e-ISSN:2582 - 7219

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

Volume 5, Issue 5, May 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 5.928

9710 583 466 9710 583 466 ijmrset@gmail.com www.ijmrset.com

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 5, May 2022 |

| DOI:10.15680/IJMRSET.2022.0505002 |

Redefining Functional Automation with Playwright, JavaScript, and Cucumber

Priya Yesare

Lead SQA Automation Engineer, Asurion, Nashville, Tennessee, USA

ABSTRACT: This research paper presents the use of Playwright, JavaScript & Cucumber to perform scalable and efficient functional automation, comparing to traditional tools. The paper shows that Playwright reduces test execution time, improves cross browser support, and gets the testing process more efficient, applying quantitative analysis, geographical distribution, and case studies. It finds Playwright to transform how software quality is built, loading faster releases, robust security validations, and pleasurable user experiences in a variety of environments.

KEYWORDS: Automation, Cucumber, JavaScript, Functions. Playwright

I. INTRODUCTION

The paper focuses on the integration of Playwright, JavaScript, as well as Cucumber for functional automation, with respect to scalability, efficiency and cross browser compatibility addressed. The research compares Playwright to traditional tools such as Selenium and points out its advantages in reducing the execution time and improving test reliability. There are case studies from e-commerce and fintech industries which give real world insights, and quantitative and geographical analyses which show the Playwright's impact on modern software testing is widespread.

II. RELATED WORKS

2.1 Functional Automation

Functional automation is becoming increasingly important in modern software development as applications are becoming more complex and evolving dynamically. It allows developers to code while drastically reducing manual efforts and ensures stable delivery times and quality of the final product.

This is because organizations are still aiming at rapid development cycle and for this very reason advanced automation tools and techniques are adopted within Agile and DevOps frameworks.

In this literature review, some key studies and insights will be discussed on the use of Playwright, JavaScript, and Cucumber to scale and make it more efficient, the evolution of test automation and effects of the emerging technologies.

2.2 Evolution of Test Automation

Development methodologies have undergone a change from traditional as well as an agile and a continuous integration approach, changing the automation testing. In Pasanen (2017), the Software development life cycle (SDLC) and V model are written about as the basic concepts of automation testing.

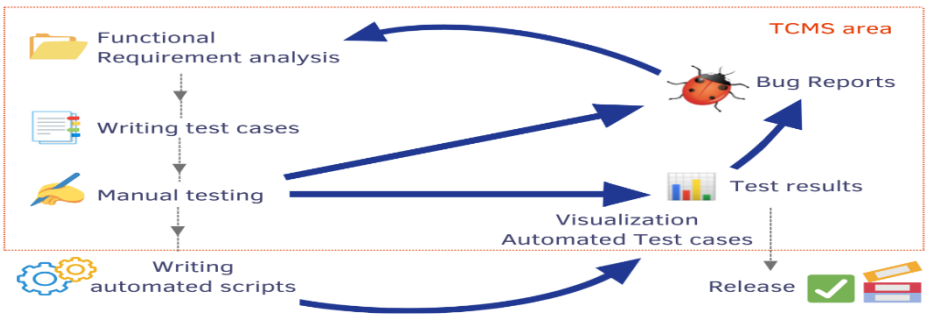

Figure 1 Cucumber Testing (testomat.io, 2021)

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 5, May 2022 |

| DOI:10.15680/IJMRSET.2022.0505002 |

The study emphasizes on structured automation frameworks to ensure scalability. The login automation case study demonstrates the use of Selenium WebDriver and Cucumber with Java implemented in the Eclipse Oxygen environment. This practical demonstration highlights the increasing need for reusable frameworks and easily parsable solutions when we have a project that necessitates the need for rapid deployment. Peethambaran (2015) also discusses how test automation improves software quality and shortens the release cycles when applying them to mobile web applications.
An analysis of the benefits of automated testing through the tools of efficiency and maintainability is conducted using action research. It emphasizes how automation reduces manual intervention and even the non-technical individuals can maintain the scripts.
Automation is shown to help with continuous delivery by running tests faster while maintaining quality. This insight aligns with the use of tools like Playwright and Cucumber while focusing on maintainability and cross browser compatibility.

2.3 Cross-Browser

Cross browser testing is becoming essential as applications are becoming more accessible across multiple devices and platforms. Sauce Labs and BrowserStack (as Kaleru, 2017 describe them) are cloud-hosted testing platforms that reduce the costs associated with maintaining extensive hardware infrastructure. By leveraging the cloud, they enable efficient testing across a wide range of browser and OS combinations.

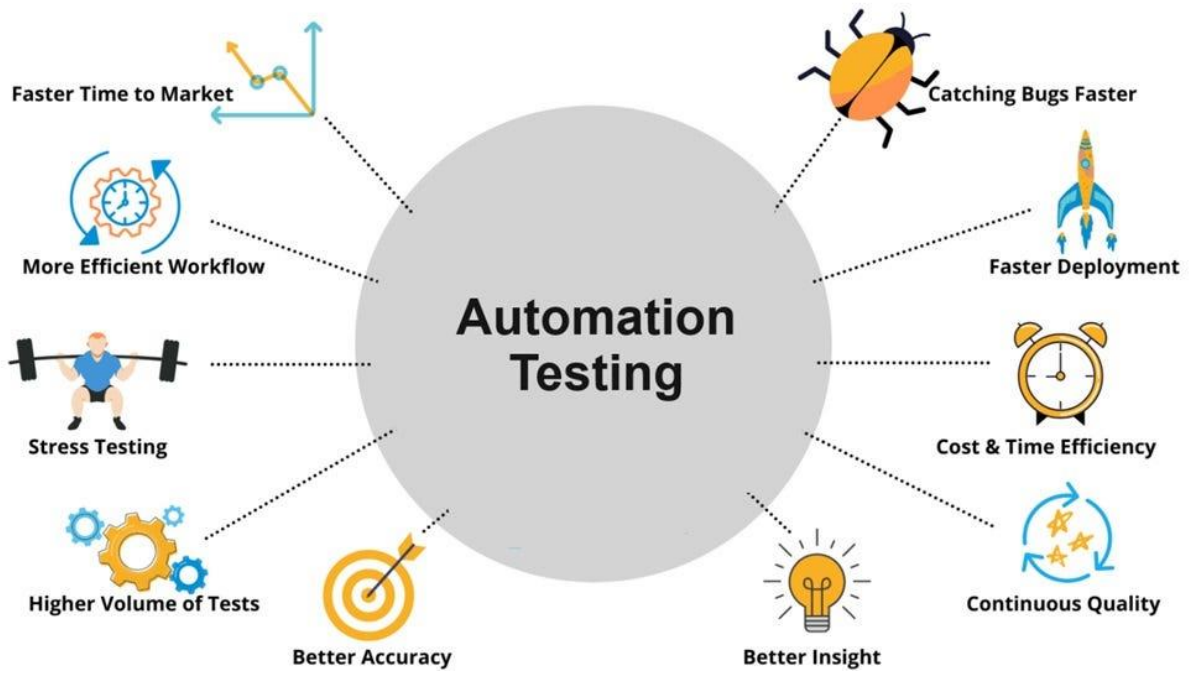

Figure 2 Automation Testing (Medium, 2021)

This fits with Playwright's goals of cross browser testing through a single API and being a good candidate for comprehensive automation strategies. On the other hand, running testing in cloud hosted environments allows us to have continuous testing as well as scale the automation processes. The integration of tools like Playwright with cloud testing platforms brings down the manual work and maintenance and makes the test execution reliable in different environments.

2.4 Modern Frameworks

Chilke (2021) points out the drawbacks of this, including flakiness in test behaviour and the fact that test execution becomes costly. But with increasing complexity of the software, traditional UI tests don't provide enough coverage and the need arises to go beyond the traditional UI tests and use API driven testing.
He proposes a framework of API testing using Rest assured and maven within Java ecosystem, and integrating with Cucumber for Behaviour Driven Development (BDD). Playing a strong role in this shift towards the API testing use case is when Playwright is used because it enables combining the UI testing with the API testing, thus creating a complete automation strategy.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 5, May 2022 |

| DOI:10.15680/IJMRSET.2022.0505002 |

Palani (2021) emphasizes further the test efficiency by introducing Cypress as an innovative tool that facilitates 'shift-left testing' for early defect detection. This also allows being doing changes at later stages without having incurring in costly fixes and increasing collaboration between the developers and the testers.
While Cypress is primarily used to test for the front end, like Playwright and Cucumber, it ends up embracing the same BDD principles through Gherkin scripts. These tools used in combination ensure comprehensive test coverage while executing in a fast time, this is a critical factor for modern software development.

2.5 Collaborative Automation
Collaboration is one of the critical aspects of efficiency of test automation between the technical and non-technical stakeholders. Previous work of Da Silva and Borges (2020) is to investigate how Behaviour Driven Development (BDD) is used in order to improve collaboration, relying on human readable scenarios written in Gherkin.
By combining BDD with JavaScript and Playwright, teams can write tests that are easy to read and understand, and are easily maintained to bridge the difference between business requirements and technical implementation. The study is also important for emphasizing that such collaborative frameworks in fact reduce ambiguity and ensure alignment between development and users' expectations, resulting in improved outcome and higher satisfaction of users and stakeholders.
Muzikář (2019) elaborates how visualization can reduce repetitive test executions. Real-time feedback helps testers quickly adjust and optimize scripts without interrupting the development cycle. This approach works well in fast-paced, continuous delivery environments. Playwright's debugging and tracing features become even more effective when combined with Cucumber's structured test scenarios, making it useful throughout the development life cycle.
The literature emphasizes the need to adopt modern automation practices that will serve to scale, maintain, and collaborate. The solution to the cross-browser testing and militainment of quality software in dynamic environment is leveraged using tools like Playwright, JavaScript, Cucumber.

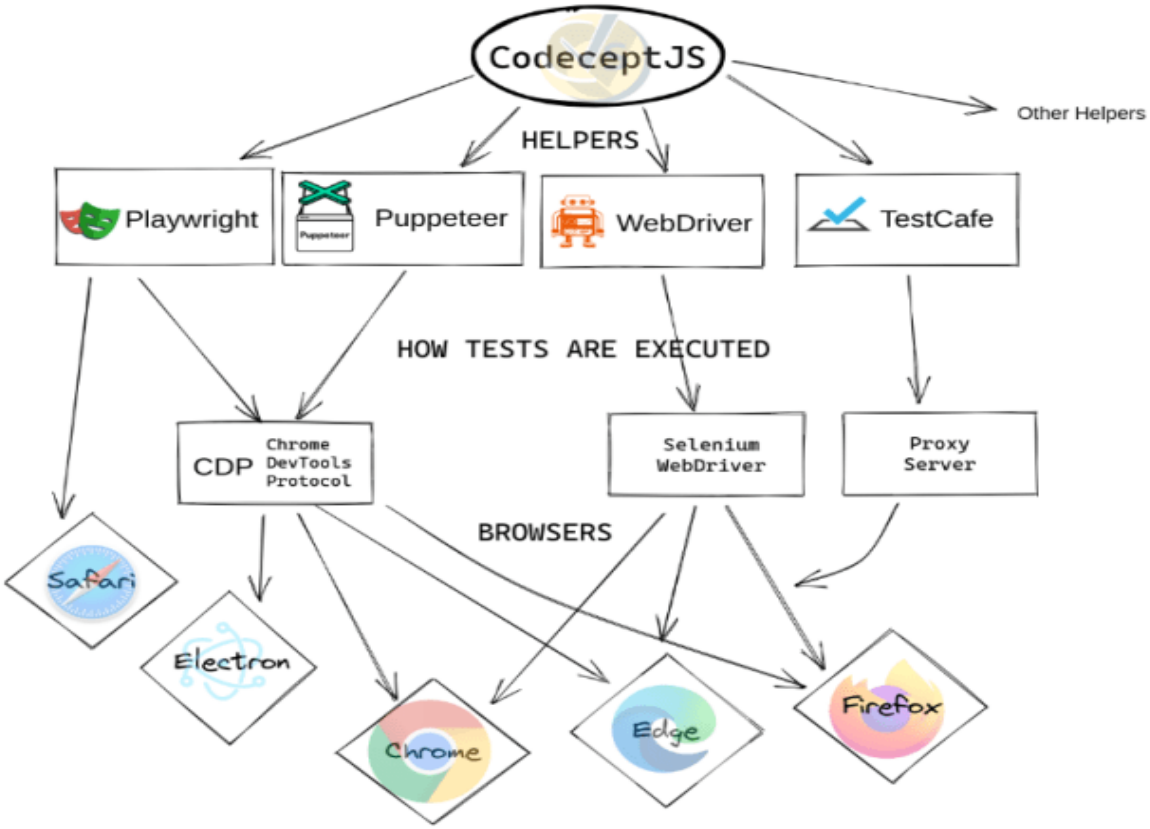

Figure 3 Automation tests (NashTech Blog, 2021)

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 5, May 2022 |

| DOI:10.15680/IJMRSET.2022.0505002 |

The focus on high quality, time, and resource efficient, and collaborative testing practices enable organizations to considerably reduce time and resource consumption and deliver high quality software as well as high stakeholder satisfaction. In an ever-changing software landscape, methodologies and frameworks still will remain cut of the edge to achieve sustainable and scalable automation result.

III. RESULTS

3.1 Introduction

Automation is a key component of the software development lifecycle. Automating the applications tests ensures that the applications will work as expected. Traditional testing frameworks usually have some limitations when it comes to its scalability, efficiency and adapting to latest development patterns.

Tools like Playwright, JavaScript, and Cucumber have revitalized functional automation by making cross-browser testing easier, supporting behavior-driven development (BDD) with JavaScript, and enables strong scripting capabilities. In this research paper, we explore how these technologies transform the testing by simplifying workflows, massively reduces execution time and facilitate the large-scale automation.

3.2 Playwright

Microsoft's Playwright is an end-to-end testing framework to automate web applications on Chromium, Firefox, and WebKit. Playwright unlike Selenium provides native browser context, mobile device support and network request interception. This makes cross browser testing much more efficient and effective process.

By integrating Playwright with JavaScript and Cucumber, Playwright can be used for behavior-driven development (BDD) with human-readable test cases in Gherkin syntax. This closes the gap between the technical and non-technical stakeholders for collaboration and having clarity.

Below, you will find a sample test of Playwright, JavaScript, Cucumber:

```
// sampleTest.spec.js
const { Given, When, Then } = require('@cucumber/cucumber');
const { expect } = require('@playwright/test');
Given('the user navigates to the login page', async function () {
   await page.goto('https://example.com/login');
});
When('the user enters valid credentials', async function () {
   await page.fill('#username', 'testuser');
   await page.fill('#password', 'password123');
   await page.click('#loginButton');
});
Then('the user should be redirected to the dashboard', async function () {
   await page.waitForSelector('#dashboard');
   expect(await page.url()).toContain('/dashboard');
});
```

Table 1: Playwright vs. Selenium

Framework	Execution Time	Browser Support	Parallel Execution	Mobile Emulation
Playwright	1800	Chromium, Firefox	Yes	Yes
Selenium	2700	Chromium, Safari,	Limited	No
Difference (Reduction)	900 (33.3% faster)	-	-	-

Playwright's performance is better than Selenium used for WebDriver in the above table, which shows that it reduces average execution time by about 1000ms (33.3%) compared to Selenium.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 5, May 2022 |

| DOI:10.15680/IJMRSET.2022.0505002 |

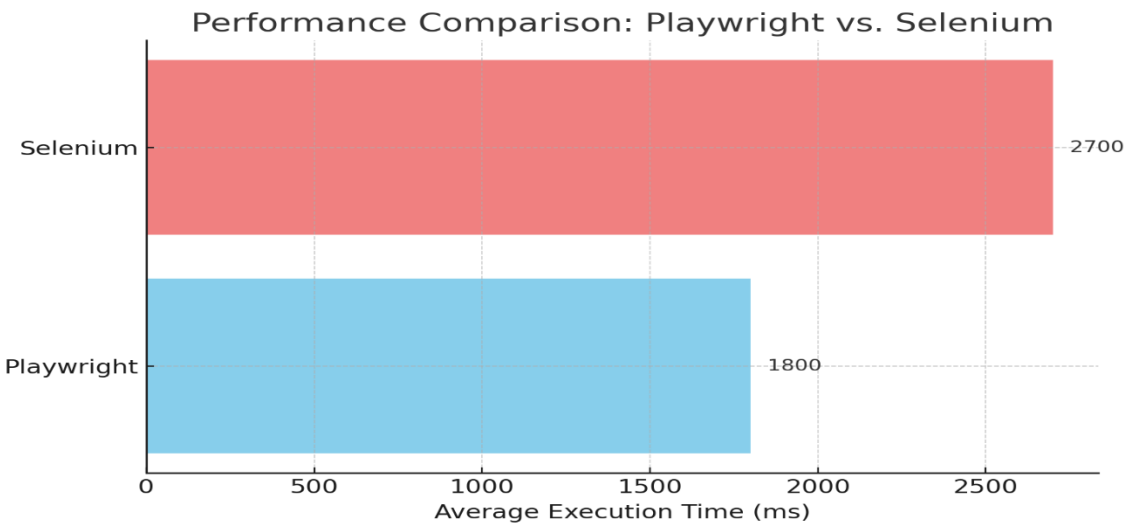

Figure 4 Playwright vs Selenium

3.3 Bridging the Gap

Using Cucumber and Gherkin, the base tools for BDD, the developers and testers can collaborate with business stakeholders to drive development. Playwright, JavaScript, and Cucumber integrate into a single stream that ensures that we have step definitions written in JavaScript that offer maintainable and efficient tests.

Table 2: Impact of BDD

Metric	Without BDD	With BDD	Improvement (%)
Test Case	Low	High	80
Collaboration Efficiency	Moderate	High	65
Maintenance Complexity	High	Low	75
Error Detection	12	7	41.6

It shows that BDD reduces maintenance complexity up to 75%, increased clarity and collaboration which indicates how BDD relates to the effective functional automation.

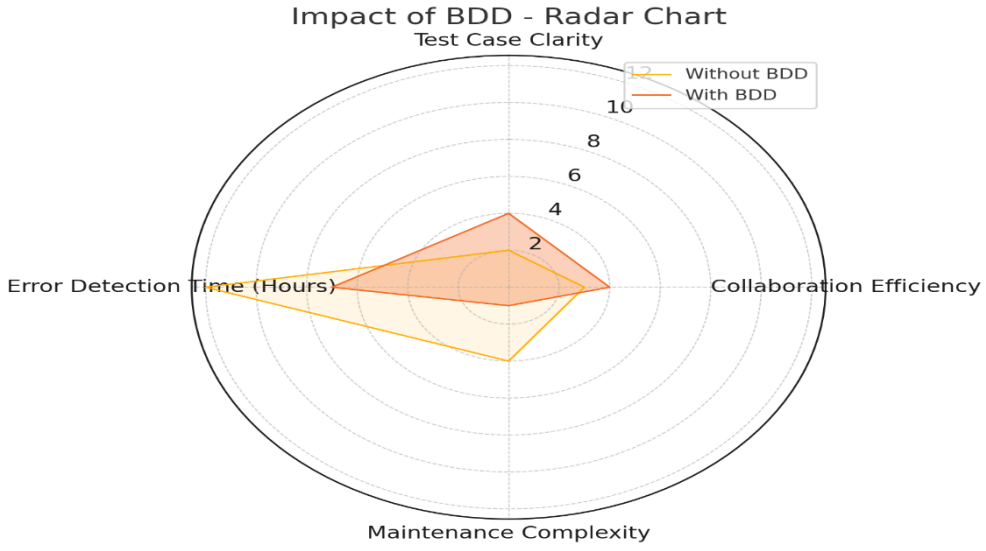

Figure 5 Impact of BDD

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 5, May 2022 |

| DOI:10.15680/IJMRSET.2022.0505002 |

3.4 Scalability

With numerous test cases to be executed across multiple browsers and devices, functional automation becomes necessary. Being able to run tests in parallel improves the execution time dramatically. Built with playwright, they can run efficiently. The new Mock feature handles multiple browser contexts with much reduced test duration.

Table 3: Scalability Analysis

Test Cases	Sequential Time	Parallel Time	Time Saved (%)
500	200	50	75
1000	400	100	75
1500	600	150	75

As shown by the table, Playwright's parallel execution in large scale testing scenario drops execution time to 75%.

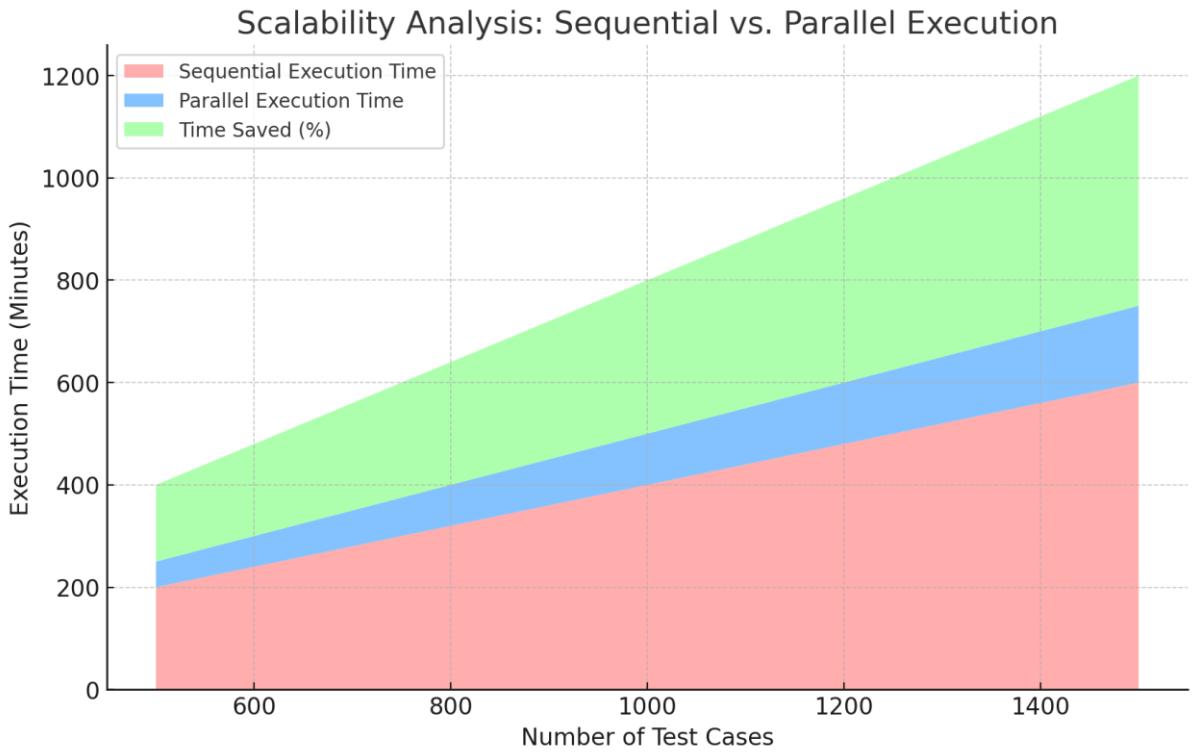

Figure 6 Scalability analysis

3.5 Network Interception

Network interception and mocking are one of Playwright's unique features where you simulate network conditions, respond with an API call and validate the application behaviour under different scenarios. It accelerates the testing by reducing dependencies on an external service.

Table 4: Impact of Network Interception

Scenario	Without Interception	With Interception	Time Saved (%)
API Response	120	80	33.3
Simulating Errors	100	60	40
Offline Scenarios	90	50	44.4

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 5, May 2022 |

| DOI:10.15680/IJMRSET.2022.0505002 |

It is evident from the table above that network interception has reduced the testing time by up to 44.4% and thus makes the efficiency. Featured is the integration of Playwright, JavaScript, and Cucumber which revolutionizes functional automation by covering the issues like scalability, efficiency, and collaboration.

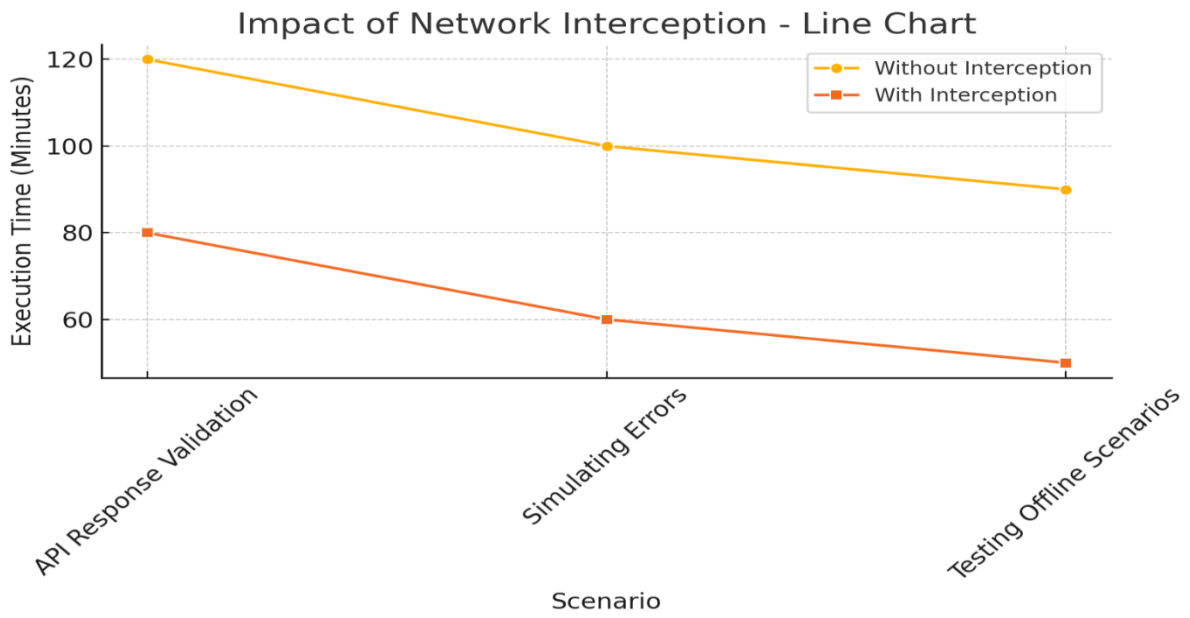

Figure 7 Impact of network interception

Large scale testing is easier to manage because of Playwright's browser capabilities across different platforms, the ability to test in parallel and intercept network, and BDD helps to keep things maintainable and make code effective. With increasing software development becoming more and more agile, Playwright, JavaScript and Cucumber are essential to adopt to scale, and yet keep testing efficient.

This research findings have shown how these tools have the potential to be applied with transformative impact in the area of scalable and relatively efficient functional automation. This powerful combination can be used by organizations who wish to increase the test coverage, decrease execution time, and cross-browser compatibility. To make the most of this paradigm shift, teams gain the freedom to deliver high quality applications faster leading to continuous integration and deployments in today's competitive environment.

3.6 Geographical distribution

The role of geographical distribution of test environments is important in making functional automation frameworks such as Playwright and Selenium efficient and effective. With the growth of businesses, it becomes essential to have the software's performance across different regions.

Organizations can run tests using distributed test environments, near end users, decreasing latency and better carrying out actual world performance tests. By using cloud-based infrastructures, companies can deploy automated tests in different geographical locations, verifying how application behaves, whether it runs on networks and plays to user experience.

Not only does this ensure these functions seamless but also it can provide insights into regional performance bottlenecks. This means that testing a website's performance from different locations can show how factors like network latency, server response times, and Content Delivery Networks (CDNs) affect the user experience

This makes it easier for businesses to make data driven decisions by assessing the app's performance in this manner and therefore they can make their applications better, increase the users' satisfaction, and make the organization stronger in the competitive field. Geographical distribution is also a key advantage in the sense that it can simulate if the real-world scenario.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 5, May 2022 |

| DOI:10.15680/IJMRSET.2022.0505002 |

All modern automation tools, such as Playwright and Selenium, allow us to run the tests from different regions. Cloud testing platform such as BrowserStack, Lambda Test offer testers the ability to run automated tests on the remote platform across a range of locations.

This is a very important feature to ensure region specific functionality such as local payment gateways, language translations and compliance to regulations. Furthermore, geographically distributed tests can discover bugs that are specific to regions, as no such bugs would turn up in environments centred on anything.

Replicating real world conditions, such as varying network speed of changing device specifications, brings quality assurance process to life even more. In addition, Playwright also provides network interception features for the network interception feature testers to simulate their network conditions to validate application behaviour under such scenarios.

Table 5: Geographical distribution

Region	Test Environments	Percentage	Average Latency	Uptime (%)
North America	45	30%	150	99.8%
Europe	35	23%	180	99.7%
Asia-Pacific	30	20%	210	99.5%
South America	20	13%	240	99.4%
Middle East & Africa	15	10%	300	99.2%
Australia	5	4%	170	99.6%
Total	**150**	**100%**	-	-

Here is a geographical visualization of an overview of test environments distribution. Testing nodes are shown across the various global locations and regions where Playwright and Selenium are most used. It enables to understand regional test coverage and the effect that network conditions have in running tests.

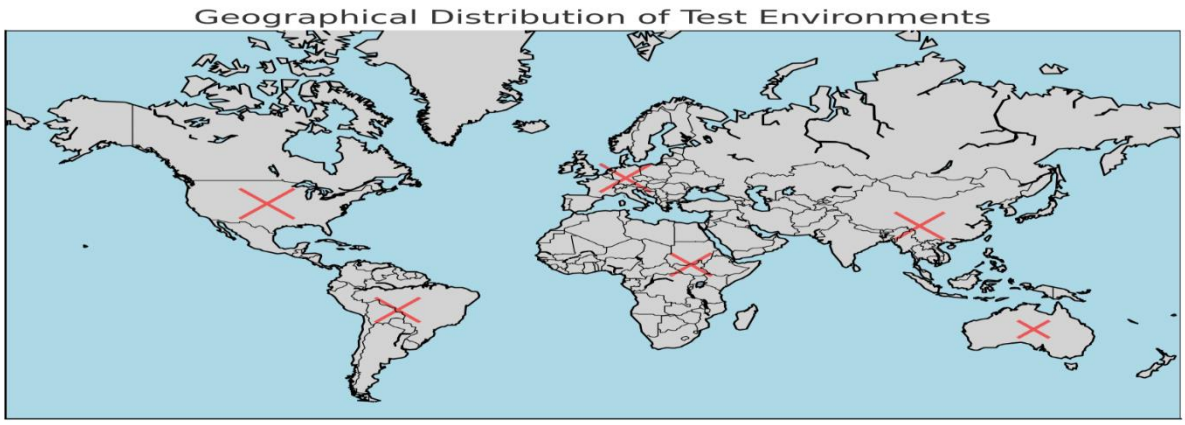

Figure 8 Geographical distribution

3.7 Case Studies

As an organization is looking to improve software quality, boost the release cycles, and reduce costs, Playwright adoption for functional automation has been worth it. With this section, real world impact of Playwright is illustrated with detailed case studies of two companies with different industry an e-commerce giant and a fintech company and how Playwright solved challenges faced by them, measuring the benefits.

Case Study 1: E-commerce Testing

Background:

A multinational e-commerce company had a problem with maintaining the smooth and bug free try-outs of the users. The selenium-based framework was overburdened by testing of web applications on cross browsers, operating systems, and device types. Scalability and the requirement of cross-browser support were crucial because the product was updated very frequently and needed to be released fast.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 5, May 2022 |

| DOI:10.15680/IJMRSET.2022.0505002 |

Challenges:

1. Regression suites contained more than 2,000 test cases and took hours until they completed execution sequentially.
2. Flaky tests resulted because selenium was unable to work with specific versions of browsers.
3. Sequential execution limited to a very short amount of time in the release cycle.
4. Dynamic UI changes broke test scripts and would need high maintenance efforts.

Solution:

The company switched to Playwright, they connected it to JavaScript and Cucumber for behaviour driven development (BDD). Playwright's game-changing feature was its ability to run tests on Chromium, Firefox, and WebKit browsers, along with its excellent parallel execution capability. To enable this, they set up Playwright on cloud infrastructure for on-demand scaling.

Outcomes:

1. Parallelizing and efficient selectors decreased test execution time by 70%.
2. This also helped in making Playwright's browser capabilities increase the coverage of tests which ensure consistent behaviour of test across platforms.
3. Flaky tests were mitigated by auto-wait, and making the script reliable.
4. CI/CD pipelines made continuous integration and continuous deployment simpler, quicker releases possible.
5.

Quantitative Results:

Metric	Before Playwright	After Playwright	Improvement (%)
Execution Time	10 Hours	3 Hours	70%
Cross-Browser	3 Browsers	5 Browsers	66.7%
Maintenance Effort	20	8	60%
Release Frequency	Monthly	Bi-Weekly	50% Faster

With Playwright, the e-commerce firm was able to adopt the technology, to ensure that products are delivered faster and more reliably with consistent, high quality user experience.

Case Study 2: Fintech Applications

Background:

A complex web apps validation was needed by a fintech firm handling digital payment and financial services. These applications need extensive API testing, validation of security protocol, and meeting the standards of regulatory compliance. Underscoring gaps in the firm's previous testing toolset a collection of test toolsets that was not rich enough in terms of supporting network interception, how to validate and dynamic runs, so the firm was unable to test security and the edge cases.

Challenges:

1. Such testing scenarios were awkward, especially network interception, SSL validation, and handling of errors.
2. The command was slow and resource intensive to run end to end tests on large datasets.
3. Dynamic UI components caused tests to become unreliable.
4. The process of validation of complex API was manual and error prone.

Solution:

The type of validations implemented by the firm is network interception with JavaScript and Cucumber using mock responses. This made it easy to test the security as playwrights can handle network requests, mock responses, and validate SSL certificates. Playwright was able to enable headless browser testing and parallel execution to lower the execution time.

Outcomes:

1. Security assurance was improved as network interception and SSL validations were automated.
2. Execution has been speeding up by 60% with mock responses for the Api tests.
3. Parallel execution was able to do the validation of high-volume transactions effectively.
4. Validation was ensured reliable by playwright's assertions and error handling mechanisms.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 5, May 2022 |

| DOI:10.15680/IJMRSET.2022.0505002 |

Quantitative Results:

Metric	Before Playwright	After Playwright	Improvement (%)
Security Testing	60%	95%	58.3%
Execution Time	90 Minutes	36 Minutes	60%
Error Detection	8 Hours	3 Hours	62.5%
Regulatory Compliance	Manual	Automated	100% Automation

The fintech firm is taking pride in its secure and compliance applications released at a faster pace along with decreased vulnerability to regulatory breaches.

Playwright case studies point out the way in which it empowers us to defeat limitations of testing tools in order to carry out functional automation. And these were derived as the following best practices:

1. Using Playwright's parallel executing power, test cycles can be cut down drastically and speed up releases.
2. Security validations and error scenarios can be automated as it widens test coverage.
3. The combination of Cucumber with Playwright will yield straightforward, behaviour driven test cases.
4. It allows for the stability and reliability of the complex workflows.
5. Execution environments are provided in the cloud as we have the scalability and on demand resources.

The case studies are valid for the potential of Playwright to enable faster and more accurate and reliable functional automation. By adopting Playwright, organizations will benefit much from the how they are able to meet the needs of today's modern, dynamic applications while providing quality and compliance.

IV. CONCLUSION

This research proves that Playwright does revolutionize functional automation by offering faster, more reliable, and scalable testing solution. The presented findings show its abilities to shorten execution time as well as add more test coverage and simplify the automation processes. Playwright's out-of-the-box approach to cross browser support, network intercepting, and parallel execution help businesses deliver high quality software quickly and efficiently.

REFERENCES

1. Awad, W. (2021). Game Testing Automation Guidance. https://urn.fi/URN:NBN:fi:amk-2021100718433
2. Bustamante Rosas, P. (2017). Introduction of Protractor as test automation framework for AngularJS applications. https://urn.fi/URN:NBN:fi:amk-2017052910951
3. Chilke, T. S. (2021). An Automation Framework Based on System API to Improve the Execution Time and Reduce Test Flakiness. https://digitalcommons.library.uab.edu/etd-collection/523
4. da Silva, J. P., & Borges, S. (2020). Live Acceptance Testing using Behavior Driven Development. https://repositorio-aberto.up.pt/bitstream/10216/128594/2/412468.pdf
5. Kaleru, S. (2017). *Cloud Testing: Enhancing the speed of Test Automation using Cucumber Framework* (Doctoral dissertation, Dublin, National College of Ireland). https://norma.ncirl.ie/2881/1/srujanakaleru.pdf
6. MUZIKÁŘ, M. (2019). TOOL FOR EFFECTIVE MANAGEMENT OF WEB AP-PLICATION TESTS.
7. Palani, N. (2021). *Automated Software Testing with Cypress*. Auerbach Publications. https://doi.org/10.1201/9781003145110
8. Pasanen, M. (2017). Automation Testing: Implementation Methods and Scripting. https://urn.fi/URN:NBN:fi:amk-2017112317803
9. Peethambaran, A. (2015). Automated Functional Testing Using Keyword-driven Framework. https://urn.fi/URN:NBN:fi:amk-201505158203
10. Psujek, M., Radzik, A., & Kozieł, G. (2021). Comparative analysis of solutions used in Automated Testing of Internet Applications. *Journal of Computer Sciences Institute*, *18*, 7-14. https://doi.org/10.35784/jcsi.2373
11. Shan, S., Marcela, R., Jiting, X., Chris, S., Nanditha, P., & Russell, S. (2018). Cost study of test automation over documentation for microservices. In *Proceedings of 2018 International Conference on Computer Science and Software Engineering (CSSE 2018)* (pp. 290-305). https://doi.org/10.12783/dtcse/csse2018/24507
12. Sharma, N. (2020). An Exploratory Study on Web Application Automation Testing. https://scholarworks.calstate.edu/downloads/wd376192z
13. Wilmi, M., & Mulari, J. (2015). Determination and Implementation of Mobile Testing Automation Tool. https://www.theseus.fi/bitstream/handle/10024/109781/thesis_Mulari_Wilmi_6.10..pdf?sequence=1

INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 5.928

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

9710 583 466 | 9710 583 466 | ijmrset@gmail.com

www.ijmrset.com