the driver script first interprets data from an external file before executing it, we refer to this system as data driven.

Keyword Driven

The term "keyword" refers to a step, and these frameworks are executed step by step. There are various types of Keyword Driven Frameworks, and as it is highly comparable to manual testing, test scripts in these frameworks are also similar.

JUNIT

This framework, called Java Unit Testing (junit), is used to write scripts in the Java programming language and then run them. After the scripts are run, junit automatically provides a report containing the results.

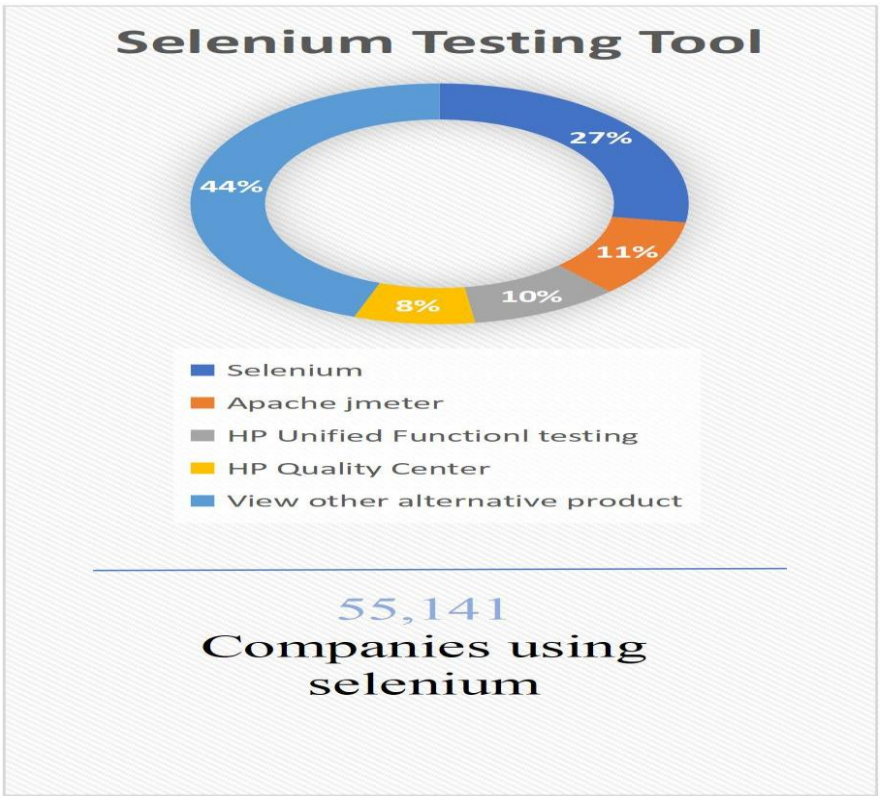
Test Driven

TestNG is an add-on for Eclipse that provides a testing framework built on top of Junit. It overcomes all of Junit's constraints and introduces some additional capabilities.

4. Initial analysis of Selenium with other tools

4.1.Huge community

The data-driven research company Enlyft (formerly iDataLabs) provides an interesting viewpoint on the market for software testing solutions. Surprising statistics reveal that Selenium commands a staggering 27.48 percent of the market share for software testing tools, with Apache Jmeter trailing closely behind at little over 10 percent.



Selenium, being a trailblazer in the realm of contemporary automated testing, has amassed a following of engineers spanning both large corporations like Google and startups. Proficiency with Selenium is listed as a necessary skill in job postings for quality assurance experts.
The alternatives to Selenium, which we previously discussed, range in price from a few thousand to ten thousand dollars. When a great free tool is available, existing testers stay faithful to Selenium, and new testers become enthusiasts.

4.2.Pros of using Selenium

t's finally time to talk about how the reliable Selenium continues to be viable in spite of the abundance of high-quality testing tools that are released annually.

4.2.1. Free
4.2.2. Works with Continuous Delivery, DevOps, and Agile workflows
4.2.3. Extensive add-on and plugin library
4.2.4. Supports a wide range of languages, platforms, and browsers
4.2.5. Enables mobile testing

4.3.Cons of using Selenium

We never award software a high rating without also highlighting some of its shortcomings. Selenium also has a number of them.

4.3.1. Does not have an integrated picture comparison
4.3.2. Has a steep learning curve
4.3.3. Is limited to web-based applications;

5. Literature Review

Several researchers have examined software test automation from architectural, methodological, and tool-based perspectives. Early studies focus on the limitations of manual testing, highlighting challenges related to scalability, time consumption, human error, and increased cost in large-scale web applications. These works emphasize the growing need for automated testing solutions to ensure quality assurance in modern software development environments.

Survey-based research on automated testing tools compares open-source and commercial solutions, identifying Selenium as one of the most widely adopted frameworks due to its flexibility, cross-browser compatibility, and language independence. These studies underline Selenium's suitability for web-based applications while also noting the lack of built-in reporting and technical support as key limitations.

Recent research explores framework-based automation approaches, including data-driven, keyword-driven, and hybrid frameworks. These approaches aim to improve test reusability, maintainability, and execution efficiency. While such frameworks enhance automation effectiveness, researchers identify challenges related to framework complexity, learning curve, and long-term maintenance.

Several studies focus on Selenium WebDriver and its evolution, emphasizing improvements in browser interaction, stability, and support for modern web technologies. Comparative analyses demonstrate that Selenium WebDriver outperforms earlier automation tools by providing direct browser control and better integration with Agile and DevOps practices.

Literature on continuous testing highlights the integration of Selenium with CI/CD pipelines, enabling rapid feedback and regression testing. These studies suggest that automation is critical for continuous delivery models; however, issues such as flaky tests, environment dependency, and synchronization failures remain significant concerns.

Further research emphasizes best practices in test automation, including modular design, separation of test logic and test data, and effective use of assertions and waits. These works argue that test automation alone does not guarantee quality unless supported by proper framework design and disciplined testing strategies.

Despite extensive research on Selenium and automation frameworks, limited work addresses application-level challenges such as test script maintainability, handling dynamic web elements, and optimizing automation strategies for complex user interfaces. This gap highlights the need for structured automation approaches that balance test coverage, execution speed, and maintainability.

Although many studies discuss Selenium and automation frameworks, most of them focus on theoretical comparison of tools rather than practical implementation and performance measurement. There is limited research that provides structured methodology, measurable results, and real-time execution analysis. Therefore, this paper focuses on practical implementation of Selenium WebDriver and evaluates its effectiveness using experimental analysis.

While previous research has analyzed automation tools and frameworks, most studies emphasize tool capabilities rather than real-time implementation and measurable performance outcomes. This paper builds upon existing literature by applying Selenium WebDriver to automate a real-world application and evaluating its effectiveness through experimental analysis. Thus, the study bridges the gap between theoretical research and practical implementation.

6. Proposed Work and Contribution

Finance-based web applications require high reliability, security, and accuracy because even minor defects can lead to significant financial losses. Frequent updates and regulatory requirements make manual testing impractical. Automation testing using Selenium provides an efficient solution for validating critical functionalities such as user authentication, transaction processing, and data handling.

This research presents a practical implementation of Selenium-based automation testing on a real-world web application. Unlike previous studies that primarily focus on theoretical comparison of automation tools, this work demonstrates the design, development, and execution of automated test cases using Selenium WebDriver.

The selected application belongs to the finance domain and includes critical modules such as user login, account access, form validation, and transaction workflows. Automating these modules helps evaluate the effectiveness of Selenium in handling real-time user interactions.

The major contributions of this paper are:

- Development of an automation framework using Selenium WebDriver
- Execution of automated test cases on key application modules
- Comparative analysis between manual and automation testing
- Evaluation of execution time, accuracy, and repeatability

This implementation demonstrates how automation testing can significantly improve testing efficiency in practical software development environments.

7. Research Methodology

This study follows an experimental research methodology to evaluate the effectiveness of Selenium-based automation testing.

Step 1: Selection of Application
A web-based application from the finance domain was selected for testing.

Step 2: Identification of Test Scenarios
Critical functionalities such as login validation, navigation, data entry, and form submission were identified.

Step 3: Test Case Design
Test cases were prepared for both manual testing and automated testing.

Step 4: Manual Testing Execution
All test cases were executed manually, and execution time and results were recorded.

Step 5: Automation Implementation
The same test cases were automated using Selenium WebDriver with Java and TestNG framework.

Step 6: Automated Test Execution
Automated scripts were executed on the Chrome browser.

Step 7: Data Collection and Comparison
Execution time, accuracy, and repeatability were analyzed and compared between manual and automated testing.

The automation framework consists of Selenium WebDriver interacting with the browser through Java-based test scripts. TestNG is used for test execution and reporting. The framework simulates real user actions such as clicking, typing, navigation, and validation of outputs. This architecture enables automated testing of web application workflows efficiently.

8. Experimental Setup

The automation framework was developed using the following environment:

• Programming Language: Java
• Automation Tool: Selenium WebDriver
• Testing Framework: TestNG
• IDE: Eclipse
• Browser: Google Chrome
• Operating System: Windows

A total of 50 test cases were executed for comparison analysis.

9. Results and Analysis

The experimental results indicate that automation testing using Selenium significantly improves efficiency compared to manual testing.

Manual testing of 50 test cases required approximately 4 hours to complete, whereas automated testing completed the same set within 30 minutes. Automation testing eliminated human errors and ensured consistent execution across multiple runs.

Regression testing was performed repeatedly without additional effort, demonstrating the reusability of automated scripts. Cross-browser testing confirmed successful execution on different browser environments.

Overall, Selenium-based automation achieved higher accuracy, reduced execution time, and improved test coverage compared to manual testing.

A total of 50 test cases were executed to evaluate performance. Manual testing required approximately 4 hours, whereas automated testing completed execution within 30 minutes. Automation achieved nearly 85% reduction in testing time while maintaining consistent accuracy across multiple runs.

Table 1: Comparison of Manual and Automation Testing

Parameter	Manual Testing	Automation Testing
Execution Time	High	Low
Accuracy	Moderate	High
Human Error	Possible	Minimal
Reusability	Low	High
Test Coverage	Limited	Extensive
Regression Testing	Time-consuming	Fast and Efficient

10. Practical Implications and Future Scope

The findings of this study demonstrate that Selenium-based automation can be effectively implemented in real-world software development environments, especially in Agile and DevOps practices. Organizations can significantly reduce testing effort, improve product quality, and accelerate release cycles by adopting automated testing frameworks.

Automation is particularly beneficial for large-scale web applications where frequent updates require repeated testing. The use of Selenium enables continuous testing and supports integration with CI/CD pipelines.

Future research can explore the integration of Selenium with artificial intelligence-based testing tools, self-healing test scripts, and automation of complex dynamic web applications.

11. Limitations of the Study

Although Selenium provides powerful automation capabilities, it is limited to web-based applications and does not support desktop applications directly. Automation scripts require programming expertise and regular maintenance when application interfaces change. Handling dynamic web elements and synchronization issues remains challenging. Therefore, Selenium should be combined with proper framework design and testing strategies for optimal results.

Future research can explore the integration of Selenium with artificial intelligence-based testing tools, self-healing automation frameworks, and cloud-based testing platforms. Automation of highly dynamic applications and mobile platforms also presents promising research opportunities.

12. Conclusion

The study provided a number of testing principles and methodologies. Gaining knowledge about Selenium test automation and its significance will be beneficial. Software application testing is made easier with the use of automated testing techniques. Selenium and HP-QTP were only two of the many web automation tools available. Researchers studying Selenium will be able to refer to our study paper as a review of Selenium's web test automation technology in the future. Testing professionals should weigh a tool's features before choosing one and make sure it's the right fit for the job.

This research not only reviews Selenium automation but also provides practical implementation and experimental validation. The findings confirm that automation testing using Selenium enhances productivity and reduces execution time compared to manual testing.

13. Reference

[1] Fei Wang and Wencaai Du, "A Test Automaton Framework Based on WEB" proc. IEEE 11th International Conference on Computer and Information (ACIS 12),IEEE Press, 2012, pp. 683- 687, doi:10.1109/ICIS.2012.21 .
[2] Ms. RigzinAngmo, Mrs. Monika Sharma,"Selenium Tool: A Web based Automation Testing Framework", (IJETCAS),2014.
[3] Sherry single, Harpreetkaur,"Selenium keyword automation testing framework", International Journal of Advanced Research in Computer Science and Software Engineering, Vol.4,2014
[4] Monika Sharma and RigzinAngmo, "Web based Automation Testing and Tools", of Computer Science (IJCSIT), Vol. 5(1),2014, ISSN:0975-9646, pp. 908-912.
[5] Mohammad Imran, Dr. Mohamed A.Hebaishy, Dr. Abdullah Shawan Alotaibi, A Comparative Study of QTP and Load Runner Automated Testing Tools and their Contributions to Software Project Scenario, Vol. 4, Issue 1, January 2016
[6] Niranjanamurthy M, Arun Kumar R , Sahana Srinivas, Manoj RK, Research Study on Web ApplicationTesting using Selenium Testing Framework, IJCSMC, Vol. 3,Issue. 10, October 2014.
[7] Harpreet Kaur, Dr.Gagan Gupta, Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete, ISSN: 2248-9622, Vol. 3, Issue 5, Sep-Oct 2013.
[8] Rafi, "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey", Automation of Software Test, IEEE, pp. 36-42, 2012.
[9] Marback Aaron, Do Hyunsook and Ehresmann Nathan. An effective regression testing approach for php web applications. 2012..
[10] I. Singh, B. Tarika, "Comparative Analysis of Open Source Automated Software Testing Tools: Selenium, Sikuli and Watir" International Journal of Information & Computation Technology, vol 4, pp. 1507-1518, 2015.
[11] G. Meszaros, *xUnit Test Patterns: Refactoring Test Code*, Addison-Wesley, 2007.
[12] E. Dustin, J. Garrett, and B. Gauf, *Implementing Automated Software Testing*, Addison-Wesley, 2009.
[13] M. Fewster and D. Graham, *Software Test Automation: Effective Use of Test Execution Tools*, Addison-Wesley, 1999.
[14] J. Whittaker, *Exploratory Software Testing*, Addison-Wesley, 2009.
[15] P. Ammann and J. Offutt, *Introduction to Software Testing*, Cambridge University Press, 2016.
[16] S. Elbaum, A. Malishevsky, and G. Rothermel, "Test Case Prioritization: A Family of Empirical

Studies," *IEEE Transactions on Software Engineering*, Vol. 28, No. 2, 2002.
[17] R. Pressman and B. Maxim, *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 2014.
[18] A. Bertolino, "Software Testing Research: Achievements, Challenges, and Dreams," *Future of Software Engineering*, IEEE, 2007.
[19] K. Beck, *Test-Driven Development: By Example*, Addison-Wesley, 2003.
[20] D. Graham, E. van Veenendaal, and I. Evans, *Foundations of Software Testing*, Cengage Learning, 2008.