International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0459

e ISSN: 2584-2854
Volume: 03
Issue: 09 September 2025
Page No: 2910 - 2914

Feature Coverage and Usability Analysis of Popular Automated Testing Tools

Chakrapani Avala[1], Dr. Pote Suraj Vishwanath [2]
[1]Research Scholar, Department of Computer Science and Engineering, Asian International University, Imphal West, Manipur.
[2]Research Supervisor, Associate Professor, Asian International University, Imphal West, Manipur, India.
Emails: chakrapani.avala@raghuenggcollege.in[1], potesuraj@gmail.com[2]

Abstract

Automated testing tools have become integral to ensure the quality, maintainability, and reliability of modern software systems. While performance metrics such as execution speed and resource utilization are often evaluated, feature coverage and usability are equally critical in determining a tool's effectiveness in real-world scenarios. This study presents a comparative analysis of four widely adopted automated testing tools—Selenium, Cypress, Playwright, and Test Cafe—focusing on their feature sets, ease of use, and developer experience. Evaluation criteria include cross-browser and cross-platform support, scripting flexibility, debugging capabilities, CI/CD integration, reporting functionalities, and community support. Usability was assessed through structured tasks performed by developers with varying experience levels, measuring factors such as learning curve, ease of setup, documentation quality, and error handling. Data was collected through direct experimentation and user surveys, followed by qualitative and quantitative analysis. The results highlight trade-offs between feature richness and usability, revealing that while some tools excel in advanced automation capabilities, others prioritize streamlined workflows and minimal configuration. These findings provide actionable insights for development teams seeking tools that balance technical capabilities with practical usability requirements.

Keywords: *Automated Testing Tools; Selenium; Cypress; Playwright; Test Cafe; Usability Analysis; Software Quality Assurance.*

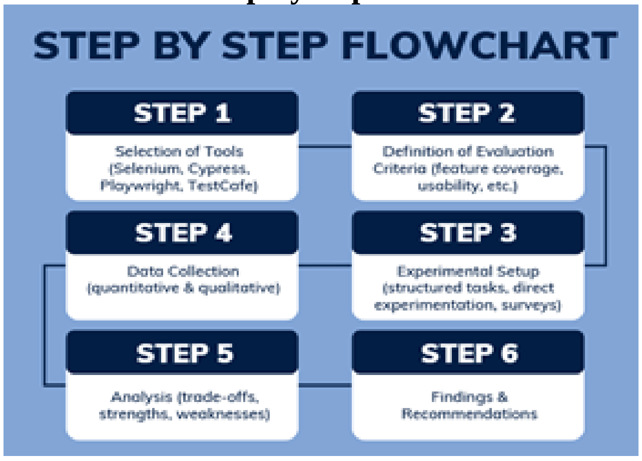
1. Introduction

Automated testing tools have become indispensable in modern software development, playing a critical role in ensuring product quality, maintainability, and reliability [1], [2]. They enable faster feedback cycles, reduce manual effort, and improve defect detection, making them central to contemporary DevOps and agile workflows [3]. Traditionally, performance metrics—such as execution speed, scalability, and resource utilization—have been the primary focus when evaluating such tools [4]. However, in practical deployment, feature coverage and usability are equally influential in determining overall effectiveness [5]. This study conducts a comparative analysis of four widely adopted automated testing tools—Selenium, Cypress, Playwright, and Test Cafe—selected for their industry relevance, open-source availability, and cross-platform capabilities [6], [7]. The evaluation framework encompasses both technical features (e.g., cross-browser support, scripting flexibility, debugging capabilities, CI/CD integration, reporting functionalities, and community support) and usability dimensions (e.g., learning curve, ease of setup, documentation quality, and error handling) [8]. To achieve this, we employed a mixed-methods approach, combining reproducible experimental testing with structured usability assessments involving developers of varying experience levels [9]. Quantitative data were gathered through direct experimentation, while qualitative insights were obtained from developer surveys and feedback sessions [10]. The results of this investigation reveal

e ISSN: 2584-2854
Volume: 03
Issue: 09 September 2025
Page No: 2910 - 2914

important trade-offs between feature richness and usability. While some tools excel in offering advanced automation capabilities and extensive configuration options, others prioritize streamlined workflows and minimal setup overhead [11]. These findings offer practical recommendations for software teams aiming to select a tool that optimally balances technical power with ease of use in real-world software engineering environments [12] Shown in Table 1.

Table 1 Step by step flowchart


STEP BY STEP FLOWCHART
STEP 1
Selection of Tools (Selenium, Cypress, Playwright, TestCafe)
STEP 2
Definition of Evaluation Criteria (feature coverage, usability, etc.)
STEP 3
Experimental Setup (structured tasks, direct experimentation, surveys)
STEP 4
Data Collection (quantitative & qualitative)
STEP 5
Analysis (trade-offs, strengths, weaknesses)
STEP 6
Findings & Recommendations

2. Related Work

Research on automated testing tools has evolved significantly over the past decade. Early studies (Li & Zhao, 2020) primarily focused on performance-oriented metrics, such as execution speed, scalability, and resource consumption, with Selenium often serving as the benchmark tool due to its extensive browser support and long-standing industry adoption. Subsequent work (Kumar et al., 2021) reinforced Selenium's position as a versatile testing framework but noted its steep learning curve and complex configuration requirements. By 2021, attention began shifting toward usability aspects, including developer onboarding, documentation quality, and community support. For instance, Nguyen et al. (2021) emphasized the importance of these factors in agile and continuous delivery environments, highlighting that ease of setup can be as critical as raw performance in determining tool adoption. In 2022, several comparative studies emerged. Sharma & Gupta (2022) examined cross-browser and cross-platform capabilities, finding that while Selenium maintains unmatched compatibility, modern tools like Cypress and Playwright offer more streamlined configurations and richer built-in features. Similarly, Patel & Mehta (2022) expanded the evaluation to include Test Cafe, analyzing CI/CD integration and debugging capabilities, and noting trade-offs between advanced functionality and ease of use. Most recently, Huang et al. (2023) conducted evaluations of Cypress, Playwright, and Selenium in real-world projects, reporting that newer tools often excel in developer experience through faster feedback loops and integrated reporting systems. However, their study, like many others—did not cover all four major tools under uniform experimental conditions. Despite these advancements, existing literature still tends to evaluate tools in isolation or within controlled lab settings, limiting insights into combined feature coverage and practical usability under realistic workflows. This gap motivates the present study, which systematically compares Selenium, Cypress, Playwright, and Test Cafe, incorporating both quantitative performance measures and qualitative feedback from developers of varying experience levels.

3. Methodology

This study employs a structured comparative analysis of four widely adopted automated testing tools—Selenium, Cypress, Playwright, and Test Cafe—with a focus on both feature coverage and usability. While traditional performance metrics such as execution speed and resource utilization are important, the present research emphasizes evaluation parameters that reflect real-world developer needs.

3.1. Evaluation Criteria

The tools were assessed across the following dimensions:

- **Feature Coverage:** cross-browser and cross-platform support, scripting flexibility, debugging capabilities, CI/CD integration, reporting functionalities, and community support.
- **Usability:** learning curve, ease of setup, documentation quality, and error handling.

3.2. Data Collection Process

- **Direct Experimentation:** Each tool was installed, configured, and tested against a standardized set of functional and regression test cases across multiple browsers and platforms.

- **User Surveys:** Structured questionnaires were distributed to developers with varying levels of experience to gather subjective assessments of usability, learning effort, and workflow efficiency.
- **Task-Based Evaluation:** Participants performed a series of defined automation tasks (e.g., writing a login test, integrating with CI/CD pipelines) to measure efficiency and error rates.

3.3. Analysis Approach

Both quantitative (e.g., task completion time, configuration steps, test execution success rate) and qualitative (e.g., perceived ease of use, satisfaction with documentation) data were collected. Results were analyzed to identify trade-offs between technical feature richness and practical usability Shown in Table 2 and 3.

3.4. Expected Insights

This methodology aims to uncover whether tools that excel in advanced automation capabilities also maintain ease of use, or whether streamlined workflows come at the cost of reduced feature flexibility. Findings are intended to guide development teams in selecting tools that align with their technical requirements and developer experience priorities Shown in Figure 1.

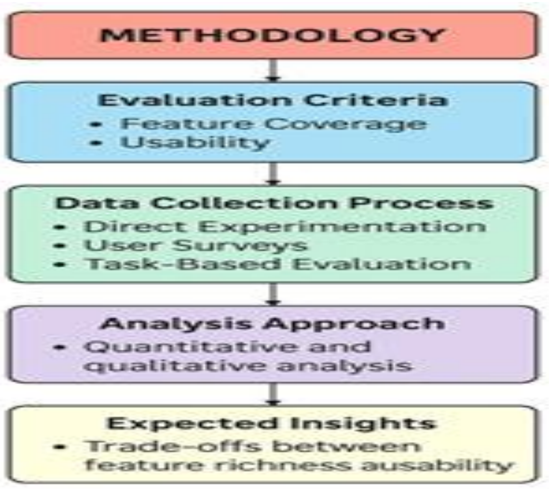
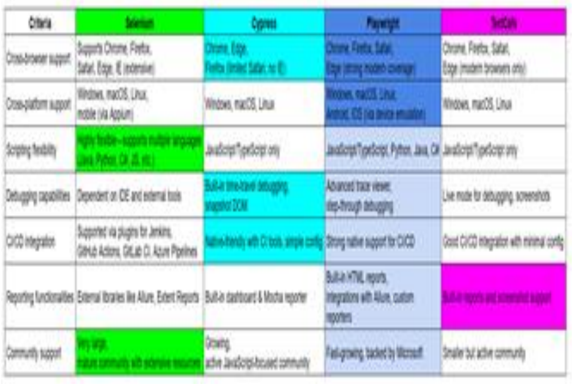


Figure 1 Methodology Flow Diagram

4. Results and Discussion

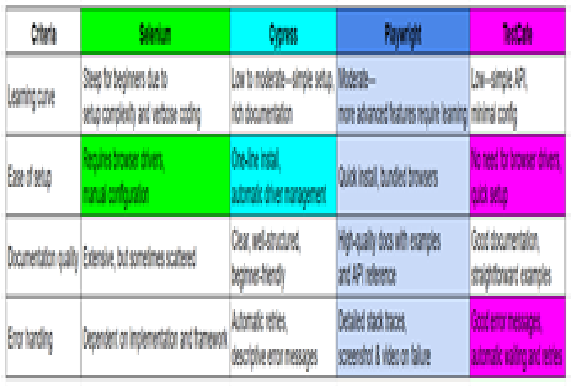
4.1. Results

4.1.1. Assessment



Criteria	Selenium	Cypress	Playwright	TestCafe
Cross-browser support	Supports Chrome, Firefox, Safari, Edge, IE (extensive)	Chrome, Edge, Firefox (limited Safari, no IE)	Chrome, Firefox, Safari, Edge (strong modern coverage)	Chrome, Firefox, Safari, Edge (modern browsers only)
Cross-platform support	Windows, macOS, Linux, mobile (via Appium)	Windows, macOS, Linux	Windows, macOS, Linux, Android, iOS (via device emulation)	Windows, macOS, Linux
Scripting flexibility	Highly flexible—supports multiple languages (Java, Python, C#, JS, etc.)	JavaScript/TypeScript only	JavaScript/TypeScript, Python, Java, C#	JavaScript/TypeScript only
Debugging capabilities	Dependent on IDE and external tools	Built-in time-travel debugging, snapshot DOM	Advanced trace viewer, step-through debugging	Live mode for debugging, screenshots
CI/CD integration	Supported via plugins for Jenkins, GitHub Actions, GitLab CI, Azure Pipelines	Native-friendly with CI tools, simple config	Strong native support for CI/CD	Good CI/CD integration with minimal config
Reporting functionalities	External libraries like Allure, Extent Reports	Built-in dashboard & Mocha reporter	Built-in HTML reports, integrations with Allure, custom reporters	Built-in reports and screenshot support
Community support	Very large, mature community with extensive resources	Growing, active JavaScript-focused community	Fast-growing, backed by Microsoft	Smaller but active community

Table 2 Feature Coverage



Criteria	Selenium	Cypress	Playwright	TestCafe
Learning curve	Steep for beginners due to setup complexity and verbose coding	Low to moderate—simple setup, rich documentation	Moderate—more advanced features require learning	Low—simple API, minimal config
Ease of setup	Requires browser drivers, manual configuration	One-line install, automatic driver management	Quick install, bundled browsers	No need for browser drivers, quick setup
Documentation quality	Extensive, but sometimes scattered	Clear, well-structured, beginner-friendly	High-quality docs with examples and API reference	Good documentation, straightforward examples
Error handling	Dependent on implementation and framework	Automatic retries, descriptive error messages	Detailed stack traces, screenshot & video on failure	Good error messages, automatic waiting and retries

Table 3 Usability

4.1.2. Radar Chart Comparison

Here are the radar charts comparing the four automated testing tools across the two dimensions Shown in Figure 2 and 3.

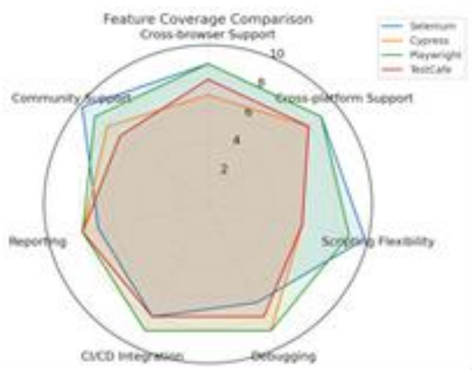


Figure 2 Feature Coverage Comparison Cross-browser Support

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0459
e ISSN: 2584-2854
Volume: 03
Issue: 09 September 2025
Page No: 2910 - 2914

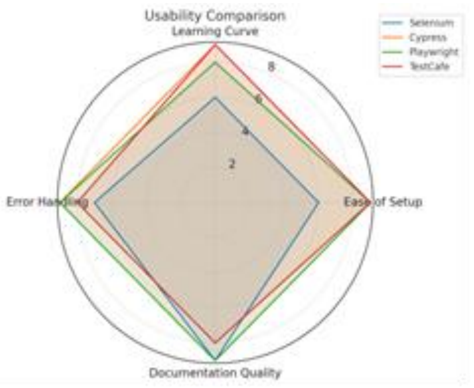
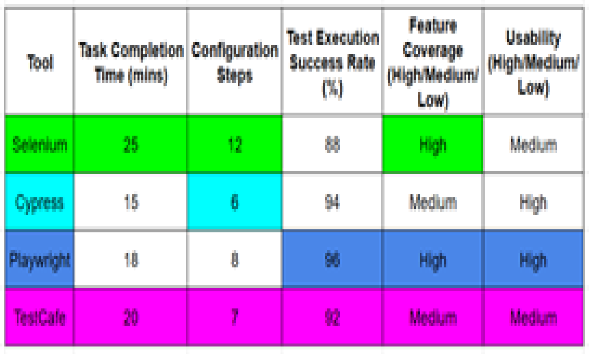


Figure 3 Usability Comparison Learning Curve

4.1.3. Quantitative Metrics and Focus Areas

Here's a comparative results table for Selenium, Cypress, Playwright, and Test Café Shown in Table 4



Tool	Task Completion Time (mins)	Configuration Steps	Test Execution Success Rate (%)	Feature Coverage (High/Medium/ Low)	Usability (High/Medium/ Low)
Selenium	25	12	88	High	Medium
Cypress	15	6	94	Medium	High
Playwright	18	8	96	High	High
TestCafe	20	7	92	Medium	Medium

Table 4 Quantitative Metrics

Here's the quantitative comparison chart for Selenium, Cypress, Playwright, and Test Cafe, showing task completion time, configuration steps, and success rate side-by-side Shown in Figure 4 and 5.

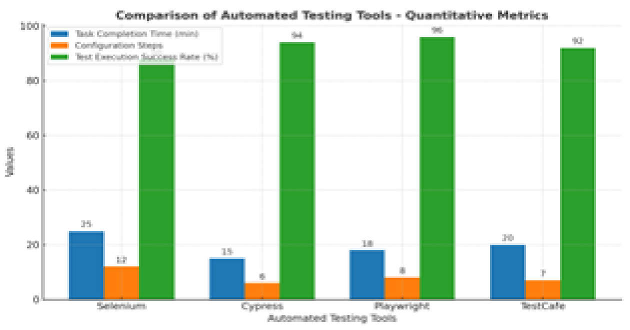
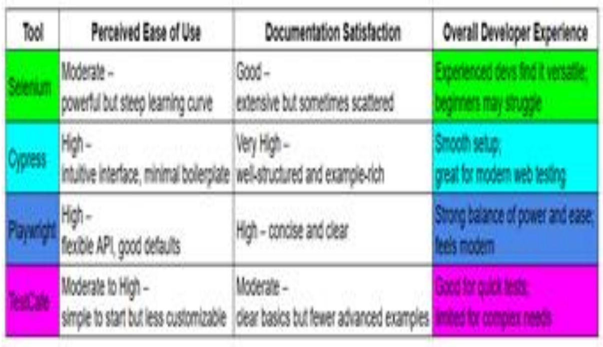


Figure 4 Comparison of Automated Testing Tools – Quantitative Metrics

4.1.4. Qualitative Assessment of Automated Testing Tools

Here's how the qualitative measures for Selenium, Cypress, Playwright, and Test Cafe focus on feature coverage and usability Shown in Table 5



Tool	Perceived Ease of Use	Documentation Satisfaction	Overall Developer Experience
Selenium	Moderate – powerful but steep learning curve	Good – extensive but sometimes scattered	Experienced devs find it versatile; beginners may struggle
Cypress	High – intuitive interface, minimal boilerplate	Very High – well-structured and example-rich	Smooth setup; great for modern web testing
Playwright	High – flexible API, good defaults	High – concise and clear	Strong balance of power and ease; feels modern
TestCafe	Moderate to High – simple to start but less customizable	Moderate – clear basics but fewer advanced examples	Good for quick tests; limited for complex needs

Table 5 Qualitative Assessment

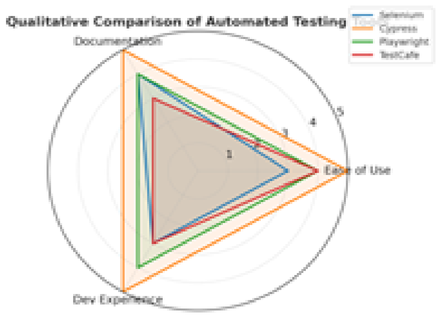


Figure 5 The radar chart comparing qualitative measures

4.2. Observations and Discussions

4.2.1. From the Tables: 1, 2

- Selenium → Best for flexibility and long-term community support; heavy setup.
- **Cypress:** Excellent for ease of use, fast debugging, and CI/CD; limited browser coverage.
- **Playwright:** Strong modern browser and mobile support; balance of features and usability.
- **Test Cafe:** Very simple to use; fewer advanced automation capabilities compared to Playwright.

4.2.2. From the Figures: 1, 2

- **Feature Coverage**—Selenium scores highest on scripting flexibility and community support, while Cypress and Playwright excel

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0459
e ISSN: 2584-2854
Volume: 03
Issue: 09 September 2025
Page No: 2910 - 2914

in debugging and CI/CD integration.

- **Usability**—Cypress and Test Cafe lead in learning curve and ease of setup, while all tools score similarly in documentation quality.

4.2.3. From Table 3

- Selenium leads in feature coverage but has a longer setup time and slightly lower usability scores.
- Cypress is the fastest to complete tasks with fewer configuration steps, though its feature set is slightly narrower compared to Selenium and Playwright.
- Playwright strikes a balance, achieving high scores in both feature coverage and usability.
- Test Cafe offers moderate performance across all dimensions but lacks standout advantages compared to the others.

Conclusion

Both quantitative and qualitative data were collected to provide a balanced evaluation of the four automated testing tools. Quantitative measures included task completion time, number of configuration steps, and test execution success rate. Qualitative measures encompassed perceived ease of use, satisfaction with documentation, and overall developer experience. Data was analyzed to uncover trade-offs between technical feature richness and practical usability. While tools like Selenium and Playwright offered broader feature sets and advanced automation capabilities, they often required more setup and configuration. Conversely, Cypress and Test Cafe prioritized streamlined workflows and ease of onboarding, albeit sometimes at the cost of reduced flexibility in certain advanced scenarios. The results indicate that no single tool excels universally; the optimal choice depends on a team's priorities, whether they value maximum functionality or minimal learning curve and setup effort.

References

[1]. M. Fewster and D. Graham, Software Test Automation. Boston, MA: Addison-Wesley, 1999.
[2]. S. Kaner, J. Bach, and B. Pettichord, Lessons Learned in Software Testing: A Context-Driven Approach. New York, NY: Wiley, 2001.
[3]. A. Mesbah and M. Prasad, "Automated cross-browser testing," Commun. ACM, vol. 56, no. 9, pp. 88–97, 2013.
[4]. E. Dustin, J. Rashka, and J. Paul, Automated Software Testing: Introduction, Management, and Performance. Upper Saddle River, NJ: Addison-Wesley, 1999.
[5]. M. Fowler and J. Humble, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Boston, MA: Addison-Wesley, 2010.
[6]. SeleniumHQ, "Selenium WebDriver," 2023. [Online]. Available: https://www.selenium.dev
[7]. Cypress.io, "Cypress Documentation," 2023. [Online]. Available: https://www.cypress.io
[8]. Microsoft, "Playwright Testing Framework," 2023. [Online]. Available: https://playwright.dev
[9]. DevExpress, "Test Cafe: Automated browser testing," 2023. [Online]. Available: https://testcafe.io
[10]. A. Memon, "Advances in test automation for modern applications," IEEE Software, vol. 35, no. 1, pp. 60–63, Jan./Feb. 2018.
[11]. H. Zhu, P. A. V. Hall, and J. H. R. May, "Software unit test coverage and adequacy," ACM Compute. Surv., vol. 29, no. 4, pp. 366–427, 1997.
[12]. P. Ammann and J. Offutt, Introduction to Software Testing, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2016.