 ISSN:2302-8149
e-ISSN:2540-9719

Analisis *Framework* Pengujian Otomatis untuk Aplikasi *Web* berdasarkan Kinerja dan Kegunaan

Analysis of Automated Testing Frameworks for Web Applications based on Performance and Usability

[1]Shidqi Adiatma*, [2]Ana Kurniawati
[1,2]Program Studi Magister Manajemen Sistem Informasi, Fakultas Teknologi dan Rekayasa, Universitas Gunadarma
[1,2]Jl. Margonda Raya No.100, Pondok Cina, Beji, Kota Depok, Jawa Barat, Indonesia
*e-mail: adiatma944@gmail.com

(***received*:** 24 April 2026, ***revised*:** 5 May 2026, ***accepted*:** 15 May 2026)

Abstrak

Kemajuan teknologi informasi dalam beberapa tahun terakhir telah mendorong percepatan transformasi digital di berbagai sektor bisnis. Hal ini ditandai dengan meningkatnya penggunaan aplikasi berbasis *web* yang menuntut kinerja sistem yang responsif, stabil dan andal. Namun, proses pengujian perangkat lunak masih sering dilakukan secara manual sehingga kurang efisien karena memerlukan waktu yang lama, menguras sumber daya, serta berpotensi menimbulkan kesalahan, terutama saat aplikasi mengalami pembaruan fitur secara berkelanjutan. Oleh karena itu, diperlukan pendekatan pengujian yang lebih efektif melalui *automation testing*. Penelitian ini bertujuan untuk menganalisis dan membandingkan beberapa *framework automation testing* guna menentukan *framework* yang paling sesuai untuk pengujian aplikasi *web*. Skenario pengujian disusun menggunakan pendekatan *Domain Specific Language* berbasis *Cucumber Gherkin* agar lebih mudah dipahami dan terstruktur. Evaluasi dilakukan berdasarkan dua aspek utama, yaitu *Automated Testing Progress* dan *Tools Usability*. Aspek *performance* meliputi *time complexity*, *covered test case*, *execution start* dan *element inspection*. Sementara itu, aspek *usability* mencakup *platform compatibility*, *supported browser*, *scripting language* dan *parallel execution*. Studi kasus dilakukan pada aplikasi logistik berbasis *web muatmuat.com* dengan fokus pada fitur *register*, *login* dan *reset password*. Data dianalisis menggunakan metode *Multiple Attribute Decision Making* (MADM) dengan pendekatan *The Distance to the Ideal Alternative* (DIA). Hasil penelitian menunjukkan bahwa Selenium WebDriver memiliki nilai Ri sebesar 0.25799, diikuti oleh Cypress (0.31747), serta Playwright dan WebDriverIO (0.33426), sehingga menjadi *framework* paling optimal.

Kata kunci: *automation testing, framework, performance, usability, the distance to the ideal alternative*

Abstract

Advances in information technology in recent years have accelerated digital transformation across various business sectors. This development is reflected in the increasing use of web-based applications that demand responsive, stable, and reliable system performance. However, software testing processes are still often performed manually, making them less efficient due to the significant time and resources required, as well as the potential for human error, particularly when applications undergo continuous feature updates. Therefore, a more effective testing approach through automation testing is required. This study aims to analyze and compare several automation testing frameworks to determine the most suitable framework for testing web applications. The testing scenarios were designed using a Domain Specific Language (DSL) approach based on Cucumber Gherkin syntax to ensure better readability and structured test cases. The evaluation was conducted based on two main aspects: Automated Testing Performance and Tools Usability. The performance aspect included time complexity, covered test cases, execution start time, and element inspection. Meanwhile, the usability aspect covered platform compatibility, supported browsers, scripting languages, and parallel execution capabilities. The case study was conducted on the web-based logistics application muatmuat.com, focusing on the register, login, and password reset features. The data were analyzed

 ISSN:2302-8149
e-ISSN:2540-9719

using the Multiple Attribute Decision Making (MADM) method with the Distance to the Ideal Alternative (DIA) approach. The results indicate that Selenium WebDriver achieved the lowest Ri value of 0.25799, followed by Cypress (0.31747), and both Playwright and WebdriverIO (0.33426), making Selenium WebDriver the most optimal framework among the evaluated alternatives.

Keywords: *automation testing, framework, performance, usability, the distance to the ideal alternative*

1 Pendahuluan

Perkembangan teknologi informasi dan digitalisasi telah membawa perubahan signifikan dalam berbagai aspek kehidupan, termasuk dalam cara manusia berinteraksi, bekerja dan mengakses informasi. *Website* sebagai salah satu komponen utama internet kini berperan penting sebagai media informasi, transaksi dan layanan digital. Peningkatan jumlah pengguna internet yang mencapai lebih dari 5,35 miliar orang secara global menunjukkan tingginya ketergantungan masyarakat terhadap aplikasi berbasis *web* [1]. Di Indonesia sendiri, pertumbuhan ekonomi digital, khususnya sektor *e-commerce* dan layanan berbasis aplikasi, turut memberikan kontribusi besar terhadap pendapatan negara dan mendorong transformasi digital di berbagai sektor industri [2].

Seiring dengan meningkatnya penggunaan aplikasi berbasis *web*, tuntutan terhadap kualitas perangkat lunak juga semakin tinggi. Aplikasi dituntut untuk memiliki kinerja yang cepat, stabil, aman, serta mampu memberikan pengalaman pengguna yang baik. Oleh karena itu, proses pengujian perangkat lunak *(software testing)* menjadi tahapan yang krusial dalam siklus pengembangan sistem. Namun, pengujian manual yang masih banyak digunakan dinilai kurang efisien karena membutuhkan waktu yang lama, rentan terhadap kesalahan manusia, serta sulit untuk menangani skala pengujian yang besar dan berulang [3]. Keterbatasan pengujian manual mendorong munculnya *automation testing* sebagai solusi untuk meningkatkan efisiensi dan akurasi pengujian. *Automation testing* memungkinkan eksekusi pengujian secara cepat, berulang dan konsisten dengan meminimalkan intervensi manusia. Saat ini, terdapat berbagai *framework automation testing* seperti Playwright, Cypress, WebdriverIO dan Selenium WebDriver yang digunakan dalam pengujian aplikasi berbasis *web*. Masing-masing *framework* memiliki karakteristik, keunggulan dan keterbatasan yang berbeda, sehingga pemilihan *framework* yang tepat menjadi faktor penting dalam menentukan keberhasilan proses pengujian [4].

Meskipun berbagai penelitian sudah dilakukan untuk membandingkan kinerja *framework automation testing*, hasil yang diperoleh menunjukkan bahwa variasi kinerja dan kemudahan penggunaan antar *framework* tergantung pada konteks penggunaan, parameter pengujian, serta lingkungan sistem. Hal ini menimbulkan permasalahan dalam menentukan *framework* yang paling optimal, khususnya jika mempertimbangkan lebih dari satu kriteria penilaian seperti kinerja dan kemudahan penggunaan [5]. Oleh karena itu, diperlukan sebuah metode pengambilan keputusan yang mampu mengevaluasi berbagai alternatif secara objektif dan komprehensif. Berdasarkan permasalahan tersebut, penelitian ini berfokus pada analisis perbandingan kinerja empat *framework automation testing*, yaitu Playwright, Cypress, WebdriverIO dan Selenium WebDriver, pada aplikasi berbasis *web*. Evaluasi dilakukan dengan mempertimbangkan aspek *performance* dan *usability* menggunakan metode *The Distance to the Ideal Alternative* (DIA). Metode ini dipilih karena mampu memberikan penilaian berdasarkan kedekatan setiap alternatif terhadap solusi ideal, sehingga menghasilkan keputusan yang lebih terukur dan sistematis [6].

Tujuan dari penelitian ini adalah untuk menganalisis kinerja keempat *framework automation testing* berdasarkan parameter *performance* dan *usability*, serta menentukan *framework* terbaik melalui metode DIA. Penelitian ini diharapkan dapat memberikan kontribusi bagi praktisi *Software Quality Assurance* dalam memilih *framework* yang sesuai dengan kebutuhan proyek, serta menjadi referensi bagi akademisi dan peneliti dalam pengembangan studi terkait evaluasi *automation testing* dan sistem pendukung keputusan.

2 Tinjauan Literatur

Perkembangan *automation testing* dalam beberapa tahun terakhir menunjukkan peningkatan signifikan dalam pemanfaatan berbagai *framework* untuk meningkatkan efisiensi dan kualitas

pengujian perangkat lunak. Studi empiris menunjukkan bahwa penggunaan *framework modern* mampu mempercepat proses pengujian, khususnya dalam aspek waktu eksekusi. Penelitian oleh Luh Putu Melya Wati menunjukkan bahwa Playwright memiliki waktu eksekusi sebesar 4,9 menit, lebih cepat dibandingkan Selenium yang membutuhkan 8,3 menit, sehingga menunjukkan keunggulan Playwright dalam efisiensi pengujian otomatis [7]. Temuan ini mengindikasikan bahwa pemilihan *framework* memiliki pengaruh signifikan terhadap kinerja pengujian, terutama dalam pengujian aplikasi berbasis *web* yang membutuhkan proses validasi cepat dan berulang.

Selain itu, variasi kinerja antar *framework* juga dipengaruhi oleh parameter evaluasi yang digunakan. Shtokal dan Smołka menyatakan bahwa WebdriverIO memiliki kecepatan eksekusi yang lebih tinggi dibandingkan TestNG, namun dengan konsumsi memori yang lebih besar, sehingga menunjukkan adanya *trade-off* antara kecepatan dan efisiensi sumber daya [8]. Penelitian lain oleh Profesio Putra dan Melia juga menunjukkan bahwa Selenium lebih unggul dalam kecepatan eksekusi dibandingkan Katalon Studio, tetapi Katalon memiliki kelebihan dalam fitur bawaan serta kemudahan penggunaan [9]. Hal ini menunjukkan bahwa evaluasi *framework* tidak dapat hanya didasarkan pada satu parameter, melainkan perlu mempertimbangkan berbagai aspek secara komprehensif.

Dalam upaya mengatasi kompleksitas tersebut, beberapa penelitian telah mengadopsi pendekatan *multi-criteria decision making*. Metode *The Distance to the Ideal Alternative* (DIA) menjadi salah satu metode yang banyak digunakan karena mampu memberikan pemeringkatan alternatif secara objektif. Penelitian oleh Fakhrii dan Setiani menunjukkan bahwa Github Copilot memperoleh peringkat terbaik berdasarkan kombinasi parameter teknis dan *usability* menggunakan metode DIA [10]. Selain itu, penelitian oleh Riza juga menemukan bahwa Katalon Studio memiliki nilai terbaik dalam evaluasi *framework automation testing* menggunakan metode yang sama, karena memiliki keseimbangan antara kinerja dan fitur [11]. Hal ini menunjukkan bahwa metode DIA efektif dalam mengintegrasikan berbagai kriteria evaluasi dalam satu model analisis.

Pada konteks pengujian berbasis *web*, penelitian oleh Prasetyo menunjukkan bahwa Cypress memiliki kinerja terbaik dibandingkan Selenium dan Playwright, khususnya dalam aspek kecepatan dan efisiensi pengujian [12]. Namun demikian, Penelitian dari Fatima menyatakan bahwa Selenium tetap unggul dalam fleksibilitas, dukungan *multi-platform*, serta komunitas yang luas, meskipun memerlukan konfigurasi tambahan untuk fitur tertentu [13]. Di sisi lain, penelitian oleh Ari Rifqi menunjukkan bahwa *tools* seperti Repeato mampu meningkatkan efisiensi pengujian berbasis visual, tetapi memiliki keterbatasan dalam menangani elemen yang kompleks [14]. Perbedaan hasil penelitian ini menegaskan bahwa setiap *framework* memiliki karakteristik yang berbeda sesuai dengan kebutuhan pengujian.

Meskipun berbagai penelitian telah dilakukan, masih terdapat beberapa keterbatasan yang signifikan. Sebagian besar penelitian hanya membandingkan dua atau tiga *framework*, sehingga belum memberikan gambaran yang komprehensif terhadap berbagai alternatif *framework* dalam satu studi. Selain itu, banyak penelitian yang hanya berfokus pada aspek kinerja teknis seperti waktu eksekusi dan penggunaan sumber daya, tanpa mengintegrasikan aspek *usability* secara menyeluruh. Beberapa penelitian juga menggunakan pendekatan deskriptif tanpa metode pengambilan keputusan yang kuat, sehingga hasilnya kurang objektif dan sulit untuk digeneralisasi. Selain itu, keterbatasan pada studi kasus yang digunakan juga menjadi kendala karena belum sepenuhnya mencerminkan kondisi nyata pada aplikasi berbasis *web* yang kompleks.

Berdasarkan celah penelitian tersebut, penelitian ini difokuskan pada analisis komparatif empat *framework automation testing*, yaitu Playwright, Cypress, WebdriverIO dan Selenium WebDriver, dengan pendekatan yang lebih komprehensif. Penelitian ini mengintegrasikan dua perspektif utama, yaitu *performance* dan *usability*, serta menggunakan metode *The Distance to the Ideal Alternative* (DIA) untuk menghasilkan pemeringkatan yang lebih objektif. Berbeda dengan penelitian sebelumnya, studi ini tidak hanya menilai aspek teknis seperti *time complexity* dan *covered test case*, tetapi juga memasukkan parameter *usability* seperti *platform compatibility*, *supported browser*, *scripting language* dan *parallel execution* dalam satu model evaluasi terpadu.

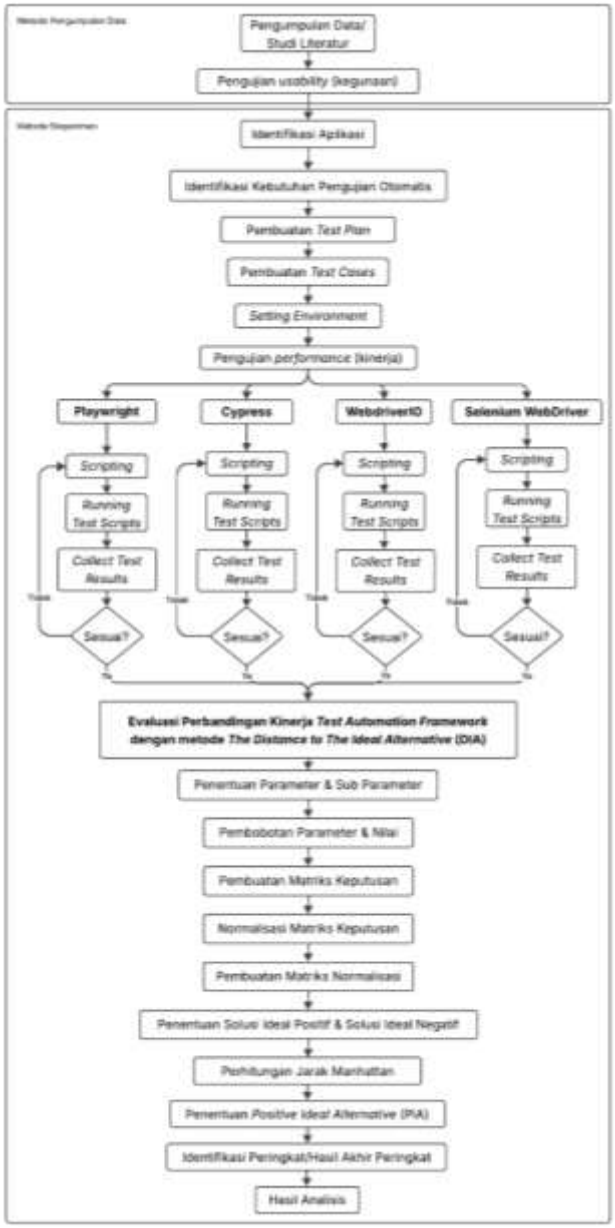
Dengan demikian, hipotesis yang diajukan dalam penelitian ini adalah bahwa terdapat perbedaan signifikan dalam kinerja *framework automation testing* ketika dievaluasi secara simultan dari perspektif *performance* dan *usability*, serta terdapat satu *framework* yang memiliki tingkat kedekatan paling optimal terhadap solusi ideal berdasarkan metode DIA. Penelitian ini diharapkan dapat memberikan kontribusi dalam bentuk model evaluasi yang lebih komprehensif serta menjadi referensi

 ISSN:2302-8149
e-ISSN:2540-9719

dalam pemilihan *framework automation testing* yang optimal pada pengujian aplikasi berbasis *web modern*.

3 Metode Penelitian

Metode atau tahapan penelitian yang dilakukan pada penelitian ini dapat dilihat pada Gambar 1.



Gambar 1 Tahapan penelitian

Tahapan penelitian disusun secara sistematis mulai dari pengumpulan data hingga penarikan kesimpulan untuk menentukan *framework automation testing* terbaik berdasarkan aspek *performance* dan *usability*. Berikut ini penjelasan dari tahapan penelitian yang dilakukan:

a. Pengumpulan data

Pada tahap ini dilakukan pengumpulan data melalui studi literatur dan eksperimen. Studi literatur dilakukan dengan mengkaji jurnal, buku, serta penelitian terdahulu yang relevan dengan *automation testing*, *framework* pengujian dan metode pengambilan keputusan. Selain itu, data eksperimen diperoleh dari hasil pengujian otomatis pada aplikasi berbasis *web* yang meliputi waktu eksekusi, hasil *test case*, serta parameter *usability* dari masing-masing *framework* [15].

b. **Identifikasi objek dan kebutuhan penelitian**

Tahap ini bertujuan untuk menentukan objek penelitian yaitu aplikasi berbasis *web muatmuat.com*, serta fitur yang akan diuji meliputi *register*, *login* dan *reset password*. Selain itu,

 ISSN:2302-8149
e-ISSN:2540-9719

dilakukan identifikasi kebutuhan pengujian seperti perangkat lunak (Visual Studio Code, Node.js, JDK, Git, *browser*) dan *framework* yang digunakan yaitu Playwright, Cypress, WebDriverIO dan Selenium WebDriver. Pendekatan yang digunakan adalah *Behavior Driven Development* (BDD) dengan *Domain Specific Language* (DSL) *Cucumber Gherkin*.

c. Perancangan pengujian *(Test Plan dan Test Case)*

Pada tahap ini dilakukan perancangan pengujian dengan menyusun *test plan* sebagai acuan pelaksanaan pengujian. Selanjutnya dibuat *test case* yang berisi skenario pengujian, langkah-langkah pengujian, data uji, serta *expected result*. *Test case* yang dibuat mencakup pengujian positif dan negatif untuk memastikan seluruh fitur berjalan sesuai kebutuhan [15].

d. Persiapan lingkungan *(Setup Environment)*

Tahap ini meliputi instalasi dan konfigurasi seluruh perangkat yang diperlukan untuk *automation testing*, termasuk pengaturan *environment variable* dan pengecekan kompatibilitas sistem. Tujuan dari tahap ini adalah memastikan seluruh *tools* dan *framework* dapat berjalan dengan optimal sebelum pengujian dilakukan [11].

e. Implementasi *automation testing*

Pada tahap ini dilakukan pembuatan *test script* berdasarkan *test case* yang telah disusun. *Script* ditulis menggunakan bahasa pemrograman JavaScript dengan sintaks Gherkin. Setelah itu, dilakukan eksekusi *test script* pada masing-masing *framework* secara berulang untuk mendapatkan hasil pengujian yang konsisten [15].

f. Pengumpulan hasil pengujian *(Test Result)*

Hasil dari eksekusi *test script* dikumpulkan dalam bentuk laporan pengujian yang berisi informasi seperti waktu eksekusi, jumlah *test case* berhasil dan gagal, serta detail hasil pengujian lainnya. Pengujian dilakukan dalam beberapa iterasi untuk memastikan validitas dan reliabilitas data [15].

g. Penentuan variabel penelitian

Variabel penelitian terdiri dari dua parameter utama, yaitu *Automated Testing Progress* dan *Tools Usability*. *Automated Testing Progress* meliputi *time complexity*, *covered test case*, *execution start* dan *element inspection*. Sedangkan *Tools Usability* meliputi *platform compatibility*, *supported browser*, *scripting language* dan *parallel execution*. Variabel ini digunakan sebagai dasar dalam mengevaluasi kinerja masing-masing *frameworkI* [11].

h. **Analisis data menggunakan metode *The Distance to the Ideal Alternative* (DIA)**

Analisis data dalam penelitian ini menggunakan metode *The Distance to the Ideal Alternative* (DIA) yang merupakan bagian dari pendekatan *Multiple Attribute Decision Making* (MADM). Metode ini digunakan untuk menentukan alternatif terbaik berdasarkan kedekatan terhadap solusi ideal positif dan menjauhi solusi ideal negatif [6]. Tahapan metode DIA dijelaskan sebagai berikut:

1. **Penentuan parameter dan sub-parameter**
   Penentuan parameter dan sub-parameter bertujuan untuk mengidentifikasi seluruh kriteria yang digunakan dalam evaluasi, sehingga proses analisis dapat mencerminkan kondisi nyata kinerja *framework* secara menyeluruh [6].
2. **Pembobotan parameter**
   Setiap parameter diberikan bobot sesuai tingkat kepentingannya untuk memastikan kontribusi masing-masing kriteria terhadap hasil akhir [6].
   $$\sum_{j=1}^{n} w_j = 1 \qquad (1)$$
3. **Penyusunan matriks keputusan**
   Nilai hasil pengujian dari masing-masing *framework* disusun ke dalam matriks keputusan sebagai dasar analisis [6].
   $$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \qquad (2)$$
4. **Normalisasi matriks keputusan**
   Normalisasi matriks keputusan dilakukan untuk menyamakan skala antar kriteria sehingga dapat dibandingkan secara objektif [6].

 ISSN:2302-8149
e-ISSN:2540-9719

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \quad (3)$$

5. **Pembobotan matriks normalisasi**
   Pembobotan matriks normalisasi bertujuan mengintegrasikan nilai kinerja dengan bobot kriteria untuk menghasilkan matriks berbobot [6].
   $$v_{ij} = w_j \cdot r_{ij} \quad (4)$$
6. **Penentuan solusi ideal positif dan negatif**
   Penentuan solusi ideal positif dan negatif digunakan dalam menentukan nilai terbaik dan terburuk sebagai acuan evaluasi pada tahapan perhitungan jarak Manhattan [6].
   $$A^+ = \{\max(v_{ij})\}, A^- = \{\min(v_{ij})\} \quad (5)$$
7. **Perhitungan jarak Manhattan**
   Mengukur kedekatan setiap alternatif terhadap solusi ideal:
   $$D_j^+ = \sum_{i=1}^{m} | v_{ij} - A_i^+ \quad (6)$$
   $$D_j^- = \sum_{i=1}^{m} | v_{ij} - A_i^- | \quad (7)$$
8. **Penentuan *Positive Ideal Alternative* (PIA)**
   Penentuan PIA digunakan untuk mengukur alternatif terbaik berdasarkan jarak minimum ke solusi ideal positif dan maksimum dari solusi ideal negatif [6].
   $$PIA = \min(D_j^+), \max(D_j^-) \quad (8)$$
9. **Perhitungan nilai preferensi dan peringkat (R_i)**
   Menentukan peringkat alternatif berdasarkan kedekatan terhadap solusi ideal:
   $$R_i = \sqrt{(D_j^+ - \min D_j^+)^2 + (D_j^- - \max D_j^-)^2} \quad (9)$$
   Nilai R_i yang lebih kecil menunjukkan bahwa alternatif tersebut memiliki kinerja terbaik.

i. Penentuan hasil dan kesimpulan

Tahap akhir penelitian adalah menentukan peringkat pada masing-masing *framework* berdasarkan nilai R_i *framework* dengan nilai terbaik dipilih sebagai alternatif yang paling optimal [15]. Selanjutnya dilakukan penarikan kesimpulan serta pemberian rekomendasi berdasarkan hasil analisis yang telah dilakukan.

4 Hasil dan Pembahasan

4.1 Pengumpulan data

Berikut hasil pengumpulan data yang mempresentasikan kinerja *framework* dari aspek *usability* diperoleh studi literatur dan disajikan dalam sebuah matriks seperti yang terlihat pada Tabel 1.

Tabel 1 Matriks perbandingan *usability*

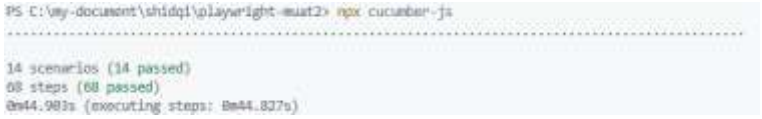
Kategori	Playwright	Cypress	WebDriverIO	Selenium WebDriver
Web Testing	Iya	Iya	Iya	Iya
Android Testing	Tidak	Tidak	Iya	Iya
iOS Testing	Tidak	Tidak	Iya	Iya
Desktop Testing	Tidak	Tidak	Iya	Tidak
API Testing	Iya	Iya	Iya	Tidak
Bahasa Pemrograman	TypeScript, JavaScript, Python, .NET, Java	JavaScript, TypeScript	JavaScript, TypeScript	Java, C#, Python, Ruby, JavaScript
Parallel Execution	Iya	Iya	Iya	Iya
Supported Browser	Chromium, Firefox, WebKit	Chrome, Firefox, Safari (eksperimental)	Chrome, Firefox, Safari, Edge	Chrome, Firefox, Safari, Edge

 ISSN:2302-8149
e-ISSN:2540-9719

4.2 Pengumpulan Hasil Pengujian *(Test Result)*

Setelah dilakukan perancangan *test plan*, penyusunan *test case* dan implementasi *automation testing*, tahap selanjutnya adalah pengumpulan hasil pengujian. Data diperoleh dari eksekusi *test script* pada masing-masing *framework* terhadap fitur *register*, *login* dan *reset password*. Hasil pengujian digunakan untuk merepresentasikan kinerja *framework* dari aspek *performance*. Berikut hasil pengujian dari *framework* Playwright dari iterasi 1 sampai dengan iterasi 10 dapat dilihat pada Gambar 2 dan Gambar 3.

Gambar 2 *Test result* playwright iterasi 1 sampai 9



Gambar 3 *Test result* playwright iterasi 10

Berikut hasil dari pengujian untuk keempat *framework* yang sudah dijumlahkan dan direkap dapat dilihat pada Tabel 2.

Tabel 2 Ringkasan hasil *automation testing*

Nama *framework*	*Time Complexity*	*Covered Test Case*	*Execution Start*
Cypress	675 detik 84 ms	140	48 detik 08 ms
Playwright	444 detik 87 ms	140	6 detik 95 ms
WebDriverIO	346 detik 31 ms	140	2 detik 19 ms
Selenium WebDriver	457 detik 82 ms	140	8 detik 46 ms

 ISSN:2302-8149
e-ISSN:2540-9719

4.3 Penentuan variabel penelitian

Pada tahap ini ditentukan variabel penelitian yang digunakan sebagai dasar dalam mengevaluasi kinerja *framework automation testing*. Variabel yang digunakan terdiri dari dua parameter utama, yaitu *Automated Testing Progress* dan *Tools Usability*. Kedua parameter ini dipilih karena mewakili aspek kinerja teknis dan kemudahan penggunaan dalam proses *automation testing*.
Parameter *Automated Testing Progress* memiliki pengaruh yang lebih signifikan sehingga diberikan bobot sebesar 0.6, sedangkan parameter *Tools Usability* diberikan bobot sebesar 0.4. Penentuan bobot ini didasarkan pada tingkat kontribusi masing-masing parameter dalam menilai efektivitas pengujian otomatis [11].

Selanjutnya, masing-masing parameter memiliki sub-parameter yang juga diberikan bobot sesuai tingkat kepentingannya, dengan ketentuan total bobot setiap parameter bernilai 1. Berikut bobot untuk setiap sub parameter dapat dilihat pada Tabel 3 dan Tabel 4.

Tabel 3 Bobot sub-parameter *automated testing progress*

Sub-Parameter	Bobot
Time Complexity	0.4
Covered Test Case	0.3
Execution Start	0.2
Element Inspection	0.1

Tabel 4 Bobot sub-parameter *tools usability*

Sub-Parameter	Bobot
Platform Compatibility	0.4
Supported Browser	0.3
Scripting Language	0.2
Parallel Execution	0.1

Berdasarkan pembobotan tersebut, terlihat bahwa *time complexity* dan *platform compatibility* menjadi faktor dominan dalam masing-masing parameter. Hal ini menunjukkan bahwa kecepatan eksekusi pengujian serta fleksibilitas penggunaan *framework* menjadi aspek utama dalam menentukan kinerja *framework automation testing* [15]. Bobot ini selanjutnya digunakan dalam proses analisis menggunakan metode DIA untuk menghasilkan pemeringkatan alternatif secara objektif.

4.4 Penentuan solusi ideal positif dan negatif

Setelah dilakukan penyusunan dan normalisasi matriks keputusan serta pembobotan, langkah selanjutnya adalah menentukan solusi ideal positif dan solusi ideal negatif. Tahap ini bertujuan untuk mengidentifikasi nilai terbaik dan terburuk dari setiap kriteria yang digunakan sebagai acuan dalam proses evaluasi alternatif.

Solusi ideal positif (A^+) merupakan nilai maksimum dari setiap kriteria, sedangkan solusi ideal negatif (A^-) merupakan nilai minimum. Perhitungan dilakukan menggunakan formula berikut:

$A^+ = \{\max(v_{ij})\}$

$A^- = \{\min(v_{ij})\}$

Berdasarkan hasil perhitungan, diperoleh nilai solusi ideal positif dan negatif dari parameter *Automated Testing Progress* yang disajikan pada Tabel 5 dan Tabel 6.

Tabel 5 Solusi ideal positif *automated testing progress*

Positive Ideal	**Sub Parameter**	$A^+ = \max V_{ij} = [V_1^+, V_2^+, \ldots, V_n^+]$
V^+	*Time Complexity*	0.2970
V^+	*Covered Test Case*	0.15
V^+	*Execution Start*	0.1373
V^+	*Inspect Element*	0.07072

 ISSN:2302-8149
e-ISSN:2540-9719

Tabel 6 Solusi ideal negatif *automated testing progress*

Negative Ideal	Sub Parameter	$A^- = \min V_{ij} = [V_1^-, V_2^-, \dots, V_n^-]$
V^-	*Time Complexity*	0
V^-	*Covered Test Case*	0.15
V^-	*Execution Start*	0
V^-	*Inspect Element*	0

Berikut adalah hasil perhitungan solusi ideal positif dan negatif untuk parameter *Tools Usability* yang dapat dilihat pada Tabel 7 dan Tabel 8.

Tabel 7 Solusi ideal positif *tools usability*

Positive Ideal	Sub Parameter	$A^+ = \max V_{ij} = [V_1^+, V_2^+, \dots, V_n^+]$
V^+	*Platform Compatibility*	0.2828
V^+	*Supported Browser*	0.15
V^+	*Scripting Language*	0.1414
V^+	*Parallel Execution*	0.05

Tabel 8 Solusi ideal negatif *tools usability*

Negative Ideal	Sub Parameter	$A^- = \min V_{ij} = [V_1^-, V_2^-, \dots, V_n^-]$
V^-	*Platform Compatibility*	0
V^-	*Supported Browser*	0.15
V^-	*Scripting Language*	0
V^-	*Parallel Execution*	0.05

Nilai solusi ideal ini menjadi acuan dalam tahap selanjutnya, yaitu perhitungan jarak Manhattan untuk menentukan tingkat kedekatan masing-masing *framework* terhadap kondisi ideal.

4.5 Perhitungan jarak manhattan

Pada tahap ini dilakukan perhitungan jarak Manhattan untuk mengukur kedekatan setiap alternatif terhadap solusi ideal positif (A^+)dan solusi ideal negatif (A^-). Perhitungan ini menggunakan nilai pada matriks berbobot sebagai dasar evaluasi. Jarak Manhattan dihitung menggunakan formula (6) dan (7), di mana nilai alternatif dibandingkan terhadap solusi ideal dan dijumlahkan untuk setiap parameter.

Hasil perhitungan jarak Manhattan menghasilkan dua nilai, yaitu *Maximum Manhattan Distance* (D^+)yang menunjukkan jarak terhadap solusi ideal positif, serta *Minimum Manhattan Distance* (D^-)yang menunjukkan jarak terhadap solusi ideal negatif. Hasil perhitungan jarak Manhattan untuk parameter *Automated Testing Progress* disajikan pada Tabel 9 dan Tabel 10.

Tabel 9 Jarak manhattan maksimum *automated testing progress*

Alternatif	D^+ (*Maximum*)
Cypress	-0.4343
Playwright	-0.1085
WebDriverIO	0
Selenium WebDriver	-0.2535

Tabel 10 Jarak manhattan minimum *automated testing progress*

Alternatif	D^- (*Minimum*)
Cypress	0.07072
Playwright	0.39652
WebDriverIO	0.4343
Selenium WebDriver	0.2515

 ISSN:2302-8149
e-ISSN:2540-9719

Berdasarkan hasil tersebut, WebDriverIO memiliki nilai D^+terkecil sedangkan Cypress memiliki nilai D^-paling kecil (mendekati nol),. Hal ini menunjukkan adanya variasi kinerja antar *framework* dalam aspek *Automated Testing Progress*.

Hasil perhitungan jarak Manhattan untuk parameter *Automated Testing Progress* disajikan pada Tabel 11 dan Tabel 12.

Tabel 11 Jarak manhattan maksimum *tools usability*

Alternatif	D^+ (*Maximum*)
Cypress	-0.4242
Playwright	-0.2828
WebDriverIO	-0.1414
Selenium WebDriver	0

Tabel 12 Jarak manhattan minimum *tools usability*

Alternatif	D^- (*Minimum*)
Cypress	0
Playwright	0.1414
WebDriverIO	0.2828
Selenium WebDriver	0.4242

Dari hasil tersebut, Selenium WebDriver memiliki nilai D^+terbaik (mendekati nol) serta nilai D^-terbesar, yang menunjukkan bahwa *framework* ini memiliki tingkat *usability* yang paling mendekati kondisi ideal dibandingkan alternatif lainnya.

4.6 Penentuan *Positive Ideal Alternative* (PIA)

Penentuan *Positive Ideal Alternative (PIA)* dilakukan dengan memilih nilai terkecil dari *Maximum Manhattan Distance* (D^+) dan nilai terbesar dari *Minimum Manhattan Distance* (D^-) berdasarkan formula (8).

Berdasarkan hasil perhitungan, diperoleh nilai PIA sebagai berikut:

- Parameter *Automated Testing Progress*: PIA = (-0.4343 ; 0.4343)
- Parameter *Tools Usability*: PIA = (-0.4242 ; 0.4242)

Nilai tersebut merepresentasikan kondisi ideal untuk masing-masing parameter dan digunakan sebagai acuan dalam perhitungan nilai preferensi R_i pada tahap selanjutnya.

4.7 Perhitungan nilai preferensi dan peringkat (R_i)

Tahap ini merupakan tahap akhir dalam metode DIA, yaitu menentukan peringkat alternatif berdasarkan nilai preferensi (R_i). Nilai $R_i$dihitung dengan mengombinasikan nilai *Positive Ideal Alternative (PIA)* dengan jarak Manhattan maksimum (D^+)dan minimum (D^-)sesuai dengan formula (9). Alternatif dengan nilai $R_i$terkecil menunjukkan kedekatan tertinggi terhadap solusi ideal.

4.7.1 Hasil Ranking Parameter *Automated Testing Progress*

Perhitungan nilai R_i pada parameter *Automated Testing Progress* menghasilkan peringkat sebagai berikut dapat dilihat pada Tabel 13.

Tabel 13 *Ranking automated testing progress*

Framework	R_i	Peringkat
Cypress	0.3636	3
Playwright	0.3279	2
WebDriverIO	0.4343	4
Selenium WebDriver	0.2572	1

 ISSN:2302-8149
e-ISSN:2540-9719

Hasil menunjukkan bahwa Selenium WebDriver memiliki nilai R_i terkecil sehingga menjadi alternatif terbaik pada parameter ini.

4.7.2 Hasil Ranking Parameter *Tools Usability*

Perhitungan nilai R_i pada parameter *Tools Usability* menghasilkan hasil sebagai berikut dapat dilihat pada Tabel 14.

Tabel 14 *Ranking tools usability*

Framework	R_i	**Peringkat**
Cypress	0.4242	2
Playwright	0.3162	1
WebDriverIO	0.3162	1
Selenium WebDriver	0.4242	2

Hasil menunjukkan bahwa Playwright dan WebDriverIO memiliki nilai terbaik pada aspek *usability*.

4.7.3 Hasil Ranking Akhir

Setelah diperoleh hasil pada masing-masing parameter, dilakukan penggabungan untuk menentukan peringkat akhir berdasarkan keseluruhan parameter dapat dilihat pada Tabel 15.

Tabel 15 *Final ranking result*

Framework	R_i	**Peringkat**
Cypress	0.31747	2
Playwright	0.33426	3
WebDriverIO	0.33426	3
Selenium WebDriver	0.25799	1

Hasil akhir menunjukkan bahwa Selenium WebDriver memiliki nilai R_i terkecil sehingga menjadi *framework* terbaik secara keseluruhan. Hal ini menunjukkan bahwa Selenium memiliki keseimbangan terbaik antara aspek *performance* dan *usability* dibandingkan *framework* lainnya.

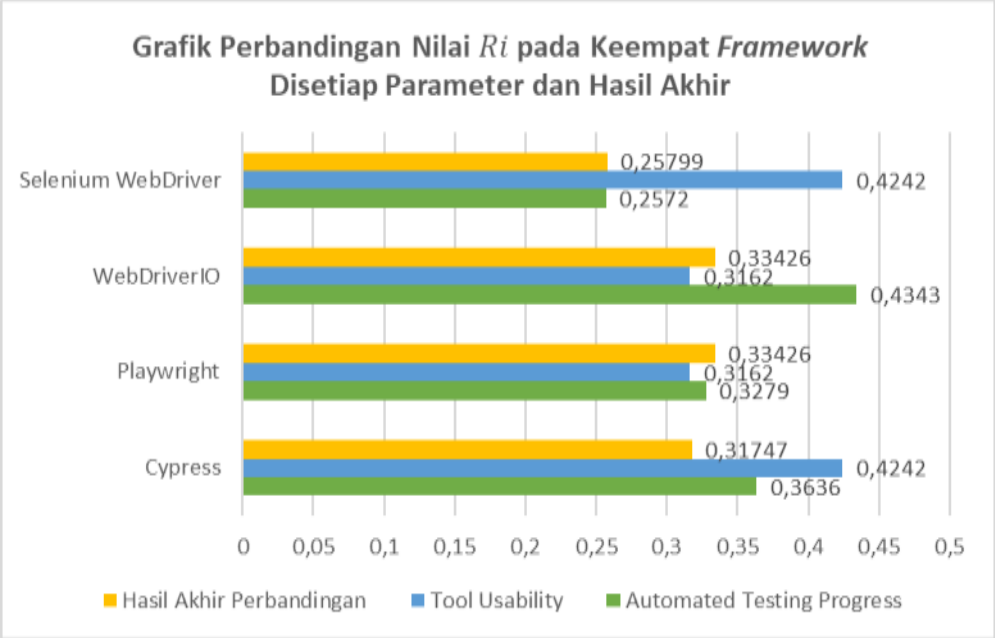
5 Kesimpulan

Berdasarkan hasil pengolahan data menggunakan metode *The Distance to the Ideal Alternative (DIA)*, diperoleh bahwa Selenium WebDriver merupakan *framework automation testing* terbaik secara keseluruhan. Pada parameter *Automated Testing Progress*, Selenium WebDriver memiliki nilai $R_i$terkecil yaitu 0.2572, yang menunjukkan kedekatan tertinggi terhadap solusi ideal. Hal ini dipengaruhi oleh kemampuannya dalam efisiensi waktu eksekusi, cakupan pengujian yang luas, inisialisasi eksekusi yang cepat, serta kemudahan dalam inspeksi elemen. Playwright menempati peringkat kedua dengan nilai 0.3279, diikuti oleh Cypress (0.3636) dan WebDriverIO (0.4343) sebagai alternatif dengan jarak terjauh dari solusi ideal.

Sementara itu, pada parameter *Tools Usability*, Playwright dan WebDriverIO menempati peringkat terbaik dengan nilai R_i yang sama yaitu 0.3162. Hal ini menunjukkan bahwa kedua *framework* tersebut memiliki tingkat *usability* yang tinggi, ditinjau dari aspek kompatibilitas *platform*, dukungan *browser*, fleksibilitas bahasa pemrograman, serta kemampuan eksekusi paralel.

Secara keseluruhan, hasil penelitian menunjukkan bahwa tidak terdapat satu *framework* yang unggul pada seluruh aspek, namun Selenium WebDriver memiliki keseimbangan terbaik antara *performance* dan *usability*. Dengan demikian, Selenium WebDriver direkomendasikan sebagai *framework* yang paling optimal untuk digunakan dalam pengujian aplikasi berbasis *web*. Berikut adalah grafik perbandingan nilai alternatif ideal untuk setiap *framework* dapat dilihat pada Gambar 4.

 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 4 Grafik perbandingan nilai alternatif *Ri* pada keempat *framework*

Referensi

[1] S. Kemp, "Digital 2024: *Global Overview Report*," Jan. 2024. Accessed: Dec. 26, 2024. [Online]. Available: https://datareportal.com/reports/digital-2024-global-overview-report

[2] T. Santia, "Penerimaan Pajak dari Sektor Digital Sentuh Rp 28,91 Triliun hingga September 2024," Liputan6.

[3] D. Wahyono, "*Automation Regression Testing* pada Aplikasi *Ifocus Mobile* menggunakan Katalon Studio Studi Kasus PT Gue," Aug. 2020.

[4] Rianfati, "Pentingnya *Automation Testing* dalam Pengembangan Perangkat Lunak," Jenius co.create.

[5] A. Aburas, "*Choosing the Right Automated Software Testing Tools*," in *2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA)*, IEEE, May 2024, pp. 31–35. DOI: 10.1109/MI-STA61267.2024.10599723.

[6] Cahya, "Implementasi *DSS (Decision Support System)* dengan metode *DIA (Distance to the Ideal Alternative)* menggunakan *PHP* dan *MySQL* untuk penentuan penerima beasiswa," *Decision Support System* Series. Accessed: Jan. 01, 2025. [Online]. Available: https://extra.cahyadsn.com/dia?access=20241230091950

[7] N. Luh Putu Melya Wati, I. Made Dwi Putra Asana, N. Wayan Suardiati Putri, K. Jaya Atmaja, and I. Gede Iwan Sudipa, "*Comparison of Automation Testing on Card Printer Project using Playwright and Selenium Tools*," *Architecture and High Performance Computing*, Vol. 6, No. 3, 2024, DOI: 10.47709/cnapc.v6i3.4362.

[8] A. Shtokal and J. Smołka, "*Comparative Analysis of Frameworks used in Automated Testing on Example of TestNG and WebdriverIO*," 2021. DOI: https://doi.org/10.35784/jcsi.2595.

[9] F. Profesio Putra and S. Melia, "*Comparative Analysis of Automated Testing Tools on GUI WEB-based Applications*," *SENTIMAS: Seminar Nasional Penelitian dan Pengabdian Masyarakat*, Aug. 2023, [Online]. Available: https://journal.irpi.or.id/index.php/sentimas

[10] R. Fakhrii and N. Setiani, "Analisis Perbandingan Unit *Test Automation Framework* dengan Metode *the Distance to the Ideal* Alternatif," *Edusaintek: Jurnal Pendidikan, Sains dan Teknologi*, Vol. 12, No. 1, 2024, DOI: 10.47668/edusaintek.v12i1.1489.

[11] F. Riza, B. Berliyanto, A. Nurrohman, and R. Setiabudi, "*Comparative Analysis of Automation Functional Testing Tools Performance for Playstore Apps with DIA Method*," *Jurnal Techno Nusa Mandiri*, Vol. 21, No. 1, pp. 9–14, Mar. 2024, DOI: 10.33480/techno.v21i1.5363.

 ISSN:2302-8149
e-ISSN:2540-9719

[12] G. A. Prasetyo, "Analisis Perbandingan *UI Test Automation Framework* dengan Metode *the Distance to the Ideal Alternative*," *Universitas Islam Indonesia*, 2024.

[13] S. Fatima, S. F. Nasim, N. G. Haider, M. Rasheed, and Z. Akram, "*Comparative Study of Software Automation Tools: Selenium and Quick Test Professional*," *Journal of Independent Studies and Research Computing*, 2023, DOI: 10.31645/JISRC.23.21.1.6.

[14] M. Ari Rifqi, S. Endang Anjarwani, and A. Hernawan, "*Analysis of Automation Testing using Repeato for Functional Testing of the Yess Nutrition Application based on Flutter*," *E3S Web of Conferences*, Vol. 465, p. 02036, Dec. 2023, DOI: 10.1051/e3sconf/202346502036.

[15] C. Merina, N. Anggraini, S. H. Afrizal, and N. Hakiem, "*A Comparative Analysis of Test Automation Frameworks Performance for Functional Testing in Android-based Applications using the Distance to the Ideal Alternative Method*," *IEEE*, 2018.