Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

Comparison of Automation Testing On Card Printer Project Using Playwright And Selenium Tools

Ni Luh Putu Melya Wati[1)], I Made Dwi Putra Asana[2)*], Ni Wayan Suardiati Putri[3)], Ketut Jaya Atmaja[4)], I Gede Iwan Sudipa[5)]
[1) 2*)3)4)5)]Program Studi Teknik Informatika, Fakultas Teknologi dan Informatika, Institut Bisnis Dan Teknologi Indonesia, Bali, Indonesia
[1)]luhlia75@gmail.com, [2)*]dwiputraasana@instikia.ac.id, [3)]suardiatiputri@instiki.ac.id, [4)]ketutjayaatmaja@instiki.ac.id, [5)]iwansudipa@instiki.ac.id

ABSTRACT
The quality of the software is greatly determined by the testing phase, which involves various test cases that can be conducted through manual testing and automation testing. Manual testing is performed manually without using automation scripts, whereas automation testing is conducted using automation scripts. ABC is a company that operates globally in the field of access control, with the Card Printer being one of the menus used in access control. In the development process of this software, both manual and automation testing phases are carried out. The automation testing process employs the Selenium tool, which has proven to be time-consuming and poses challenges when running numerous test cases. This research aims to develop automation testing using Playwright to address the long execution time issue encountered with Selenium. The research utilizes the Card Printer project in the development of automation testing and adopts the Agile methodology. The result of developing automation testing using Playwright was successfully applied to 12 test cases. Additionally, the time analysis between Playwright and Selenium showed that Playwright has a total execution time of 4.9 minutes, which is faster compared to Selenium's total execution time of 8.3 minutes. With faster execution times, Playwright can be considered a tool in the development of automation testing.

Keywords: Automation Testing; Playwright; Selenium; Agile

1. INTRODUCTION

Software quality is determined from the testing stage with several test cases. The testing stage can be done in two ways, namely manual testing and automation testing, manual testing is a software testing method where a tester tests manually without the help of automated tools or scripts, while automation testing is a software testing method where the tester uses automated tools and scripts to automate the testing process. The testing stage becomes a challenge when there are quite a lot and complex test cases. In this case, of course, a method is needed to create a software testing stage that automatically runs several test cases, so this method is needed in the era of dynamic, complex and fast software development.

ABC is a leading global provider of access control, time & attendance, and biometrics solutions. To control this abc company has a platform. Where this platform has web-based, open, and integrated security that provides comprehensive functionality for access control and time & attendance. In this platform there are many menus that have various functions each in access control, time and attendance. Card Printer is one of the menus contained in this platform. Card Printer is a feature that allows users to print user information via a card printer. In the development process of each menu, the testing stage is carried out based on the test cases that have been prepared by the Quality Assurance section of each team. The testing process uses two methods, namely manual testing and automation testing. In automation testing itself uses Selenium and Protractor tools.

The use of selenium to run automation on the Card Printer project takes a long time to run 12 test cases on average for 5 - 6 minutes, often also experiencing delays in executing automation code or the page does not respond to anything. This is included in the slow category for running 12 test cases, this is an obstacle if there are many test cases run and the deployment time is close, it will take more time than this. In research (Bhagat dkk., 2020) on the comparison of automation testing tools which states that programming skills in Selenium are needed in addition to making tests as well as integrating with other tools. It is also explained that the script creation time in Selenium falls

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

into the slow category. The same thing is also found in research (Mobaraya dan Ali, 2019) that there is a difference in the running time of automation testing for 10 test cases using selenium and cypress, although it is not too significant the difference in time between using selenium and cypress but this is certainly a challenge when it must be deployed in the near future and the number tested is large. So that tools that have speed in testing are very calculated.

Playwright is the latest open source cross-browser automation testing tool developed by Microsoft that provides APIs for controlling web browsers. In the research journal (Pogudin, 2023) displays a comparison table where Playwright has faster performance than Selenium and is very easy to use as an automation testing tool. In addition, in research (Amalia dan Cahyono, 2019) on Playwright Utilization Analysis for Automated Testing of Web-Based Applications getting results, the percentage of passing the test is 100%. The ease of Playwright Codegen in creating automated test scripts directly when recording test cases is very helpful for testers. In this study also mentions the comparison of using selenium with playwright, one of which is about the ease of use of playwright for testers who are new to automation testing. So that playwright can be a solution in overcoming the weaknesses of selenium.

From the above problems and literature review, the researcher proposes research on "Comparison of Automation Testing on Project Card Printer Using Playwright and Selenium".

2. LITERATURE REVIEW

This research is inseparable from previous studies and research. As reference material in this study, several summaries of previous research related to the research to be conducted by the author will be included:

Research (Biju dan Ali, 2020) entitled "Automation Of Purchase Order In Microsoft Dynamics 365 By Deploying Selenium". This research discusses the importance of regression testing in dynamic verification, with a focus on automation of manufacturing orders in Microsoft Dynamics 365. The results in using Selenium in the automation testing process get a test result pass, so that the application of this automation can help empower companies to reduce resource allocation.

The research (Budiman dkk., 2023) entitled "Selenium Effectiveness in Testing the Functionality of Web-Based Cashier Applications with the Blackbox Method". Discussing the impact of computer use in increasing efficiency and productivity in various aspects of human life, as well as to evaluate the effectiveness of using Selenium in testing the functionality of web-based cashier applications using blackbox. The test results show that the use of Selenium in testing the functionality of web-based cashier applications with the blackbox method can increase the effectiveness and efficiency of testing, and ensure the application can function properly.

Furthermore, research (Pogudin, 2023)that discusses "End-To-End Test Implementation For A Saas Platform". This research explores the implementation of End-to-End (E2E) test protocols for the Software-as-a-Service (SaaS) platform ordered by Mindhive Oy. The result of implementing Playwright for automation testing is very good, although this development is still basic, but at the same time very versatile. It is very useful for testers to ensure that the functionality required by users is achieved and works, so as to minimize the appearance of new bugs.

Research (Anjum dkk., 2023) "Automation of Grocery Store Web Application Testing Using Selenium IDE". In this study, it discusses how to carry out the testing process automatically using the Selenium IDE tool. From the research by testing the Monotaro.id Sembako shop web application software automatically using the Selenium IDE application on Google Chrome that has been done, it can be concluded that when we do the test we can find out the functions of the menus or features contained in the page on the website whether it really works or there is a feature / interface error on the website.

Research (Amalia dan Cahyono, 2019) entitled "Analysis of Playwright Utilization for Automatic Testing of Web-Based Applications". This research discusses the results of analyzing the implementation of automated testing in web-based applications. The test tools that will be discussed are Playwright and Selenium. After analysis and testing, the results of the Network Management System in the Create Link process were successfully carried out by Playwright with a percentage of 100% test pass. The ease of Playwright Codegen in creating automated test scripts directly when recording test cases is very helpful for testers.

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

3. METHOD

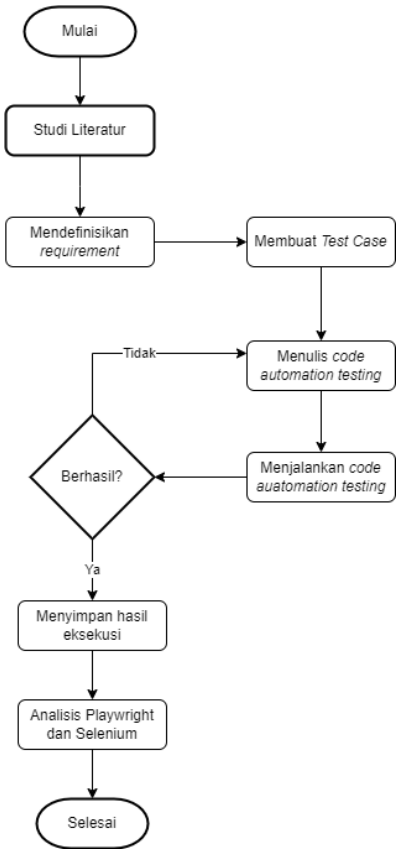

Fig 1. Automation Testing Overview

The first stage is a literature study where looking for theories used in this research, then entering into the process of defining this requirement will begin with an explanation of the flow of the Card printer project and at the same time to collect all requirement documents regarding the Card Printer project. After all the requirements are clear and collected, continue with making test cases in accordance with the requirements and scenarios that may occur when users use the Card Printer, after all test cases are completed, proceed to the process of writing code automation testing in accordance with the test cases that have been determined. After the process of writing code automation testing is complete, all code automation testing based on the test case will be executed, if the execution results are successful it will proceed to the process of saving the execution results, if the results fail it will return to the process of writing code automation testing, in this process it will be seen whether there are errors in the code automation testing that has been made or there are updates from the Card Printer project. The execution results that have been successfully saved will proceed to the process of analyzing the comparison of the use of Playwright and Selenium tools, in this process the performance of which tools is faster will be analyzed.

Automation Testing

Automation testing is a software testing technique that is performed using specialized software to run a series of test cases. Automated testing software can also feed test data into the Tested System, compare expected and actual results, and generate detailed test reports. Successive development cycles will require the execution of the same test suite repeatedly. By using automated testing tools, it is possible to record these test suites and replay them as needed. The goal of Automation is to reduce the number of manually executed test cases and not to eliminate Manual Testing (Thooriqoh dkk., 2021).

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

Automated testing, automates not only test case execution but also test case generation and test result verification. Fully automated testing systems can run software without user participation. Instead, they often provide test coverage, a list of classes to be tested in other words, they only need to indicate what is tested, not how (Brahmbhatt, 2023).

Selenium

Selenium is an open-source software that works for web-based applications in all browsers and also supports various scripting languages and programming languages such as Java, Python, PHP, and JS. (Fatima dkk., 2023) This test script can be run on various browsers such as Chrome, Safari, and Firefox and supports various operating system platforms such as Windows, Linux, and Mac OS.
Selenium is not just one tool, but a package of tools consisting of (Lathwal, 2019):

1. Selenium Integrated Development Environment (IDE)
2. Selenium Remote Control (RC)
3. Selenium WebDriver
4. Selenium Grid

Integrating artificial intelligence (AI) into Selenium-based automation has the potential to improve the efficiency, adaptability, and intelligence of test procedures. Moreover, it empowers testing to focus on activities that provide significant value, such as formulating test scenarios and supervising results, while routine and repetitive tasks are delegated to AI-driven automation (Khankhoje, 2023).

Selenium is an open source tool recommended by W3C in 2018. It is used for test automation of various types of online applications across different browsers and platforms. Selenium is not just one tool but a software package. Selenium tools consist of Selenium IDE, Selenium RC, Selenium WebDriver, and Selenium Grid (Tibell dan Kholi, 2023).

Playwright

Fig 2. Playwright Logo

Playwright is a relatively new open source automated testing tool developed by Microsoft. It entered the market in 2020, and since then, its active user count has reached the 23,200-user mark. From the first version of the tool to date, a total of 94 versions have been released showing how active Microsoft is in fixing bugs and adding new features (Microsoft, 2023).

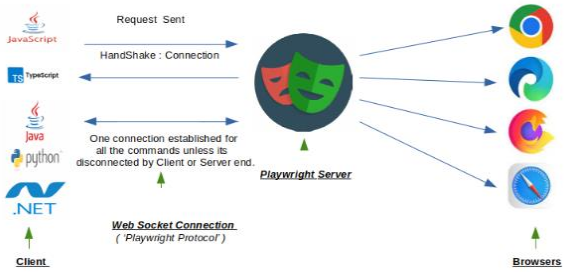

Fig 3. Playwright Architecture

Playwright has a cross-browser, cross-platform, and cross-language test platform that supports Chromium, WebKit, and Firefox. Its flexibility allows testing on Windows, Linux, and macOS, both locally and on CI (Continuous Integration). The Playwright API can be used in TypeScript, JavaScript, Python, .NET, and Java.

With a focus on stability, Playwright eliminates the need for artificial wait times with features such as auto-wait and web assumption first. Auto-wait allows Playwright to wait for elements before performing actions, while web-first assumption is designed for the dynamic web with automatic iteration until conditions are met.

Playwright provides robust tracking, including test repetition strategy configuration, execution trace capture,

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

video, and screenshots to address instability. The browser runs web content in a separate process, following modern browser architecture to be free from the constraints of in-process test runners.
With full isolation, Playwright creates a browser context for each test, equivalent to a new browser profile, providing fast execution speed. New browser contexts can be created on the fly, providing full isolation without overhead.

Other features include a coder, Playwright inspector, and trace viewer. The code generator enables test generation by recording user actions, while the Playwright inspector facilitates page and step checks through test execution. The trace viewer provides comprehensive information for investigating test failures, including execution screencast recordings, live DOM snapshots, action explorers, and test sources.

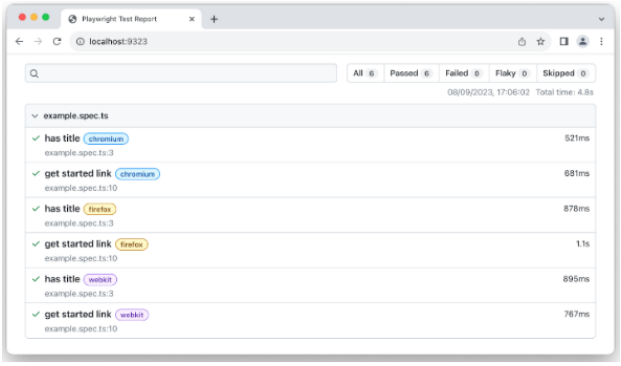

Fig 4. HTML reporter

Upon completion of the test, Playwright generates an HTML Reporter that creates a full report of the test. This report allows filtration by browser, test status (passed, failed, skipped, not passed), and allows users to explore test errors as well as each step taken. Automatically, the HTML report is opened if there are failures in multiple tests.

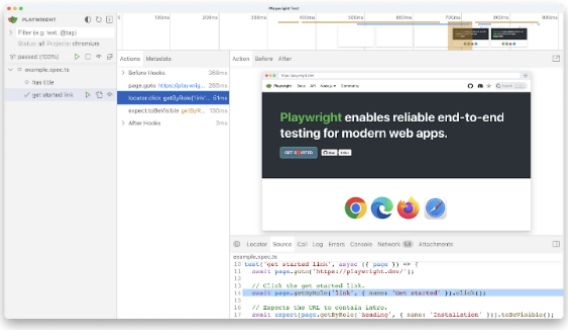

Fig 5. Example Test With *UI Mode*

For a better developer experience, Playwright supports testing in UI Mode. This mode allows time-travel debugging, watch mode, and various other features to improve development efficiency.

In a very short time Playwright became very popular as it allows writing tests in various languages such as TypeScript, JavaScript, Python and C# and testing on web browsers such as Chrome, Firefox, Safari, and Edge, which confirms how flexible the tool is. The tool supports testing on various platforms such as Windows, Linux and Mac-OS. Playwright supports functional testing, end-to-end software testing (eng. end-to-end) and API testing and various types of test cases. The mentioned tool already contains reporting components such as JSON, Line, Dot or HTML, and creates precise reports on the tests performed (GH, 2020).

Metode Agile

Agile is a software development method that emphasizes team collaboration, flexibility, and adaptation to changes that occur during the development process. It allows teams to work iteratively and incrementally, with a focus on delivering high-value products in a short period of time. Agile also emphasizes the use of tools and techniques that enable teams to communicate and collaborate more effectively, such as daily stand-up meetings, sprint planning, and

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

retrospective meetings. In software development, Agile has proven effective in improving product quality, speeding up delivery time, and increasing customer satisfaction (Hakam dkk., 2022).

Agile is one of the methods in the software development process, this method is very flexible and emphasizes adaptation to changes that occur during the software development process.

In this research, the author uses the Agile method as a reference in comparing automation testing on the Card Printer project using Playwright and Selenium. The use of agile methods is very flexible in the software development process and emphasizes the process of adapting to changes in the software development process so as to speed up the process of delivering a software. The same thing is conveyed in research conducted by (Nurzaman, 2020) the use of agile methods can accelerate the delivery of software gradually and is fast, lightweight, flexible.Figure 5. shows the agile method in this research.

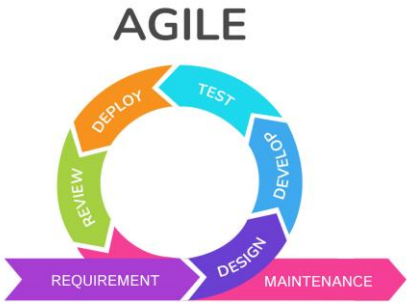

Fig 6. Agile Method

Requirement

In the first stage is the requirement stage, which compiles various needs such as requirement documents from the Card Printer project, tools that will be used in making code automation testing, namely Playwright and creating user stories that can be achieved in short sprints. This document will be a reference in creating test cases and test plans for execution into code automation testing.

Design

The second stage in this research is the design stage, where after the stage of compiling the requirement document, it continues to design detailed and clear test cases and test plans, as well as establishing a strategy for creating code automation testing. In the design stage, it must ensure that the test cases made are in accordance with the specifications and have covered all possible scenarios so that no problems will arise after the test cases are executed in automation testing. So that this stage can minimize the appearance of bugs in the Card Printer project.

Develop

The third stage in this research is the develop stage, where at this stage we will start to code automation testing for the Card Printer project according to the test cases and test plans that have been made previously at the design stage. After the develop stage is complete, before proceeding to the test stage, initial testing or initial trials must be carried out on each test case to ensure that the code created is functioning properly and is in accordance with the test cases and test plan.

Test

The fourth stage in this research is the test stage, where after the develop stage is completed and initial testing is carried out on each test case, then testing is carried out as a whole of the existing test cases, if the results of the execution of code automation testing pass, a statement is given in the test case that the test case has passed the execution using code automation testing. And the function of this test is to see if the code automation testing collides or nudges the code from the Card Printer project development team so that it causes bugs in the system.

Deploy

The fifth stage in this research is the deploy stage, where the testing automation code that has been tested can be combined with the code from the development team. So that each team can execute the code and see the results whether on their device the Card Printer project passes or fails.

Review

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

The sixth stage in this research is the review stage, where the code automation testing for the Card Printer project is reviewed by the team lead of this project to ensure that the code created is in accordance with the structure and easy to maintain in the future. The review stage is also carried out to evaluate the results of automation testing together with the development team, to get feedback from all teams for continuous improvement and improvement of code automation testing.

4. RESULT

In research on the comparison of automation testing on card printer projects using playwright and selenium tools, using agile methods for the implementation stage of automation testing using playwright. In addition to describing the implementation using agile methods, there will be a discussion of the analysis of the comparison time between playwright and selenium tools and a discussion of the challenges in implementing playwright.

Requirement

In this stage, the collection and discussion of the requirements of the card printer project are carried out. Requirements are given in the form of pdf files and designs of card printers that will be made, after getting the requirements followed by a discussion of the card printer section in the requirements to better understand the expectations to be achieved in each process of the card printer project, so that during the process of making test cases and test plans there are no step errors or expected results that cause delays in the automation testing process.

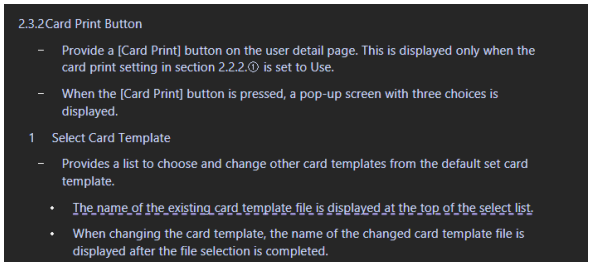

Fig 7. Printer Card Requirement

Figure 4.1 shows the contents of the requirement project card printer, where the card print button will be in the user details. The card print button will appear when the card printer settings in the settings menu are changed to use. When pressing the print card button, a pop-up screen will appear to select a card template, this card template will be provided when setting up the card printer in the settings menu. In addition to collecting requirements regarding the card printer project, setup tools playwright will be used to create code automation testing. The following are the steps in setting up the playwright tools using visual studio code:

1. Install the Playwright test for VS Code extension from masketplace or from the extensions tab in VS Code.

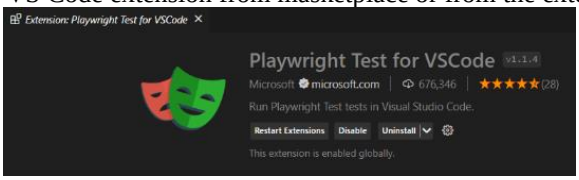

Fig 8. nstall Playwright Test Extension

2. Once installed, open the command panel and type: **Install Playwright**

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

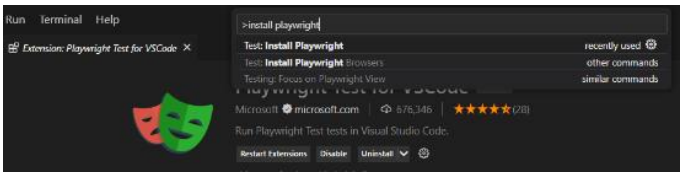

Fig 9. Install Playwright

3. Select Test: Install Playwright and select the chromium browser as the temporary browser to install playwright. This can later be further configured in the playwright file.

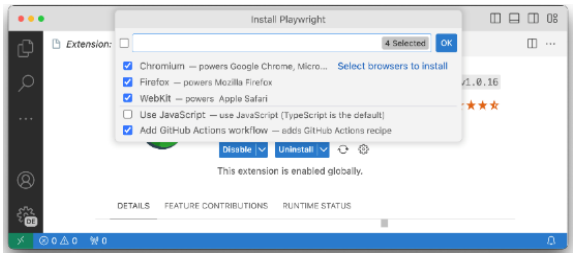

Fig 10. Select Browser to run Playwright

4. After successfully installing **Playwright**, open the **playwright.config.ts** file. In the projects property, comment the browser that is not used to run automated tests. In this research, the author uses the Chrome browser, so it does not need to be commented.

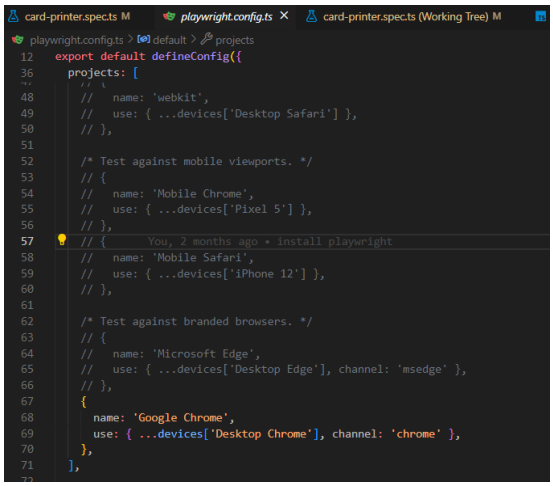

Fig 11. Chrome browser usage

Design

At the design stage, after collecting and discussing the requirements in the printer card project, it is continued with the test case design process, in the process of making test cases it must be in accordance with the specifications and include all scenarios that may occur when testing using automation testing, where the design of this test case will contain name, objective, precondition, step by step, test data, result, e2e test code, api test case. From the requirements given, it produces 12 test cases as a reference in making automation testing.

Develop

After the design stage is complete, the next stage will begin the creation of code automation testing based on the test cases that have been made previously in the design stage. This code automation testing uses playwright which has previously been set up and uses the typescript programming language. One of the code automation testing card

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

printers using playwright can be seen in Figure 4.7.

```
test('1. Login to BioStar 2 and access Badge setting', async ({ page }) => {
  await page.locator('button:has-text("Settings")').click();
  const isCardPrinterBtnPresent = await page.locator('li.cardPrinter a').isVisible();
  expect(isCardPrinterBtnPresent, 'Card printer button should be available').toBeTruthy();
  await page.locator('li.cardPrinter a:has-text("Card Printer")').click();
  let isTooltipPresent = await page.locator('[tooltip-class="customTooltip"]').isVisible();
  let comboboxElement =  page.locator("//div[@ng-model='testPrint.selectedCard']");
  let isCardComboboxDisabled = await comboboxElement.getAttribute('disabled');
  let btnTestPrintDisable = page.locator('button.btnTestPrint');
  let isBtnTestPrintDisabled = await btnTestPrintDisable.getAttribute('disabled');
  let isApplyButtonPresent = await page.locator('button[ng-click="doApplyCardPressoSetting();"]')
  let isCancelButtonPresent = await page.locator('button[ng-label="button.cancel"]').isVisible();
  const ipAddressElement = await page.$('input[model="cardPressoSetting.ipAddress"]');
  if (ipAddressElement) {
    const ipAddressAttributeValue = await ipAddressElement.getAttribute('value');
    if (ipAddressElement === null) {
      throw new Error('Element not found');
    }
   expect(ipAddressAttributeValue).toBe(testCardPressoConfig.ipAddress);
  }
```

Fig 12. Code Login and Access Badge Setting

Test

At the test stage, we will start running the initial test for each code automation testing that has been made, based on the existing test cases. After each successful test case is continued to run the overall code automation testing based on the 12 test cases made, if the results of this execution are successful, a statement is given in the test case that the test case made into the code automation testing has run according to the step by step in the test case. This test stage also includes verifying whether the code automation testing that has been made provides consistent and reliable results from testing a system. The execution results of the code automation testing login and access badge settings can be seen in Figure 12

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TEST RESULTS  TERMINAL  PORTS  GITLENS

Running 1 test using 1 worker
[Google Chrome] › card-printer.spec.ts:72:7 › Card Printer › 1. Login to BioStar
 2 and access Badge setting
combobox should be disabled
btn test print should be disabled
  Slow test file: [Google Chrome] › card-printer.spec.ts (18.3s)
  Consider splitting slow test files to speed up parallel execution
  1 passed (19.9s)
```

Fig 13. Execution result of test case login to biostar 2 and access badge setting

The image above displays the results of the execution of the Login and access badge setting test case, the results of the execution above pass with a time of 18.3s. The image above also displays several elements that have been successfully verified, namely combobox should be disabled and btn test print should be disabled.

Deploy

After all the tests have been made into code automation testing, proceed to the deploy process where this process combines the code automation testing that has been made with the code from the development team to be executed on each team device so that it can find out the results of this code automation testing combination pass or fail.

In addition to the process of combining code automation testing, a presentation of the test results was made to the development team. This presentation is to ensure the quality and functionality of the card printer project. This presentation shows the logs of the tests and displays the test results on the development team's device.

Review

At the review stage, apart from the development team, the method used is blackbox testing. Blackbox testing is used to ensure that the automation testing code created is in accordance with the expected scenario.
The results of blackbox testing for the 12 test cases that have been made are as expected from each test case.

Playwright and Selenium Analysis

In this research, execution time is a comparative factor in the automation testing process between playwright

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

and selenium. After the process of making automation testing using playwright, run automation testing for playwright and selenium to test the results of the run time of each tool. From the results of running automation testing project card printer using playwright and selenium, the time used as a comparison is made in a table 1.

Table 1
Execution Time of Playwright and Selenium

No	Scenario	Expected Result	Result Selenium (Execution time)	Result Playwright (Execution time)	Difference in Testing Time
1	Login to BioStar 2 and access Badge setting	CardPresso Setting toggle on default (Not Use) IP Address value 127.0.0.1 Port value 632 Tooltip on Printer Name Test Print Dropdown and Button disabled Apply and Cancel button on footer	7.4 second	6.2 second	1.2 second
2	Verify Print Card button	Print Card button not available below Add Photo button	5.4 second	5.1 second	0.3 second
3	Fill in Badge Setting	Inputbox can be filled Typed in card template added to Card Template Print Test combo box Use as default radio button selected	21.5 second	7.1 second	14.4 second
4	Verify Print Card button	Print Card button available below Add Photo button	5.0 second	5.0 second	0
5	Print Card on an empty New User page	Card is printed with default user information (ID)	10.3 second	10.9 second	0.6 second
6	Print Card on a filled New User page	Card is printed with filled user information (on step 3) User information is saved Return to User page	80.4 second	24.9 second	55.5 second
7	Print Card on existing User	Card is printed with selected user's information Printer card uses template 2 Return to User page	18.9 second	12.4 second	6.5 second
8	Print Card on existing User after modify information	Card is printed with updated user's information Printer card uses template 2 Updated user information saved Return to User page	54.1 second	31.0 second	23.1 second
9	Verify default template changed on Add and Edit	Print Card template selection window shows up Template 2 selected as	22.0 second	20.0 second	2 second

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

	User page after changing default template	default Print and Cancel button on footer			
10	Verify Print Card button not available after CardPresso setting is changed	Print Card button not available below Add Photo button	10.3 second	7.5 second	2.8 second
11	Verify Print Card setting with network environment	Success messagebox shows up Prepared card template can be printed	23.8 second	11.8 second	12 second
12	Verify Print Card with network environment	Card is printed with updated user's information Printer card uses template 2 Updated user information saved Return to User page	249.7 second	150 second	99.7 second
13	Average test time difference				18.17 second

From the time analysis table using playwright and selenium, the results of the time difference between playwright and selenium are very significant in testing automation testing on the printer card project. In the first test case between selenium has an execution time of 7.4 seconds and playwright has an execution time of 6.2 seconds with a time difference of 1.2 seconds While in the second test case, selenium has an execution time of 5.4 seconds and playwright has an execution time of 5.1 seconds with a time difference of 0.3 seconds, with these results it can be said that playwright has a faster execution time than selenium.

In the fourth test case, selenium and playwright have the same execution time of 5.0 seconds and in the fifth test case, selenium has an execution time of 10.3 seconds and playwright has an execution time of 10.9 seconds with a time difference of 0.6 seconds. This time selenium has a faster execution time. The twelfth test case has the longest execution time because the contents of the test case are verifying the print card with updated user information, verifying the print card using template 2, verifying the updated user information is saved back to the user page, so this one test case has a long line of code and requires a longer execution time than the other test cases. With the result of execution time on selenium is 249.7 seconds and the result of execution time on playwright is 150 seconds, with a difference in test time of 99.7 seconds.

The process of implementing Playwright for automation testing on card printer projects has the challenge of handling dynamic elements in card printer projects. Playwright needs to handle elements that may not always be present in the DOM at the same time, or elements that are dynamically loaded at different times. In overcoming this challenge, in each line of code playwright has wait loading, this serves to wait for data to be sent to the server and in the process of filling in the data, because if it does not have wait loading, the code automation testing will fail to execute because it is too fast for playwright to find the selector of the automation testing code created and the process of sending data to the server takes a few seconds to be stored. Another challenge that is often faced is the use of the typescript language in the creation of these scripts requires time to adjust in the creation of code automation testing.

DISCUSSIONS

From the results of the execution time data between playwright and selenium above, it can be concluded that there is one test case, namely the fourth test case which has the same execution time between selenium and playwright, one test case, namely the fifth test case which displays selenium has a faster execution time compared to playwright and there are 10 test cases that display that playwright has a faster execution time than selenium on the printer card project. Playwright has a total execution time of 294 seconds and selenium has a total execution time of 498 seconds,

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 3, July 2024
https://doi.org/10.47709/cnapc.v6i3.4362

Submitted : 22 July 2024
Accepted : 27 July 2024
Published : 31 July 2024

with an average test time difference of 18.17 seconds.

5. CONCLUSION

The development of automation testing for the card printer project using Playwright tools followed the stages of requirement, design, development, testing, deployment, and review. This approach was successfully applied to 12 test cases with a total execution time of 4.9 minutes. The comparative time analysis revealed that Playwright offers faster execution times for automation testing compared to Selenium. Additionally, Playwright's configuration is more beginner-friendly. However, a challenge with Playwright is handling elements that are dynamically loaded at different times, necessitating the addition of wait commands in the code. Moreover, adjusting to the TypeScript programming language for creating automation scripts is required.

6. REFERENCES

Amalia, A., dan Cahyono, A. B. 2019. "Analisis Pemanfaatan Playwright untuk Pengujian Otomatis Aplikasi Berbasis Web (Studi Kasus: Sistem Manajemen Jaringan)". Jurnal Universitas Islam Indonesia, *2*(2), 1–8.

Anjum, H., Imran, M., dkk. 2023. "Otomatisasi Pengujian Aplikasi Web Toko Sembako Menggunakan Selenium IDE". LOGIC : Jurnal Ilmu Komputer dan Pendidikan, *1*(2), 303–309. diambil dari https://journal.mediapublikasi.id/index.php/logic/article/view/1654.

Bhagat, B., Bhattacharjee, S., dkk. 2020. "Software Testing Techniques & Automation Tools". Mukt Shabd Journal, *9*(5), 5957–5962.

Biju, V., dan Ali, S. 2020. "Automation of Purchase Order in Microsoft Dynamics 365 by Deploying Selenium", 101–114. https://doi.org/10.5121/csit.2020.100610.

Brahmbhatt, K. H. 2023. "Comparative analysis of selecting a test automation framework for an e-commerce website E-kaubanduse veebisaidi testimise automatiseerimise raamistiku valimise võrdlev analüüs".

Budiman, A., Nur Rahman, M., dkk. 2023. "Efektivitas Selenium dalam Pengujian Fungsionalitas Aplikasi Kasir Berbasis Web dengan Metode Blackbox", *01*(01), 1–10. diambil dari https://journal.mediapublikasi.id/index.php/jriin.

Fatima, S., Nasim, S. F., dkk. 2023. "Comparative Study Of Software Automation Tools: Selenium And Quick Test Professional". Journal of Independent Studies and Research Computing, *21*(1), 52–60. https://doi.org/10.31645/jisrc.23.21.1.6.

GH, C. C. 2020. "Playwright Automation Framework: Tutorial". diambil 15 Desember 2023, dari https://www.browserstack.com/guide/playwright-tutorial.

Hakam, M. A., Triayudi, A., dkk. 2022. "Implementasi Metode Agile pada Sistem Manajemen Zakat Berbasis Website dengan Framework Laravel". Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi), *6*(1), 111–116. https://doi.org/10.35870/jtik.v6i1.393.

Khankhoje, R. 2023. "WEB PAGE ELEMENT IDENTIFICATION USING SELENIUM AND CNN : A NOVEL", (October), 0–17. https://doi.org/10.13140/RG.2.2.17110.42569.

Lathwal, A. 2019. "A Literature Review on Automation Testing Using Selenium+Sikuli". International Journal of Distributed Artificial Intelligence, *11*, 35–40. https://doi.org/10.4018/IJDAI.2019070104.

Microsoft 2023. "Playwright enables reliable end-to-end testing for modern web apps.". diambil 15 Desember 2023, dari https://playwright.dev/.

Mobaraya, F., dan Ali, S. 2019. "Technical Analysis of Selenium and Cypress as Functional Automation Framework for Modern Web Application Testing", 27–46. https://doi.org/10.5121/csit.2019.91803.

Nurzaman, F. 2020. "Pengembangan Sistem Otomatisasi Tagihan Menggunakan Metode Agile Software Development". Jurnal IKRA-ITH Informatika, *4*(1), 46–57.

Pogudin, V. 2023. "Vladislav Pogudin End-To-End Test Implementation Fos A Saas Platform".

Thooriqoh, H. A., Annisa, T. N., dkk. 2021. "Selenium Framework for Web Automation Testing: A Systematic Literature Review". JUTI: Jurnal Ilmiah Teknologi Informasi, *19*(2), 65–76.

Tibell, S., dan Kholi, M. 2023. "Choosing the Right Automated UI Testing Tool-A Comparative Study of Selenium and TestComplete".

* Corresponding author