JIKOMTI:
Jurnal Ilmiah Ilmu Komputer dan Teknologi Informasi
e- ISSN: 3089-2996
Vol. 3 No. 1, Mei 2026

# Penerapan Automation Testing dengan Cypress pada Website Healthcare Berbasis STLC Agile: Integrasi E2E dan Regression Testing

Ghaida Sandie Hayatus Sahla[1], Farhan Hamdallah[2], Muhamad Dzaky Ashidqi[3]
[1,2,3]Universitas Sains Indonesia, Kabupaten Bekasi, Jawa Barat, Indonesia

E-mail:
ghaidasahla123@gmail.com[1], farhan.hamdallah@lecturer.sains.ac.id[2*],
muhamad.dzaky@lecturer.sains.ac.id[3]

***Abstract***

*Advances in information technology have led to increased complexity in information systems for clinical services, particularly in patient data management, doctor scheduling, and healthcare transactions. This situation requires systems that are not only stable at initial release, but also remain reliable after feature updates. Therefore, software testing is necessary to ensure that each feature runs according to system requirements and does not interfere with existing functionality. This study applies automated testing using Cypress with an Agile-based Software Testing Life Cycle (STLC) approach, using end-to-end testing and regression testing methods. Testing was conducted on the clinic's website with the admin role as the main user who manages user authentication, master data, patient visits, and medication transactions. The test results showed that all test cases achieved a pass status after system improvements and adjustments were made. Thus, the implementation of Agile STLC-based automated testing is capable of supporting a structured, adaptive, and sustainable testing process in maintaining the quality of the clinic's information system.*

***Keywords***: *automation testing; cypress; end-to-end testing; regression testing; STLC Agile*

**Abstrak**

Perkembangan teknologi informasi mendorong meningkatnya kompleksitas sistem informasi pada layanan klinik, khususnya dalam pengelolaan data pasien, penjadwalan dokter, dan transaksi layanan kesehatan. Kondisi ini menuntut sistem yang tidak hanya stabil saat rilis awal, tetapi juga tetap andal setelah dilakukan pembaruan fitur. Oleh karena itu, pengujian perangkat lunak diperlukan untuk memastikan setiap fitur berjalan sesuai kebutuhan sistem serta tidak menimbulkan gangguan pada fungsionalitas yang telah ada. Penelitian ini menerapkan pengujian otomatis menggunakan *Cypress* dengan pendekatan *Software Testing Life Cycle* (STLC) berbasis *Agile*, dengan metode *end-to-end testing* dan *regression testing*. Pengujian dilakukan pada website klinik dengan peran admin sebagai pengguna utama yang mengelola autentikasi pengguna, data master, kunjungan pasien, dan transaksi obat. Hasil pengujian menunjukkan bahwa seluruh *test case* mencapai status *pass* setelah dilakukan perbaikan dan penyesuaian sistem. Dengan demikian, penerapan pengujian otomatis berbasis STLC *Agile* mampu mendukung proses pengujian yang terstruktur, adaptif, dan berkelanjutan dalam menjaga kualitas sistem informasi klinik.

**Kata kunci**: pengujian otomatis; *cypress;* pengujian *end-to-end*; pengujian regresi; *STLC Agile*

## 1. PENDAHULUAN

Era perkembangan digital mendorong percepatan pengembangan teknologi informasi di berbagai sektor, termasuk layanan kesehatan. Dalam cetak biru Kementerian Kesehatan, transformasi digital menekankan pentingnya integrasi data kesehatan yang bersifat rutin dan periodik agar berbagai aplikasi kesehatan dapat saling terhubung dan digunakan secara terpadu. Namun, proses ini masih menghadapi tantangan, salah satunya karena sekitar 80% fasilitas pelayanan kesehatan di Indonesia belum sepenuhnya memanfaatkan teknologi digital [1]. Kondisi tersebut menghambat terwujudnya sistem pelayanan kesehatan yang terintegrasi dan berkelanjutan.

Lingkup layanan klinik yang luas menyebabkan tingginya kompleksitas proses pelayanan, khususnya dalam pengelolaan data pasien dan operasional klinik. Oleh karena itu, dibutuhkan sistem informasi yang mampu mengelola data secara teratur, akurat, dan efisien [2]. Dengan berkembangnya teknologi informasi, pekerjaan manusia semakin mudah, terutama dalam pengelolaan data [3]. Aplikasi perlu melalui proses pengujian untuk memastikan kesesuaiannya dengan spesifikasi dan standar yang ditetapkan. Pengujian perangkat lunak bertujuan untuk menemukan kesalahan yang berpotensi menyebabkan kegagalan sistem serta menghasilkan produk dengan kualitas dan produktivitas yang tinggi [4][5].

Untuk mendukung proses pengujian yang terstruktur dan adaptif, pengujian perangkat lunak perlu diintegrasikan ke dalam kerangka *Software Testing Life Cycle* (STLC) berbasis *Agile*. STLC mencakup tahapan *Requirement Analysis, Test Planning, Test Case Development, Environment Setup, Test Execution, dan Test Cycle Closure*[6]. Pendekatan *Agile* memungkinkan kolaborasi yang lebih intens antara pengembang dan penguji melalui iterasi singkat, sehingga mendukung perbaikan sistem secara berkelanjutan berdasarkan umpan balik [7].

Pada penelitian ini, metode pengujian yang digunakan merupakan kombinasi *end-to-end testing* dan *regression testing*. Pengujian *end-to-end* umum diterapkan untuk memastikan bahwa alur sistem secara keseluruhan berjalan sesuai dengan kebutuhan pengguna [8]. Proses pengujian otomatis dilakukan menggunakan *Cypress*, yang memiliki keunggulan dalam hal eksekusi pengujian yang stabil serta dukungan *dashboard* pengujian yang terintegrasi [9]. Selain itu, *regression testing* diperlukan untuk memastikan bahwa perubahan atau penambahan fitur baru tidak menyebabkan gangguan pada fungsi yang telah berjalan sebelumnya. Pengujian regresi berperan penting dalam mendukung proses integrasi berkelanjutan (CI) dengan mendeteksi kesalahan sedini mungkin dalam siklus pengembangan [10].

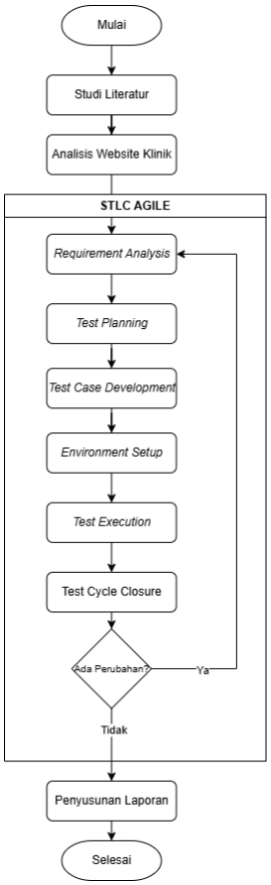
Dalam pengembangan perangkat lunak modern, integrasi pengujian otomatis ke dalam *pipeline Continuous Integration/Continuous Deployment* (CI/CD) menjadi kebutuhan penting. *Pipeline* CI/CD memungkinkan proses pembangunan, pengujian, dan penyebaran aplikasi dilakukan secara berkelanjutan sehingga dapat meningkatkan kualitas produk dan mempercepat siklus rilis [11]. Integrasi *automation testing* menggunakan *Cypress* memungkinkan proses pengujian dijalankan secara otomatis setiap kali terjadi perubahan kode.

Meningkatnya kompleksitas sistem informasi klinik, terutama dalam pengelolaan data pasien, penjadwalan dokter, dan transaksi layanan kesehatan, menuntut sistem yang tidak hanya stabil saat rilis awal, tetapi juga tetap andal setelah pembaruan berkala. Pengujian perangkat lunak diperlukan untuk mengidentifikasi kerentanan dan menjaga kualitas sistem [12]. Selain itu, pendekatan iteratif dalam pengembangan sistem informasi terbukti efektif dalam menyesuaikan sistem terhadap perubahan kebutuhan pengguna [13]. Oleh karena itu, penelitian ini bertujuan untuk mengimplementasikan pengujian otomatis pada *website Healthcare* guna memastikan setiap pembaruan sistem tidak menimbulkan gangguan

pada fungsionalitas yang telah ada. Penelitian ini berjudul "Penerapan *Automation Testing* dengan *Cypress* pada *Website Healthcare* Berbasis STLC *Agile*: Integrasi *End-to-End* dan *Regression Testing*".

## 2. METODE

Alur pelaksanaan penelitian pengujian perangkat lunak dijelaskan di sini, dengan penekanan khusus pada penggunaan *Cypress*, *framework* berbasis *JavaScript* yang mendukung pengujian *end-to-end real-time* dengan integrasi CI/CD serta performa yang ringan [14], untuk pengujian otomatis di *website Healthcare*. Pendekatan STLC berbasis *Agile* digunakan dalam proses pengujian, yang memungkinkan pengujian *end-to-end* dan *regression*. Setelah perubahan, seperti perbaikan *bug*, peningkatan, atau pembaruan sistem, pengujian regresi dilakukan untuk memastikan bahwa fitur yang sudah ada tetap berfungsi dengan baik [15]. Alur penelitian tersebut disajikan dalam bentuk diagram alir pada Gambar 1.




**Gambar 1.** Tahapan Penelitian

### 2.1. Studi Literatur

Penelitian ini dimulai dengan melakukan penelusuran dan studi literatur tentang pengujian perangkat lunak berbasis web. Sumber yang dikaji termasuk jurnal nasional yang membahas pengujian otomatis, teknik STLC, pengujian berbasis *Agile*, dan penerapan pengujian *end-to-end* dan *regression* pada sistem yang terus berkembang. Kajian ini meningkatkan pemahaman kita tentang pentingnya pengujian terstruktur dan berulang untuk menjaga stabilitas fungsional aplikasi, terutama untuk sistem yang belum digunakan secara operasional dan masih dalam tahap penyempurnaan fitur.

### 2.2. Analisis *Website*

Tujuan dari tahap analisis sistem adalah untuk mendapatkan pemahaman tentang fitur, tujuan, dan cara kerja *website Healthcare* yang menjadi subjek penelitian. Admin menggunakan *website* ini untuk mengelola semua data dan aktivitas klinik, termasuk data laboratorium, obat, penyakit, dan pencatatan kunjungan pasien. Struktur menu, hubungan antar modul, dan proses penggunaan sistem dari awal hingga akhir adalah topik utama analisis. Tahap ini penting untuk menentukan ruang lingkup pengujian dan menyusun skenario *end-to-end* yang menunjukkan proses bisnis yang sebenarnya terjadi dalam sistem. Hasil analisis sistem digunakan sebagai dasar untuk menentukan fitur prioritas yang akan diuji melalui pengujian otomatis.

### 2.3. *Requirement Analysis*

Tahap awal dalam STLC adalah analisis kebutuhan, yang bertujuan untuk menentukan kebutuhan pengujian berdasarkan fungsi sistem yang telah dianalisis. Peneliti menentukan fitur yang akan diuji, skenario penggunaan utama, dan kriteria keberhasilan pengujian. Metode *Agile* memungkinkan perubahan atau penambahan kebutuhan pengujian selama proses pengembangan sistem. Oleh karena itu, analisis persyaratan tidak tetap, tetapi dapat disesuaikan dengan keadaan sistem.

### 2.4. *Test Planning*

Tujuan dari tahap perencanaan pengujian adalah untuk membuat strategi pengujian berdasarkan apa yang diperlukan sistem untuk berfungsi. Pada titik ini, diputuskan jenis pengujian, cakupannya, dan cara pengujian otomatis dilakukan di situs *website Healthcare*. Untuk memastikan bahwa setiap fitur berjalan sesuai kebutuhan sistem, terutama setelah penambahan atau perubahan fitur, pengujian difokuskan pada fungsi utama yang dikelola oleh admin, seperti autentikasi pengguna, pengelolaan data master, pencatatan kunjungan pasien, dan transaksi obat. Strategi pengujian menggunakan pendekatan fungsional berbasis skenario penggunaan sistem, yang mencakup pengujian *end-to-end* untuk memastikan alur proses berjalan secara konsisten.

### 2.5. *Test Case Development*

Pada tahap ini, disusun dan didefinisikan kasus uji yang akan digunakan sebagai dasar pelaksanaan pengujian otomatis menggunakan *Cypress*. Setiap kasus disusun secara sistematis dan mencakup skenario pengujian, urutan langkah pengujian, data uji yang digunakan, dan hasil yang diharapkan dari sistem. Tahap ini mengacu pada kebutuhan sistem yang telah diidentifikasi selama tahap analisis kebutuhan dan perencanaan kasus uji.

Variasi kondisi *input* dan alur penggunaan sistem dirancang untuk menunjukkan bagaimana admin klinik menggunakan sistem baik dalam kondisi normal maupun dalam kondisi tertentu. Pengujian belum dilakukan sampai saat ini karena kasus uji coba masih berada pada tahap perancangan. Pada tahap pelaksanaan pengujian, hasil aktual dan status keberhasilan atau kegagalan (*pass*/*fail*) akan diperoleh.

### 2.6. *Environment Setup*

Tujuan dari langkah ini adalah untuk menyiapkan lingkungan pengujian untuk memenuhi kebutuhan eksekusi kasus pengujian. Lingkungan pengujian harus disesuaikan dengan kondisi pengembangan sistem untuk memastikan bahwa hasil pengujian relevan dan konsisten. Perangkat lunak yang digunakan meliputi *Node js*, *Windows*, *Chrome*, *Visual Studio Code*, *Cypress*, *Google Spreadsheet* dan *server hosting website Healthcare*.

### 2.7. *Test Execution*

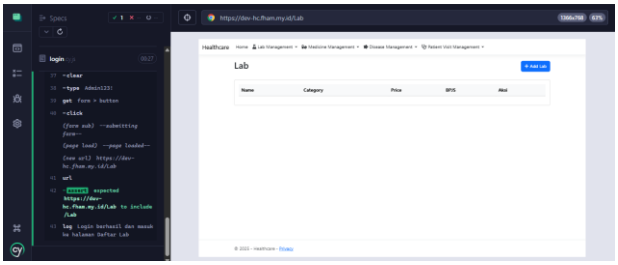
Setelah *test case* yang telah disusun sebelumnya, tahap pelaksanaan pengujian dilakukan. Pada tahap ini, pengujian dilakukan dengan dua metode, yaitu pengujian manual dan pengujian otomatis menggunakan *Cypress*. Pengujian manual terbatas pada skenario *create* data pada modul-modul utama sebagai pembanding, sedangkan pengujian otomatis mencakup *regression testing* dan *end-to-end testing* untuk memastikan bahwa seluruh fungsi sistem berjalan dengan benar.

a. Skrip Pengujian Otomatis

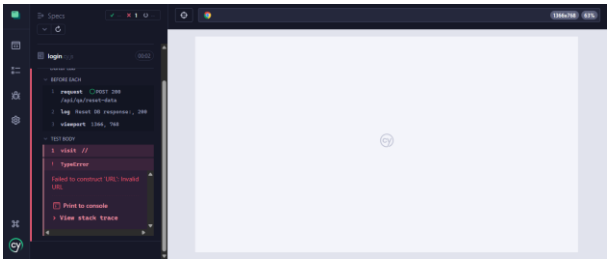
Pada tahap eksekusi awal adalah penulisan skrip pengujian otomatis, karena *website Healthcare* ini hanya ada *role admin*, jadi semua skrip ditujukan untuk *role* tersebut dengan total 113 *test case*. Semua skrip pengujian sudah di*publish* pada GitHub penulis, berikut link nya : *https://github.com/ghaidasandie/healthcare.git*.

b. Eksekusi Pengujian

Pengujian dilakukan secara berulang setiap kali ada perubahan atau rilis fitur pada sistem *website Healthcare* klinik, yang dicatat pada tanggal pelaksanaan pengujian. Untuk memastikan bahwa fungsi yang telah ada sebelumnya tidak mengalami gangguan, *test case* yang relevan dijalankan kembali. Proses pengujian ulang ini menunjukkan penggunaan STLC berbasis *Agile*, yang memungkinkan pengujian berulang dan menyesuaikan diri dengan perubahan sistem. Untuk contoh *test passed* terdapat pada Gambar 2 dan untuk *test failed* terdapat pada gambar 3.



**Gambar 2.** *Test Passed*



**Gambar 3.** *Test Failed*

### 2.8. *Test Cycle Closure*

Pada tahap akhir dari *Software Testing Life Cycle* (STLC), semua aktivitas pengujian dievaluasi. Tujuan dari tahap ini adalah untuk memastikan bahwa semua kasus pengujian telah dilakukan sesuai rencana dan tujuan.

## 3. HASIL DAN PEMBAHASAN

Pada bagian ini pengujian otomatis dilakukan menggunakan *Cypress* yang mencakup peran admin sebagai *role* utama dalam *website Healthcare* ini. Terdapat total 14 *test scenario* dan 113 *test case*.

Pada tahap pengujian, *test case* dengan status *fail* menunjukkan adanya ketidaksesuaian antara hasil aktual dan hasil yang diharapkan. Hasil fail tersebut kemudian dianalisis untuk mengidentifikasi penyebabnya, yang dapat berupa *bug* pada sistem atau perubahan kebutuhan fungsional pada sistem *website* Healthcare selain mencatat hasil keberhasilan pengujian. Hasil dari kesalahan ini diperoleh selama pelaksanaan pengujian otomatis maupun manual, terutama selama pengujian *regression* dan *end-to-end*. Dokumentasi *bug* bisa dilihat pada Tabel 1.

**Tabel 1.** Temuan Bug

| Skenario Pengujian | Deskripsi Bug | Solusi/ Akhir | Status |
|---|---|---|---|
| Edit data Kategori Lab | Tombol edit terjadi error request, tidak bisa masuk halaman form edit | Diperbaiki developer, diuji ulang | Pass |
| Hapus data Kategori Lab | Tombol delete terjadi error request, tidak bisa masuk halaman delete | Diperbaiki developer, diuji ulang | Pass |
| Hapus data Patients | Fitur tidak error tapi data tidak terhapus | Diperbaiki developer, diuji ulang | Pass |
| Hapus data departments | Fitur tidak error tapi data tidak terhapus | Diperbaiki developer, diuji ulang | Pass |
| Hapus data Doctors | Fitur tidak error tapi data tidak terhapus | Diperbaiki developer, diuji ulang | Pass |
| Hapus data obat | Fitur tidak error tapi data tidak terhapus | Fitur delete tidak diimplementasikan pada versi final | Pass |
| Detail Medicine stocks | Tombol detail terjadi error request, tidak bisa masuk halaman detail medicine stock | Diperbaiki developer, diuji ulang dan hasil passed | Pass |
| Edit data Medicine Stocks | Tombol edit terjadi error request, tidak bisa masuk halaman form edit | Diperbaiki developer, diuji ulang dan hasil passed | Pass |
| Hapus data Medicine Stocks | Tombol delete terjadi error request, tidak bisa masuk halaman delete | Fitur delete tidak diimplementasikan pada versi final | Pass |
| Edit data Stock Transactions | Tombol edit sebelum nya ada dan berfungsi, tetapi jadi hilang | Fitur edit tidak diimplementasikan pada versi final | Pass |
| Hapus data Stock Transactions | Fitur tidak error tapi data tidak terhapus | Fitur delete tidak diimplementasikan pada versi final | Pass |
| Hapus data Diseases | Fitur tidak error tapi data tidak terhapus | Diperbaiki developer, diuji ulang | Pass |

Semua *bug* didokumentasikan berdasarkan modul, skenario pengujian, dan bagaimana mereka memengaruhi proses bisnis sistem.

Selanjutnya, hasil disampaikan kepada pengembang untuk evaluasi dan pemilihan solusi terbaik. Dalam beberapa kasus, solusi bukan perbaikan fungsi, tetapi penyesuaian kebutuhan sistem, seperti menghapus fitur yang dianggap tidak lagi berguna. Sebelum penyusunan laporan akhir penelitian, semua hasil penanganan *bug* tersebut dimasukkan ke dalam evaluasi kualitas sistem.

Berikutnya dilakukan perbandingan antara pengujian otomatis dan pengujian secara manual. Pada tahap ini dilakukan evaluasi terhadap hasil pengujian dengan membandingkan waktu eksekusi pengujian otomatis dan manual menggunakan *Cypress*. Tujuan dari perbandingan ini adalah untuk mengevaluasi seberapa efektif pengujian otomatis mendukung proses pengujian berbasis STLC *Agile*, terutama dalam hal pengujian *regression* dan *end-to-end* (E2E). Data yang dianalisis diperoleh dari pengukuran waktu yang digunakan untuk menjalankan pengujian pada beberapa modul utama, yang sering mengalami perubahan selama proses pengembangan sistem.

**Tabel 2.** Perbandingan Waktu Manual & Otomatis

| Modul | Otomatis (detik) | Manual (detik) | Selisih (detik) |
|---|---|---|---|
| Login | 10 | 19 | 9 |
| Create Medicine Stock | 8 | 149 | 141 |
| Create Stock Transaction | 11 | 59 | 48 |
| Create Doctor | 15 | 230 | 215 |
| Create Patients | 19 | 215 | 196 |
| Create Patient Visit | 22 | 376 | 354 |

Pengujian otomatis membutuhkan waktu eksekusi yang jauh lebih singkat dibandingkan dengan pengujian manual pada seluruh modul yang diuji, seperti yang ditunjukkan oleh Tabel 2. Modul *Create* Kunjungan Pasien, *Create* Dokter, dan *Create* Stok Obat membutuhkan waktu yang lebih lama karena melibatkan banyak langkah *input*, validasi data, dan perpindahan halaman, sementara pengujian otomatis memiliki kemampuan untuk mengeksekusi langkah yang sama secara konsisten dan cepat tanpa dipengaruhi faktor kelelahan atau kesalahan manusia.

**Tabel 3**. Manual & Otomatis E2E

| Jenis Pengujian | Modul | Otomatis | Manual | Selisih |
|---|---|---|---|---|
| **E2E** | Login-Patient-Patient Visit-Stock Transac-tion Update | 33 | 409 | 376 |

Selanjutnya, hasil pengujian *end-to-end*, yang ditampilkan pada Tabel 2, menunjukkan bahwa pengujian otomatis sangat efektif dalam menguji alur bisnis yang luas, mulai dari proses *login* hingga pembaruan stok obat setelah kunjungan pasien. Pengujian E2E otomatis dapat menjalankan alur yang sama dalam waktu yang jauh lebih singkat dengan hasil yang konsisten, sedangkan pengujian E2E manual membutuhkan waktu yang cukup lama karena seluruh alur harus dijalankan secara berurutan dan teliti. Hal ini menunjukkan bahwa menggunakan *automation testing* sangat membantu kebutuhan *regression testing*, terutama dalam kasus di mana sistem diperbarui secara berulang dalam pendekatan *Agile*.

Berdasarkan hasil perbandingan pada tabel 4, baik pengujian manual maupun otomatis tidak menemukan bug sistem pada modul yang diuji, yang ditunjukkan dengan hasil pengujian yang sebagian besar berstatus pass. Hal ini menunjukkan bahwa automation testing yang diterapkan mampu merepresentasikan kondisi sistem secara akurat dan menghasilkan output yang konsisten dengan pengujian manual. Dengan demikian, dari sisi validitas hasil, pengujian otomatis dapat dinilai setara dengan pengujian manual dalam mendeteksi kesesuaian fungsi sistem.

**Tabel 4.** Perbandingan Kualitas Pengujian Manual & Otomatis

| Modul | Metode | Hasil | Kendala Pengujian | Keterang-an |
|---|---|---|---|---|
| **Login** | Otomatis | Pass | Tidak ada | Eksekusi stabil tanpa kendala |
| | Manual | Pass | Tidak ada | Tidak terdapat hambatan pengujian |
| **Doctor** | Otomatis | Pass | Tidak ada | Data konsisten setiap pengujian |
| | Manual | Pass | Tidak ada | Tidak terdapat hambatan pengujian |
| **Patient** | Otomatis | Pass | Tidak ada | Konsisten setiap eksekusi |
| | Manual | Pass | Tidak ada | Tidak terdapat hambatan pengujian |
| **Medi-cine Stock** | Otomatis | Pass | Tidak ada | Data di-reset sebelum pengujian |
| | Manual | Kurang optimal | Ada (berdam-pak) | Tidak ada delete, data menumpuk |
| **Stock Tran-saction** | Otomatis | Pass | Tidak ada | Kondisi data selalu bersih |
| | Manual | Kurang optimal | Ada (berdam-pak) | Tidak ada delete, data menumpuk |
| **Patient Visit** | Otomatis | Pass | Tidak ada | Alur pengujian stabil |
| | Manual | Kurang efisien | Ada (berdam-pak) | Tidak ada delete, sulit mengulang skenario |

Namun demikian, terdapat perbedaan yang cukup signifikan pada aspek pelaksanaan pengujian, khususnya pada modul *medicine stock, stock transaction*, dan *patient visit*. Pada pengujian manual, tidak tersedianya fitur *delete* menyebabkan terjadinya penumpukan data yang berdampak pada kesulitan dalam mengontrol kondisi data uji serta mengulang skenario pengujian. Hal ini menjadi kendala yang cukup berpengaruh, terutama ketika pengujian dilakukan secara berulang dalam konteks *regression testing*. Sebaliknya, pada pengujian otomatis, kondisi data dapat dikendalikan melalui mekanisme reset sebelum proses pengujian dijalankan, sehingga setiap skenario dapat dieksekusi dalam kondisi awal yang konsisten.

Kondisi tersebut menunjukkan bahwa pada sistem yang masih dalam tahap pengembangan dan menggunakan pendekatan *Agile*, yang ditandai dengan adanya perubahan dan pengujian berulang, *automation testing* memiliki keunggulan dalam menjaga konsistensi dan efisiensi proses pengujian. Integrasi antara *automation testing*, *end-to-end testing*, dan *regression testing* dalam kerangka STLC berbasis *Agile* terbukti mampu mendukung proses pengujian yang lebih adaptif, terstruktur, dan berkelanjutan. Oleh karena itu, dapat disimpulkan bahwa metode yang digunakan dalam penelitian ini efektif dalam meningkatkan kualitas proses pengujian, khususnya pada sistem yang bersifat dinamis dan terus berkembang.

Untuk menilai tingkat keberhasilan keseluruhan proses pengujian, tahap hasil akhir pengujian digunakan data hasil eksekusi *test case* pada setiap modul sistem *website* klinik. Test case dibuat sebanyak 113 jenis yang tersebar di 14 modul, yaitu modul *login, lab categories, lab, patients, department, doctor, clinic profile, medicine categories, medicines, medicine stock, stock transactions, desease ICD-10, patient visit,* dan *healtcare flow*.

Pengujian dilakukan secara bertahap hingga tujuh kali uji yang berjalan pada beberapa tanggal pengujian seiring dengan pembaruan dan penyesuaian fitur. Selanjutnya dilakukan evaluasi terhadap setiap modul berdasarkan jumlah *test case* yang diuji dan jumlah *test case* yang mengalami kegagalan (*fail*) sebelum perbaikan atau penyesuaian fitur dilakukan.

Secara keseluruhan, 113 tes telah dieksekusi pada semua modul sistem, menurut tabel hasil pengujian. Pada tahap awal pengujian, ada sejumlah tes yang berstatus *fail* pada beberapa modul, termasuk Lab *Categories, Patients, Departments, Doctors, Medicines, Medicine Stock, Stock Transactions*, dan *Patient Visit*. Hal tersebut seperti dapat dilihat pada Tabel 5 berikut.

**Tabel 5.** *Test Pass*

| Modul | Jumlah Test Case | Fail | | | | | | | Status Akhir |
|---|---|---|---|---|---|---|---|---|---|
| | | 10-Oct | 14-Oct | 20-Oct | 23-Oct | 29-Oct | 30-Oct | 1-Jan | |
| Login | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pass |
| Lab Categories | 7 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | Pass |
| Lab | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pass |
| Patients | 9 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | Pass |
| Department | 9 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | Pass |
| Doctor | 9 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | Pass |
| Clinic Profile | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pass |
| Medicine Categories | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pass |
| Medicines | 9 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | Pass |
| Medicine Stock | 8 | 0 | 0 | 0 | 2 | 2 | 2 | 1 | Pass (Fitur Dihapus) |
| Stock Transactions | 6 | 0 | 0 | 0 | 2 | 2 | 2 | 2 | Pass (Fitur Dihapus) |
| Disease ICD-10 | 8 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | Pass |
| Patient Visit | 14 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | Pass (Fitur Dihapus) |
| Healthcare Flow | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pass |
| **TOTAL** | **113** | **5** | **5** | **0** | **5** | **7** | **5** | **4** | **Pass** |

Ketidaksesuaian fungsi dengan persyaratan sistem, kesalahan dalam proses edit atau detail data, dan perubahan kebijakan pengembangan sistem adalah beberapa sumber kegagalan sistem. Developer memperbaiki sebagian besar kegagalan selama proses pengujian berbasis *Agile*. Di sisi lain, fitur seperti *Stock Transactions, Medicine Stock*, dan *Patient Visit* diputuskan untuk ditiadakan karena tidak lagi digunakan dalam proses bisnis klinik. Penentuan tingkat keberhasilan ditentukan dari rumus berikut.

$$TK = \frac{TC\ Pass}{TC\ Total} \times 100\% \qquad (1)$$

Keterangan:
TK : Tingkat Keberhasilan
TC Pass : Jumlah test case yang berhasil
TC Total : Jumlah test case yang diuji

Berdasarkan persamaan (1), diperoleh,

$$TK = \frac{113}{113} \times 100\% = 100\% \qquad (2)$$

Hasil ini menunjukkan bahwa penerapan *automation testing* menggunakan *Cypress* dalam pendekatan STLC *Agile* dan metode pengujian *end-to-end, regression* mampu memastikan seluruh fungsi yang dipertahankan dalam sistem berjalan sesuai dengan kebutuhan. Selain itu, proses pengujian juga berperan dalam membantu pengambilan keputusan pengembangan, seperti penghapusan fitur yang tidak relevan, sehingga sistem akhir yang dihasilkan lebih stabil, sesuai kebutuhan proses bisnis klinik, dan siap untuk tahap implementasi selanjutnya.

## 4. KESIMPULAN

Penelitian ini menunjukkan bahwa penerapan pengujian otomatis menggunakan *Cypress* pada *website* administrasi klinik berhasil dilakukan dengan menggunakan metode *Software Testing Life Cycle* (STLC) berbasis *Agile*. Pengujian *end-to-end* dan *regression* digunakan untuk memastikan seluruh fungsi sistem tetap berjalan dengan baik setelah adanya perubahan fitur. Berdasarkan hasil pengujian terhadap 113 *test case*, seluruh modul memperoleh status *pass* dengan tingkat keberhasilan mencapai 100%. Selain itu, pengujian otomatis dinilai lebih konsisten dan efisien dibandingkan pengujian manual, khususnya pada pengujian yang dilakukan secara berulang. Penerapan STLC berbasis *Agile* juga membantu proses pengujian menjadi lebih terstruktur dan mudah menyesuaikan perubahan sistem. Pengembangan selanjutnya dapat dilakukan melalui integrasi CI/CD serta penerapan pada berbagai jenis sistem informasi guna mengetahui efektivitas pengujian otomatis secara lebih luas.

## 5. DAFTAR PUSTAKA

[1] D. A. Kusuma, K. N. Siregar, A. Prabawa, P. Yuniar, Diana, and E. Yuliana, "Rancang Bangun Aplikasi Rekam Medis Elektronik Di Klinik Medika Lestari

Jakarta Pusat," *J. Indones. Manaj. Inform. dan Komun.*, vol. 4, no. 3, pp. 1758–1769, 2023, doi: 10.35870/jimik.v4i3.400.

[2] H. D. Putra, M. I. Syarif, and Asriyadi, "Rancang Bangun Aplikasi Sistem Informasi Klinik Kesehatan Berbasis Web dan Mobile," *Semin. Nas. Tek. Elektro dan Inform.*, pp. 128–135, 2020.

[3] F. Hamdallah *et al.*, "Sistem Manajemen Basis Data Pada Sistem Perpustakaan ( Studi Kasus : SMK AL-WAFA )," pp. 30–32, 2020.

[4] A. R. Rambe, "Pengujian Otomatis Aplikasi Mobile Dengan Teknik Black-Box Menggunakan Appium," 2022.

[5] R. A. Kamisr, "Studi Komparatif Pengujian Manual dan Pengujian Otomatis Dengan Cypress Pada Website," *J. Sains, Teknol. Dan Kesehat.*, vol. 2, no. 4, p. 355, 2025.

[6] D. N. Arjuna, Mohamad Irwan Afandi, and Abdul Rezha Efrat Najaf, "Testing of Thesis Management Application Using STLC Framework and Automation Tools in Physics Study Program UPN 'Veteran' Jatim," *J. Teknol. Dan Open Source*, vol. 8, no. 1, pp. 208–216, 2025, doi: 10.36378/jtos.v8i1.4413.

[7] K. Pal and B. Karakostas, "Software Testing Under Agile, Scrum, and DevOps," *Res. Anthol. Agil. Software, Softw. Dev. Test.*, vol. 2, no. September, pp. 1059–1076, 2022, doi: 10.4018/978-1-6684-3702-5.ch053.

[8] A. U. Afidah and D. G. P. Putri, "Studi Komparasi Teknik Pengujian End-to-End pada E-payment: Studi Kasus Travelink (Sistem Web E-ticketing)," *J. Internet Softw. Eng.*, vol. 5, no. 2, pp. 50–58, 2024, doi: 10.22146/jise.v5i2.11741.

[9] B. Prakoso, "Pemanfaatan Cypress Untuk Pengujian End-To-End (Studi Kasus: Pengembangan Aplikasi Indicar)," *J. Ilm. Teknol. Inf. dan Robot.*, vol. 4, no. 1, 2023.

[10] R. Pan, M. Bagherzadeh, T. A. Ghaleb, and L. Briand, "Test Case Selection and Prioritization Using Machine Learning: A Systematic Literature Review," *Empir. Softw. Eng.*, vol. 27, no. 2, pp. 1–34, 2021, doi: 10.1007/s10664-021-10066-6.

[11] N. Singh, D. Patel, A. Raj, and M. S. Kour, "CI / CD Pipeline for Web Applications," no. May, 2023.

[12] F. Hamdallah, "ANALISIS KERENTANAN BROKEN ACCESS CONTROL , CRYPTOGRAPHIC FAILURES DAN INJECTION PADA APLIKASI BIMBINGAN KONSELING MENGGUNAKAN WHITE-BOX TESTING," vol. 03, no. 03, pp. 528–535, 2025.

[13] F. Hamdallah, "PERANCANGAN SISTEM INFORMASI BIMBINGAN KONSELING ( STUDI KASUS : SMA X ) Pendahuluan," vol. 01, no. 04, pp. 832–845, 2023.

[14] I. V. Krasnokutska and O. S. Krasnokutskyi, "Implementing E2E tests with Cypress and Page Object Model: evolution of approaches," *CEUR Workshop Proc.*, vol. 3662, no. December, pp. 101–110, 2024.

[15] Y. F. Putri, A. B. P. Irianto, and S. Sharma, "Comparison of Automatic and Manual Regression Testing on Mobile Application Health Technology with Black Box Testing Method," *Indones. J. Inf. Syst.*, vol. 7, no. 2, pp. 218–230, 2025, doi: 10.24002/ijis.v7i2.6850.