JIKOMTI:
Jurnal Ilmiah Ilmu Komputer dan Teknologi Informasi
e- ISSN: 3089-2996
Vol. 2 No. 2, Des 2025

Implementasi Pengujian Otomatis *End-To-End* Menggunakan Cypress dengan Metode *Black Box Testing* untuk Meningkatkan Kualitas Aplikasi EdTech XYZ Berbasis Website

Rina Septiani[1], Budi Jejen Zaenal Abidin[2], Angge Firizkiansah[3]
[1,2,3]Universitas Sains Indonesia, Kabupaten Bekasi

E-mail:
septianirina05@gmail.com[1*], budi.jejen@lecturer.sains.ac.id[2],
angge.firizkiansah@lecturer.sains.ac.id[3]

Abstract

Software quality is an important aspect of development because it affects user satisfaction. Therefore, testing is necessary to ensure that the software runs according to specifications and minimizes errors. However, manual testing tends to be time-consuming, resource-intensive, and prone to human error. This study focuses on improving software quality through the implementation of end-to-end automated testing using Cypress with the black box testing method and following the Software Testing Life Cycle (STLC) stages. The test results show that all test cases were successfully executed with a pass status after improvements were made to the program code, which means that the application is functioning according to specifications. Thus, automated testing has proven to be effective and efficient in improving application quality. Further research is recommended to integrate Cypress with the CI/CD pipeline to support continuous testing automation.

Keywords: *automated testing*; *black box; cypress*; *end-to-end testing; website*

Abstrak

Kualitas perangkat lunak merupakan aspek penting dalam pengembangan karena berpengaruh terhadap kepuasan pengguna. Oleh karena itu, pengujian diperlukan untuk memastikan perangkat lunak berjalan sesuai spesifikasi dan meminimalkan kesalahan. Namun, pengujian manual cenderung memakan waktu yang lama, memerlukan banyak sumber daya, dan rentan terhadap kesalahan manusia. Penelitian ini berfokus pada peningkatan kualitas perangkat lunak melalui penerapan pengujian otomatis *end-to-end* menggunakan *Cypress* dengan metode *black box testing* serta mengikuti tahapan *Software Testing Life Cycle* (STLC). Hasil pengujian menunjukkan seluruh *test case* berhasil dijalankan dengan status *pass* setelah dilakukan perbaikan pada kode program, yang berarti bahwa aplikasi telah berfungsi sesuai spesifikasi. Dengan demikian, pengujian otomatis terbukti efektif dan efisien dalam meningkatkan kualitas aplikasi. Penelitian selanjutnya disarankan untuk mengintegrasikan *Cypress* dengan CI/CD pipeline guna mendukung otomatisasi pengujian berkelanjutan.

Kata kunci: pengujian otomatis; *black box*; *cypress; end-to-end testing*; *website*

1. PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi telah memberikan dampak pada banyak aspek kehidupan, termasuk sistem informasi. Sistem informasi memberikan informasi yang relevan dan cepat, sehingga dapat membantu dalam proses pengambilan keputusan[1]. Salah satu teknologi informasi yang paling sering digunakan saat ini adalah website. Website digunakan sebagai media untuk menyajikan informasi dari data yang telah diolah, dengan demikian dapat diakses oleh banyak pengguna pada waktu yang sama[2]. Keberhasilan sistem sangat dipengaruhi oleh kualitasnya, karena kualitas yang baik dapat meningkatkan kepuasan pengguna. Sistem yang andal, mudah digunakan, dan minim kesalahan akan membuat pengguna merasa nyaman saat menggunakannya. Hal ini memberikan pengalaman yang positif dan meningkatkan kepercayaan pengguna.

Secara umum, salah satu masalah yang dihadapi dalam pengembangan perangkat lunak saat ini adalah rendahnya kualitas pada perangkat lunak. Agar kualitas bebas dari bug, perlunya dilakukan pengujian pada perangkat lunak[3]. Dalam proses pengembangan sistem atau aplikasi, tim pengembang umumnya perlu melakukan pengujian terlebih dahulu sebelum sistem diluncurkan ke publik[4]. Jika pengujian tidak dilakukan dengan baik, perangkat lunak berisiko mengalami error yang dapat muncul dalam berbagai bentuk, seperti kegagalan fungsi, data yang tidak akurat, atau sistem yang berhenti bekerja secara tiba-tiba. Kondisi tersebut berpotensi menimbulkan gangguan pada operasional, mempersulit pengelolaan sistem, dan memerlukan banyak waktu untuk memperbaiki masalah. Untuk memastikan perangkat lunak yang dikembangkan berfungsi sesuai harapan, pengujian ini sangat penting dilakukan[5].

Website EdTech (*Educational Technology*) XYZ adalah platform pemesanan dan konsultasi online untuk kelas bimbingan dan pengayaan, yang memudahkan pengguna untuk menemukan dan mengakses layanan pendidikan secara efisien. Salah satu tantangan terbesar dalam mengembangkan website XYZ adalah memastikan kualitasnya yang konsisten di seluruh fitur. Kompleksitas fitur menuntut proses pengujian yang dilakukan secara berulang untuk memastikan setiap fitur dapat berfungsi dengan baik. Namun, dalam praktiknya, proses pengujian yang dilakukan masih menghadapi kendala, seperti keterbatasan dalam menjalankan seluruh skenario pengujian secara konsisten, kebutuhan waktu dan tenaga yang cukup besar, dan potensi adanya skenario tertentu yang tidak teruji. Hal ini menyebabkan kualitas aplikasi belum dapat dipastikan secara menyeluruh.

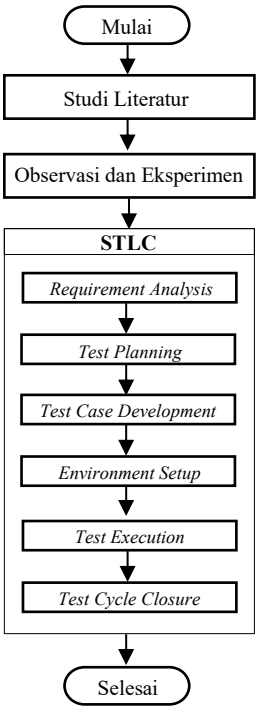
Kondisi tersebut tidak hanya terjadi pada website EdTech XYZ, tetapi juga menjadi permasalahan umum dalam pengembangan perangkat lunak. Banyak yang kemudian beralih dari pengujian manual ke pengujian otomatis untuk mengurangi biaya dan waktu yang diperlukan, sehingga proses pengujian menjadi lebih cepat hingga peluncuran sistem[4]. Berdasarkan uraian tersebut, permasalahan dalam penelitian ini berkaitan dengan bagaimana pengujian otomatis dapat dijalankan secara konsisten sesuai dengan skenario pengujian yang telah dirancang, bagaimana penerapannya memengaruhi efisiensi proses pengujian dibandingkan dengan pengujian manual, dan bagaimana hasil pengujian otomatis dapat berkontribusi terhadap peningkatan kualitas website EdTech XYZ.

Untuk menjawab permasalahan tersebut, penelitian ini menerapkan pengujian otomatis menggunakan Cypress, sebuah alat otomatisasi pengujian *end-to-end* untuk Antarmuka Pengguna Grafis (GUI) aplikasi web yang dirancang untuk mempermudah kehidupan pengembang dan *quality assurance*[6]. Metode yang digunakan adalah *black box testing* karena pengujian difokuskan pada respons sistem terhadap data masukan tanpa memerhatikan struktur internalnya[7]. Tahapan pengujian dilakukan berdasarkan *Software Testing Life Cycle* (STLC), yaitu rangkaian proses untuk menguji suatu perangkat lunak dengan langkah-langkah yang terencana dan sistematis[8].

Penelitian sebelumnya menunjukkan keberhasilan pengujian *end-to-end* menggunakan Cypress, namun cakupannya masih terbatas pada skenario tertentu[9], sementara penelitian lain dengan Katalon Studio menunjukkan peningkatan kecepatan eksekusi tetapi belum menerapkan teknik khusus dalam perancangan skenario uji[10]. Oleh karena itu, penelitian ini menggunakan variasi *test case* yang lebih banyak dengan teknik *use case testing* dan *equivalence partitioning* untuk memperluas cakupan pengujian dan mengurangi potensi *human error*. Penelitian ini bertujuan untuk menilai efektivitas Cypress dalam menjalankan skenario pengujian, membandingkan efisiensi proses pengujian otomatis dengan pengujian manual, dan mengevaluasi kontribusinya terhadap peningkatan kualitas website EdTech XYZ.

2. METODE

Penelitian ini melakukan pengujian fungsional *end-to-end* menggunakan Cypress dengan metode *black box* dan tahapan *Software Testing Life Cycle* (STLC) agar pengujian berjalan terstruktur dan sistematis. Seluruh proses penelitian ditunjukkan pada Gambar 1.



Gambar 1. Tahapan Penelitian

2.1. Studi Literatur

Tahap penelitian diawali dengan menelaah berbagai jurnal dan buku yang membahas penerapan pengujian otomatis pada aplikasi berbasis website. Dalam penelitian ini, Cypress dipilih karena mendukung pengujian *end-to-end*, mudah digunakan, bersifat *open-source*, memiliki tampilan yang interaktif, dan didukung oleh komunitas yang luas. Metode pengujian yang digunakan adalah *black box testing*, karena berfokus pada pengujian fungsionalitas aplikasi tanpa perlu memahami kode program.

2.2. Observasi dan Eksperimen

Pada tahap observasi, dilakukan pengamatan terhadap aplikasi versi 1 untuk memahami alur sistem dan menentukan cakupan pengujian pada aplikasi versi 2. Aplikasi memiliki dua peran utama, yaitu admin dan customer. Tahap eksperimen dilakukan untuk menyusun dan memvalidasi skenario pengujian melalui uji coba manual sebelum diotomatisasi. Sebanyak 145 skenario pengujian ditetapkan, terdiri dari 138 skenario admin, 5 skenario customer, dan 2 skenario integrasi, yang selanjutnya akan digunakan pada tahap *requirement analysis*.

2.3. *Requirement Analysis*

Tahap *requirement analysis* bertujuan untuk menganalisis kebutuhan pengujian berdasarkan hasil observasi dan eksperimen yang telah dilakukan sebelumnya. Proses pada tahap ini diawali dengan memahami kebutuhan perangkat lunak yang akan diuji[11]. Hasil analisis menghasilkan skenario pengujian, tingkat prioritas, serta hasil yang diharapkan dari setiap skenario. Perbedaan tingkat prioritas ditentukan berdasarkan peran dan tingkat kepentingan fitur dalam sistem. Hasil analisis ini menjadi dasar dalam perancangan *test plan* dan pengembangan *test case*, yang selanjutnya dieksekusi pada tahap *Test Execution*.

2.4. *Test Planning*

Pada tahap ini, dilakukan perencanaan pengujian yang mencakup penentuan strategi, ruang lingkup, jadwal, sumber daya, dan lingkungan pengujian yang akan digunakan. Tahap ini bertujuan agar proses pengujian dapat

dilaksanakan secara terarah dan efisien sesuai dengan rencana pengujian dan tujuan penelitian yang telah ditetapkan. Hasil dari tahap ini berupa dokumen yang mencakup rencana pengujian yang akan digunakan dalam *test case*, kemudian penguji akan mulai mengembangkannya[12].

2.5. *Test Case Development*

Pada tahap ini, dilakukan pengembangan kasus uji sebagai acuan untuk pelaksanaan pengujian otomatis menggunakan Cypress. Setelah rencana pengujian selesai, *test case* dirancang oleh penguji untuk setiap fitur yang akan diuji[13]. Skenario pengujian dirancang lebih rinci, mencakup langkah-langkah pengujian, data uji, dan hasil yang diharapkan. Dalam perancangannya, teknik pengujian *black box* seperti *use case testing* dan *equivalence partitioning* diterapkan untuk menentukan variasi skenario serta data uji yang relevan. Hasil dari tahap ini menjadi acuan untuk penulisan skrip otomatis pada tahap *Test Execution*, sehingga setiap langkah pengujian dapat diimplementasikan dengan tepat sesuai *test case* yang telah dirancang.

2.6. *Environment Setup*

Pada tahap ini, dilakukan persiapan dan konfigurasi lingkungan pengujian agar mendukung pelaksanaan pengujian sesuai dengan rencana yang telah ditetapkan[14]. Tujuannya agar lingkungan pengujian siap digunakan dan dapat mendukung proses pengujian menggunakan Cypress. Perangkat lunak yang digunakan meliputi Node.js, XAMPP, Visual Studio Code, Cypress, Google Chrome, dan Google Sheets. Konfigurasi perangkat lunak dilakukan untuk menyesuaikan lingkungan pengujian dengan kebutuhan pengujian otomatis.

2.7. *Test Execution*

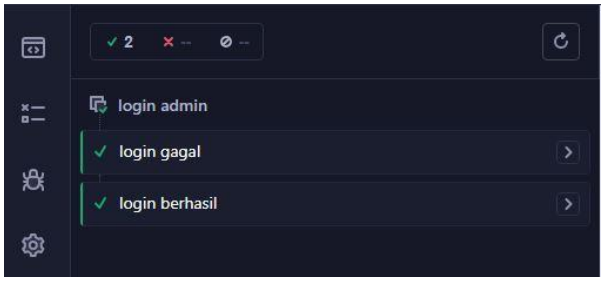
Pada tahap ini, dilakukan pengujian perangkat lunak yang didasarkan pada *test plan* dan *test case* yang telah dirancang sebelumnya[15]. Selanjutnya, pengujian otomatis dilakukan dengan langkah-langkah berikut.

a. Penulisan Skrip Pengujian Otomatis

Penulisan skrip dilakukan menggunakan Visual Studio Code dengan bahasa pemrograman JavaScript. Skrip ini disusun berdasarkan hasil *Test Case Development*, sehingga setiap langkah pengujian direpresentasikan dalam bentuk kode. Secara keseluruhan, terdapat 145 skenario pengujian dan 228 *test case* yang mencakup peran admin, customer, dan integrasi keduanya.

b. Eksekusi Pengujian

Setelah seluruh skrip pengujian disusun, tahap selanjutnya adalah menjalankan pengujian menggunakan Cypress melalui Visual Studio Code dengan metode *black box testing*. Proses eksekusi dimulai dengan membuka *End-to-End Testing*, kemudian memilih file pengujian yang akan dijalankan pada Cypress Test Runner menggunakan browser Google Chrome. Pada saat pengujian berlangsung, Cypress menampilkan setiap langkah yang dieksekusi sesuai dengan skrip dan memberikan status hasil berupa *pass* jika pengujian berhasil atau *fail* jika terjadi kegagalan. Gambar 2 dan Gambar 3 menunjukkan contoh hasil pengujian otomatis menggunakan Cypress.



Gambar 2. Hasil Pengujian *Pass*

Gambar 2 menunjukkan hasil pengujian untuk skenario login admin, di mana kedua *test case* yang dijalankan berhasil dan mendapatkan status *pass*. Hasil ini menunjukkan bahwa proses login sudah berjalan sesuai dengan *expected result*. Adapun gambar berikut menunjukkan contoh hasil pengujian dengan status *fail*.

Gambar 3. Hasil Pengujian *Fail*

Pada Gambar 3 menunjukkan kegagalan yang terjadi karena elemen yang dicari tidak berhasil ditemukan dalam batas waktu yang telah ditentukan, sehingga Cypress menghentikan proses eksekusi. Kondisi ini biasanya muncul ketika halaman membutuhkan waktu lebih lama untuk memuat data atau elemen tertentu. Kendala tersebut tidak termasuk bug pada aplikasi karena tidak berkaitan dengan kesalahan fungsional sistem yang disebabkan oleh kode program. Pada kondisi normal, ketika waktu tunggu sudah cukup dan elemen berhasil muncul tepat waktu, pengujian tidak akan menghasilkan error seperti ini.

2.8. *Test Cycle Closure*

Pada tahap akhir ini, dilakukan evaluasi dengan membandingkan hasil pengujian manual dan pengujian otomatis yang meliputi total waktu eksekusi setiap *test case* dan keseluruhan pengujian, jumlah *test case* dengan status *pass* dan *fail*, serta tingkat keberhasilan pengujian sebelum dan setelah dilakukan perbaikan pada kode program.

3. HASIL DAN PEMBAHASAN

Pengujian otomatis dilakukan berdasarkan skrip yang telah disusun dan dieksekusi menggunakan Cypress, yang mencakup peran admin, customer, dan pengujian integrasi. Pengujian ini terdiri atas 138 skenario pengujian dengan 214 *test case* untuk peran admin, 5 skenario pengujian dengan 8 *test case* untuk peran customer, serta 2 skenario pengujian dengan 6 *test case* untuk pengujian integrasi. Selama proses pengujian, ditemukan 16 bug, yang terdiri atas 14 bug pada peran admin dan 2 bug pada peran customer. Selanjutnya, hasil pengujian dilaporkan kepada tim pengembang untuk dilakukan perbaikan pada kode program. Setelah proses perbaikan selesai, seluruh *test case* diuji kembali untuk memastikan bahwa tidak terdapat bug yang tersisa.

Waktu eksekusi pengujian manual dan pengujian otomatis dihitung berdasarkan setiap *test case* dan keseluruhan pengujian. Waktu eksekusi setiap *test case* pada masing-masing metode pengujian dijumlahkan untuk memperoleh total waktu pengujian manual dan pengujian otomatis. Selanjutnya, total waktu dari kedua metode tersebut dibandingkan untuk menilai efisiensi pengujian otomatis menggunakan Cypress. Tabel 1–3 menunjukkan waktu eksekusi setiap *test case* pada pengujian manual, pengujian otomatis menggunakan Cypress, dan perbandingannya.

Tabel 1. Total Waktu Manual per *Test Case*

Peran	Total Waktu
Admin	1 jam 36 menit 35 detik
Customer	4 menit 47 detik
Terintegrasi	8 menit 26 detik
Total	1 jam 49 menit 48 detik

Tabel 2. Total Waktu Otomatis per *Test Case*

Peran	Total Waktu
Admin	1 jam 14 menit 51 detik
Customer	3 menit 30 detik
Terintegrasi	6 menit 25 detik
Total	1 jam 24 menit 46 detik

Tabel 3. Perbandingan Waktu per Test Case

Metode Pengujian	Total Waktu
Manual	1 jam 49 menit 48 detik
Otomatis	1 jam 24 menit 46 detik
Selisih	25 menit 2 detik

Berdasarkan total waktu eksekusi setiap *test case* pada Tabel 1–3, pengujian otomatis menunjukkan efisiensi yang lebih tinggi dibandingkan pengujian manual, dengan selisih waktu sebesar 25 menit 2 detik.

Selanjutnya, waktu eksekusi dihitung berdasarkan keseluruhan pengujian untuk masing-masing metode pengujian manual dan pengujian otomatis, di mana seluruh proses dari awal hingga akhir dicatat sebagai satu durasi

total, bukan dihitung berdasarkan waktu setiap *test case*. Tabel 4–6 menunjukkan waktu eksekusi keseluruhan pengujian untuk masing-masing metode dan perbandingannya.

Tabel 4. Total Waktu Pengujian Manual

Peran	Total Waktu
Admin	3 jam 56 menit 27 detik
Customer	10 menit 20 detik
Terintegrasi	10 menit 30 detik
Total	4 jam 17 menit 17 detik

Tabel 5. Total Waktu Pengujian Otomatis

Peran	Total Waktu
Admin	3 jam 31 menit 31 detik
Customer	8 menit 32 detik
Terintegrasi	7 menit 39 detik
Total	3 jam 47 menit 42 detik

Tabel 6. Perbandingan Waktu Total

Metode Pengujian	Total Waktu
Manual	4 jam 17 menit 17 detik
Otomatis	3 jam 47 menit 42 detik
Selisih	29 menit 35 detik

Berdasarkan total waktu eksekusi keseluruhan pada Tabel 4–6, pengujian otomatis menunjukkan efisiensi yang lebih tinggi dibandingkan pengujian manual, dengan selisih waktu sebesar 29 menit 35 detik.
Perbedaan waktu eksekusi pengujian dapat dipengaruhi oleh kondisi sistem, jaringan, dan server pada saat pengujian dilakukan.

Setelah diperoleh perbandingan waktu eksekusi antara pengujian manual dan otomatis, analisis dilanjutkan dengan melihat hasil pengujian otomatis berupa jumlah *pass* dan *fail* serta tingkat keberhasilan. Jumlah *test case pass* dan *fail* sebelum dan setelah perbaikan kode program disajikan pada Tabel 7 dan Tabel 8.

Tabel 7. Hasil Pengujian Sebelum Perbaikan

Peran	Test Skenario	Test Case	TC Pass	TC Fail
Admin	138	214	200	14
Customer	5	8	6	2
Terintegrasi	2	6	6	0
Total	145	228	212	16

Tabel 8. Hasil Pengujian Setelah Perbaikan

Peran	Test Skenario	Test Case	TC Pass	TC Fail
Admin	138	214	214	0
Customer	5	8	8	0
Terintegrasi	2	6	6	0
Total	145	228	228	0

Berdasarkan Tabel 7 dan Tabel 8, tingkat keberhasilan pengujian otomatis dianalisis dengan membandingkan kondisi sebelum dan setelah perbaikan pada kode program oleh tim pengembang untuk melihat peningkatan kualitas aplikasi. Tingkat keberhasilan dapat dihitung menggunakan persamaan (1).

$$\text{TK} = \frac{\text{TC Pass}}{\text{TC Total}} \text{ x } 100\% \quad (1)$$

Dimana:
TK : Tingkat keberhasilan
TC Pass : Jumlah *test case* yang berhasil
TC Total : Total *test case* yang diuji

Berdasarkan persamaan (1), tingkat keberhasilan pengujian otomatis dihitung menggunakan data hasil pengujian sebelum dan setelah perbaikan kode.

Sebelum perbaikan: $\text{TK} = \frac{212}{228} \text{ x } 100\% = 93\%$

Setelah perbaikan: $\text{TK} = \frac{228}{228} \text{ x } 100\% = 100\%$

Hasil perhitungan menunjukkan peningkatan tingkat keberhasilan pengujian otomatis, yang berarti kualitas aplikasi secara fungsional juga meningkat. Lebih banyak fitur berjalan sesuai harapan dan bug yang sebelumnya ditemukan telah diperbaiki. Peningkatan ini terjadi karena pengujian otomatis mampu menemukan bug lebih cepat, sehingga proses perbaikan kode dapat dilakukan dengan lebih efisien.

4. KESIMPULAN

Berdasarkan evaluasi total waktu eksekusi, pengujian otomatis umumnya lebih efisien dibandingkan pengujian manual. Meskipun beberapa *test case* tertentu dapat lebih cepat dijalankan manual, otomatisasi tetap memberikan efisiensi waktu yang lebih

signifikan karena mengurangi intervensi manusia serta mempermudah pengujian berulang dan pengecekan bug. Seluruh skenario pengujian telah berhasil dijalankan menggunakan Cypress.

Dari 228 *test case* yang diuji, ditemukan 16 bug, sehingga tingkat keberhasilan awal sebesar 93%. Setelah perbaikan kode, seluruh *test case* berhasil dijalankan dengan tingkat keberhasilan 100%. Hal ini menunjukkan bahwa aplikasi telah berfungsi sesuai spesifikasi dan kualitas fungsionalnya meningkat. Dengan demikian, pengujian otomatis tidak hanya efisien dari segi waktu dan tenaga, tetapi juga efektif dalam meningkatkan kualitas aplikasi.

Saran untuk penelitian selanjutnya adalah melakukan integrasi pengujian otomatis menggunakan Cypress dengan CI/CD pipeline, sehingga proses pengujian dapat berjalan secara berkelanjutan, lebih efisien, dan konsisten dalam memastikan kualitas aplikasi.

5. DAFTAR PUSTAKA

[1] A. L. Kalua, "Penerapan Extreme Programming Pada Sistem Informasi Keuangan Sekolah Berbasis Website," *J. Ilm. Inform. DAN ILMU Komput.*, vol. 1, no. September, pp. 69–76, 2022.

[2] Y. Wahyudin and D. N. Rahayu, "Analisis Metode Pengembangan Sistem Informasi Berbasis Website: A Literatur Review," *J. Interkom J. Publ. Ilm. Bid. Teknol. Inf. dan Komun.*, vol. 15, no. 3, pp. 119–133, 2020, doi: 10.35969/interkom.v15i3.74.

[3] Mintarsih, "Pengujian Black Box Dengan Teknik Transition Pada Sistem Informasi Perpustakaan Berbasis Web Dengan Metode Waterfall Pada SMC Foundation," *J. Teknol. Dan Sist. Inf. Bisnis*, vol. 5, no. 1, pp. 33–35, 2023, doi: 10.47233/jteksis.v5i1.727.

[4] B. D. Saputra and A. Stefanie, "Automation Testing Api, Android, dan Website Menggunakan Serenity Bdd Pada Software Sistem Manajemen Rumah Sakit," *J. Ilm. Wahana Pendidik.*, vol. 9, no. 10, pp. 114–126, 2023, [Online]. Available: https://doi.org/10.5281/zenodo.7983405.

[5] D. Agustin, Herlinah, and Nasrullah, "Implementasi Pengolahan Citra Digital Untuk Mengidentifikasi Penyakit Tanaman Anggrek Menggunakan Algoritma K-Nearest Neighbor (K-NN)," *Semin. Nas. Tek. Elektro*, vol. 1, pp. 86–91, 2024.

[6] M. T. Taky, "Automated Testing with Cypress," 2021.

[7] R. A. Al Ayyubi, M. K. Anam, and H. Permatasari, "Pengujian Sistem Informasi Penjualan Menggunakan Pengujian Black Box Testing dan White Box Testing," *Senatib*, pp. 529–536, 2023.

[8] R. Effendi Ibrahim, N. Rachma Sari, and A. Wahyu, "Analisis dan Pengujian dengan Menggunakan Value Analysis dan Metode Equivalence Partitioning," *J. Rein (Rekayasa Inform.*, vol. 1, no. 2, pp. 110–115, 2024.

[9] M. G. Alkhairi, S. P. A. Alkadri, and P. Y. Utami, "Implementasi Unit Testing Dan End-To-End Testing Pada Sistem Informasi Akademik Teknik Informatika," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 9, no. 4, pp. 2208–2219, 2024, doi: 10.29100/jipi.v9i4.5626.

[10] A. Zulianto *et al.*, "Pemanfaatan Katalon Studio untuk Otomatisasi Pengujian Black-Box pada Aplikasi iPosyandu," *Jepin (Jurnal Edukasi Dan Penelit. Inform.*, vol. 7, no. 3, pp. 370–378, 2021.

[11] P. V. Maharani, V. Z. Nazah, M. Sholiha, and A. Haikal, "Pengujian Black Box Pada Aplikasi Access By KAI Menggunakan Teknik Equivalence Partitioning," *Biner J. Ilmu Komputer, Tek. dan Multimed.*, vol. 2, no. 2, pp. 189–195, 2024.

[12] Ruliansyah, Tukinob, B. Huda, and A. L.

Hananto, "Penerapan Software Testing Life Cycle Pada Pengujian Otomatisasi Platform Dzikra," *CSRID J.*, vol. 15, no. 1, pp. 1–11, 2023, [Online]. Available: https://www.doi.org/10.22303/csrid.15.1.2023.01-11

[13] T. F. K. Rahman Dilan Syaukanie, Wahjoe Witjaksono, "Implementasi Automated Testing untuk Proses Pengujian pada SAP ECC dengan Menggunakan Software TestComplete (Studi Kasus PT Telkom Indonesia)," *e-Proceeding Eng.*, vol. 11, no. 4, pp. 4327–4332, 2024.

[14] A. Arfan and Hendrik, "Penerapan STLC dalam Pengujian Automation Aplikasi Mobile (Studi kasus: LMS Amikom Center PT.GIT Solution)," *Jur. Inform. Univ. Islam Indones.*, vol. 3, no. 2, pp. 1–6, 2022, [Online]. Available: https://jurnal.uii.ac.id/AUTOMATA/article/view/24127/

[15] F. W. Azhari and D. F. Suyatno, "Pengujian Otomatis GUI dengan Katalon Studio pada Situs Lowongan Kerja Jobstreet dan Glints," *J. Emerg. Inf. Syst. Bus. Intell.*, vol. 05, no. 03, pp. 205–213, 2024.