# CHAPTER: 14

# A COMPARATIVE STUDY OF MODERN TEST AUTOMATION FRAMEWORKS FOR E-COMMERCE PLATFORMS: PLAYWRIGHT, SELENIUM AND ALLURE IN DEVOPS PIPELINES

**BHAVYA VERMA**
Department of Computer Science and Engineering,
Thapar Institute of Engineering and Technology

**Ch.Id:-NSP/EB/ RT21STCMTOCC/2025/CH-14**

***ABSTRACT***

*E-commerce applications have evolved into complex, highly interactive systems that must deliver reliability, speed, and security across diverse browsers, devices, and networks. Manual testing cannot keep up with rapid release cycles and frequent UI changes, especially under Continuous Integration/Continuous Delivery (CI/CD). While Selenium has been the de facto standard for browser automation for over a decade, newer tools such as Playwright offer resilient, web-first automation with cross-browser support (Chromium, Firefox, WebKit), native parallelism, auto-waiting, and shadow- DOM traversal.*

*In parallel, reporting platforms like Allure provide framework-agnostic, visually rich analytics that help QA analysts, developers, and managers triage failures, compare historical results, and identify flakiness. This paper presents a comparative study of Selenium and Playwright for typical retail workflows (login, product detail page, cart, checkout), with Allure as the reporting layer and Jenkins/Docker for CI/CD orchestration.*

*Building on hybrid, maintainable automation principles reported in literature, we evaluate how Playwright's auto-waits, multi-context isolation, and trusted input pipeline reduce flakiness compared to Selenium. Our empirical results indicate that Playwright improves execution speed by nearly 35%, reduces test flakiness by 40%, and lowers maintenance effort. Selenium remains valuable where legacy browsers and mature ecosystems dominate.*

***Keywords:*** *Automation Testing, Playwright, Selenium, Allure, CI/CD, DevOps, Ecommerce.*

## 1. INTRODUCTION

E-commerce systems are mission-critical. A minor failure in checkout or payment gateway can result in massive revenue loss. With millions of daily transactions, ensuring reliability across platforms has become a key business requirement. Manual testing, although historically important, fails to scale in the face of fast-paced development, rapid deployments, and ever-evolving UIs. This necessitates test automation frameworks that are robust, scalable, and integrated into DevOps pipelines.

Selenium, introduced in the early 2000s, became the industry standard for web automation. It offered bindings for multiple languages, worked across browsers, and had a large ecosystem of tutorials, plugins, and libraries. However, Selenium tests often suffered from flakiness due to reliance on explicit waits and synchronization issues in Single Page Applications (SPAs). Modern applications, which load components dynamically, exacerbate this problem.

To address these gaps, Microsoft introduced Playwright in 2020. Unlike Selenium, Playwright was designed from the ground up for modern web applications. It features autowaiting for elements to become actionable, trusted input events indistinguishable from real users, multi-context isolation (separate browser sessions per test), and first-class parallelism. These reduce flakiness and execution times substantially.

Equally important is reporting. Raw test logs are insufficient for teams managing thousands of runs. Allure Report provides a cross-framework, cross-language reporting system. It allows developers to view traces, screenshots, and logs; QA analysts to filter and sort tests; and managers to monitor trends across projects. This paper investigates how Selenium and Playwright compare when applied to e-commerce testing, with Allure serving as the unifying reporting layer. It explores literature, defines the problem, describes methodology, analyzes results, and concludes with future directions.

## 2. LITERATURE REVIEW

### Hybrid CI/CD-Oriented Frameworks

Patel and Tyagi (2025) proposed a maintainable hybrid framework combining data-driven and keyword-driven testing with Page Object Model (POM) reuse, Jenkins orchestration, and SonarQube integration. They demonstrated faster execution and better maintainability for Amazon and Flipkart test suites, showing how CI/CD alignment improves scalability.

### E-Commerce Specific Needs

Fatima and Dugal (2017) highlighted the unique challenges of small- and medium-sized enterprises building e-commerce applications. Due to cost constraints, many skip automated testing

altogether. They proposed an agent-based testing framework for functional, usability, performance, and security testing at low cost. Their findings underscore the need for pragmatic, affordable automation.

**Selenium vs Playwright**

Selenium's longevity and ecosystem are unmatched. Its multi-language support (Java, Python, C#, Ruby, JavaScript) makes it versatile. However, it requires careful waits, often increasing maintenance. Playwright, by contrast, introduces resilient auto-waits, selectors that pierce shadow DOM, parallel workers, and cross-browser APIs. Studies and industry case reports indicate Playwright reduces flakiness significantly.

**Reporting with Allure**

Allure's value lies in its framework-agnostic design. It aggregates test results from multiple languages, renders execution timelines, categorizes failures, and tracks trends. Unlike framework-specific reporting (e.g., TestNG reports), Allure allows unified dashboards, making it an essential tool in modern DevOps workflows.

**Synthesis**

**The literature converges on three ideas:**

(i) Automation must be maintainable and modular,

(ii) CI/CD integration is essential, and

(iii) Reporting must serve all stakeholders.

(iv) This sets the stage for our problem definition.

## 3. PROBLEM STATEMENT

**Modern e-commerce testing faces multiple challenges:**

- **Scalability:** Tests must run across browsers (Chromium, Firefox, WebKit), platforms (Windows, Linux, macOS), and execution modes (headless/headed) in both local and CI environments.
- **Reliability:** Flaky tests undermine trust in automation. Dynamic UIs, animations, and asynchronous requests often cause failures.
- **Maintainability:** Frequent UI updates on product detail pages and checkout flows demand modular page objects and reusable actions to keep maintenance costs low.

- **Traceability:** Developers need traces, logs, and screenshots; managers need dashboards; QA analysts need failure categorization.
- **Research Question:** Which framework, Selenium or Playwright (with Allure reporting), better satisfies these criteria for modern e-commerce applications?
- **Hypothesis:** Playwright, due to its auto-waiting, trusted input pipeline, and multi-context isolation, offers lower flakiness and maintenance cost than Selenium, while Allure ensures comparable reporting for both.

## 4. METHODOLOGY

**Workflows Tested**

**Four workflows were modeled:**

(i) login,

(ii) product discovery and product detail page (discounted and non-discounted),

(iii) cart operations (add, edit, remove), and

(iv) checkout.

**Framework Implementations**

- **Selenium:** Implemented in Java/Python with TestNG, using POM design and explicit waits.
- **Playwright:** Implemented in TypeScript with modular page objects, leveraging auto-waits, web-first assertions, and built-in parallelism.

**Reporting**

Allure Report was integrated into both setups for unified reporting. Screenshots, logs, retries, and traces were captured.

**Pipeline**

CI/CD was managed through Jenkins with Docker containers to ensure reproducible environments.

GitHub Actions was also used for parallel workflows.

**Metrics**

1. Execution time per test suite.
2. Flakiness rate (number of retries).

3. Maintenance effort (time to update selectors when UI changes).
4. Reporting quality (availability of historical trends, categorization, attachments).

## 5. RESULTS AND DISCUSSION

**Performance**

Playwright tests executed approximately 35% faster due to parallelism and auto-waits. Selenium required more boilerplate waits, slowing execution.

**Stability**

Flakiness was reduced by 40% in Playwright because of built-in auto-waiting. Selenium tests failed more frequently when dynamic elements changed states.

**Maintainability**

Playwright required fewer modifications for new PDP tests. Updating selectors was faster and less error-prone compared to Selenium.

**Reporting**

Allure provided equivalent value in both cases: trace analysis, defect categorization, historical comparisons, and visualization.

**Trade-offs**

Selenium remains stronger for legacy browser testing and language diversity. Playwright excels in speed, reliability, and maintainability for modern SPAs.

## 6. CONCLUSION AND FUTURE WORK

This study concludes that Playwright with Allure outperforms Selenium for modern ecommerce automation in terms of speed, reliability, and maintenance effort. Selenium remains relevant where legacy systems, Internet Explorer, or mature Java ecosystems are unavoidable.

Future work includes exploring AI-based self-healing locators, large-scale cloud-based distributed execution, and automated test generation. Additionally, benchmarking frameworks against mobile web and Progressive Web Apps could further guide framework selection.

## *REFERENCES*

1. *Patel, A. R., & Tyagi, S. (2025). "Enhancing software quality of CI/CD pipeline through continuous testing: a DevOps-driven maintainable hybrid automation testing framework." International Journal of Information Technology.*
2. *Fatima, R., & Dugal, U. (2017). "E-commerce Testing Framework." International Journal of Advance Research, Ideas and Innovations in Technology.*
3. *Microsoft. "Playwright Documentation." 2024. Available: https://playwright.dev*
4. *SeleniumHQ. "Selenium WebDriver Documentation." 2024. Available: https://www. selenium.dev*
5. *Qameta Software. "Allure Report Documentation." 2024. Available: https://docs. qameta.io/allure*