A comparative study of automation testing tools for web applications

Elis Pelivani
South East European University, Tetovo, Macedonia
Contemporary Sciences and Technologies
ep24253@seeu.edu.mk

Betim Cico
Metropolitan Tirana University, Tirana, Albania
Faculty of Computer Science & IT
bcico@umt.edu.al

2021 10th Mediterranean Conference on Embedded Computing (MECO) | 978-1-6654-3912-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/MECO52532.2021.9460242

Abstract— **In today's world, websites and web applications have become a necessity for all kinds of businesses and commercial enterprises. Software development is an important step of designing the software, ensuring that the designed software fulfills its practical needs and that it is free of bugs and errors. Software testing plays an important role in the software development life cycle because it contributes to the overall quality of the product. This guarantees the consistency and standard of the applications. It is important to be mindful of time and costs when evaluating a program. Therefore most testers have moved from a manual testing to automated testing to reduce time and cost. Then selecting a software-testing tool for automated testing that best fit a project is important yet a challenging task. The aim of this paper is evaluation and comparison of automation testing tools used for web application testing. In this paper will be analyzed and compared different features of two testing tools: The Katalon Studio and Selenium.**

Keywords-testing, automation, manual, test cases, Katalon, framework, tool

I. INTRODUCTION

In the world today, peoples' jobs, comforts, safety, entertainment, decisions, and their very lives depends on computer software, so it is better to be done correctly. This is one reason why software testing is incredibly necessary. [1] According to International Software Testing Qualifications Board (ISTQB), testing consists only on running tests, i.e. executing the software. Although this can be count as part of it, it does not mean that it defines the entire process. All test activities exist before and after test execution. Part of these activities are choosing test conditions, evaluation exit criteria, designing and executing five test cases, planning and control, reporting on the testing process, checking results, and completing closure activities after a test phase has been completed [2].

There are many methods open to analyzing applications. Static tests refer to reviews, walk-throughs, or inspections, whereas dynamic tests refer to software code execution with a collection of test cases [3], [4]. Dynamic checking takes place while the software itself is running. Dynamic testing can start before the software is fully completed to test all unique parts of code and extend to discrete functions or modules.

Static testing is an implicit process, comparable to proofreading. Similarly, to how programming tools or text editors inspect the source code structure or how compilers (pre-compilers) control the syntax and data flow as a stable software analysis. Static testing main goal is to achieve authentication, while dynamic testing main goal is to get a confirmation [4].

Passive monitoring means evaluating the actions of the device without any contact with the software product. In comparison to active monitoring, testers do not have any test results, but look at device logs and traces. They mine trends and individual actions to make some form of decisions [5]. This is related to offline runtime verification and log analysis.

- Exploratory Approach

Exploratory testing is another software testing method, which is concisely defined as simultaneous learning, test design, and test execution. Cem Kaner, who "coined the concept in 1984" [6], describes exploratory testing as "a form of software testing that stresses the personal independence and duty of the individual tester to constantly enhance the consistency of his/her work by treating test-related learning, test design, test execution, and test result analysis as mutually supportive tasks that run parallel to each other".

- The "box" approach

Traditionally, software-testing techniques are split into white-and black box testing. The method of testing where the functionalities of an application are tested without having the knowledge of backend logic is called Black Box Testing, and the technique where the backend logic like internal structure, information regarding internal code and design are tested to verify the flow of input-output, the improvement of security, usability and design is called White Box Testing. These two methods are used to characterize the perspective held by the tester when constructing test cases. A mixed approach, called gray-box testing, can also be extended to software testing methodology [7], [8]. With the definition of gray-box testing, which produces experiments focused on particular design features, gaining popularity, this "arbitrary distinction" between black-and-white-box testing has diminished slightly [9]. Different ways of grouping testing and labels may be software testing tactics or techniques and testing types [10].

- Installation Testing

Most software systems have installation procedures that are required before they can be used for their main purpose. With the installation tests, we understand the process of testing the procedures to achieve an installed software that can be used.

- Compatibility Testing

This failure is caused by the lack of synchronization of the operating systems and the software applications. Conversely, these failures might come from the environments that vary from testing to production one.

- Smoke and Sanity Testing

Sanity testing decides whether further testing is necessary. Smoke testing consists of brief attempts to run the software to decide if there are any simple bugs that preclude it from running at all. These tests can be used to build a verification test.

- Regression Testing

Regression testing focuses on finding flaws after significant code changes have happened. Specifically, it aims to discover technological regressions such as degraded or lost functionality, including old vulnerabilities that have returned. Such regressions occur if program functionality that has been running correctly has ceased to perform as intended.

- Alpha Testing

Alpha testing is a virtual or real running test conducted by prospective users/customers or by an autonomous test party at the developers' site. Alpha testing is also used with off-the-shelf applications as a form of internal approval testing before the software moves through beta testing [11].

- Beta Testing

Beta testing is conducted after alpha testing and may be viewed as a form of external user acceptance testing. Versions of the software, known as experimental versions, are distributed to a small audience outside the programming team known as beta testers. The software is distributed to groups of people so that additional tests will guarantee that the application has few defects or glitches.

- Non-functional vs Functional testing

Functional testing is the activities that verify a specific action or function of the code. These are typically included in the code specification documentation, but certain implementation methodologies are based on usage cases or user stories. Functional tests appear to address the issue of "can the user do this" or "does this particular feature work." Non-functional testing refers to features of the software that may not be linked to a particular feature or activity of the user, such as scalability or other performance, actions under certain restrictions, or protection.

- Continuous testing

The process of running automated test cases as part of the pipeline of product development to gain immediate input on the market threats involved with a software release candidate, is known as Continuous Testing [12]. Continuous testing requires validation of all functional requirements and non-functional criteria; the scope of testing ranges from validation of bottom-up requirements or usage history.

- Destructive testing

Destructive testing aims to force the program or subsystem to malfunction. It verifies that the software operates correctly even though it accepts invalid or unintended inputs, thus establishing the robustness of input validation and error management routines. Software fault injection, in the context of fuzzing, is an example of malfunction checking.

- Software performance testing

Performance testing is generally conducted to assess how the device or subsystem functions in terms of responsiveness and reliability under a specific workload. The attributes can be investigated and then calcite to validate or verify the system.

- Usability testing

Usability testing is to verify that the user interface is easy to use and understand. It concerns primarily the use of the program. This is not a form of research that can be automated; real individual users need to be supervised by trained UI programmers.

- Accessibility testing

  The accessibility testing makes the application usable for everyone. Three standards for accessibility testing are listed below:

  - Americans with Disabilities Act of 1990.
  - Section 508 Amendment to the Rehabilitation Act of 1973.
  - Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C)

- The process planned to reveal flaws in the security mechanisms of an information system that protect data and maintain functionality as intended is called security testing.

Security testing is essential for software that handles sensitive data to deter hackers from accessing the device.

- Property testing

Property testing is a testing technique where, instead of asserting that particular inputs produce specific predicted outputs, the practitioner arbitrarily produces several inputs, runs the software on all of them, and asserts the truth of any "property" that should be valid for each input and output pair.

- VCR testing

VCR testing, also known as, "playback testing" or "record/replay" testing, is a test technique that aims to increase efficiency and speed up the regression tests. It includes a variable that is slow or unreliable to connect with, typically a third-party API outside the reach of the tester.

II. Background

Automation test is getting better and more efficient every day. Today in the market, there are a lot of frameworks and tools available to test the application. Their architecture and methodology of testing is different between each other making one technology better at one point than the other does.

A. Test Automation Framework

Test automation framework is a well-defined architecture for creating the tests based on the business logic. Developers uses the automation frameworks to develop a structure code, which test every feature of the application. Test Automation framework provides higher reusability of test components and easily maintainable scripts. According to Steve McConnell, there are four types of framework [13]:

- Modular Framework

This framework contains reusable code that can be called from multiple scripts.

- Data-driven Framework

This framework provides reusable steps to be executed on different test scripts.

- Hybrid Framework

The framework is a mix of the data-driven and modular framework.

- Keyword-driven Framework

Keyword-driven Framework is useful for non-technical users as it identifies a keyword to be performed with the details from the spreadsheet.

This paper analyzes the most popular and best frameworks that are used in important projects. They show a major differences and improvement. Their architecture is unique and very powerful to test the application in efficient way. The other frameworks are build using these frameworks [14].

- **Selenium**

Selenium is the most popular automation framework for testing web applications [15].

- ***Advantages***
  - It is open source.
  - Works on multiple operation system and languages.
  - Cross-browser testing. Selenium supports all type of browsers, making it used in any application.
  - Possibility to integrate with many different tools.
- ***Disadvantages***
  - Configuring the project is done from scratch and it takes time.
  - Timing. Developer needs to ensure the CSS elements are visible.
  - Not smart to give specific reason of what makes the test fail.

- **Cypress**

Cypress is a next generation front end Automation testing tool [16].

- ***Advantages***
  - Operates inside the browser. In this way knows everything that happens in the browser if the page loads.
  - Modifies the browser behavior at run time by manipulating DOM and altering Network requests and responses on the fly.
  - Ability to test Edge Test Cases by mocking the server response. (Out of box testing)
  - Architecture design makes cypress delivers fast and reliable test execution. (does not need any driver for connection)
  - Supports unit testing, integration testing and end-to-end testing.
  - It provides his own CI.
- ***Disadvantages***
  - Supports only some of the browsers (Chrome, Electron).
  - It can be developed only in JavaScript, using Mocha and Chai.
  - It supports CSS selectors only.

- **Robot Framework**

Robot Framework is a powerful tool that uses driven testing approach [17].

- ***Advantages***
  - It has a modular architecture with self-made libraries.
  - Easy. Robot Framework helps in creating test cases.
  - Supports keyword driven, behavior driven and data driven.
- Disadvantages
  - No nested loops.
  - No features like skip test or author tag.

- **WebDriverIO**

WebDriverIO is a selenium framework that uses NODE.js [18].

- ***Advantages***
  - Easy to setup and run by using CI tools.
  - Fast and stable.
  - Supports different extension and plugin.
  - Does not require to implement anything from scratch.
- ***Disadvantages***
  - It is not customized.
  - It is less object-oriented API.

- **Gauge**

Gauge is lightweight cross-platform test automation framework to writ test cases in business language [19].

- ***Advantages***
  - Simple, flexible, and rich syntax based on Markdown.
  - Modular architecture.
  - Supports data driven execution.
  - Gauge takes a screenshot on a test failure allowing you to get a visible picture of what went wrong.

B. Automation Tools

Automation tool is a software that is based mostly on the frameworks. It improves and extends the tester's ability to test the application, by even providing features. Various testing automation tools have his own methodology for testing the application. This paper shows the most unique automation tools that shows a significant evolvement of the technology [20],[21].

- **Katalon Studio**

Katalon Studio is free of charge test automation tool [22].

- ***Advantages***
  - Features for quick test cases creation.

- No need for advance programming skills.
- ***Disadvantages***
  - Lack of choices for scripting languages.

- **UFT**

Unified Functional Testing is an automated functional testing tool. UFT has two main components, GUI testing and API testing [23].

- ***Advantages***
  - Required only basic programming skills to get started with test creation and execution.
  - Comprehensive automated testing features integrated into single system.
- ***Disadvantages***
  - Costly solution.
  - Supporting only VBScript.

Test Automation Framework separates the raw code from the testing code, generates logs and provides a list of common libraries of functions. On the other side automation tools are the utilities that testers are using while the actual code is being used in the background.

Following sections of this paper will be focused on comparing Selenium and Katalon Studio, as they have a significant difference in architecture and methodology of testing the application. It shows the weak and strength points of each of them and how those can be combined to have the best automation test.

III. SIMULATIONS

a) Computer type

Same computer type used for both Selenium+ Cucumber and Katalon Studio executions, Table 1.

TABLE I. THE SIMULATED TEST ENVIRONMENT

	Selenium	Katalon Studio
System Manufacturer	HP	HP
System Model	HP EliteBook 850 g3	HP EliteBook 850 g3
RAM	16.0 GB	16.0 GB
Operating System	Windows 10 64bit	Windows 10 64bit
Tool version	3.141.59	Katalon 7.9.1
Browser used	Chrome 80.0	Chrome 80.0

b) Test Cases

The same test cases are implemented and executed for both Selenium and Katalon Studio Tool.

No. of test cases = 5
No. of steps ~ 120 steps
Testing Type: UI Testing

c) Execution time

Obviously, the execution time for Katalon Studio is higher than Selenium , Table 2 and Figure 1. The main reason for that is the programming language used. Katalon Studio uses Groovy, a language built on top of Java, and must load many libraries for parsing test data, test objects, logging while for Selenium framework is used Java itself.

TABLE II. THE SIMULATED TEST EXECUTION TIME RESULTS

Test Case	Selenium	Katalon Studio
T1	15.850s	29.934s
T2	18.127s	40.922s
T3	21.873s	42.448s
T4	16.165s	33.599s
T5	17.841s	34.811s
Total	1m – 29.856s	3m - 16.293s

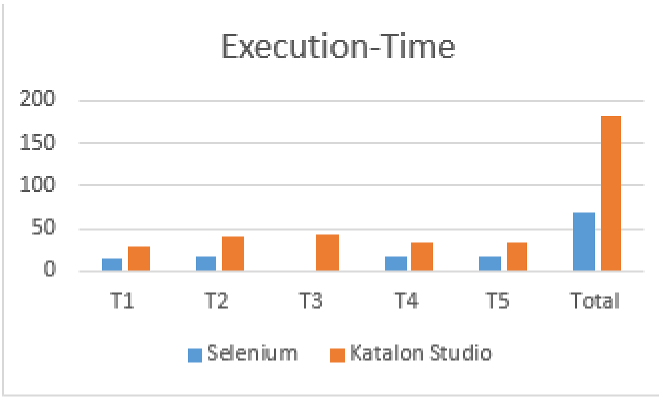


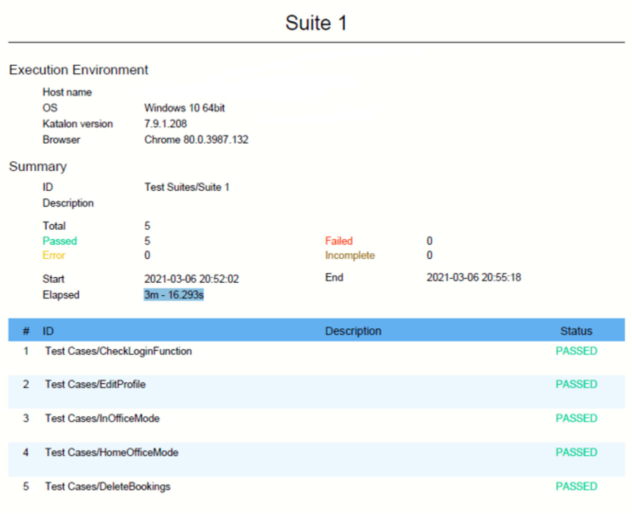
Figure 1. Execution time of Katalon studio and Selenium

d) Reporting

Reporting can be considered as one of advantages of Katalon Studio, Figure 2 over Selenium framework, Figure 3.

On one hand, Katalon provides detailed, user friendly and easy to read reports that can be exported into HTML, CSV and PDF files and shared with team members or manager anytime. Screenshots in case of failures are also included without any extra configuration needed.

On the other hand, Selenium do not have reporting tools, extra configurations are needed to be done based on the framework used like TestNG, Cucumber and so on. which have some simple options of reporting templates.



Suite 1

Execution Environment

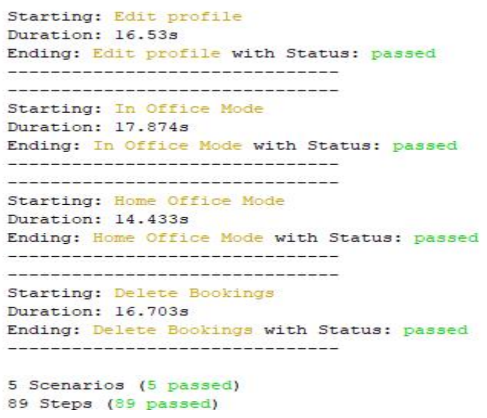
Host name	
OS	Windows 10 64bit
Katalon version	7.9.1.208
Browser	Chrome 80.0.3987.132

Summary

ID	Test Suites/Suite 1		
Description			
Total	5		
Passed	5	Failed	0
Error	0	Incomplete	0
Start	2021-03-06 20:52:02	End	2021-03-06 20:55:18
Elapsed	3m - 16.293s		

#	ID	Description	Status
1	Test Cases/CheckLoginFunction		PASSED
2	Test Cases/EditProfile		PASSED
3	Test Cases/InOfficeMode		PASSED
4	Test Cases/HomeOfficeMode		PASSED
5	Test Cases/DeleteBookings		PASSED

Figure 2. The report generated from Katalon studio



```
Starting: Edit profile
Duration: 16.53s
Ending: Edit profile with Status: passed
-------------------------------
-------------------------------
Starting: In Office Mode
Duration: 17.874s
Ending: In Office Mode with Status: passed
-------------------------------
-------------------------------
Starting: Home Office Mode
Duration: 14.433s
Ending: Home Office Mode with Status: passed
-------------------------------
-------------------------------
Starting: Delete Bookings
Duration: 16.703s
Ending: Delete Bookings with Status: passed
-------------------------------

5 Scenarios (5 passed)
89 Steps (89 passed)
```

Figure 3. The results from Selenium framework

e) Easy of setup

In general, frameworks are harder than tools to set up. For this reason, even why Selenium is packaged into a ready-to-use stuff, it needs extra configuration depends on which IDE, programming language, testing frameworks are being used. In this case study is used a framework which uses Selenium, Java, and Cucumber.

Katalon has a much easier installation process and does not require technical knowledge. After downloading it from the Katalon official web site, we just need to run the .exe file than we are ready for using it.

f) Capture and recording versus programing

Katalon Studio is an easy tool, which does not require programming knowledge for implementing simple test cases. It has three main ways to form tests: Record and Replay, Manual mode, Script mode. Using record/replay, mode user can record actions that have to be performed in the application or functionalities that need to be tested, and after saving the test, he can run the test when needed or change some of the recorded steps depending on the idea of test.

Scripting mode allows the user to write Groovy/Java code for creating a test or, more often, to customize a recorded test and to create advance functions called custom keywords.

While for Selenium testers programming knowledge is required to build the framework, create the structure of the testing project, and implement functions and scenarios, Table 3.

g) Programming language

Katalon Studio: Groovy/Java Selenium: Java, C#, Perl, Python, JavaScript, Ruby, PHP (Java used in our study), Table 3.

TABLE III. DIFFERENCES BETWEEN KATALON STUDIO AND SELENIUM [23],[24]

Features	**Katalon Studio**	**Selenium**
Test development platform	Cross-platform	Cross-platform
Application under test	Web and mobile apps	Web apps
Scripting languages	Java/Groovy	C#, Python, JavaScript, Ruby, Perl, PHP , Java
Programming skills	Recommended for advanced test scripts, Not required	To integrate various tools advanced skills needed
Learning curves	Medium	High
Ease of installation and use	Easy to run and setup	Require installing and integration various tools
Script creation time	Quick	Slow
Object storage and maintenance	Build-in object repository, object re-identification, Xpath	Xpath, UI Maps
Image-based testing	Built-in support	Require installing additional libraries
Continuous integrations	The most used CI tools (e.g. Teamcity, Jenkins)	CI tools (e.g. Cruise Control , Jenkins)
Product support	Ticketing support community	Open source community
License type	Freeware	Open source (Apache 2.0)
Cost	Free	Free

IV. CONCLUSIONS

Commonly, manual testing and automated testing are considered and used as two different testing approaches. In fact, these two testing approaches depend on one another. The former is performed in a manual way, by navigating through application as an end user. Manual testing is more useful in projects where software requirements change continually. On the other hand, test automation makes the testing process faster, provides code reusability, ensures consistency, and overcomes weaknesses of manual testing. There are many frameworks and tolls that support and help test automation. It takes time and practice to be able to master each tool. In addition, it takes time to determine which tool is more appropriate for the type of testing used. In my view, to successfully fulfill the testing needs, a tool should have the following characteristics: To begin with, the tool should be installed easy and quickly, and it should support both users with no programming skills, and those with good programming skills. Secondly, the tool should support integration with other frameworks and reporting tools, to make it easy to understand the cause of the failure. It is very important to have record, playback, and script maintainability capabilities. In this paper some of the most powerful and used ones are reviewed and compared with each other. Selenium and Katalon studio are analyzed in a real web application with real test cases. This review will be followed by a wider range of simulation in test cases numbers and in other tools and frameworks. On a final note, for future work, I would like to extend this paper by adding some other testing tools, as well as more evaluation factors for comparison, Table 3.

V. FUTURE WORK

In recent years, automation test has made a major evolution. Many different tools and framework can be analyzed, as they use different architecture to test the application. Comparing Selenium framework and Katalon Studio tool are important because they show the topics where the automation test is improved. However, Artificial Intelligence is not integrated too much. The future work of this paper is implementing the artificial intelligence in software testing frameworks for making the software development lifecycle easier. Artificial intelligence can make significant improvement on:

- Reasoning and problem solving.
- Updating all CSS elements.
- CI – run only the affected test from the changes.

The goal of integrating the AI is to make automation test more efficient and faster. AI can greatly improve testing process of an application. There are many features we can add and make the future of automation test improved.

REFERENCES

[1] F. Okezie, I. Odun-Ayo, S. Bogle. "A Critical Analysis of Software Testing Tools", Journal of Physics: Conference Series, 2019

[2] Kaner, Cem (November 17, 2006). Exploratory Testing (PDF). Quality Assurance Institute Worldwide Annual Software Testing Conference. Orlando, FL. Retrieved November 22, 2014.

[3] Graham, D.; Van Veenendaal, E.; Evans, I. (2008). Foundations of Software Testing. Cengage Learning. pp. 57–58. ISBN 9781844809899.

[4] Oberkampf, W.L.; Roy, C.J. (2010). Verification and Validation in Scientific Computing. Cambridge University Press. pp. 154–5. ISBN 9781139491761.

[5] Lee, D.; Netravali, A.N.; Sabnani, K.K.; Sugla, B.; John, A. (1997). "Passive testing and applications to network management". Proceedings 1997 International Conference on Network Protocols. IEEE Comput. Soc: 113–122. doi:10.1109/icnp.1997.643699. ISBN 081868061X. S2CID 42596126.

[6] Cem Kaner (2008), A Tutorial in Exploratory Testing (PDF)

[7] Limaye, M.G. (2009). Software Testing. Tata McGraw-Hill Education. pp. 108–11. ISBN 9780070139909.

[8] Saleh, K.A. (2009). Software Engineering. J. Ross Publishing. pp. 224–41. ISBN 9781932159943.

[9] Ammann, P.; Offutt, J. (2016). Introduction to Software Testing. Cambridge University Press. p. 26. ISBN 9781316773123.

[10] Kaner, Cem; Bach, James; Pettichord, Bret (2001). Lessons Learned in Software Testing: A Context-Driven Approach. Wiley. pp. 31–43. ISBN 9780471081128.

[11] Standard Glossary of Terms used in Software Testing" (PDF). Version 3.1. International Software Testing Qualifications Board. Retrieved January 9, 2018.

[12] Mojtaba Shahina , Muhammad Ali Babara , Liming Zhub. Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices

[13] McConnell, Steve (2004). Code Complete (2nd ed.). Microsoft Press. p. 29. ISBN 978-0735619678.]

[14] 11 top open-source test automation frameworks: How to choose; https://techbeacon.com/app-dev-testing/top-11-open-source-testing-automation-frameworks-how-choose (2021)

[15] "Selenium Documentation: Test Automation for Web Applications," www.Seleniumhq.org. [Online]. Available: http://www.Seleniumhq.org/docs/01_introducing_Selenium.jsp (2020)

[16] An Introduction to Cypress; https://docs.cypress.io/guides/overview/why-cypress.html# In-a-nutshell (2021)

[17] An Introduction to Robot Framework; https://robotframework.org/#documentation (22 Aug, 2019)

[18] An Introduction to WebDriverIO; https://webdriver.io/

[19] An Introduction to Gauge; https://docs.gauge.org/?os=windows&language=java&ide=vscode (14 Dec, 2019)

[20] A Study of Automated Software Testing: Automation Tools and Frameworks; Mubarak Albarka Umar Changchun (University of Science and Technology), Zhanfang Chen Changchun (University of Science and Technology) (December 2019)

[21] A Comparison of Automated Testing Tools, https://www.katalon.com/resources-center/blog/comparison-automated-testing-tools/ (Jun 2020)

[22] The Good and the Bad of Katalon Studio Automation Testing Tool; https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-katalon-studio-automation-testing-tool/ (18 Jan, 2019)

[23] A Comparative Study of Automated Software Testing Tools, Nazia Islam, St.Cloud State University. (2016)

[24] Test Automation Tool comparison – HP UFT/QTP vs. Selenium, Aspire (25 Nov, 2013)