SOCIETY: Jurnal Pengabdian Masyarakat,
Vol. 5, No. 2 (2026): Januari, pp. 73- 79
E-ISSN:2827-878X (Online -Elektronik)

Implementation of Playwright as an Automation Tool for Functional Web Application Testing

Gina Nurbilkis[1]*, Alun Sujjada[2]
[1,2] Program Studi Teknik Informatika, Universitas Nusa Putra Sukabumi
gina.nurbilkis_ti22@nusaputra.ac.id

ARTICLE INFO

Article history
Received : 12-6-2025
Revised : 5-1-2026
Accepted : 26-1-2026

Keywords
Playweigt, automated Testing, functional Testing, QA improvements, Web applications

ABSTRACT

The rapid development of web technology has increased the complexity of modern applications, requiring efficient and accurate testing methods. PT. Neuronworks Indonesia previously used manual and semi-automated testing through the Cypress framework, which has limitations in cross-browser support and script maintenance. To address these issues, a community service activity was carried out by implementing the Playwright framework as a tool for functional web application test automation. The activity included training, test environment preparation, and direct implementation on the company's web project. The results showed an improvement in regression testing efficiency from around three hours to one hour per cycle. Test stability and consistency increased by approximately 20% with the support of multi-tab features and cross-browser testing. This activity not only improved testing performance but also strengthened the QA staff's technical competence in using modern automation tools. The implementation of Playwright proved effective in optimizing the quality assurance process and is recommended for broader adoption in web application testing within the industry.

A. INTRODUCTION

The rapid development of web technology has led to increasing complexity in web applications(Jazayeri, 2007) developed by PT. Neuronworks Indonesia, particularly those designed to support partner business operations. As a company that provides digital solutions for various partners, PT. Neuronworks Indonesia is required to ensure that the applications delivered operate reliably, stably, and in accordance with predefined business requirements. Functional issues in web applications may directly affect partner business processes and reduce system reliability.

In the software development process, PT. Neuronworks Indonesia frequently manages applications that undergo continuous feature enhancements and adjustments based on partner demands. These ongoing changes increase the potential risk of

functional defects(Inzitari et al., 2007) if testing activities are not conducted in a structured and effective manner. Therefore, a well-defined testing process is essential to ensure that system updates do not negatively impact existing functionalities and that all features continue to function as expected.

Functional testing is an essential activity to validate whether application features operate correctly(Bühler & Wegener, 2008) according to specified requirements and partner business needs. This type of testing focuses on verifying system behavior from the user perspective, including the correctness of input handling, process execution, and output generation. Through proper functional testing, PT. Neuronworks Indonesia can ensure that the delivered applications support partner business operations effectively. However, the manual testing approach previously implemented required considerable time(Taipale et al., 2011), involved extensive human effort, and was vulnerable to human error, particularly when repetitive testing was required across multiple application versions.

The automated testing approach emerges as a solution(Portolan, 2019) capable of overcoming the limitations of manual testing. Automated testing allows the verification of application functions to be performed quickly, repeatedly, and consistently every time a code change occurs. One of the rapidly growing frameworks in this field is Playwright, an open- source framework developed by Microsoft.

Playwright is designed to perform end-to-end testing(Talakola, 2024) on various modern browsers such as Chrome, Firefox, and WebKit. Its main advantages lie in its ability to handle multi-tab, multi-page, and more stable cross-platform support compared to previous frameworks such as Selenium or Puppeteer. Its simple and efficient API makes Playwright easier for developers to use and maintain.

The implementation of Playwright in the context of functional testing for web applications is still relatively new in educational institutions and small-scale development projects. Many developers still rely on older frameworks such as Selenium, which are more widely known. This implementation holds significant value in demonstrating the effectiveness of Playwright in improving the efficiency and accuracy of functional testing processes.

B. METHODS

This community service activity was conducted in collaboration with PT. Neuronworks Indonesia as an industrial partner in the field of software development. The implementation method focused on the application and training of the Playwright framework as an automation tool for functional testing of web applications. The web applications tested in this activity were primarily built using modern JavaScript-based frameworks, such as React and Vue.js, which are commonly used in PT. Neuronworks Indonesia's internal and partner-oriented systems. These frameworks rely heavily on dynamic rendering and asynchronous interactions, requiring a testing tool capable of handling complex user interface behaviors across different browsers. The activity was carried out through the following stages:

1. Partner Needs Analysis

This stage aimed to identify the testing processes currently applied at PT. Neuronworks Indonesia, including the challenges encountered in manual and semi-automated testing approaches. Particular attention was given to applications developed

with React and Vue.js, where dynamic components and frequent state changes often complicate manual testing. The analysis results were used to determine the most relevant areas for implementing Playwright, especially in user interface (UI) testing and validation of critical system functionalities.

2. Solution Design and Test Environment Preparation

Based on the analysis results, automated testing scenarios were designed using Playwright to accommodate the characteristics of modern web applications. This stage included installing the Playwright framework, configuring the testing environment, and developing test scripts tailored to the React- and Vue-based applications used by PT. Neuronworks Indonesia. Playwright was selected due to its ability to handle asynchronous operations, dynamic elements, and cross-browser testing efficiently, which aligns with the technical specifications of the applications being tested.

3. Training and Technical Assistance

The team provided direct training to technical staff or QA engineers at PT. Neuronworks Indonesia. The training covered the introduction of basic Playwright concepts, test script development, execution of automated tests, and integration with the CI/CD pipeline so that testing can run continuously.

4. Implementation and Evaluation

The Playwright framework was applied directly to one of the web projects currently being developed by the partner. Evaluation was carried out by comparing testing time efficiency, bug detection accuracy, and script maintenance convenience with the previous methods. The evaluation data served as the basis for measuring the success of the community service activity.

5. Report Preparation and Recommendations

The results of the activity were documented in a final report along with strategic recommendations for PT. Neuronworks Indonesia to implement Playwright sustainably in future testing projects. The recommendations also included potential integration of Playwright with other automation systems used by the company.

C. RESULTS AND DISCUSSION

This community service activity was carried out in collaboration with PT. Neuronworks Indonesia with a focus on implementing Playwright as an automation tool for functional testing of web applications that previously used Cypress. The main objective of this activity was to improve the efficiency, stability, and flexibility of the testing processes used by the company.

1. Initial Conditions and Problems

Before the implementation of Playwright, the Quality Assurance (QA) team at PT. Neuronworks Indonesia used Cypress as the main framework for end-to-end testing. Cypress proved helpful for test automation but had limitations in several technical aspects, such as support for multiple browsers, difficulty in multi-tab testing, and constraints when integrated with large-scale applications involving many asynchronous interactions.

SOCIETY: Jurnal Pengabdian Masyarakat,
Vol. 5, No. 2 (2026): Januari, pp. 73- 79
E-ISSN:2827-878X (Online -Elektronik)

The QA team reported that some testing scenarios required manual adjustments to run stably in Cypress, resulting in time-consuming script maintenance. This situation became one of the reasons for implementing Playwright as a more modern and flexible alternative.

2. Playwright Implementation Process

The Playwright framework was implemented through a series of activities including installation, configuration, creation of test scripts, and integration with the Continuous Integration/Continuous Deployment (CI/CD) pipeline used by PT. Neuronworks Indonesia.

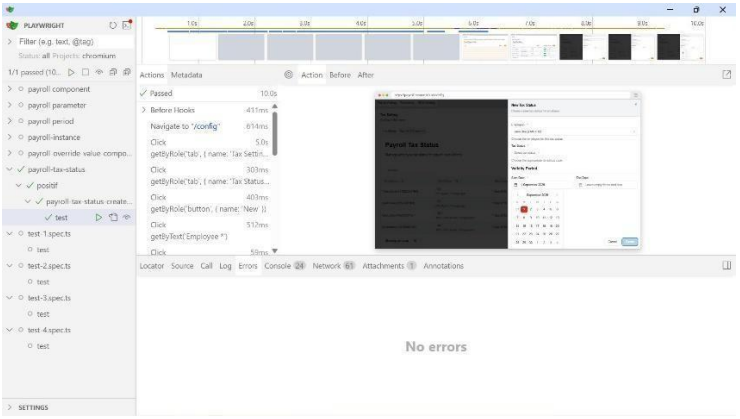

Figure 1 Results of the Playwright Implementation

The community service team also provided direct training to the QA staff on the basics of using Playwright, writing test scripts, and integrating test results into the existing system. The training was conducted in two sessions and attended by several QA members who were previously familiar with Cypress, making the adaptation process relatively fast.

3. Implementation Results and Evaluation

After the implementation process, Playwright demonstrated improved performance compared to Cypress in several aspects.

a. Execution time efficiency: Regression testing that previously required around 3 hours using Cypress could be completed in 1 hour with Playwright for equivalent scenarios.
b. Execution stability: Playwright was more stable in handling cross-browser testing such as Chrome, Firefox, and WebKit without requiring additional configuration.
c. Ease of script maintenance: Playwright's simple API and built-in wait mechanisms reduced the likelihood of test failures due to timing issues that frequently occurred in Cypress.
d. Support for multi-tab and parallel testing: These features increased testing efficiency for complex applications with multiple user interaction paths.

The evaluation results also showed that functional error detection increased by approximately 15–20% after the implementation of Playwright, particularly in tests involving simultaneous user interaction simulations.

4. Impact and Benefits for the Partner

The implementation of Playwright provided positive impacts on the quality

assurance process at PT. Neuronworks Indonesia. The QA team now has an alternative framework that is faster and more flexible for handling large-scale projects. In addition, this community service activity enhanced the technical capabilities of QA staff in mastering modern testing tools relevant to current industry needs.

The adoption of Playwright helped the company achieve time efficiency, reduce manual workload, and improve the consistency of testing results. These outcomes indicate that Playwright-based automated testing is feasible to be established as a new standard to support the software development process within the company.

D. CONCLUSION

The community service activity carried out in collaboration with PT. Neuronworks Indonesia successfully demonstrated that the implementation of Playwright as an automation tool for functional testing of web applications can improve the efficiency and reliability of the quality assurance process. This framework provides more stable, faster, and easier-to-maintain results compared to Cypress, which was previously used by the company.

The implementation of Playwright also provided added value for the QA team in understanding modern testing technologies that support multiple browsers and complex scenarios such as multi-tab and parallel testing. The increase in execution time efficiency and consistency of test results helped the company accelerate the software development cycle without reducing the quality of the products produced.

This activity not only focused on the implementation of technology but also on enhancing human resource capacity through training and guidance. Its impact can be seen in the improved ability of QA staff to independently develop and manage automated testing.

Overall, the implementation of Playwright proved effective as an automated testing solution that can be widely adopted in industrial environments. This framework can serve as a viable alternative to strengthen software testing strategies at PT. Neuronworks Indonesia as well as in similar companies seeking to transition to a more efficient and modern testing approach.

E. ACKNOWLEDGEMENTS

The author expresses sincere gratitude to the lecturers of the Informatics Engineering Study Program at Nusa Putra University, Sukabumi, for their guidance, direction, and support during the implementation of this community service activity. Appreciation is also extended to PT. Neuronworks Indonesia for the collaboration and the opportunity provided, which enabled the implementation of Playwright as an automation tool for functional testing of web applications to be carried out successfully.

F. AUTHOR CONTRIBUTIONS

All authors contributed significantly to the implementation of this community service activity. Gina Nurbilkis was responsible for designing the activity concept, implementing the Playwright framework, and preparing the final project report. Alun Sujjada contributed to the supervision process, technical validation, as well as the

preparation and editing of the manuscript for publication. Both authors are jointly responsible for the content and authenticity of this work.

G. REFERENCES

Bühler, O., & Wegener, J. (2008). Evolutionary functional testing. *Computers & Operations Research*, *35*(10), 3144–3160.

Inzitari, D., Simoni, M., Pracucci, G., Poggesi, A., Basile, A. M., Chabriat, H., Erkinjuntti, T., Fazekas, F., Ferro, J. M., & Hennerici, M. (2007). Risk of rapid global functional decline in elderly patients with severe cerebral age-related white matter changes: the LADIS study. *Archives of Internal Medicine*, *167*(1), 81–88.

Jazayeri, M. (2007). Some trends in web application development. *Future of Software Engineering (FOSE'07)*, 199–213.

Portolan, M. (2019). Automated testing flow: The present and the future. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *39*(10), 2952–2963.

Taipale, O., Kasurinen, J., Karhu, K., & Smolander, K. (2011). Trade-off between automated and manual software testing. *International Journal of System Assurance Engineering and Management*, *2*(2), 114–125.

Talakola, S. (2024). Automated end to end testing with Playwright for React applications. *International Journal of Emerging Research in Engineering and Technology*, *5*(1), 38–47.