Implementasi Automation Testing Menggunakan Playwright Pada Project di PT Itsavirus

Eliazer Jevon Carlos Loelan[1)], I Gede Suardika[2)], Riza Wulandari[3)],
[1]Teknologi Informasi, [2,3]Sistem Informasi
Institut Teknologi dan Bisnis STIKOM Bali
Denpasar, Indonesia
e-mail: [1]210040075@stikom-bali.ac.id, [2]suardika@stikom-bali.ac.id, [3]rizawulandari@stikom-bali.ac.id

Abstrak

Pengujian perangkat lunak merupakan hal yang penting dalam suatu pengembangan aplikasi untuk memastikan kualitas produk sebelum dirilis kepada publik. PT Itsavirus Bali Development, sebagai salah satu software house di Bali masih melakukan pengujian secara manual yang memiliki beberapa keterbatasan seperti membutuhkan waktu yang lebih lama dan rentan terhadap kesalahan manusia. Penelitian ini akan mengimplementasikan automation testing menggunakan Playwright untuk meningkatkan efesiensi waktu pengerjaan terhadap fitur yang berulang. Dalam implmentasi Playwright akan menggunakan pendekatan Page Object Model (POM) untuk meningkatkan keterbacaan dan pemeliharaan dari automation script. Automation script akan dieksekusi pada browser Chromium dan hasil pengujian menunjukkan bahwa semua skenario yang diotomatisasikan berhasil dengan baik. Waktu yang dibutuhkan untuk mengeksekusi automation script adalah selama tiga menit dan semua test skenario berhasil. Otomatisasi pengujian menggunakan Playwright dapat mengurangi ketergantungan pengujian manual dan menjaga konsistensi dari hasil pengujian. Namun, tidak semua fitur dapat diotomatisasikan karena beberapa fitur masih membutuhkan penilaian manusia atau masih terdapat banyak perubahan yang dilakukan.

***Kata kunci**: Playwright, Pengujian, Otomatisasi.*

1. Pendahuluan

Pada era digitalisasi saat ini, pengembangan perangkat lunak baik itu berupa *webapp* maupun *mobile app* telah menjadi salah satu pilar dalam menunjang berbagai sektor industri. Dengan adanya teknologi digital, pelaku bisnis dapat terhubung dengan pelanggan mereka secara lebih mudah dan efisien [1]. Kecepatan dan ketepatan dalam proses pengembangan dan pengujian perangkat lunak menjadi hal yang krusial untuk memastikan produk yang dihasilkan memiliki kualitas yang tinggi dan memenuhi kebutuhan pengguna. Perusahaan-perusahaan berlomba-lomba untuk mengadopsi teknologi terbaru untuk meningkatkan produktivitas tenaga kerja mereka. Dengan adanya teknologi dapat membantu meningkatkan produktivitas tenaga kerja melalui berbagai cara, seperti peningkatan efisiensi, peningkatan inovasi, dan penciptaan lapangan kerja baru [2].

PT Itsavirus merupakan salah satu perusahaan *software house* yang fokus pada pengembangan proyek-proyek berbasis website dan aplikasi. PT Itsavirus memberikan kualitas tinggi sambil memangkas biaya hingga 60% [3]. Namun, dalam proses pengujian perangkat lunak, perusahaan ini masih mengandalkan pengujian manual untuk kebutuhan *end-to-end testing*. Pengujian manual memiliki beberapa keterbatasan, antara lain memerlukan waktu yang lebih lama dan bisa jadi membosankan, membutuhkan investasi besar pada manusia, dan seringkali tidak bisa menguji semua fitur karena keterbatasan waktu [4]. Selain itu, pengujian secara manual rentan terhadap kesalahan manusia, dan tidak efisien dalam menghadapi perubahan kode yang sering terjadi. Permasalahan lainnya adalah tidak jarang terjadi perubahan atau penambahan fitur pelengkap pada masa akhir pengembangan aplikasi yang disampaikan oleh orang yang berkepentingan (*stakeholder*) yang mengakibatkan beberapa dokumen pengujian belum sempat diperbaharui karena berfokus pada pengembangan dan pengujian fitur yang di ubah atau ditambahkan tersebut. Oleh karena itu, dibutuhkan suatu metode yang lebih efektif dan efisien dalam melakukan pengujian perangkat lunak untuk mempermudah pekerjaan dari *quality assurance* (QA) tester sendiri.

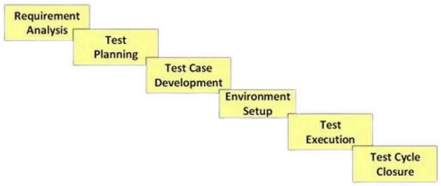
Test Automation adalah penggunaan perangkat lunak khusus untuk melakukan pengujian yang biasanya dilakukan secara manual [5]. Seiring dengan perkembangan teknologi, otomatisasi pengujian (*automation testing*) telah menjadi solusi yang semakin populer untuk mengatasi keterbatasan pengujian

manual. Secara umum, automation test diharapkan dapat menghemat waktu pengujian [6]. Dengan pengimplementasian *automation*, ketersediaan untuk menguji seluruh *test* cases meningkat dan waktu pengujian berkurang [7]. Masih banyak keuntungan yang didapatkan dalam *automation test* seperti menjaga standar dari produk tetap terpenuhi dan meningkatkan kualitas dari produk [8]. Salah satu alat yang dapat digunakan untuk *automation test* adalah Playwright. Playwright merupakan kerangka kerja *open-source* yang dikembangkan oleh Microsoft yang memungkinkan pengujian *end-to-end* yang andal, cepat, dan fleksibel pada aplikasi berbasis web. Playwright Test dibuat khusus untuk mengakomodasi kebutuhan pengujian menyeluruh. Playwright mendukung semua mesin *rendering* modern termasuk Chromium, WebKit, dan Firefox, uji pada Windows, Linux, dan macOS, secara lokal atau pada CI, *headless* atau *headed* dengan emulasi seluler asli Google Chrome untuk Android dan Mobile Safari [9]. Dengan menggunakan Playwright, proses pengujian dapat dilakukan dengan lebih efisien.

Melalui implementasi *automation testing* menggunakan Playwright pada salah satu proyek di PT Itsavirus, diharapkan dapat memberikan beberapa manfaat. Pertama, meningkatkan efisiensi karena *automation test* dapat mempercepat proses pengujian dan mengurangi waktu yang diperlukan untuk pengujian regresi. Kedua, meningkatkan akurasi dengan mengurangi ketergantungan pada pengujian manual, otomatisasi dapat mengurangi kemungkinan kesalahan manusia dan memastikan konsistensi hasil pengujian. Ketiga, memungkinkan skalabilitas, dimana *automation testing* memungkinkan pengujian yang lebih mudah skalanya seiring dengan bertambahnya kompleksitas dan ukuran proyek. Namun, tidak semua hal dapat diotomatisasi meskipun automation test berkembang dengan cepat [10].

2. Metode Penelitian

STLC (*Software Testing Life Cycle*) adalah suatu proses sistematis yang di dalamnya terdapat serangkaian langkah-langkah untuk memastikan kualitas suatu aplikasi telah tercapai [11]. Proses ini dimulai dari menganalisis persyaratan, merancang kasus uji, menyiapkan lingkungan pengujian, hingga mengeksekusi dan mengevaluasi hasil pengujian. Tujuan utama STLC adalah untuk menemukan dan memperbaiki *bug* atau kesalahan dalam perangkat lunak sebelum dirilis, sehingga dapat memenuhi kebutuhan pengguna dan memenuhi standar kualitas yang telah ditetapkan. Dengan menerapkana STLC, alur pengujian menjadi lebih terstruktur dan setiap tahp uji memiliki fokus yang lebih [12]. STLC memiliki beberapa tahap yang setiap tahap memiliki kriteria untuk memulai dan hasilnya [13]. STLC akan digunakan sebagai acuan pada penelitian ini dengan sedikit modifikasi untuk menyesuaikan dengan situasi dan keadaan yang ada.



Gambar 1. *Software Test Lfie Cycle [11]*

2.1. Analasis Kebutuhan

Analisis kebutuhan adalah tahapan yang dilakukan oleh seorang penguji dimana pengumpulan data dilakukan dan menganalisis data yang telah dikumpulkan dari sudut pandang pengujian dan melihat secara garis besar apa saja yang akan diuji [11]. Analisis kebutuhan ini akan melihat kriteria yang dibutuhkan.

2.2. Perencanaan Pengujian

Tahap selanjutnya adalah melakukan perencanaan untuk pengujian dimana penguji akan menentukan apa saja yang akan diuji, menentukan pengujian apa saja yang bisa di otomatisasi, limitasi dari pengujian, *environment* yang dibutuhkan, dan estimasi dari upaya dan pengerjaan yang dibutuhkan.

2.3. Pengembangan *Test Cases*

Setelah perencanaan selesai, pembuatan *test cases* akan dilakukan yang dalam hal ini penguji akan merinci dengan lebih spesifik tentang pengujian yang akan dilakukan otomatisasi berdasarkan perencanaan pengujian. Rincian tersebut dapat berupa bagaimana suatu fitur dapat berjalan, apa saja tahap-tahap untuk mengeksekusinya dan ekspektasi hasil yang akan ditampilkan. *Automation tool yang akan digunakan adalah* Playwright. Playwright merupakan alat otomasi dan pengujian web yang dibuat oleh Microsoft, dan Test Engine menerjemahkan syntax Power Fx ke dalam pengujian Playwright [14].

2.4. Pengembangan *Automation Script*

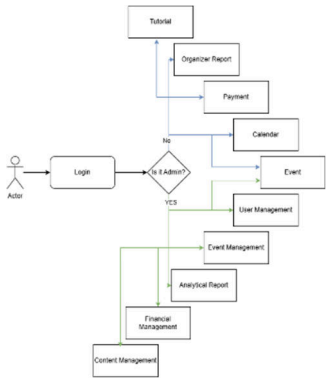
Setelah pembuatan test cases selesai dilakukan, penguji akan mulai menerjemahkan test cases yang ditentukan ke dalam script yang dapat dieksekusi oleh komputer. Pada dasarnya, script ini melakukan hal yang sama seperti pengujian secara manual, namun ini diterapkan ke dalam bahasa mesin agar tidak perlu ketergantungan manusia secara penuh. Pengembangan *script* akan menggunakan Visual Studio Code sebagai *text editor*, dan akan menggunakan JavaScript dan *framework* Playwright.

2.5. Eksekusi Pengujian

Pada tahap ini, *automation script* yang telah dibuat akan dieksekusi untuk menguji aplikasi secara otomatis. Eksekusi pengujain *script* dilakukan melalui *local environment* dikarenakan *script* belum terintegrasi dalam pipeline CI/CD. Selain itu, penelitian ini akan mendokumentasikan terkait dengan berapa lama waktu yang dibutuhkan untuk mengeksekusi *test cases* dan dokumentasi terkait tingkat keberhasilan pengujian secara menyeluruh.

3. Hasil dan Pembahasan

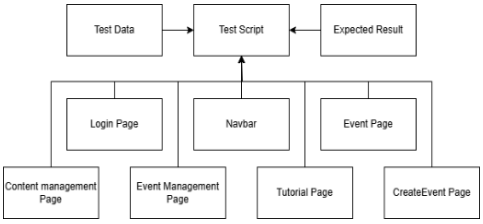
Implementasi *automation testing* menggunakan Playwright, memberikan fitur yang mengotomatisasikan pengujian-pengujian yang sebelumnya dilakukan secara manual untuk mengurangi beban kerja dari QA dan memungkinkan seorang QA dapat melakukan pengujian dengan lebih luas karena hal-hal kecil sudah diotomatisasikan. Analisa dari sistem yang akan diotomatisasikan cukup penting untuk mempertimbangkan fitu-fitur apa saja yang dapat diotomatisasikan baik itu fitur-fitur kecil yang repetitif atau fitu yang sangat panjang dan belum membutuhkan penilaian manusia. Beberapa fitur lainnya belum dapat diotomatisasikan karena bergantung dari *environment* lain sehingga sulit untuk menentukan ekspetasi yang dihasilkan, atau fitur tersebut masih sering terjadi perubahan berdasarkan dari *feedback* dari pihak yang bersangkutan.



Gambar 2. Gambaran Umum Sistem

3.1 Perencanaan Pengujian

Pengujian akan difokuskan pada fitur-fitur kecil yang berulang seperti *filter*, *sort, search*, membuat admin baru, membuat tutorial, dan melihat detail dari *event*, *user* atau *tutorial*. Pengembangan *auotmation script* akan menggunakan pendekatan *Page Object Model* (POM) meningkatkan pemeliharaan, perluasan dan fleksibilitas *automation script* [15].



Gambar 3. Gambaran Penggunaan POM

3.2 Pengembangan *Test Cases*

Test cases dikembangkan dengan tujuan untuk dijadikan sebagai acuan terkait skenario apa saja yang akan diotomatisasikan. Test cases yang dikembangkan dapat dilihat pada tabel berikut.

Tabel 1. Pengembangan *Test Cases*

Fitur	Test Name	Pre-condition	Expected Result
Authentication	Login to the web	Pengguna telah memiliki akun	Berhasil masuk ke halaman dashboard
Event	Search events	Event sudah dibuat di akun	Event yang dicari berhasil ditemukan
	Filter event	Event sudah dibuat di akun	Event terfilter sesuai dengan kategori yang dipilih
	Filter event based on status	Event sudah dibuat di akun	Event terfilter sesuai dengan status yang dipilih
	View Event Detail	Event sudah dibuat di akun	Detail dari event muncul dengan benar
(Admin) User Management	Create a new admin	Pengguna telah memiliki akun admin lainnya	Admin berhasil dibuat
	Search for a user	Pengguna telah memiliki akun admin	User yang dicari ditemukan
	Filter User based on Role	Pengguna telah memiliki akun admin lainnya	Daftar user terfilter berdasarkan role yang dipilih
	View a user's detail	Pengguna lain telah terdaftar	Detail dari user muncul dengan benar
(Admin) Event Management	View detail event	Event telah di *submit* untuk direview	Detail dari event muncul dengan benar
	Admin search for an event	Event telah di *submit* untuk direview	Event yang dicari ditemukan
	Filter event based on category	Event telah di *submit* untuk direview	Event terfilter sesuai dengan filter yang dipilih
	Filter event based on status	Event telah di *submit* untuk direview	Event terfilter sesuai dengan filter yang dipilih
(Admin) Content Management	Create new tutorial	Pengguna telah memiliki akun admin	Tutorial berhasil ditambahkan
	Edit tutorial	Tutorial sudah dibuat oleh admin	Tutorial berhasil di edit
	Delete tutorial	Tutorial sudah dibuat oleh	Tutorial berhasil di hapus dan menghilang dari daftar
Tutorial	View Tutorials	Tutorial sudah dibuat oleh admin	Tutorial berhasil ditampilkan

3.3 Implementasi Sistem

Instalasi awal dari Playwright dapat dilakukan dengan perintah "npm init playwright@latest" Perintah ini akan menginisiasi proses konfigurasi dari Playwright yang dilakukan secara interaktif yang memungkinkan pengguna untuk memilih bahasa pemrograman yang akan digunakan dan juga instalasi browser yang akan digunakan dalam pengujian.

Pengumpulan page object dilakukan menggunakan fitur dari Playwright yaitu Playwright Codegen. Untuk menggunakan Codegen, perintah yang dijalankan pada terminal adalah "npx playwright codegen <URL>". Playwright akan membuka halaman web yang telah ditentukan menggunakan browser dari Playwright yang telah dipasang sebelumnya dan secara otomatis merekam bagaimana pengguna berinteraksi dengan halaman tersebut menggunakan fitur *record and play*. Dari interaksi yang dilakukan, Playwright akan membuat *script* yang berisi pilihan untuk elemen yang akan digunakan dalam pengujian. Namun, pada penelitian ini tidak menggunakan fitur *record and play* tetapi hanya menggunakan fitur untuk mendapatkan *locator* dari *object* yang diinginkan. Setelah itu, *locator* yang dibuat ini dapat digunakan dalam pendekatan *page object* untuk membuat kode lebih modular dan dapat digunakan berulang. Contoh dari *locator* yang telah di dapat dilihat pada gambar berikut.

Gambar 4. Contoh Pengumpulan *Page Object*

Pengimplementasian *source code* menggunakan pendekatan *Page Object Model* (POM) untuk meningkatkan pemeliharaan, perluasan dan fleksibilitas *automation script* [15]. Dengan pendekatan ini, elemen-elemen yang digunakan dalam pengujian dikelompokkan ke dalam kelas tersendiri, sehingga penulisan *locator* tidak berlebihan (*redundant*), serta memudahkan perubahan jika ada pembaruan pada antarmuka pengguna.

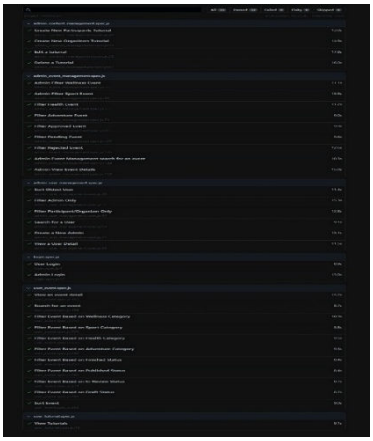
Dalam pengujian ini, setiap skenario otomatisasi memanggil *Page Object* yang telah dibuat untuk menangani interaksi dengan halaman. *Script* pengujian menggunakan metode yang telah didefinisikan dalam kelas *Page Object*, seperti memasukkan email, mengisi kata sandi, dan mengklik tombol login. Dengan cara ini, kode pengujian menjadi lebih modular dan dapat digunakan kembali dalam berbagai skenario pengujian lainnya. Contoh implementasi *automation script* untuk pengujian login dapat dilihat pada gambar berikut.

```
import { test, expect } from '@playwright/test'
import { loginPage } from '../page-object/login_page'

test('User Login', async({page, baseURL}, testInfo) => {
    const email_user = testInfo.project.use.emailUser
    const password_user = testInfo.project.use.passwordUser

    await page.goto(baseURL);

    const loginPage = new loginPage(page);
    await loginPage.enterEmail(email_user);
    await loginPage.enterPassword(password_user);
    await loginPage.clickLogin();
    await expect(page.getByRole('button', { name:
'An awesome user avatar' })).toBeVisible();
})

test('Admin Login', async({page, baseURL}, testInfo) => {
    const email_admin = testInfo.project.use.emailAdmin
    const password_admin = testInfo.project.use.passwordAdmin

    await page.goto(baseURL);

    const loginPage = new loginPage(page);
    await loginPage.enterEmail(email_admin);
    await loginPage.enterPassword(password_admin);
    await loginPage.clickLogin();
    await expect(page.getByRole('button', { name:
'An awesome user avatar' })).toBeVisible();
})
```

Gambar 5. Contoh *Automation Script*

3.4 Eksekusi & Hasil Pengujian

Pada tahap ini, penguji mengeksekusi seluruh *automation script* sekaligus perintah "npx playwright test" yang dijalankan pada *browser* berbasis Chromium. Hasil yang didapatkan adalah sebagai berikut.



Gambar 6. Hasil Eksekusi *Automation Script*

Hasil pada gambar 6 menunjukkan bahwa total waktu yang dibutuhkan untuk menjalankan keseluruhan pengujian pada browser Chromium adalah 3 menit dari 32 *Test Cases* yang ada dan seluruh *test cases* berhasil. Hasil tersebut dapat dilihat dengan lebih detail menggunakan perintah "npx playwright show-report". Kemudian peneliti melakukan eksekusi lagi pada *test script* yang sudah dibuat sebanyak tiga kali dan mendapatkan hasil yang serupa dan membuktikan bahwa hasil dari *automation test* ini konsisten.

4. Kesimpulan

Dari penelitian ini telah dihasilkan *automation script* untuk menggantikan pengujian manual yang repetitif yang dieksekusi dalam 3 menit. Secara keseluruhan, penggunaan *automation testing* menggunakan Playwright terbukti dapat mengurangi ketergantungan pada pengujian manual, dan memastikan kualitas perangkat lunak tetap terjaga serta berhasil menjalankan test dengan waktu yang singkat secara konsisten.

Hal ini membuat pengujian menjadi lebih efesien karena fitur-fitur kecil sudah diotomatisasikan. Namun berdasarkan penelitian ini, tidak semua fitur tidak dapat diotomatisasi sehingga pengujian manual tetap diperlukan dalam beberapa kasus tertentu dan hal-hal yang membutuhkan penilaian manusia dan masih dalam pengembangan.

Daftar Pustaka

[1] A. Widya Astuti, S. Sayudin, and A. Muharam, "Perkembangan Bisnis Di Era Digital," *Jurnal Multidisiplin Indonesia*, vol. 2, no. 9, pp. 2787–2792, Sep. 2023, doi: 10.58344/jmi.v2i9.554.
[2] S. R. Ningsih, "Pengaruh Teknologi Terhadap Produktivitas Tenaga Kerja di Indonesia," *Benefit: Journal of Bussiness, Economics, and Finance*, vol. 2, no. 1, pp. 1–9, Feb. 2024, doi: 10.37985/benefit.v2i1.341.
[3] "Itsavirus." Accessed: Dec. 21, 2024. [Online]. Available: https://www.itsavirus.com/
[4] R. Patidar, A. Sharma, and R. Dave, "Survey on Manual and Automation Testing strategies and Tools for a Software Application," 2017. [Online]. Available: www.ijarcsse.com
[5] G. Ukkuru, *Test Automation Best Practices*. George Ukkuru, 2015.
[6] A. Axelrod, *Complete Guide to Test Automation*. Apress, 2018.
[7] D. Kumar and K. K. Mishra, "The Impacts of Test Automation on Software's Cost, Quality and Time to Market," in *Procedia Computer Science*, Elsevier B.V., 2016, pp. 8–15. doi: 10.1016/j.procs.2016.03.003.
[8] Kanglin Li and Mengqi Wu, *Effective Software Test Automation*. Wiley, 2006.
[9] "Installation | Playwright." Accessed: Dec. 21, 2024. [Online]. Available: https://playwright.dev/docs/intro
[10] B. Oliinyk and T. Volodymyr, "Automation in software testing, can we automate anything we want?," 2019.
[11] A. Nordeen, *Learn Software Testing in 24 Hours*. Guru99, 2020.
[12] B. Huda and A. Lia Hananto, "Penerapan Software Testing Life Cycle Pada Pengujian Otomatisasi Platform Dzikra Application of Software Testing Life Cycle in Automated Testing of Dzikra Platform," vol. 15, no. 1, pp. 1–11, 2023, doi: 10.22303/csrid.15.1.2023.01-11.
[13] S. Bhor, S. Randhave, V. Thakare, S. Nanaware, and A. Butalia, "A SURVEY ON STLC: NEW TESTING TECHNIQUES IN SMART VOTING SYSTEM," *Open Access International Journal Science & Enginering*, vol. 3, Mar. 2018.
[14] E. Kinsbruner, *A A Frontend Web Developer's Guide to Testing*. Packt Publishing, 2022.
[15] A. Rani and R. N. Chandrika, "Designing a Robust Page Object Model (POM) for Cross-Browser and Cross-Platform Testing," 2020. [Online]. Available: http://jtipublishing.com/jti