ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

Penerapan Black Box Automation Testing Menggunakan Playwright Untuk Pengujian Fungsional Aplikasi Vmedis

Qolbi Adi Lumintang

Sistem Informasi, Universitas Pembangunan Nasional "Veteran" Jawa Timur
21082010096@student.upnjatim.ac.id

***Abstrak*— Pengujian perangkat lunak adalah salah satu langkah penting dalam Software Development Life Cycle (SDLC) yang bertujuan untuk memastikan kualitas dan fungsionalitas aplikasi sebelum sampai kepada *end user*. PT Virtual Medis Internasional, merupakan salah satu perusahaan startup yang berbasis di Surabaya yang mengembangkan aplikasi kesehatan bernama Vmedis. Dalam proses penggembangan Vmedis tantangannya yaitu padaa proses pengujian yang masih dilakukan secara manual. Pengujian dengan metode manual ini memakan waktu lama, tidak konsisten pada hasil pengujiannya, dan juga meningkatkan risiko kesalahan yang tidak terdeteksi. Penelitian ini bertujuan untuk mengimplementasikan automation testing dengan menggunakan tools Playwright untuk pengujian fungsional pada aplikasi Vmedis Web. Metodologi yang diterapkan yaitu *Software Testing Life Cycle* (STLC) yang mencakup *requirement analysis, test planning, test case development, test environment setup, test execution*, dan *test closure*. Pengujian dilakukan pada tiga modul dari proyek *Vmedis Intelligence* yakni modul Jurnal Keuangan Shift, Pembayaran Hutang, dan Pembayaran Piutang dengan total 27 test case yang dieksekusi secara manual dan otomatis menggunakan Playwright. Hasil penelitian ini menunjukkan bahwa kedua metode menghasilkan eksekusi yang konsisten yaitu 25 Pass dan 2 Fail, dengan pengujian menggunakan Playwright membutuhkan waktu sebanyak 191,2 detik dibandingkan pengujian manual yang memerlukan waktu sebanyak 297,04 detik. Penggunakan Playwright dapat menghemat waktu sebesar 105,84 detik dengan peningkatan mencapai 35,63% daripada menggunakan metode secara manual. Selain itu, Playwright menunjukkan Bug Detection Rate 100% dan tingkat maintainability yang baik dengan skor rata-rata 3,8/5, sehingga terbukti secara efektif mempercepat proses pengujian dan meminimalkan kesalahan yang ditimbulkan dari pengujian secara manual.**

Kata Kunci— ***Automation Testing*, *Black Box Testing*, Playwright, STLC, *Functional Testing*, Vmedis**

I. PENDAHULUAN

Kemajuan teknologi informasi yang semakin cepat mendorong hadirnya berbagai inovasi. Salah satunya dengan mengembangan perangkat lunak untuk mempermudah dan memperlancar proses bisnis. Setiap perangkat lunak dibangun melalui suatu tahapan yang dikenal sebagai *Software Development Life Cycle* (SDLC), yang berisi rencana terstruktur mencakup proses pengembangan, perubahan, pemeliharaan, dan pembaruan. Suatu perangkat lunak akan berkualitas tinggi apabila seluruh tahapan tersebut dijalankan dengan baik [1]. SDLC juga diterapkan di PT Virtual Medis Internasional, sebuah perusahaan startup di Surabaya yang bergerak dalam pengembangan aplikasi kesehatan. Perusahaan ini menghadirkan solusi digital berupa aplikasi Vmedis dengan platform berbasis *web* dan *mobile* untuk apotek serta klinik yang kini telah dipercaya oleh lebih dari 3.400 pengguna di seluruh Indonesia. Tidak hanya itu, PT Virtual Medis Internasional juga mengembangkan ekosistem terintegrasi melalui aplikasi untuk Perusahaan Besar Farmasi (PBF) dan layanan praktik dokter mandiri yang saling terhubung dengan sistem apotek dan klinik mereka.

Salah satu siklus pengembangan dalam SDLC yaitu fase pengujian. Fase ini merupakan fase krusial yang bertujuan untuk melakukan verifikasi dan validasi kesesuaian dengan kebutuhan pengguna, kualitas perangkat lunak, sekaligus membantu pengembang mengidentifikasi kesalahan (*bug*). Pada aktivitas pengujian juga mempunyai siklus yang lebih spesifik dan terstruktur, yaitu *Software Testing Life Cycle* (STLC).

Dalam mendesain pengujian perangkat lunak, terdapat dua teknik yang dapat digunakan yaitu *black box* dan *white box*. *Black box testing* adalah teknik pengujian yang memeriksa fungsionalitas aplikasi tanpa mengetahui struktur kode internal. Dalam *black box testing*, *tester* fokus pada input dan output sistem serta menguji aplikasi berdasarkan spesifikasi atau persyaratan yang telah ditentukan. *White box testing*, di sisi lain, adalah teknik pengujian yang memeriksa struktur internal aplikasi. *Tester* berfokus pada struktur kode, logika internal, dan algoritma aplikasi untuk menguji program [2].

Tantangan utama dalam pengembangan aplikasi Vmedis terletak pada metode pengujian yang masih menggunakan proses manual oleh tim *tester*. Dalam menguji aplikasi secara manual sangat bergantung pada kompetensi tiap individu, sementara kompleksitas sistem Vmedis kerap menyulitkan prosesnya. Akibatnya, beban kerja *tester* meningkat dan hasil uji sering kurang konsisten. Selain itu, pengujian manual juga memerlukan proses yang lama dan tidak sistematis. Lalu, alur pengembangan aplikasi Vmedis yang harus melewati beberapa *staging* pengembangan yang membuat *tester* perlu mengulang skenario secara monoton juga meningkatkan risiko adanya kesalahan (*error/bug*) yang luput terdeteksi.

Studi menunjukkan bahwa *automated testing* merupakan teknik paling efisien untuk pengujian fungsional karena keunggulannya dalam hal kecepatan eksekusi dan konsistensi hasil dibandingkan manual testing [3]. Melalui penggunaan *tool* otomatis yang menggantikan intervensi manusia, proses pengujian dapat diselesaikan 2-3 kali lebih cepat sambil mempertahankan tingkat konsistensi dan akurasi dalam menemukan kesalahan sistem.

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

Playwright adalah alat *testing* otomatis yang dirancang untuk mempercepat dan menyederhanakan proses pengujian end-to-end. Dari hasil riset yang sudah dilakukan menunjukkan bahwa dengan menggunakan Playwright terbukti secara efektif dapat meningkatkan kecepatan dan juga kemudahan untuk melakukan pengujian terhadap sebuah aplikasi [4]. Selain itu, Playwright juga memiliki beberapa keunggulan dari fitur-fitur yang ditawarkan seperti dapat melakukan pengujian *multi-browser*, sistem *auto-wait* yang secara otomatis menunggu elemen agar siap digunakan, eksekusi test yang dilakukan secara paralel, dan fitur *debugging* serta *tracing* yang memudahkan *tester* untuk menganalisa error dari kode yang dibuat. Selain itu, Playwright juga mampu untuk menguji *test case* yang kompleks dengan lebih baik dan mengurangi test yang tidak konsisten. Selain itu penggunaan Playwright dapat mengurangi waktu eksekusi hingga 60% dan menurunkan *bug* di *production staging* kurang lebih 40% dengan mendeteksi *error* lebih awal. Dengan beberapa keunggulan tersebut, Playwright layak dijadikan *automation testing tools* untuk meningkatkan kualitas aplikasi [4]. Oleh karena itu, Penelitian ini bertujuan untuk mengimplementasikan sistem *testing* otomatis menggunakan Playwright pada aplikasi Vmedis *Web* untuk menghasilkan proses testing yang lebih cepat dan efisien. Dengan penggunaan Playwright, diharapkan dapat memastikan bahwa aspek fungsional aplikasi berjalan sesuai ekspektasi serta dapat mendeteksi bug dengan lebih cepat. Laporan hasil dari pengujian ini akan diserahkan kepada tim developer Vmedis sebagai bahan referensi dan evaluasi untuk pengembangan aplikasi kedepannya.

II. TINJAUAN PUSTAKA

A. *Automation Testing*

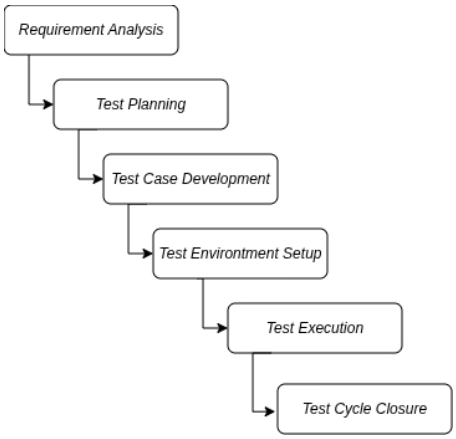
Automation Testing adalah proses pengujian yang menggunakan perangkat lunak atau alat *testing* otomatis untuk mengeksekusi skenario dan test case tanpa proses manual [3]. Keunggulannya yaitu efisiensi dalam pengujian fungsional, tingkat konsistensi dan akurasi yang tinggi, kemampuan mendeteksi masalah serta bug lebih awal, dan pengurangan kesalahan manusia yang sering terjadi pada proses pengujian manual. Selain itu, pengujian otomatis ini dapat menghasilkan respon yang cepat terhadap perubahan, meningkatkan kinerja tim, dan menjamin kualitas perangkat lunak melalui proses pengujian otomatis yang berulang.

B. *Black Box Testing*

Metode Black Box adalah proses pengujian yang menilai perangkat lunak dari perilaku *input* dan *output* sistem tanpa meninjau struktur internal atau *source code*. Tujuan pengujian ini berfokus pada kebutuhan fungsional sistem [5]. Berbagai teknik yang termasuk dalam pengujian black box ini yaitu Equivalence Partitioning, Boundary Value Analysis, Decision Table Testing, State Transition, dan Error Guessing. Contohnya, *tester* tidak perlu memahami proses kinerja yang dilakukan oleh sistem dan proses pengujian ini dilakukan dengan berbagai macam kombinasi nilai masukan yang berfokus pada fungsional normal aplikasi[2].

C. *Software Testing Life Cycle (STLC)*

Software Testing Life Cycle (STLC) yaitu *framework* terstruktur yang menjadi bagian dari proses pengembangan perangkat lunak, yang difokuskan pada tahap pengujian dalam *Software Development Life Cycle* (SDLC). Berbeda dari tahapan lain di SDLC, STLC menitikberatkan upaya untuk menjamin kualitas dan fungsionalitas perangkat lunak sebelum diserahkan kepada *end user* [7]. STLC mempunyai tahapan yang masing-masing terdapat kriteria dan hasil yang telah ditetapkan. Terdapat 6 tahapan STCL antara lain *Requirement Analysis, Test Planning, Test Case Development, Test Environment Setup, Test Execution,* dan *Test Cycle Closure.*



Gbr. 1 Tahapan STLC

D. *Functional Testing*

Functional testing (pengujian fungsional) adalah metode pengujian yang memusatkan perhatian pada fungsi-fungsi perangkat lunak—yaitu memastikan apa saja yang dapat dilakukan oleh perangkat lunak tersebut, layanan apa yang disediakan, dan persyaratan apa saja yang harus dipenuhi [6]. Pengujian fungsional menjadi instrumen yang sangat penting untuk memverifikasi bahwa sistem bekerja sesuai harapan dan konteks bisnis atau manajerial pengguna.

E. *Bug*

Bug merupakan salah satu penyebab utama menurunnya kualitas, keandalan, dan kinerja perangkat lunak [7]. *Bug* biasanya muncul pada tahap awal saat *developer* baru saja melakukan *deliver* modul yang selesai dikerjakan. Selain itu, adanya *coding* yang kompleks dan perubahan *code* yang sering tanpa refaktorisasi juga berkontribusi menimbulkan *bug* yang seharusnya dapat dicegah. Untuk mengurangi dampak dari bug maka diperlukan praktik *development* yang baik mulai dari pencegahan, deteksi, hingga perbaikan *defect*.

F. *Browser Automation Testing Tools*

Browser Automation Testing Tools merupakan suatu *software* yang berfungsi untuk menjalankan tugas-tugas *tester* secara otomatis pada web browser, seperti navigasi antar halaman web dan interaksi dengan form. Kemudian *tools* ini juga dapat digunakan untuk automated testing aplikasi berbasis web, cloud, atau microservices [8]. *Browser Automation Testing*

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

Tools dapat dibagi ke dalam beberapa jenis, yakni *record-and-playback tool, scripting tool, hybrid tool*, dan *automation framework* [2]. *Record-and-playback tool* bekerja dengan cara merekam apa yang *user* lakukan dan memutar ulang hasil dari perekaman tersebut untuk mengeksekusi *test script*, sehingga memudahkan pembuatan pengujian sederhana seperti regresi atau smoke testing. Namun, *tools* ini tidak efektif untuk menangani *test case* yang rumit sehingga perlu sering diperbarui jika aplikasi mengalami perubahan. Di sisi lain, *scripting tool* memberikan kebebasan lebih besar karena *tester* dapat menulis skrip sendiri untuk pengujian yang kompleks, meskipun memerlukan kemampuan teknis yang lebih tinggi dan perawatan yang lebih intensif. Hybrid tool hadir sebagai solusi tengah yang menggabungkan kemudahan penggunaan record-and-playback dengan fleksibilitas *scripting*, sehingga cocok untuk proyek yang membutuhkan berbagai pendekatan otomatisasi.

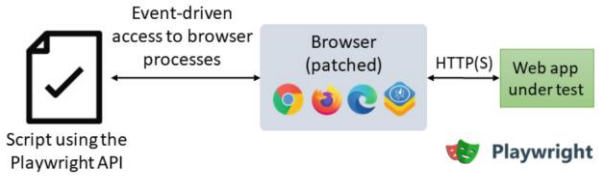
TABEL I
KOMPARASI KEMAMPUAN AUTOMATION TESTING TOOLS DARI SELENIUM, CYPRESS, PUPPETEER, AND PLAYWRIGHT (✓ = SUPPORTED, ✗ = NOT SUPPORTED, P = PARTIAL SUPPORT)[8]

Features	**Selenium**	**Cypress**	**Puppeteer**	**Playwright**
Multilanguage	✓	✗	✗	✓
Cross-browser	✓	P	P	P
Automatic waiting	✗	✓	✗	✗
Tabs handling	✓	✗	✓	✓
Frames and iframes	✓	P	✓	✓
Console log gathering	P	✗	✓	✓
Assertions	✗	✓	✓	✓
Live reload	✗	✓	✗	✓
Test retries	✗	✓	✗	✗
Visual testing	✗	✓	✗	✗
Component testing	✗	P	✗	✓
REST API testing	✗	✓	✗	P
	✗	✓	✗	✓

Tabel di atas membandingkan alat otomasi browser seperti Selenium, Cypress, Puppeteer, dan Playwright. Keempatnya memiliki kemampuan yang mirip seperti pencarian elemen, mensimulasikan mouse/keyboard, dan mengelola cookie. Perbedaannya ada pada kelengkapan fiturnya. Selenium kuat dengan dukungan banyak bahasa pemrograman dan kompatibilitas lintas browser dengan satu API. Cypress unggul pada fitur *auto-waiting* dengan beberapa batasan. Puppeteer dan Playwright menawarkan fitur modern dengan pendekatan non-standar, dengan Puppeteer yang bergantung pada Chrome DevTools Protocol (CDP) dan Playwright yang menggunakan *patched browser*. Selain itu, Cypress dan Playwright sudah disiapkan sebagai *framework* lengkap (test runner, assertion, reporter bawaan), sedangkan Selenium dan Puppeteer biasanya perlu alat tambahan untuk pengujian end-to-end [8].

G. *Playwright*

Playwright yaitu salah satu *automation testing tools* berbasis *browser yang* mendukung tiga jenis *browser engine*, antara lain Chromium, Firefox, dan WebKit. Playwright tidak menggunakan browser asli yang diinstal di komputer, melainkan menggunakan versi browser yang sudah dimodifikasi khusus oleh tim Playwright. Modifikasi ini dilakukan agar browser dapat dikendalikan secara otomatis melalui sistem berbasis event (*event-driven architecture*) yang memungkinkan akses ke berbagai proses internal browser seperti proses rendering tampilan, koneksi jaringan, kontrol browser, maupun service worker. Tim Playwright secara berkala memelihara dan memperbarui versi-versi browser yang telah dimodifikasi ini. Ketika kita menggunakan Playwright untuk pertama kali, tools ini akan secara otomatis mengunduh versi browser yang sudah dimodifikasi tersebut ke komputer lokal kita, sehingga skrip pengujian dapat langsung dijalankan tanpa perlu konfigurasi tambahan yang rumit [8].



Gbr. 2 Arsitektur dari Playwright [8]

H. *Vmedis*

Vmedis adalah platform aplikasi web dan mobile yang ditujukan untuk Apotek, Klinik, dan PBF. Sejak mulai dikembangkan pada 2017, sistem ini terus diperbarui hingga sekarang. Aplikasi web dibangun dengan PHP (framework Yii2) dan sebagian modul menggunakan React.js, sementara aplikasi mobile dikembangkan dengan React Native yang juga memanfaatkan komponen React.js. Saat ini, *branch staging master* Vmedis Web berada pada versi v1.8.0.7, sedangkan *branch staging master-rilis* yang tengah diuji yaitu Vmedis v2.0.

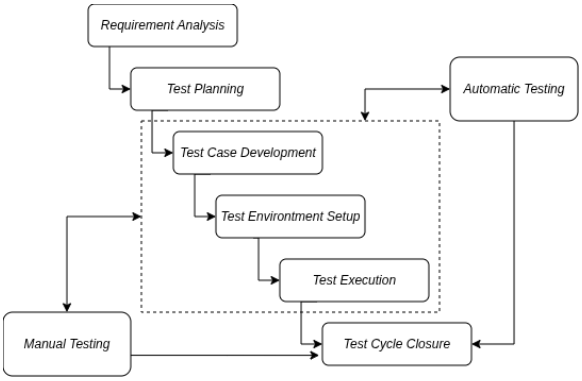
III. METODOLOGI PENELITIAN

Penelitian ini menerapkan metodologi *Software Testing Life Cycle* (STLC). Pada tahap pengembangan test case, penyiapan lingkungan uji, dan eksekusi pengujian, proses dilakukan dengan dua cara yaitu secara manual dan otomatis.

1) Requirement Analysis: Requirement Analysis merupakan tahap pertama dan paling krusial dalam STLC, di mana tim *tester* mempelajari kebutuhan sistem dari sudut pandang testing untuk mengidentifikasi komponen-komponen yang dapat diuji. Pada tahap ini, tim *tester* berinteraksi dengan para stakeholder seperti *business analyst*, *product manager*, dan *developer* untuk memahami kebutuhan fungsional maupun non-fungsional sistem. Tim juga mengidentifikasi kondisi

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

pengujian beserta prioritasnya, menyiapkan Requirement Traceability Matrix (RTM) untuk memastikan bahwa upaya pengujian sejalan dengan tujuan bisnis, sehingga mencegah perluasan ruang lingkup dan pengerjaan ulang di kemudian hari [9].



Gbr. 3 Metodologi Penelitian

2) Test Planning: *Test Planning* adalah tahap penyusunan rencana pengujian yang mendefinisikan ruang lingkup, tujuan, anggaran, dan jadwal waktu pengujian. Pada tahap ini, tim menyusun dokumen strategi pengujian, melakukan alokasi sumber daya dan pembagian peran, memilih pendekatan pengujian (otomatis atau manual), memperkirakan upaya yang dibutuhkan, serta menjadwalkan *milestone* proyek. Keputusan mengenai *tools* pengujian juga ditentukan pada tahap ini untuk memastikan kompatibilitas dengan kebutuhan proyek. Tahap ini menghasilkan dokumen Test Plan yang telah disetujui dan laporan estimasi upaya, yang berfungsi sebagai blueprint atau cetak biru dari seluruh siklus pengujian, memastikan risiko, dependensi, dan kontingensi ditangani sebelum pelaksanaan dimulai [9].

3) Test Case Development: *Test Case Development* adalah tahap yang mengubah rencana pengujian menjadi *test case* serta skrip otomasi secara sistematis. Pada tahap ini, tim merancang dan mereview *test case*, di mana setiap *case* menentukan input yang diberikan, output yang diharapkan, serta kondisi sebelum dan sesudah pengujian [9].

4) Test Environment Setup: *Test Environment Setup* adalah tahap persiapan infrastruktur deployment tempat pengujian akan dilakukan, yang berjalan paralel dengan pengembangan test case untuk efisiensi optimal. Tahap ini mendefinisikan kondisi perangkat keras dan perangkat lunak yang dibutuhkan untuk menjalankan pengujian.

5) Test Execution: Pada tahap eksekusi. Tim *tester* menjalankan pengujian baik secara manual maupun menggunakan *tools* otomatis lalu mengumpulkan hasilnya dan mencatat secara detail setiap kejadian, bug, atau kegagalan yang ditemukan selama proses pengujian berlangsung.

6) *Test Closure*: *Test closure* merupakan evaluasi akhir atas seluruh aktivitas yang telah dilakukan. Tahap ini mencakup pendokumentasian hasil uji, termasuk status masalah yang telah diselesaikan maupun yang masih belum diselesaikan. Pada tahapan ini pula, hasil dari eksekusi secara manual dan otomatis dibandingkan satu sama lain.

IV. HASIL DAN PEMBAHASAN

A. Requirement Analysis

Pada tahap *requirement analysis* untuk pengembangan aplikasi Vmedis, *Chief Technology Officer* (CTO) memberikan *briefing* kepada tim *tester* untuk memastikan arah proyek selaras dengan rencana yang telah disusun. Proyek dimaksud salah satunya *Vmedis Intelligence*, yang terdiri atas beberapa modul pendukung. Beberapa modul pendukung yang dibutuhkan adalah Jurnal Keuangan Shift, Pembayaran Hutang, dan Pembayaran Piutang. Rincian modul ini meliputi:

TABEL 2
MODULE REQUIREMENT

Modul	***Requirement***
Jurnal Keuangan Shift	Kasir dapat menambahkan jurnal keuangan shift
	Kasir dapat mengubah jurnal keuangan shift
	Kasir dapat menghapus jurnal keuangan shift
	Kasir dapat melihat detail jurnal keuangan shift
	Kasir dapat melihat daftar jurnal keuangan shift
Pembayaran Hutang	Manajer dapat membayar hutang pembelian
	Manajer dapat melihat data hutang dengan format PDF
	Manajer dapat melihat data hutang dengan format Excel
Pembayaran piutang	Manajer dapat membayar piutang penjualan
	Manajer dapat melihat data piutang dengan format PDF
	Manajer dapat melihat data piutang dengan format Excel

B. Test Planning

Pada tahap *test planning,* output yang dihasilkan yakni test plan berdasarkan *software requirement specification* (SRS) dan *requirement traceability matrix* (RTM). Untuk pengujian akan dilakukan yaitu pengujian secara manual dan otomatis. Berikut merupakan cuplikan dari *test plan*.

TABEL 3
TEST PLANNING

Modul	**Fitur**
Jurnal Keuangan Shift	Pengujian fungsional tambah jurnal keuangan shift
	Pengujian fungsional ubah jurnal keuangan shift
	Pengujian fungsional hapus jurnal keuangan shift
	Pengujian fungsional *view* jurnal keuangan shift
	Pengujian fungsional *index* jurnal keuangan shift
Pembayaran Hutang	Pengujian fungsional bayar hutang
	Pengujian fungsional cetak PDF data hutang
	Pengujian fungsional unduh Excel data hutang
	Pengujian fungsional *index* pembayaran hutang
Pembayaran Piutang	Pengujian fungsional bayar piutang
	Pengujian fungsional cetak PDF data piutang
	Pengujian fungsional unduh Excel data piutang
	Pengujian fungsional *index* pembayaran piutang

C. Test Case Development

Pembuatan *test case* dilakukan dengan aplikasi manajemen internal Vmedis yaitu VProMan. *Test case* dibuat berdasarkan dari proses bisnis, dokumen SRS dan RTM. Semua *test case* yang dibuat ada di dalam aplikasi manajemen internal Vmedis.

TABEL 3
TEST CASE

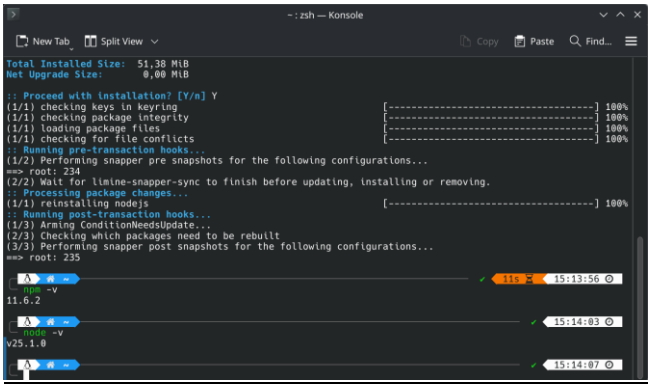
ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

Fitur	Test Case ID	Test Case
Tambah jurnal keuangan shift	TC-JRL01	Menambah jurnal keuangan shift dengan jenis pengeluaran
	TC-JRL02	Menambah jurnal keuangan shift dengan jenis pemasukan
	TC-JRL03	Menambah jurnal keuangan shift tanpa keterangan
	TC-JRL04	Menambah jurnal keuangan shift tanpa nominal
	TC-JRL05	Menambah jurnal keuangan shift dengan nomor bukti jurnalnya sama
	TC-JRL06	Menambah jurnal keuangan shift dengan nomor bukti jurnalnya kosong
Ubah jurnal keuangan shift	TC-JRL07	Mengubah jurnal keuangan shift dari jenis pemasukan menjadi jenis pengeluaran
	TC-JRL08	Mengubah jurnal keuangan shift dari jenis pengeluaran menjadi jenis pemasukan
	TC-JRL09	Mengubah keterangan jurnal keuangan shift
	TC-JRL10	Mengubah nominal jurnal keuangan shift
	TC-JRL11	Mengubah no bukti jurnal keuangan dengan no bukti yang berbeda
	TC-JRL12	Mengubah no bukti jurnal keuangan dengan no bukti yang sama
Hapus jurnal keuangan shift	TC-JRL13	Menghapus jurnal keuangan shift jenis pemasukan
	TC-JRL14	Menghapus jurnal keuangan shift jenis pengeluaran
Lihat jurnal keuangan shift	TC-JRL15	Melihat detail jurnal keuangan shift
Lihat data jurnal keuangan shift di *index*	TC-JRL16	Melakukan filter kolom jenis jurnal
	TC-JRL17	Melakukan filter kolom nomor bukti
	TC-JRL18	Melakukan filter kolom keterangan
	TC-JRL19	Melakukan sorting semua kolom
Bayar hutang	TC-HO01	Membayar hutang 1 faktur
	TC-HO02	Membayar hutang lebih dari 1 faktur
Cetak PDF data hutang	TC-HO03	Mencetak PDF data hutang
Export Excel data hutang	TC-HO04	Mengunduh Excel data hutang
Bayar piutang	TC-PU01	Membayar piutang 1 faktur
	TC-PU02	Membayar piutang lebih dari 1 faktur
Cetak PDF data piutang	TC-PU03	Mencetak PDF data piutang
Export Excel data Piutang	TC-PU03	Mengunduh Excel data piutang

D. Test Environment Setup

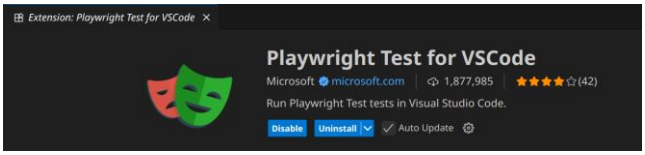
Playwright memerlukan beberapa prasyarat yang harus dipenuhi untuk dapat menjalankan skrip pengujian otomatis. Persyaratan yang harus dipenuhi yaitu:

- Instalasi NPM (Node Package Manager)
- Instalasi Node.js



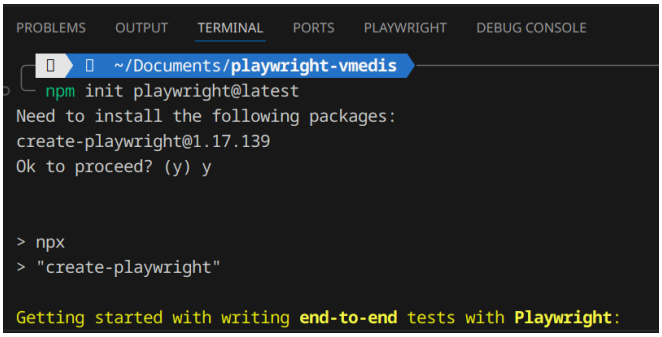
Gbr. 4 Instalasi NPM dan Node.js

- Instalasi *extension* Playwright pada teks editor Visual Studio Code



Gbr. 5 Instalasi *extension* Playwright pada Visual Studio Code

Setelah semua prasyaratnya sudah diinstal, maka selanjutnya yaitu menginstall playwright di dalam folder proyek yang akan dibuat.



Gbr. 6 Intsalasi Playwright pada folder proyek

E. Test Execution

Setelah test case selesai disusun dan lingkungan pengujian siap, langkah berikutnya adalah membuat *test script* di Playwright untuk eksekusi otomatis. Pembuatan *script* memanfaatkan fitur *Playwright Generator Agent* yang memanfaatkan model dari *generative AI* dan *Record/Codegen* pada Playwright untuk merekam alur dan memperbaiki locator yang belum akurat. Setelah script siap, jalankan pengujian melalui Visual Studio Code menggunakan test runner Playwright hingga seluruh skenario tervalidasi.

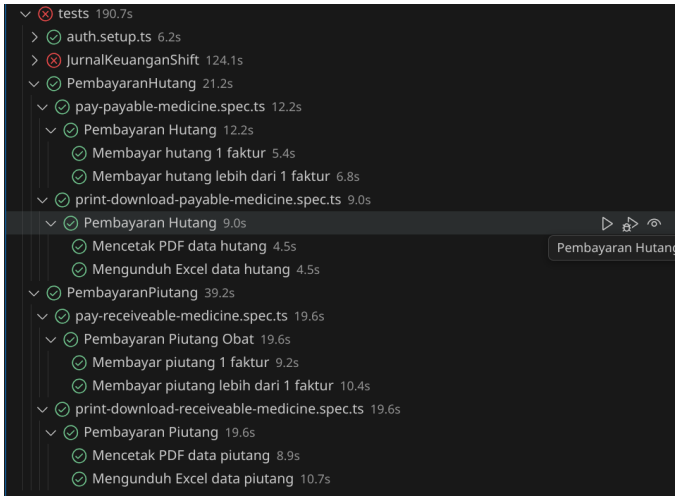
F. Test Closure

Proses pengujian telah selesai dilakukan dari awal tahap perencanaan hingga tahap eksekusi pengujian. Hasil pengujian dari pengujian manual dan pengujian otomatis akan

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

dibandingkan dengan parameter: status hasil eksekusi dan waktu eksekusi. Hasil pengujian dirangkum pada tabel 4.

Gbr. 7 Hasil pengujian modul Jurnal Keuangan Shift



Gbr. 8 Hasil pengujian modul Pembayaran Hutang dan Pembayaran Piutang

TABEL 4
HASIL PERBANDINGAN TESTING

No	Test Case	Hasil Eksekusi		Waktu Eksekusi (detik)	
		Playwright	Manual	Playwright	Manual
1	TC-JRL01	Pass	Pass	10,9	24,5
2	TC-JRL02	Pass	Pass	7,3	25,1
3	TC-JRL03	Pass	Pass	4,8	8,2
4	TC-JRL04	Pass	Pass	5,0	8,7
5	TC-JRL05	Pass	Pass	4,9	10,3
6	TC-JRL06	Pass	Pass	5,3	9,0
7	TC-JRL07	Pass	Pass	4,7	26,6
8	TC-JRL08	Pass	Pass	4,6	25,9
9	TC-JRL09	Pass	Pass	4,6	14,2
10	TC-JRL10	Pass	Pass	4,8	16,8
11	TC-JRL11	Pass	Pass	4,8	12,4
12	TC-JRL12	Pass	Pass	4.0	10,9
13	TC-JRL13	Pass	Pass	4,1	6,2
14	TC-JRL14	Pass	Pass	4,7	5,6
15	TC-JRL15	Pass	Pass	4,1	3,8
16	TC-JRL16	Fail	Fail	30,1	7,24
17	TC-JRL17	Fail	Fail	8,2	7,59
18	TC-JRL18	Pass	Pass	3,2	5,12
19	TC-JRL19	Pass	Pass	4,2	5,28
20	TC-HO01	Pass	Pass	5,4	7,3
21	TC-HO02	Pass	Pass	6,8	7,9
22	TC-HO03	Pass	Pass	4,5	6,2
23	TC-HO04	Pass	Pass	4,5	7,6
24	TC-PU01	Pass	Pass	9,2	10,6
25	TC-PU02	Pass	Pass	10,4	11,1
26	TC-PU03	Pass	Pass	8,9	9,4
27	TC-PU03	Pass	Pass	10,7	9,8
Total Durasi Waktu Pengujian				191,2	297,04

Hasil eksekusi pengujian antara *manual* dengan menggunakan Playwright, status yang dihasilkan sama yakni 25 *Pass* dan 2 Fail. Kemudian untuk waktu eksekusi *test*, akumulasi dari menggunakan Playwright yakni 191,2 detik sedangkan untuk pengujian secara manual membutuhkan waktu 297,04 detik. Dari hasil yang didapat maka penggunaan Playwright untuk pengujian otomatis memberikan penghematan waktu sebanyak 105,84 detik dan peningkatan sebesar 35,63%.

G. Analisis Bug Detection Rate dan Error Coverage

ISSN (Online) 2747-0563
Volume 5 No. 2 Tahun 2025

TABEL 5
ANALISIS BUG DETECTOR

Bug ID	Deskripsi	Manual	Playwright
BUG-01	Filter kolom jenis jurnal tidak berfungsi	✓	✓
BUG-02	Filter kolom nomor bukti tidak berfungsi	✓	✓

Dari 27 test case yang telah dieksekusi baik secara manual dan menggunakan Playwright, terdapat 2 *test case* yang mengalami *error* yakni pada TC-JRL16 dan TC-JRL17. Kedua *case* tersebut merupakan fitur filter kolom jenis jurnal dan filter kolom nomor bukti yang tidak berfungsi, jadi pada saat melakukan filter data, hasil yang tampil pada tabel masih semua data. Untuk komposisi *test case* meliputi 24 *positive case* (88,89%) dan 3 *negative case* (11,11%). Hasil yang didapatkan baik dari pengujian manual maupun menggunakan Playwright yaitu keduanya dapat menemukan bug yang sama sehingga terbukti bahwa Playwright dapat mendeteksi error secara konsisten. Dengan demikian, untuk Bug Detection Rate didapatkan hasil 100% (2/2), sementara failure rate keseluruhan adalah 2/27 = 7,41%.

H. Analisis Pengaruh Kondisi Jaringan

TABEL 6
IDENTIFIKASI TEST CASE DENGAN KETERGANTUNGAN JARINGAN

Test Case	Playwright	Manual	Selisih	Kategori
TC-JRL01	10,9	24,5	13,6	Tinggi
TC-JRL02	7,3	25,1	17,8	Tinggi
TC-JRL16	30,1	7,24	-22,86	Tidak Normal
TC-JRL17	8,2	7,59	-0,61	Tidak Normal
TC-HO03	4,5	6,2	1,7	Normal
TC-HO04	4,5	7,6	3,1	Normal

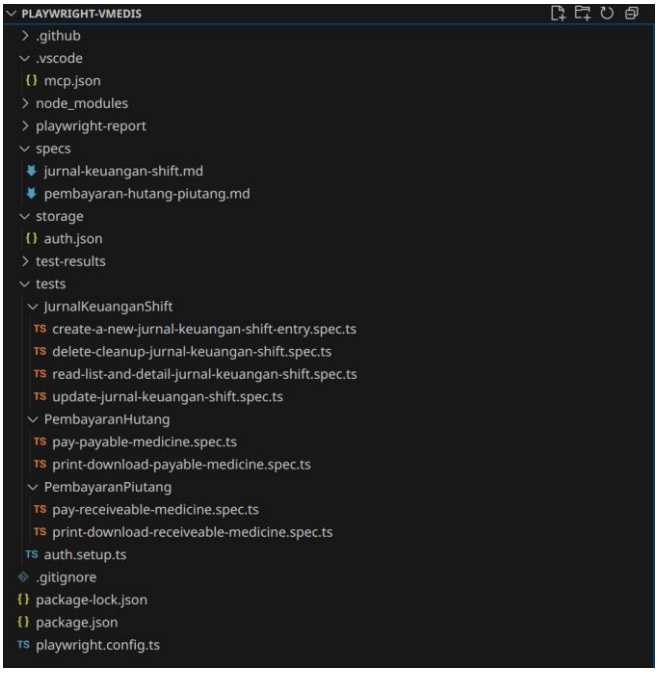
Berdasarkan tabel di atas, maka didapatkan hasil analisis terhadap waktu eksekusi yang menunjukkan adanya variasi pada beberapa test case sehingga mengindikasikan adanya pengaruh kondisi jaringan. Pada TC-JRL01 dan TC-JRL02 menunjukkan bahwa diperlukan waktu *testing* manual yang lama dikarenakan adanya *delay* jaringan saat melakukan *testing* secara manual. Lalu pada *test case* TC-JRL16 dan TC-JRL17 menunjukkan waktu *testing* Playwright yang lebih lama dikarenakan adanya mekanisme untuk *auto-waiting* dan *retry* saat menghadapi *bug*. Sedangkan *test case* pada cetak PDF dan *export excel* seperti pada TC-HO03 dan TC-HO04 menunjukkan waktu yang relatif stabil yang mengindikasikan bahwa ketergantungan jaringannya minimal.

I. Evaluasi Maintainability Skrip Otomatis dalam Jangka Panjang

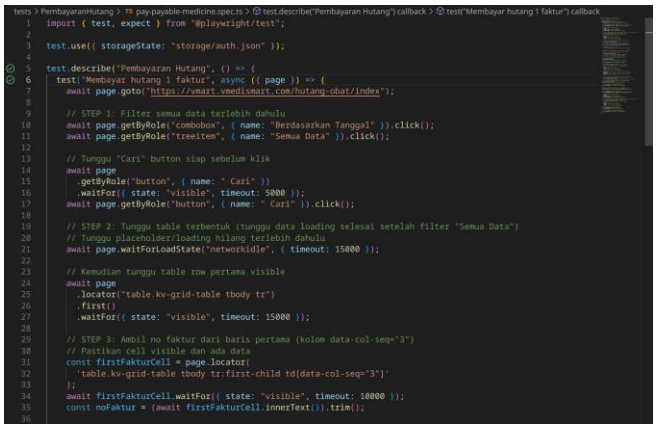
Maintainability pada *automation skrip* juga menjadi salah faktor yang penting untuk penggunaan jangka panjang. Evaluasi dilakukan berdasarkan implementasi dari 27 test case pada 3 modul aplikasi Vmedis.

TABEL 7
EVALUASI ASPEK MAINTAINABILITY

Aspek	Penilaian	Penjelasan
Code Reusability	4/5	Fungsi login sudah dapat digunakan ulang di semua test case
Readability	4/5	*Syntax* Playwright yang *descriptive* memudahkan untuk memahami fungsi dari kodenya
Modularity	4/5	*Test script* telah diorganisir dengan struktur folder yang jelas
Documentation	3/5	Terdapat inline comment pada baris code
Update Effort	4/5	Ketika terdapat perubahan locator dapat diubah dengan cepat menggunakan Playwright Inspector



Gbr. 9 Struktur direktori proyek Playwright



Gbr. 10 Hasil pengujian modul Pembayaran Hutang dan Pembayaran Piutang

Implementasi Playwright pada aplikasi Vmedis menunjukkan tingkat maintainability yang baik dengan struktur kode yang terorganisir dan modular. Penggunaan fitur bawaan Playwright seperti Inspector dan Trace Viewer mempercepat proses debugging dan update. Untuk jangka panjang, disarankan untuk menambahkan dokumentasi yang lebih komprehensif.

V. KESIMPULAN

Hasil dari penelitian ini menyimpulkan bahwa penggunaan Playwright untuk pengujian fungsional secara otomatis sangat efektif untuk mempercepat pengujian dalam pengembangan aplikasi sekaligus meminimalkan kesalahan manusia. Dari 27 test case yang dilakukan pengujian pada tiga modul yakni Jurnal Keuangan Shift, Pembayaran Hutang, dan Pembayaran Piutang, Playwright dapat menyelesaikan *testing* secara otomatis dalam waktu 191,2 detik dibandingkan *testing* yang dilakukan secara manual yang memerlukan waktu selama 297,04 detik, sehingga menghasilkan peningkatan efisiensi sebesar 35,63%. Kedua metode pengujian juga menunjukkan hasil yang konsisten yakni 25 Pass dan 2 Fail, sehingga membuktikan bahwa Playwright memiliki *Bug Detection Rate* 100%.

Kemudian untuk analisis maintainability menunjukkan bahwa implementasi Playwright pada aplikasi Vmedis memiliki tingkat *maintenance* yang baik dengan skor rata-rata 3,8/5, dengan adanya struktur kode yang modular, *reusability*, dan kemudahan untuk update kode dengan menggunakan fitur Playwright Inspector. Meskipun terdapat ketergantungan pada jaringan yang terlihat pada beberapa test case, Playwright dapat menunjukkan performa yang lebih stabil dibandingkan dengan pengujian secara manual yang lebih rentan terhadap variasi kondisi jaringan, terutama pada test case yang melibatkan *test case* yang kompleks.

Untuk memperkuat hasil penelitian, pengujian otomatis juga perlu diperluas ke modul lainnya serta dapat diimplementasikan ke aplikasi Vmedis Mobile untuk memvalidasi konsistensi dalam memanfaatkan penggunaan *automation testing*. Hal ini juga penting karena Vmedis Mobile terintegrasi dengan Vmedis Web, sehingga kualitas keduanya saling bergantung antara satu sama lain.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih setinggi-tingginya kepada Tim SANTIKA karena tlah menyediakan *template* yang sangat membantu. Keberadaan *template* ini mempermudah penulis untuk menyusun artikel dengan struktur yang lebih teratur, tampilan yang rapi, serta penyajian yang lebih mudah dimengerti. Kontribusi dan bantuan yang diberikan oleh Tim SANTIKA juga memiliki peran penting dalam memperlancar proses penyusunan dan penyelesaian dokumen ini. Semoga segala usaha dari Tim SANTIKA memberikan manfaat bagi pihak-pihak yang membutuhkan.

REFERENSI

[1] S. Pargaonkar, "A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering," IJSRP, vol. 13, no. 8, hal. 120–124, Agu 2023, doi: 10.29322/IJSRP.13.08.2023.p14015.

[2] P. Leloudas, Introduction to Software Testing: A Practical Guide to Testing, Design, Automation, and Execution. Apress, 2023.

[3] R. Fauzan, F. P. Soedjono, A. A. Permadani, dan M. A. Yakin, "Perbandingan Pengujian Manual dan Terotomasi pada Software Enterprise Resource Planning," Journal of Advances in Information and Industrial Technology, vol. 5, no. 1, hlm. 23–30, Mei 2023, doi: 10.52435/jaiit.v5i1.318.

[4] S. Talakola, "Automated end to end testing with Playwright for React applications," International Journal of Emerging Research in Engineering and Technology, vol. 5, no. 1, hlm. 38–47, Mar 2024, doi: 10.63282/3050-922X.IJERET-V5I1P106.

[5] M. Sholeh, I. Gisfas, Cahiman, dan M. A. Fauzi, "Black Box Testing on ukmbantul.com Page with Boundary Value Analysis and Equivalence Partitioning Methods," J. Phys.: Conf. Ser., vol. 1823, no. 1, hlm. 012029, Mar 2021, doi: 10.1088/1742-6596/1823/1/012029.

[6] B. Homès, Fundamentals of software testing. John Wiley & Sons, 2024.

[7] I. Forgács and A. Kovács, Modern software testing techniques: a practical guide for developers and testers. Apress, 2023.

[8] B. García, J. M. del Alamo, M. Leotta, and F. Ricca, "Exploring Browser Automation: A Comparative Study of Selenium, Cypress, Puppeteer, and Playwright," in International Conference on the Quality of Information and Communications Technology, Cham: Springer Nature Switzerland, pp. 142–149, Sep. 2024.

[9] (2025) Software Testing Life Cycle (STLC). [Online], https://www.guru99.com/software-testing-life-cycle.html, tanggal akses: 30 Oktober 2025.