JOURNAL OF COMPUTER SCIENCES INSTITUTE

JCSI 35 (2025) 159–165
Received: 21 January 2025
Accepted: 05 March 2025

A comparative analysis of web application test automation tools

Analiza porównawcza narzędzi do automatyzacji testów aplikacji internetowych

Michał Moń*, Beata Pańczyk

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The aim of this study was to compare three web application test automation tools: Playwright, Cypress and Selenium. The comparative analysis was carried out using an empirical study, which consisted of executing test scenarios and reviewing the documentation of the tested tools. The following measures were used as criteria for comparison: average test execution time, average percentage CPU usage, average percentage RAM usage. Based on the results obtained and their analysis, it was concluded that Playwright has shown higher effectiveness and flexibility in most test cases.

Keywords: Playwright; Cypress; Selenium; automated testing

Streszczenie

W artykule przeprowadzono analizę porównawczą trzech narzędzi do automatyzacji testów aplikacji internetowych: Playwright, Cypress oraz Selenium. Analizy porównawczej dokonano z wykorzystaniem badania empirycznego, które polegało na wykonywaniu scenariuszy testowych oraz przeglądzie dokumentacji badanych narzędzi. Jako kryteria do porównania przyjęto miary: średni czas wykonania testu, średni procent zużycia procesora i średni procent zużycia pamięci RAM. Na podstawie uzyskanych wyników i ich analizy najlepszą efektywność osiągnął i najbardziej elastycznym w większości przypadków testowych okazał się Playwright.

Słowa kluczowe: Playwright; Cypress; Selenium; testowanie automatyczne

*Corresponding author

Email address: michal.mon@pollub.edu.pl

1. Introduction

Software testing plays an important role in the software lifecycle. It ensures that end users receive more reliable applications and services. If the software testing phase was to be skipped, the product could not only carry a cost in financial terms but also in other aspects such as decreased end user trust.

Test automation is an area of software quality assurance that differs from manual testing in that, instead of a human, it is the software that carries out specific tests. The benefits of automation are many, especially in systems that do not undergo major changes. Automated tests are faster, more accurate and always produce the same results, which cannot be said of manual testing, where there is always some chance of human error when entering data or performing an incorrect operation in the system.

Businesses around the world are increasingly digitalising. So, it is an obvious fact that more and more tools will emerge over time to support the work of testers and software quality engineers.

The main aim of present paper is to comprehensively analyse and compare three popular tools that are used in the software test automation process for web applications: Cypress, Playwright and Selenium. These tools were chosen primarily because they are widely used. Selenium is an older, proven tool with broad support for many browsers and programming languages. Cypress stands out for its simplicity and speed but also its automatic screenshot and video capture functionality which makes the testing process easier. Playwright is a modern tool offering advanced features, stability and multi-browser support.

In particular, the aim is to identify their advantages, disadvantages and appropriate applications in different testing contexts. **The research hypothesis is that the Playwright tool will show higher effectiveness, efficiency and flexibility compared to other tools in most test cases.**

2. Related works

Since software testing is an important part of software life cycle, it has attracted significant attention when it comes to literature. The existing work about test automation can be divided into three groups.

First group [1-5] are publications regarding test automation models and theories. In [1-3, 5] several testing approaches and methods were researched and analysed including: conceptual model supporting automated testing decision-making process [1], analysis of open-source projects' automated test maturity [2], a method of prioritisation and automatisation of regression tests with use of Selenium, Puppeteer and Playwright [5], novel approach to GUI-Based Software testing with use of GPT-4 AI model and Selenium [3]. In [4] the author focused on the theory of software test automation, in particular: the selection of test automation tools, the creation of independent test automation teams, an overview of software test

automation tools, test environments and which tests should be automated.

Second group are publications regarding more specific test automation tools and frameworks. In [6-12] the authors decided to compare and analyse multiple tools and frameworks such as: Selenium, Cypress, Robot Framework, Playwright and Watir. In [6, 8] the authors developed applications to be tested with the use of the aforementioned tools. In [7, 9] already existing applications have been used. In [10-12] authors focused more on functionalities and capabilities of these tools.

Third group [5, 13-15] are publications focused on methods and practices regarding test automation. In [5] the authors presented a method for prioritising and automating software regression tests that combines change-based prioritisation and test script automation. In [13] the authors conducted an empirical study comparing three approaches to web-based test automation: natural language processing (NLP)-based, programmable and capture&replay. In [14, 15] authors described theoretical and practical foundations of software test automation.

Analysis of aforementioned publications provided a wide variety of information about both the testing process and the tools themselves. From the analysis, it is apparent that these tools enable their use in many testing contexts. Additionally, it can be seen that trends in the web application test automation tools used have changed over the years and that more modern Playwright may have an advantage over currently the most popular Selenium.

3. Materials and methods

To conduct a comparative analysis of web application test automation tools (Cypress, Playwright, and Selenium) a few research methods were used.

3.1. Selection of research methods

The first phase of research started with a literature review to gather existing information about each one of the tools, in particular about their functionalities and common uses. Based on the reviewed literature evaluation criteria were developed, including aspects such as:

- functionality: capabilities and limitations of the tools in test automation,
- performance: test execution time and system resource consumption,
- flexibility: ease of adapting the tool to specific testing requirements,
- ease of use: complexity of setup, availability of documentation, and community support.

For further analysis, experiment was conducted with use of a pre-developed open-source e-commerce web application. Each tool was applied to automate the same sets of test cases, which has enabled their comparison. The results were measured and analysed to assess whether the differences between the tools were statistically significant.

3.2. Design of experiment

The research experiment was divided into a few key stages:

1) Test environment preparation: a test environment was created for the used web application. Each of the tools was installed on the authors' test machine and set up.
2) Test cases development: test cases were developed for the web application, covering various real-life usage scenarios. These test cases included typical user interactions with the application.
3) Test scripts implementation: test scripts for each of the previously created test cases were written with use of each of the analysed tools. The test scripts were written as to be as similar as possible in terms of logic to ensure differences in implementation wouldn't disrupt the results.
4) Tests execution: tests were executed in the previously created test environment, with results collected on test execution times and system resource usage.

3.3. Analysed web application test automation tools

3.3.1. Cypress

Cypress is a popular testing framework which provides support for test automation with use of JavaScript and TypeScript programming languages. For real-time feedback and short test execution time, it operates in the same execution loop as the application being tested.

3.3.2. Playwright

Playwright is a browser automation library developed by Microsoft. It supports multiple programming languages, including JavaScript, Python, C# and Java. It provides support for parallel test execution and headless mode. Playwright is a superb tool for automating complex scenarios with features such as browser context isolation and network interceptions.

3.3.3. Selenium

Selenium is a popular browser automation framework with a wide range of tools and libraries. Selenium that operates with use of the WebDriver protocol. It supports multiple programming languages (Java, Python, C#, JavaScript and more).

4. Research method

To analyse the web application test automation tools, the chosen research method was to implement and execute test scripts using each of the tested tools, and then measure the time and usage of system resources. A comparison of the available functionalities, community support and documentation quality for each of the examined tools was also made.

4.1. Objects and scope of research

The primary objective of this research is to conduct a comparative analysis of three popular web application test automation tools: Selenium, Cypress, and Playwright. These tools have been selected due to their popularity and different architectures.

The scope of the research involves evaluating the tools based on their key functionalities, such as browser and programming language support, but also the test execution speed, system resource usage and ease of use. To analyse each one of these tools test cases resembling common web application scenarios – such as form submission and navigating the different modules of the application – were created. Additionally, the research considered aspects such as setup complexity and community support to provide a comprehensive comparison of these tools.

4.1.1. Web application

A web application that presents a fictional tool shop was selected for the study [16]. The application is mainly written in PHP and TypeScript. It also has its own server on which the research was conducted. The application is fully free and has open-source code. The description of the application's repository on the GitHub platform [17] contains all the information required to begin the research, including default test account credentials. Figure 1 shows the home page of the application.

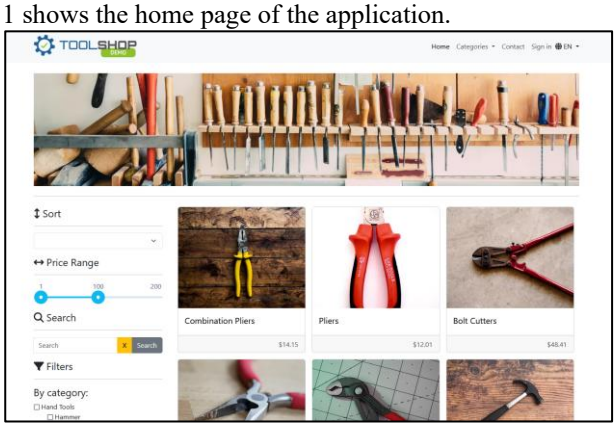


Figure 1: Home page of the test application.

4.2. Research stand

The research was conducted on a workstation with the specifications shown in Table 1.

Table 1: Research stand specification

CPU	Intel i5-10400F 2.90 GHz
GPU	GIGABYTE NVIDIA GeForce RTX 3060
RAM	Kingston KF3600C16D4 16GBx2
Motherboard	MSI B460M PRO
Hard drive	SSD Kingston A2000 1TB M.2
Display screen	1920x1080 resolution, 144 Hz refresh rate
Network	1 Gb/s LAN connection
Operating system	Windows 10 Pro 64-bit 22H2 version

4.3. Research Scenarios

A few research scenarios were prepared to properly analyse each one of the tools:

- scenario 1: **Functional Tests** - verifying that the basic functionalities of the application work correctly, such as user login, adding products to the cart, and processing payments. Created test cases are presented in Table 2.
- scenario 2: **Performance Tests** - measuring the performance of each tool by analysing the test execution time and resource usage (CPU, RAM) during test execution to compare the efficiency of the tools.
- scenario 3: **Usability Tests** - evaluating the ease of use of each tool by analysing the setup time, the quality of documentation and community support available for each tool.

Table 2: Functional test cases created for the experiment

Id	Test case
1.	Registering a new user
2.	Signing in to the application using correct credentials
3.	Adding products to shopping cart
4.	Adding and removing products from shopping cart
5.	Renting a service from “Rentals” page
6.	Submitting a message using “Contact” form and attaching a text file
7.	Signing in as standard user and adding products as favourites
8.	Signing in as standard user and changing password
9.	Signing in as standard user and downloading a PDF file from invoices history
10.	Signing in as administrator and opening “Reports” section.

5. Results analysis

The main goal of this paper was to verify the research hypothesis which stated that Playwright tool will show higher effectiveness, efficiency and flexibility compared to Selenium and Cypress in most test cases.

5.1. Functional tests

Test scripts were written and executed in Visual Studio Code IDE [18]. Each of the test scripts has been executed 10 times and the results presented in Tables 3-5 have been averaged. Sample test script written with use of Playwright is shown in Figure 2.

```
// Registering a new user
test('Registering a new user', async ({ page }) => {
    await page.goto('https://practicesoftwaretesting.com/auth/register');
    await page.locator('[data-test="first-name"]').fill('Test');
    await page.locator('[data-test="last-name"]').fill('User');
    await page.locator('[data-test="dob"]').fill('1990-11-11');
    await page.locator('[data-test="address"]').fill('Test Street 21');
    await page.locator('[data-test="postcode"]').fill('20-400');
    await page.locator('[data-test="city"]').fill('TestCity');
    await page.locator('[data-test="postcode"]').fill('11-111');
    await page.locator('[data-test="state"]').fill('Test State');
    await page.locator('[data-test="country"]').selectOption('PL');
    await page.locator('[data-test="phone"]').fill('555666777');
    await page.locator('[data-test="email"]').fill('test@email.com');
    await page.locator('[data-test="password"]').fill('testPassword1!');
    await page.locator('app-password-input').getByRole('button').click();
    await page.locator('[data-test="register-submit"]').click();
    await expect(page.getByText('Login Sign in with Google or')).toBeVisible();
});
```

Figure 2: Sample test script written with use of Playwright.

5.2. Performance Tests

Each of the test scripts has been executed 10 times and the results presented in Tables 3-5 have been averaged. The resource usage has been measured with use of Windows Performance Monitor tool [19].

Table 3: Average test execution duration in seconds

Test case id	Playwright (s)	Cypress (s)	Selenium (s)
1.	2.51	3.49	2.76
2.	1.84	1.98	2.28
3.	10.75	9.66	9.41
4.	12.69	11.97	11.56
5.	10.23	10.24	9.73
6.	2.04	2.76	2.57
7.	6.23	6.13	5.04
8.	10.39	12.12	10.45
9.	3.76	3.9	4.09
10.	2.2	2.55	3.15

Table 4: Average CPU usage during test execution

Test case id	Playwright (%)	Cypress (%)	Selenium (%)
1.	15.66	18.92	22.59
2.	14.71	19.71	24.42
3.	14.38	15.32	15.43
4.	13.37	17.48	14.71
5.	18.39	16.33	14.01
6.	15.65	19.98	22.57
7.	23.11	26.95	21.86
8.	20.89	21.73	22.35
9.	20.66	26.16	17.49
10.	18.91	23.77	24.36

Table 5: Average RAM usage during test execution

Test case id	Playwright (%)	Cypress (%)	Selenium (%)
1.	28.88	33.01	26.55
2.	28.93	33.12	26.66
3.	28.57	33.19	27.27
4.	29.17	33.27	27.39
5.	29.62	33.41	27.36
6.	29.58	33.13	26.92
7.	28.55	33.76	29.71
8.	28.31	34.88	29.81
9.	28.16	33.91	29.72
10.	27.97	33.37	29.38

Data from Tables 3-5 has also been presented in Figures 3-5.

5.3. Usability Tests

Table 6 presents a detailed comparison of supported programming languages, supported web browsers and various functionalities which the analysed tools provide. Selenium supports the highest number of programming languages but lacks built-in screenshot/video capture. Cypress supports only JavaScript and TypeScript languages which is less than Playwright. All three of the analysed tools support the same web browsers, headless mode and parallel test execution.

Setup time for both Playwright and Cypress is similar, as they can be installed simply by downloading their respective npm [20] packages. Both of them also require Node.js runtime environment [21] to be installed on the local machine. For Selenium however the setup process can be more difficult because it varies for different Selenium libraries and can involve adding additional dependencies or modifying configuration files.

The community support for all three of the analysed tools is significant but recently Playwright popularity has gone up and according to npm trends [22] Playwright has surpassed both Cypress and Selenium in number of downloaded npm packages and GitHub stars as shown in Figure 6.

Table 6: Comparison of Playwright's, Cypress's and Selenium's characteristics

	Playwright	Cypress	Selenium
Supported programming languages	JavaScript, TypeScript, C#, Ruby, Python, Java	JavaScript, TypeScript	JavaScript, Groovy, C#, Java, Perl, PHP, Python, Ruby, Scala
Supported web browsers	Chrome, Edge, Firefox, Safari	Chrome, Edge, Firefox, Safari	Chrome, Edge, Firefox, Safari
Headless mode	Yes	Yes	Yes
Parallel test execution	Yes	Yes	Yes
Screenshot/video capture feature	Yes	Yes	No built-in feature
Assertion libraries	Mocha, Chai	Mocha, Chai	JUnit, TestNG

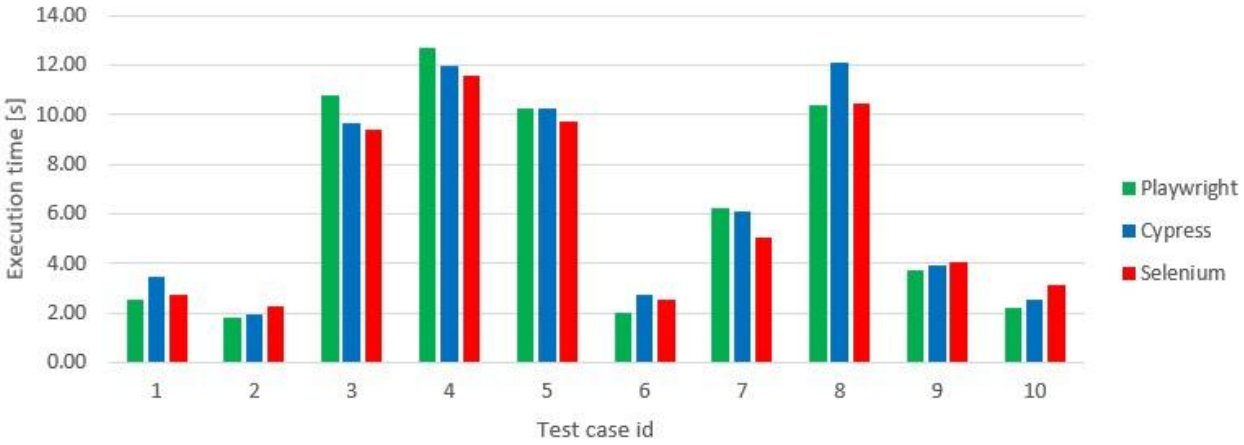


Figure 3: Average test execution duration in seconds.

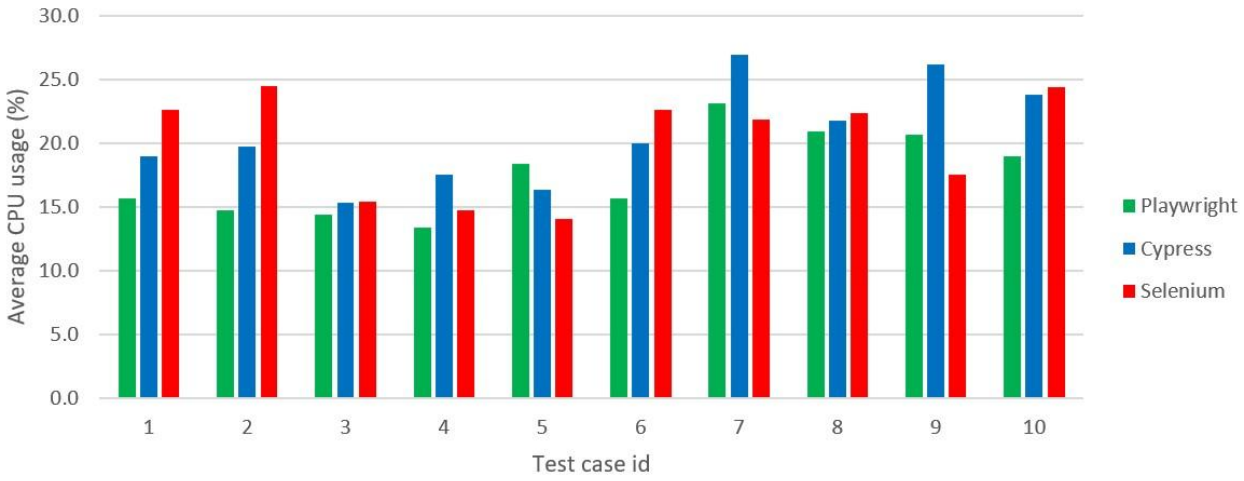


Figure 4: Average CPU usage during test execution.

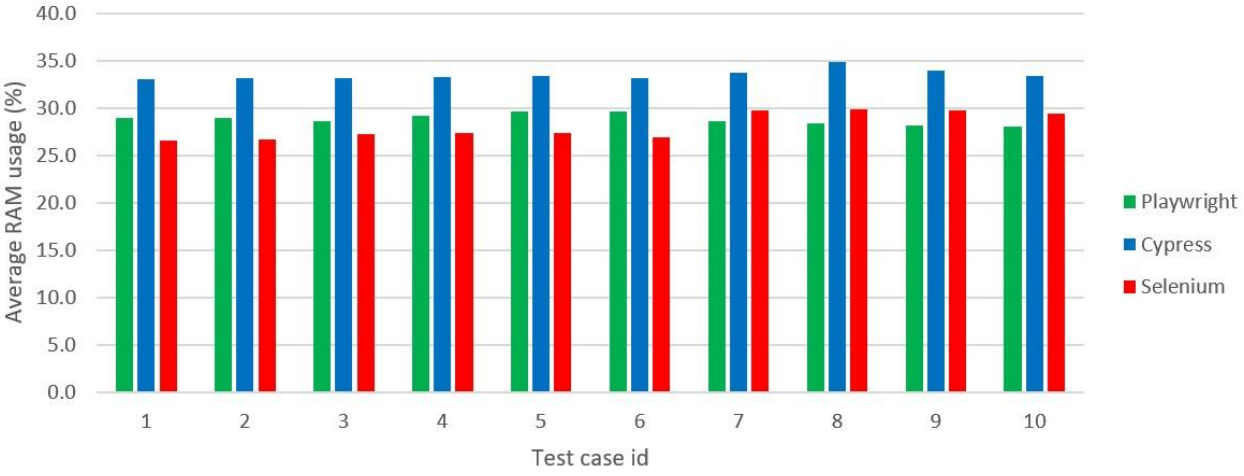
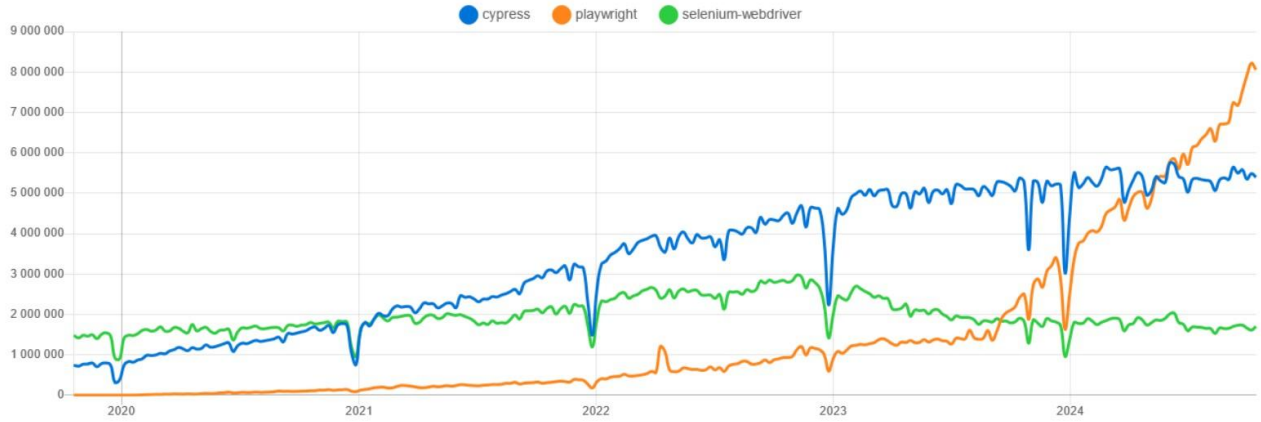


Figure 5: Average RAM usage during test execution.



Stats

	Stars	Issues	Version	Updated	Created	Size
cypress	46 867	1454	13.15.0	a month ago	11 years ago	
playwright	66 440	664	1.48.1	7 days ago	10 years ago	install size 10.3 MB
selenium-webdriver	30 557	255	4.25.0	a month ago	12 years ago	install size 19.5 MB

Figure 6: Number of downloaded packages and GitHub statistics.

Playwright, Cypress and Selenium offer comprehensive documentation to support automated testing activities. Playwright has modern, well-organised documentation making it easy to learn for both beginners and advanced testers. Cypress stands out for its intuitive guides but it could prove to be more challenging for beginners because there are less entry-level guides than Playwright documentation. Selenium offers extensive documentation, but it can seem outdated and complex in part due to its unintuitive structure and very few beginner-friendly guides.

6. Discussion

The main aim of this paper was to investigate the relative effectiveness, efficiency, and flexibility of three prominent web application test automation tools – Selenium, Cypress, and Playwright.

In case of test execution time Playwright has achieved the best result in more than half of prepared test cases (Table 3, Figure 3). Playwright performed best when applied to shorter test cases, however in more complex test cases Playwright performance was more similar to that of Cypress. Selenium, surprisingly, has had the shortest test execution time in the rest of prepared test cases (Table 2).

Both Playwright and Cypress showed similar performance on CPU (Table 4, Figure 4) in most test cases with a slight Playwright advantage, while Selenium achieved significantly higher CPU usage in more than half of executed test cases, highlighting that Cypress's design, which focuses on JavaScript/TypeScript environments, contributes to its efficiency but and Playwright's efficiency especially since it's the newest of the tools analysed in this paper.

Selenium has achieved significantly better results when it comes to RAM usage (Table 5, Figure 5)– the lowest usage percentage of all three tools in more than half all test cases, with Playwright achieving the best result in all other test cases. Cypress has performed the worst in all of the written test cases achieving statistically significant difference in memory usage.

All of the analysed tools share many common characteristics because they are often used in similar testing contexts, however Playwright stands out because of its' support for many different programming languages, which is most apparent when compared to Cypress, although not as many as Selenium. However, when compared to Selenium, Playwright has more different functionalities (Table 6) which could prove to be useful to software testers. Playwright and Cypress are similar in both the setup complexity, which is considered to be simple and intuitive, but also in quality of their documentation. Selenium's setup is more complicated and its documentation is not as intuitive or doesn't offer many beginner-friendly guides, which could discourage less experienced testers from using this automation tool.

In comparison to the findings presented in previous studies, especially to [5] results presented in this paper are similar, Playwright has proven to be superior to Selenium and Puppeteer automation tools in the context of average test case execution speed and performance.

7. Conclusions

In this study the comparative analysis of three test automation tools was realised with use of an empirical study, which consisted of writing and executing test scenarios and reviewing the documentation and functionalities of the analysed test automation tools.

The results support the hypothesis that Playwright would show superior effectiveness and flexibility, making it a strong contender for modern test automation scenarios in the context of web applications. However, the final choice of test automation tool for a project should be made with needs and constraints of that specific project in mind.

Future research could focus on similar comparative analysis of presented tools but place them in different testing context scenarios such as API testing, security testing or load testing to analyse if that change of testing context had an impact on the conclusions presented in this study.

References

[1] S. Butt, S. U. R. Khan, S. Hussain, W. Wang, A conceptual model supporting decision-making for test automation in Agile-based Software Development, Data & Knowledge Engineering 144 (2023) 102–111, https://doi.org/10.1016/j.datak.2022.102111.

[2] Y. Wang, M.V. Mäntylä, Z. Liu, J. Markkula, Test automation maturity improves product quality – Quantitative study of open source projects using continuous integration, Journal of Systems and Software 188 (2022) 1-39, https://doi.org/10.48550/arXiv.2202.04068.

[3] D. Zimmermann, A. Koziolek, GUI-Based Software Testing: An Automated Approach Using GPT-4 and Selenium WebDriver, In 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW) (2023) 171-174, https://doi.org/10.1109/ASEW60602.2023.00028.

[4] B. Jose, Test Automation; A manager's guide, BCS Learning and Development, Swindon, 2021.

[5] D. Zhyhulin, K. Kasian, M. Kasian, Combined method of prioritization and automation of software regression testing, In IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET) (2022) 751-755, https://doi.org/10.1109/TCSET55632.2022.9767034.

[6] S. Gojare, R. Joshi, D. Gaigaware, Analysis and Design of Selenium WebDriver Automation Testing Framework, Procedia Computer Science 50 (2015) 341–346, https://doi.org/10.1016/j.procs.2015.04.038.

[7] E. Pelivani, A. Besimi, B. Cico, A Comparative Study of UI Testing Framework, In 2022 11th Mediterranean Conference on Embedded Computing (MECO) (2022) 1-5, https://doi.org/10.1109/MECO55406.2022.9797165.

[8] A. Utomo, G. Wijaya, Y. Setiawan, Implementation of crowdfunding web application using AWS Amplify architecture with end-to-end testing using Playwright, Indonesian Journal of Multidisciplinary Science 11 (2023) 3888–3904, https://doi.org/10.55324/ijoms.v2i11.620.

[9] R. Angmo, M. Sharma, Performance Evaluation of Web Based Automation Testing Tools, In International Conference on Confluence The Next Generation Information Technology Summit (Confluence) (2014) 731-735, https://doi.org/10.1109/CONFLUENCE.2014.6949287.

[10] R. Gupta, Ultimate Selenium WebDriver for Test Automation, Orange Education Pvt, Delhi, 2024.

[11] W. Mwaura, End-to-End Web Testing with Cypress, Packt Publishing, Birmingham, 2021.

[12] K. Rahman, Science of Selenium, BPB Publications, Noida, 2020.

[13] M. Leotta, F. Ricca, A. Marchetto, D. Olianas, An empirical study to compare three web test automation approaches: NLP-based, programmable, and capture&replay, Journal of Software: Evolution and Process 36(5) (2024) 1-24, https://doi.org/10.1002/smr.2606.

[14] M. Sambamurthy, Test Automation Engineering Handbook, Packt Publishing, Birmingham, 2023.

[15] A. Axelrod, Complete Guide to Test Automation, Apress, Berkeley, 2018.

[16] Open-source web application used in the paper, https://practicesoftwaretesting.com, [29.12.2024].

[17] GitHub repository with source code of the used application, https://github.com/testsmith-io/practice-software-testing, [29.12.2024].

[18] Front page of Visual Studio Code IDE website, https://code.visualstudio.com/, [29.12.2024].

[19] Overview section of technical documentation of Performance Monitor, https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-r2-and-2008/cc749154(v=ws.11), [29.12.2024].

[20] Front page of npm website, https://www.npmjs.com/, [29.12.2024].

[21] Front page of Node.js runtime website, https://nodejs.org/en, [29.12.2024].

[22] Front page of npm trends website, https://npmtrends.com/, [29.12.2024].