Asian Journal of Research in Computer Science

Volume 18, Issue 1, Page 34-44, 2025; Article no.AJRCOS.128913
ISSN: 2581-8260

Comparative Reliability Analysis of Selenium and Playwright: Evaluating Automated Software Testing Tools

Saad Almabruk [a*], Samia Abdalhamid [b] and Tahani Almabruk [b]

[a] Computer Science Department, School of Science, The Libyan Academy, Libya.
[b] Computer Science Department, Faculty of Science, Omar Almukhtar University, Libya.

Authors' contributions

This work was carried out in collaboration among all authors. All authors read and approved the final manuscript.

Article Information

DOI: https://doi.org/10.9734/ajrcos/2025/v18i1546

Open Peer Review History:
This journal follows the Advanced Open Peer Review policy. Identity of the Reviewers, Editor(s) and additional Reviewers, peer review comments, different versions of the manuscript, comments of the editors, etc are available here: https://www.sdiarticle5.com/review-history/128913

Original Research Article

Received: 25/10/2024
Accepted: 29/12/2024
Published: 03/01/2025

ABSTRACT

Aims: This study aims to evaluate and compare the reliability of Selenium and Playwright, two leading frameworks for automated web testing. The assessment focuses on key reliability metrics, including uptime and the Rate of Occurrence of Failures (ROCOF).
Study Design: A comparative experimental study conducted under controlled testing conditions.
Place and Duration of Study: The study was conducted over a 24-hour continuous testing period using two laptops with distinct hardware configurations: an HDD-equipped HP laptop and an SSD-equipped Dell laptop.
Methodology: Reliability was measured using two core metrics: uptime and ROCOF. Tests were conducted on an HP laptop (HDD) and a Dell laptop (SSD). Two Python scripts — one for Selenium

**Corresponding author: E-mail: saadboksheem@gmail.com;*

Cite as: *Almabruk, Saad, Samia Abdalhamid, and Tahani Almabruk. 2025. "Comparative Reliability Analysis of Selenium and Playwright: Evaluating Automated Software Testing Tools". Asian Journal of Research in Computer Science 18 (1):34-44. https://doi.org/10.9734/ajrcos/2025/v18i1546.*

and one for Playwright — were developed to execute identical actions. For the 24-hour uptime test, Selenium ran on HP and Playwright on Dell. ROCOF was assessed at six-time intervals — 8:00 AM, 8:15 AM, 4:00 PM, 4:15 PM, 12:00 AM, and 12:15 AM — by alternating tool execution between HP and Dell, allowing for analysis of the effects of hardware and time of day on failure rates.
Results: Selenium achieved 100% uptime with no failures, while Playwright recorded 99.72% uptime with four downtimes. For ROCOF, both tools had one failure per 10-test sequence, but Selenium's higher failure rate per second (0.1208 on HP, 0.1336 on Dell) was due to faster execution times (7.93s on HP, 7.87s on Dell) compared to Playwright (36.74s on HP, 35.84s on Dell). The SSD-equipped Dell laptop outperformed the HDD-based HP, with faster completion times (43.71s vs. 44.67s).
Conclusion: Selenium is ideal for scenarios requiring uninterrupted uptime, while Playwright's consistent response times suit dynamic web application testing. The study highlights hardware's role in performance, with SSDs offering superior speed and stability. These findings guide practitioners in choosing tools based on hardware, stability, and execution needs.

Keywords: Software testing; reliability; selenium; playwright.

1. INTRODUCTION

Automated testing has become a cornerstone of modern software development, facilitating faster releases, minimizing manual effort, and enhancing test coverage. Among the various tools available, Selenium and Playwright stand out as two of the most widely used frameworks for web application testing. Selenium, with its long-standing presence and broad browser compatibility, remains a trusted industry standard. In contrast, Playwright is a more recent entrant, providing cross-browser testing with superior performance and support for modern web technologies. The development of these frameworks reflects the increasing complexity of web applications and the growing need for efficient, dependable testing solutions. Maintaining the reliability of these tools is critical for their integration into software development workflows, as testing interruptions or failures can hinder development progress and delay product releases.

Despite their widespread use, there is a lack of empirical research directly comparing the reliability of Selenium and Playwright under controlled conditions. In this context, reliability refers to the tool's ability to deliver consistent uptime, minimize test failures, and maintain stable execution. Selenium is often favored for its maturity and strong community support, while Playwright's modern web compatibility and faster execution present a compelling alternative. This creates a dilemma for developers and testers when choosing the most appropriate tool for their needs. The decision is further complicated by hardware-related factors, as variations in system architecture (e.g., SSD vs. HDD) can influence the performance and stability of both frameworks.

This study seeks to fill the research gap by providing a detailed comparative analysis of Selenium and Playwright, with a primary emphasis on reliability metrics. It evaluates key indicators such as uptime and Rate of Occurrence of Failures (ROCOF) to offer a data-driven perspective on the strengths and limitations of each tool. The analysis also considers the impact of hardware variations by assessing the performance of both frameworks on two distinct systems: an HP laptop with an HDD and a Dell laptop with an SSD. This dual-system approach ensures a comprehensive evaluation of the tools' reliability across different hardware environments.

Automated software testing is heavily influenced by tools like Selenium and Playwright, both of which are integral to modern development practices. Selenium, as one of the earliest open-source automation frameworks, has a well-established ecosystem backed by a large developer community. Altaf et al. (2015) detail Selenium's evolution from Selenium Core to WebDriver, which introduced essential features like cross-browser compatibility, support for multiple programming languages, and parallel test execution via Selenium Grid. However, limitations persist, such as its inability to manage Windows-based applications and the lack of a built-in reporting system (Altaf & Rafiq, 2015). To address these gaps, Gojare et al. (2015) proposed a structured framework that incorporates an object repository and modular utilities to improve reusability and scalability in testing (Gojare & Gaigaware, 2015). Biju & Ali

(2020) demonstrated the creation of a hybrid framework by integrating Selenium with Visual Studio and TestNG, leveraging the Page Object Model (POM) to enhance maintainability and streamline testing processes (Biju & Ali, 2020).

In contrast to Selenium, Playwright, a more recent addition to the testing landscape, addresses key limitations of its predecessor. Notable features include auto-wait functionality to reduce flaky tests and a modern architecture optimized for handling dynamic content. Wellner (2023) highlights Playwright's dynamic synchronization, which outperforms Selenium's static approach, enabling faster test execution and reducing maintenance costs (Wellner, 2024). Melyawati et al. (2024) further report that Playwright executes tests nearly twice as fast as Selenium for dynamic web applications (Melyawati, & Sudipa, 2024). dos Santos Marques (2023) emphasizes Playwright's seamless integration with CI/CD pipelines, streamlining test approvals and enhancing efficiency within DevOps workflows (dos Santos Marques, 2023).

Comparative studies highlight distinct strengths and weaknesses between Selenium and Playwright. Brahmbhatt (2023) notes that while Selenium excels in certain test execution scenarios, Playwright's advanced handling of dynamic elements, floating buttons, and automatic script synchronization provides an advantage for modern web applications (Brahmbhatt, 2023). Pasławski & Pańczyk (2024) further emphasize Playwright's superiority in headless execution, reporting faster execution times and better CPU efficiency compared to Selenium (Pasławski & Pańczyk, 2024). Blanc et al. (2022) add another dimension, pointing out that GUI test frameworks like Playwright may not fully replicate real-user interactions with the system under test (SUT), which can pose limitations for specific test cases (Blanc & Falleri, 2022).

Although Playwright offers several advantages, it remains in an earlier stage of development compared to Selenium. Its modern, streamlined API is still evolving, and community support is less mature than Selenium's well-established network. García et al. (2020) recognize Playwright as a promising alternative but emphasize Selenium's continued dominance as a stable and reliable solution (García, & Munoz-2020). Despite this, Metin (2023) illustrates Playwright's potential in testing Graphical Language Server Protocol (GLSP) web modeling tools, highlighting its ability to support semantic interactions and multi-environment testing (Metin, 2023). Khan (2023) further underscores Playwright's adaptability, showcasing its use in automated web scraping, where it enables scalability and real-time data updates — features that are more difficult to achieve with Selenium (Khan, 2023).

Numerous studies have compared several automation testing tools to assess their capabilities and limitations. Zhyhulin et al. (2022) compared Playwright with Selenium and Puppeteer, concluding that Playwright offered faster execution times. Nevertheless, their study was limited to basic tests and a simple system under test (SUT). In contrast, the current research evaluates these tools on a mature, real-world web application with diverse and complex testing scenarios (Zhyhulin, et al., 2022). Pelivani and Cico (2021) conducted a comparison between Selenium and Katalon Studio. Their findings revealed that Selenium surpassed Katalon in execution efficiency due to Katalon's reliance on Groovy, which necessitates extensive library loading. However, Katalon was recognized for its ease of setup and user-friendly reporting capabilities (Pelivani & Cico, 2021).

The scope of this work is a direct comparison of the reliability of Selenium and Playwright under controlled conditions. The main underlying measures of reliability include the following: (1) Uptime-the proportion of time for which the system has been up during a test in a 24hour and (2) the ROCOF, estimating the number of failures occurring during a test execution The hardware aspect, concerning the use of one HDD from HP and an SSD by Dell, will help illustrate how each tool's reliability depends on system architecture. Given such a focused area of research, this article has thus provided useful advice to be used by a software tester, quality assurance engineer, or development team to refine and optimize testing flows. This is justified because this work sits within the landscape of continuous integration/continuous deployment pipelines, which heavily rely on test reliability for development velocity and release timelines.

2. METHODOLOGY

2.1 Tool Selection Criteria

Selenium and Playwright were selected for this study due to their distinct capabilities and strong presence in the software testing domain.

Selenium, a veteran in automated testing, is known for its extensive browser support and compatibility with legacy systems, making it a widely used industry benchmark. Playwright, in contrast, introduces modern features like automatic browser context handling and support for contemporary web technologies, with seamless cross-browser automation across Chromium, WebKit, and Firefox. These contrasting attributes enabled a comprehensive assessment of reliability across a variety of real-world testing scenarios.

2.2 Evaluation Metrics

The assessment of reliability was based on two key metrics: uptime, which reflects the continuous availability of the target webpage during testing, and the rate of occurrence of failure, which tracks the frequency of errors or failures encountered while interacting with both dynamic and static web elements.

2.3 Experimental Setup

To assess the performance and reliability of Selenium and Playwright, testing was carried out on two laptops with distinct hardware specifications. This strategy enabled a comprehensive analysis of the extent to which hardware differences influence the operational efficiency of each framework.

2.3.1 Hardware Configuration

i. HP Laptop
- Processor: Intel Core i7-8550U CPU
- RAM: 12 GB
- Graphics: NVIDIA GeForce MX 130
- Storage: 930 GB HDD
- Operating System: Windows 10 Pro

ii. Dell Latitude 3400
- Processor: Intel Core i7-8565U CPU
- RAM: 12 GB
- Graphics: NVIDIA GeForce MX 130
- Storage: 256 GB SSD
- Operating System: Windows 10 Pro

The selected hardware configurations were intended to analyze the impact of storage type (HDD vs. SSD) and system architecture on the performance of Selenium and Playwright. It was anticipated that the SSD in the Dell Latitude 3400 would offer faster data access times compared to the HDD in the HP laptop, potentially leading to improved test execution speed and enhanced system responsiveness
.

2.3.2 Software environment

- Both laptops operated on Windows 10 Pro to maintain consistency in the testing environment. Browser versions were standardized across systems to prevent variability in performance caused by software differences. For the tests, the stable Chrome browser Version 130.0.6735.44 was used, ensuring uniformity in browser behavior and rendering. Furthermore, automatic browser updates were disabled during the testing period to eliminate discrepancies due to version changes.
- Development Environment: Python 3.x was used to execute scripts for both Selenium and Playwright, maintaining uniformity across the two frameworks for accurate performance comparison.

2.3.3 Test Implementation

Two Python scripts, one for Selenium and one for Playwright, were developed to execute identical sequences of actions, ensuring a consistent basis for comparison. These scripts were designed to perform key actions commonly encountered in web application testing, including:

- Homepage Load Time: Measuring the time taken to fully load the homepage.
- Navigation Link Response Times: Assessing the response times for clicking and loading navigation links.
- Image Loading Performance: Evaluating the efficiency and speed of rendering images on the page.

These actions were selected to simulate typical user interactions and to evaluate the frameworks' capabilities in handling both static and dynamic web elements. The scripts were executed in repeated cycles over a 24-hour period, with detailed logs capturing metrics such as response times, failures, and execution stability.

2.4 Test Scenarios

The test scenario of this study focused on assessing Selenium and Playwright reliability of the target webpage (https://nclc.ly/), specifically uptime and the ROCOF.

The uptime analysis was conducted over a 24-hour continuous testing period. During this experiment, Selenium was executed on an HP laptop, while Playwright was run on a Dell laptop. This configuration allowed for a comparative evaluation of uptime performance under distinct hardware conditions.

To measure the ROCOF, the role of hardware differences was explicitly considered. By alternating the execution of Selenium and Playwright on HP (HDD) and Dell (SSD) laptops, the study aimed to explore how system architecture impacts failure rates and overall reliability. This approach provided a deeper understanding of the potential influence of hardware on each tool's performance.

The rate of occurrence of failures was measured at specific time intervals throughout the day to evaluate the impact of time-based conditions on system stability. Testing was conducted at the following time slots:

- 8:00 AM (Selenium on HP, Playwright on Dell)
- 8:15 AM (Selenium on Dell, Playwright on HP)
- 4:00 PM (Selenium on HP, Playwright on Dell)
- 4:15 PM (Selenium on Dell, Playwright on HP)
- 12:00 AM (Selenium on HP, Playwright on Dell)
- 12:15 AM (Selenium on Dell, Playwright on HP)

This staggered approach ensured a comprehensive evaluation of system performance across peak, off-peak, and nighttime conditions.

3. RESULTS AND DISCUSSION

This study offers a comprehensive evaluation of the reliability of Selenium and Playwright under controlled testing conditions. The assessment is guided by key metrics, including uptime and the rate of occurrence of failures. Data was gathered through continuous monitoring of test executions on two hardware configurations: an HP laptop with an HDD and a Dell laptop with an SSD. This setup enabled an assessment of how hardware differences impact the performance of both frameworks.

This section provides a detailed analysis of the results, focusing on differences in uptime and failure rates between the two tools. The influence of hardware architecture on performance is also examined to highlight the role of storage type (HDD vs. SSD) in shaping the tools' reliability. By systematically comparing Selenium and Playwright, the study delivers essential insights into their strengths and limitations, offering practical guidance for testers and developers in selecting the most appropriate tool for automated testing workflows.

3.1 Uptime

Table 1 displays uptime metrics for Selenium and Playwright, monitored over a 24-hour period. The analysis focuses on assessing the reliability, response times, and overall performance of each tool.

To enable a more comprehensive analysis, the data is further elaborated in Table 2 and Table 3 offering deeper insights into the frameworks' performance. This progression allows for a more detailed examination of how system architecture, including differences in hardware configurations,

Table 1. Uptime performance of Selenium vs Playwright

Metric	Selenium	Playwright
Monitoring Duration	1 day, 0:18:05	1 day, 0:17:49
Total Checks Performed	1440	1440
Successful Checks (Website UP)	1440	1436
Failed Checks (Website DOWN)	0	4
Website Uptime	100.00%	99.72%
Website Downtime	0.00%	0.28%
Average Response Time	0.75 seconds	0.71 seconds
Response Time Standard Deviation	1.29 seconds	0.68 seconds
Maximum Response Time	30.53 seconds	9.54 seconds
Minimum Response Time	0.44 seconds	0.43 seconds

affects the reliability and responsiveness of Selenium and Playwright under controlled testing conditions.

3.1.1 Uptime and reliability

This subsection evaluates the uptime of Selenium and Playwright using the performance metrics provided in Table 2. Uptime measures the percentage of time a system remains operational. The analysis focuses on key indicators, including total checks performed, successful checks, failed checks, and the resulting uptime percentage for both frameworks. By highlighting differences in these metrics, the comparison provides insights into each tool's ability to support uninterrupted testing sessions and maintain system stability.

The data presented in Table 2 provides a clear comparison of the reliability of Selenium and Playwright, with a particular emphasis on uptime performance. Selenium demonstrated superior reliability, achieving a flawless 100% uptime across 1,440 total checks with zero failed checks. In contrast, Playwright recorded a slightly lower uptime of 99.72%, which was affected by four failed checks out of the same total of 1,440 checks. Despite the minimal downtime of 0.28% for Playwright, this small difference in reliability could be critical in contexts where uninterrupted system availability is essential. Both tools operated under nearly identical monitoring durations, which underscores the robustness of Selenium's performance. The absence of failed checks in Selenium's execution reflects its higher reliability for sustained, long-term testing without interruption. Playwright, while still highly reliable, may require additional measures to mitigate the impact of brief system downtimes, especially in environments where continuous availability is paramount.

3.1.2 Failure and recovery analysis

This subsection analyzes the failure and recovery performance of Selenium and Playwright using the metrics presented in Table 3. The analysis focuses on two key indicators: slow responses (response times exceeding 3.34 seconds) and downtime occurrences. Table 3 reveals that Selenium recorded 19 instances of slow responses, indicating occasional delays during test execution, but it maintained zero downtime, demonstrating high system availability. In contrast, Playwright experienced four downtime occurrences, reflecting brief periods of system unavailability, but no slow response data was recorded.

Table 2. Comparison of uptime between selenium and playwright in regard of and reliability

Metric	Selenium	Playwright	Observations
Monitoring Duration	1 day, 0:18:05	1 day, 0:17:49	Monitoring durations are nearly identical.
Total Checks Performed	1440	1440	Both tools executed the same number of checks.
Successful Checks	1440	1436	Selenium had no failures; Playwright missed 4 checks.
Failed Checks	0	4	Selenium had perfect uptime, while Playwright had 4 failures.
Website Uptime (%)	100.00%	99.72%	Selenium achieved perfect uptime; Playwright had slight downtime.
Website Downtime (%)	0.00%	0.28%	Playwright's downtime was minimal but present.

Table 3. Comparison of uptime between selenium and playwright in regard to failure and recovery analysis

Metric	Selenium	Playwright	Observations
Slow Responses (>3.34 seconds)	19	N/A	Selenium recorded some slow responses; no data for Playwright.
Downtime Occurrences	None	4	Playwright encountered four downtime occurrences; Selenium had none.

This table illustrates that Selenium experienced 19 instances of slow responses, each exceeding 3.34 seconds, while Playwright did not have any recorded slow responses during the observation period. However, in terms of downtime occurrences, Selenium maintained a perfect record with zero downtime, whereas Playwright encountered four separate instances of downtime. This data highlights a trade-off in reliability; Selenium may provide continuous availability but could be prone to slower response times, while Playwright offers consistent response speeds but is more susceptible to system downtimes.

This section analyzes the ROCOF for Selenium and Playwright, focusing on the frequency of failures during test execution. The ROCOF serves as a quantitative measure of system reliability, reflecting the rate at which failures occur within a specific time frame. Table 4 presents the ROCOF performance of Selenium and Playwright across different experimental timeframes and hardware setups, offering a comparison of each tool's behavior on HP and Dell laptops. To enhance the analysis, Fig. 1 visually illustrates the ROCOF trends for both frameworks across various time slots, highlighting disparities in failure rates and emphasizing Selenium's higher failure rate compared to Playwright.

The analysis presented in Table 4 highlights the failure rates and stability metrics of Selenium and Playwright, offering key insights into their reliability during test execution.

Table 4. ROCOF performance of selenium vs playwright

Experiment Time	Laptop	Tool	ROCOF (10 Tests)			
			Total Failures	Total Time (s)	Failures per second	Failures per test
8:00 am	HP	Selenium	1	8.64	0.1158	0.1000
8:00 am	Dell	Playwright	1	34.28	0.0292	0.1000
8:15 am	HP	Playwright	1	37.77	0.0265	0.1000
8:15 am	Dell	Selenium	1	9.14	0.1094	0.1000
4:00 pm	HP	Selenium	1	7.24	0.1382	0.1000
4:00 pm	Dell	Playwright	1	35.24	0.0284	0.1000
4:15 pm	HP	Playwright	1	35.95	0.0278	0.1000
4:15 pm	Dell	Selenium	1	6.44	0.1552	0.1000
12:00 am	HP	Selenium	1	8.30	0.1205	0.1000
12:00 am	Dell	Playwright	1	36.90	0.0271	0.1000
12:15 am	HP	Playwright	1	36.52	0.0274	0.1000
12:15 am	Dell	Selenium	1	6.26	0.1598	0.1000

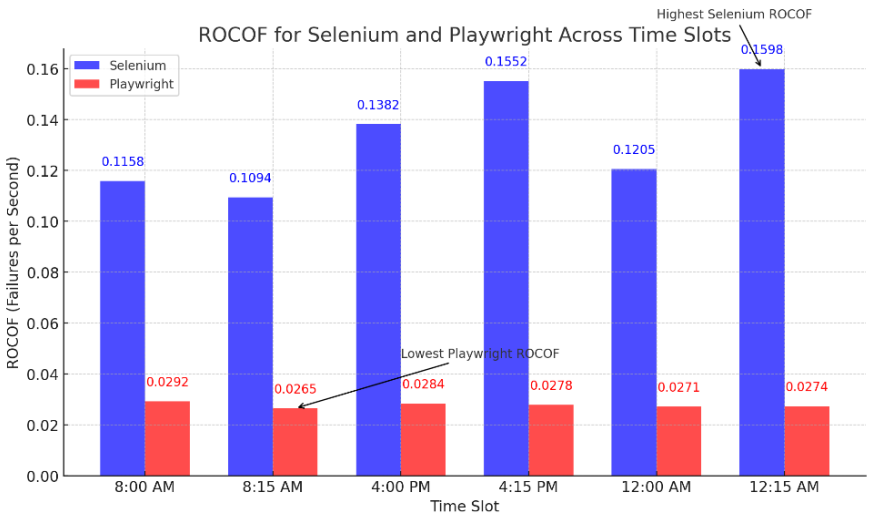

Fig. 1. Bar chart represents the ROCOF for Selenium and Playwright t across various time slots

To provide a more detailed understanding, the data is further divided into three key areas: failures and test stability (3.2.1), performance by laptop (3.2.2), and performance by experiment time (3.2.3). This structured breakdown allows for a more granular comparison of how hardware configurations and testing time slots impact the stability and performance of both frameworks under controlled testing conditions.

3.1.3 Failures and test stability

The analysis of failures and test stability is based on the data presented in Table 5, which compares the rate of occurrence of failures for Selenium and Playwright. Understanding the nature and frequency of failures provides direct insight into system robustness. This data allows for the identification of patterns and root causes, which are essential for improving software reliability.

This subsection evaluates key metrics such as total failures and failure rates for each framework, focusing on their consistency across multiple test runs. As shown in Table 5, both tools record an equal number of failures per test, with failures occurring at a steady rate of 0.1 failures per test. These findings indicate that Selenium and Playwright exhibit similar levels of stability in handling failures, offering important insights into their reliability under controlled testing conditions.

The data presented in Table 5 emphasizes the stability of Selenium and Playwright with respect to their Rate of Occurrence of Failure (ROCOF). Both frameworks exhibit identical stability metrics, with each recording a failure rate of 0.1 failures per test and one failure per test run. This indicates that neither Selenium nor Playwright holds a distinct advantage over the other in terms of overall test stability. The uniformity of their failure rates suggests that both frameworks offer consistent and predictable performance across multiple test executions.

This level of stability is particularly significant for software testing frameworks, as predictability and repeatability are essential for effective debugging and quality assurance. Stable failure patterns enable testers to identify, analyze, and address issues with greater precision. By providing a consistent testing environment, Selenium and Playwright support a structured approach to quality control, thereby reducing the uncertainty associated with random or sporadic test failures.

3.1.4 Performance by laptop

The analysis of Table 6 examines the performance of Selenium and Playwright across two hardware configurations: HP and Dell laptops. Since hardware differences (like SSDs vs. HDDs) can affect system responsiveness, dividing the analysis by laptop configuration highlights how hardware impacts the reliability of the testing process. This insight is crucial for practitioners who must select suitable hardware for their testing environments.

This subsection evaluates key metrics, including average test completion time and failure rate per second for each tool on both devices. As shown in Table 6, Selenium consistently achieves faster completion times on both laptops, with slightly better performance on the Dell laptop. However, this speed advantage is offset by a higher failure rate per second compared to Playwright. In contrast, Playwright exhibits longer test completion times but maintains a more stable and consistent failure rate across both laptop configurations. This analysis highlights the influence of hardware architecture (SSD vs. HDD) on the performance and stability of the two testing frameworks.

The data presented in Table 6 highlights notable differences in the performance and stability of Selenium and Playwright when tested on two distinct hardware configurations: HP laptops with HDDs and Dell laptops with SSDs. Selenium consistently demonstrated faster completion times on both devices, averaging 7.93 seconds

Table 5. Comparison of ROCOF between selenium and playwright in regard of stability

Metric	Selenium	Playwright	Observations
Total Failures	1 per test	1 per test	Both tools exhibit an equal number of failures per test.
Failures per Test	0.1000	0.1000	Failures are consistent across all tests.

Table 6. Comparison of ROCOF between selenium and playwright in regard of performance by laptop

Laptop	Tool	Average Total Time (s)	Average Failures per Second	Observations
HP	Selenium	7.93	0.1208	Consistently faster completion times.
HP	Playwright	36.74	0.0272	Longer completion times but lower failure rates.
Dell	Selenium	7.87	0.1336	Faster completion times with slightly higher failure rates.
Dell	Playwright	35.84	0.0279	Slower completion times but consistent failure rates.

on HP and 7.87 seconds on Dell. This performance indicates Selenium's superior speed relative to Playwright, which recorded completion times of 36.74 seconds on HP and 35.84 seconds on Dell.

However, the trade-off for Selenium's speed advantage was a higher failure rate. On average, Selenium's failure rate per second was 0.1208 on HP and 0.1336 on Dell, reflecting an increase in execution errors. By contrast, Playwright exhibited significantly lower failure rates, with 0.0272 on HP and 0.0279 on Dell. This finding indicates that Playwright offers greater performance stability, even if its execution speed is slower.

The comparison underscores a fundamental trade-off between speed and stability. Selenium's strength lies in its rapid execution, but this comes at the expense of increased failure rates. Conversely, Playwright provides more predictable and consistent performance, though its execution times are notably longer. The influence of hardware on performance is also evident, as the use of SSDs (Dell) improved completion times for both Selenium and Playwright relative to HDDs (HP). This demonstrates the critical role hardware plays in optimizing the performance of software testing frameworks.

3.1.5 Performance by experiment time

Table 7 presents the time-based ROCOF analysis for Selenium and Playwright. Examining how the time of testing affects reliability (for example, due to system load, network latency, or scheduling effects) uncovers temporal factors that could influence software behavior. This provides a more holistic understanding of system reliability in real-world, time-varying conditions.

The analysis in this study was conducted at six distinct time intervals: 8:00 AM, 8:15 AM, 4:00 PM, 4:15 PM, 12:00 AM, and 12:15 AM, with tool execution alternating between HP (HDD) and Dell (SSD) laptops. This staggered testing schedule allowed for a detailed comparison of performance across peak and off-peak hours, shedding light on how system load and time-of-day influence failure rates.

Table 7. Comparison of ROCOF between Selenium and Playwright in regard of performance by experiment time

Time Slot	Selenium Total Time (s)	Playwright Total Time (s)	Key Observations
8:00 AM	8.64	34.28	Selenium is significantly faster.
8:15 AM	9.14	37.77	Similar trend with Selenium outperforming Playwright.
4:00 PM	7.24	35.24	Selenium maintains faster performance during peak hours.
4:15 PM	6.44	35.95	Dell-Selenium achieves the fastest performance here.
12:00 AM	8.30	36.90	Nighttime results favor Selenium for speed.
12:15 AM	6.26	36.52	Selenium exhibits its fastest results.

The analysis of the data presented in Table 7 underscores the impact of testing time on the rate of occurrence of failures and the overall performance of Selenium and Playwright. Across all observed time slots, Selenium consistently demonstrated faster completion times compared to Playwright. Notably, Selenium's optimal performance was observed at 12:15 AM, where it achieved a completion time of 6.26 seconds. In contrast, Playwright recorded a significantly slower completion time of 36.52 seconds during the same time slot.

Playwright exhibited remarkable stability in its execution times, which ranged from 34.28 to 37.77 seconds across all time slots. This indicates that Playwright's performance remains stable regardless of temporal variations. Selenium, however, displayed greater variability, with completion times ranging from 6.26 to 9.14 seconds. This variation suggests that Selenium is more susceptible to time-dependent factors such as system load or hardware scheduling.

These findings underscore the key distinction between the two frameworks. Playwright's consistency in execution times makes it a more predictable option for environments where stability and uniformity are paramount. Conversely, Selenium's capacity for faster execution, particularly during off-peak hours such as 12:15 AM, positions it as a suitable choice for scenarios where time-sensitive testing is required. The balance between execution speed and predictability should therefore inform the selection of the appropriate testing framework based on specific operational priorities.

4. CONCLUSION

This study provides a comprehensive evaluation of the performance of Selenium and Playwright, with a focus on key reliability metrics such as uptime and rate of occurrence of failure. The analysis reveals that Selenium outperforms Playwright in terms of uptime, achieving a flawless 100% availability with no service interruptions. In contrast, Playwright attained a 99.72% uptime, with four instances of downtime. While Selenium demonstrated impeccable availability, Playwright exhibited greater stability by maintaining consistent response times with no instances of slow execution. Conversely, Selenium encountered 19 instances where its execution time exceeded 3.34 seconds. The influence of hardware on performance was also examined. The use of solid-state drives (SSDs), particularly Dell models, significantly enhanced the speed and stability of both tools when compared to hard disk drives (HDDs), such as those from HP. While Selenium demonstrated faster task completion, this speed came at the expense of higher failure rates. In contrast, Playwright delivered consistent execution times with minimal performance fluctuations, thereby offering greater operational stability.

The study also investigated time-based performance. It was observed that Selenium's execution speed varied depending on the time of day, with optimal performance occurring at 12:15 AM. In contrast, Playwright maintained consistent execution speeds throughout the day, unaffected by time-based variations. This consistency positions Playwright as a more predictable option for time-sensitive test scheduling.

In conclusion, Selenium is the preferred choice for testing environments where uninterrupted uptime and system availability are critical. On the other hand, Playwright is better suited for scenarios that prioritize stable response times and execution consistency. Each framework offers distinct advantages, and the selection between them should be guided by factors such as system requirements, hardware configurations, and the specific demands of the testing schedule. Future research could provide valuable insights by exploring the scalability of these frameworks for mobile and API testing, as well as their performance under varied network conditions and in cloud-based testing environments.

DISCLAIMER (ARTIFICIAL INTELLIGENCE)

Author(s) hereby declare that NO generative AI technologies such as Large Language Models (ChatGPT, COPILOT, etc.) and text-to-image generators have been used during the writing or editing of this manuscript.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

Altaf, I., Dar, J. A., ul Rashid, F., & Rafiq, M. (2015). *Survey on selenium tool in software testing.* Paper presented at the

2015 *International Conference on Green Computing and Internet of Things (ICGCIoT).*

Biju, V., & Ali, S. (2020). Automation of Purchase Order in Microsoft Dynamics 365 by Deploying Selenium. *Computer Science & Information Technology (CS & IT), 10*(4), 101-114.

Blanc, X., Degueule, T., & Falleri, J.-R. (2022). Diffing e2e tests against user traces for continuous improvement.

Brahmbhatt, K. H. (2023). Comparative analysis of selecting a test automation framework for an e-commerce website. *Tallinn University of Technology*.

dos Santos Marques, P. T. (2023). Optimization ofapprovaltimein webuitests.

García, B., Gallego, M., Gortázar, F., & Munoz-Organero, M. (2020). A survey of the selenium ecosystem. *Electronics, 9*(7), 1067.

Gojare, S., Joshi, R., & Gaigaware, D. (2015). Analysis and design of selenium webdriver automation testing framework. *Procedia Computer Science, 50*, 341-346.

Khan, A. (2023). Design and implementation of an automated web scraping system: enhancing accuracy and efficiency for Nettileasing Finland Oy.

Melyawati, N. L. P., Asana, I. M. D. P., Putri, N. W. S., Atmaja, K. J., & Sudipa, I. G. I. (2024). Comparison of Automation Testing On Card Printer Project Using Playwright And Selenium Tools. *Journal of Computer Networks, Architecture and High Performance Computing, 6*(3), 1309-1320.

Metin, H. (2023). *Testing of GLSP-based web modeling tools.* Technische Universität Wien,

Pasławski, P., & Pańczyk, M. (2024). Comparison of selected tools for automation testing of Web applications. *Journal of Computer Sciences Institute, 31*, 145-150.

Pelivani, E., & Cico, B. (2021). *A comparative study of automation testing tools for web applications.* Paper presented at the 2021 10th Mediterranean Conference on Embedded Computing (MECO).

Wellner, C. J. (2024). Comparing Static and Dynamic Synchronization of GUI-based tests: An Industrial study. In.

Zhyhulin, D., Kasian, K., & Kasian, M. (2022). *Combined method of prioritization and automation of software regression testing.* Paper presented at the 2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET).

Peer-review history:
The peer review history for this paper can be accessed here:
https://www.sdiarticle5.com/review-history/128913