Black Box Testing dengan Metode *Equivalence Partitioning* pada *Techno Expertise Academy* (TEA) Astra Credit Companies

Meiselino Ansfridus[*1], B Yudi Dwiandiyanta[2], Andi Wahju Rahardjo Emanuel[3]

[1-3] Program Studi Informatika, Universitas Atma Jaya Yogyakarta

[1] Departemen Teknik Elektro dan Teknologi Informasi, Universitas Gadjah Mada

[1] Program Profesi Insinyur, Institut Teknologi Sepuluh Nopember

E-mail: 170709222@students.uajy.ac.id[*1], yudi.dwiandiyanta@uajy.ac.id[2], andi.emanuel@uajy.ac.id[3]

Abstrak. *Techno Expertise Academy* (TEA) dikembangkan menjadi tempat pembelajaran bidang kompetensi yang berhubungan dengan pengerjaan proyek. Perangkat lunak harus melalui proses pengujian terlebih dahulu sebagai pemenuhan konsep *Software Development Life Cycle* (SDLC) yang secara umum menjadi landasan pengembangan perangkat lunak, termasuk pengembangan *Techno Expertise Academy* (TEA). Proses pengujian *black box* dalam perangkat lunak dapat lakukan dengan cara manual atau otomatis dengan hasil akhir sistem dapat menjalankan fungsi yang ada dengan semestinya. *Black box testing* menggunakan metode *equivalence partitioning* dan *automation tools* Katalon Studio. Hasil dari pengujian manual dan otomatis yang menghasilkan kesiapan aplikasi dengan *score* 100% pada setiap fungsi. Selain itu, pengujian manual dan otomatis pada total 20 fungsi memperoleh hasil 106 *passes* dan 0 *failures*. Pengujian otomatis menggunakan Katalon Studio memangkas waktu pengujian lebih cepat mencapai 64.47% dibandingkan pengujian yang dilakukan secara manual dengan kondisi yang setara. Hal ini juga memenuhi standar kualitas dari perusahaan Astra Credit Companies, dibuktikan dengan seluruh keluaran bersifat '*pass*' dan pengujian User Acceptance Test telah diterima.

Kata kunci: Pengujian *Black Box*; *Equivalence Partitioning*; Katalon Studio; Pengujian Manual; Pengujian Otomatis

Abstract. *Techno Expertise Academy (TEA) was developed as a place to learn competency areas related to project work. Software must first undergo a testing process to fulfill the Software Development Life Cycle (SDLC) concept, which is generally the basis for software development, including the development of the Techno Expertise Academy (TEA). The black box testing process in software can be carried out manually or automatically, with the final result being that the system can properly carry out its existing functions. Black box testing uses the equivalence partitioning method and automation tools Katalon Studio. Manual and automatic testing results produce application readiness with a score of 100% for each function. In addition, manual and automatic testing on 20 functions resulted in 106 passes and 0 failures. Automatic testing using Katalon Studio cuts testing time faster, reaching 64.47% compared to manual testing under equivalent conditions. This also meets the quality standards of the Astra Credit Companies company, proven by all output being 'pass' and the User Acceptance Test has been accepted.*

Keywords: *Black box testing; equivalence partitioning, Katalon Studio, manual testing, automation testing*

1. Pendahuluan

Teknologi informasi sudah menjadi kebutuhan dalam memudahkan proses bisnis suatu instansi atau organisasi [1]. Produk teknologi informasi yang diandalkan dalam pengolahan proses bisnis adalah perangkat lunak [2]. Sebagai alat bantu, perangkat lunak harus memudahkan pengguna dan memenuhi ekspektasi dari tujuan penggunaannya. Hal ini diatur dalam penjamin mutu perangkat lunak yang memastikan perangkat lunak layak untuk digunakan atau tidak [3]. Kerugian dari tidak menerapkan penjamin mutu dalam pengembangan perangkat lunak dapat menyebabkan kerugian reputasi hingga materi [4].

Aplikasi *Quest Master & Techno Expertise Academy* (QMTEA) merupakan suatu platform pelatihan yang dapat diakses oleh sumber daya manusia terkait bidang teknologi informasi. QMTEA memiliki dua bagian, yaitu *Quest Master* (QM) untuk melakukan kolaborasi pengerjaan proyek dan *Techno Expertise Academy* (TEA) yang menjadi tempat pembelajaran bidang kompetensi yang berhubungan dengan pengerjaan proyek. Pengembangan QMTEA dilakukan oleh divisi *Technocenter*, bagian dari Astra Credit Companies yang melayani digitalisasi proses bisnis dengan teknologi informasi untuk internal perusahaan maupun klien luar [5]. Adapun Astra Credit Companies merupakan *financing company* untuk kendaraan mobil dan alat berat [6]. Pengembangan perangkat lunak yang dilakukan oleh perusahaan tentu melalui proses untuk menghasilkan aplikasi yang berjalan dengan baik [7].

Perangkat lunak harus melalui proses pengujian terlebih dahulu sebagai pemenuhan konsep *Software Development Life Cycle* (SDLC) yang secara umum menjadi landasan pengembangan perangkat lunak, termasuk pengembangan *Techno Expertise Academy* (TEA) [8][9]. Pengujian pada perangkat lunak berdasarkan ranah pengujiannya terbagi menjadi *black box* yang berfokus pada pengujian secara fungsional dan *white box* yang terfokus pada pengujian struktural *code* [10]. Proses pengujian *black box* dalam perangkat lunak dapat lakukan dengan cara manual atau otomatis dengan hasil akhir sistem dapat menjalankan fungsi yang ada dengan semestinya [11]. Dalam *black box testing*, terdapat metode *equivalence partitioning* (EP) untuk menguji *input value* menghasilkan nilai keluaran *valid* atau *invalid* [12]. Selain itu, penerapan Katalon Studio sebagai *automation tools* dilakukan untuk menguji secara fungsional dalam pengujian otomatis [13]. Penelitian ini bertujuan untuk mengetahui fungsionalitas *Techno Expertise Academy* (TEA) dapat berjalan dengan semestinya serta mengetahui hasil *black box testing* secara manual dan otomatis dapat memperoleh hasil dan waktu pengujian yang sama saat kondisi setara.

Penelitian yang berkaitan dengan pengujian perangkat lunak sudah banyak dilakukan oleh para peneliti sebelumnya. Penelitian terkait pengujian perangkat lunak sering dilakukan dengan metode *black box testing* pada *website*. Penerapan metode ini bertujuan untuk menentukan kualitas dari perangkat lunak berdasarkan pengujian fungsionalitas demi memenuhi kebutuhan dari *user*. Salah satu penelitian dilakukan untuk memperkirakan kualitas dari *software* yang akan diluncurkan berdasarkan aspek *user interface*. Dengan pengujian yang sudah dilaksanakan, maka pembuat sistem dapat meyakinkan para *customer* bahwa sistem yang dibangun sudah layak untuk digunakan dan memenuhi ekspetasi mereka. Hasil yang didapatkan dengan konsep *cross browser testing*, pengujian secara *front end* dan *back end* mendapatkan hasil yang baik sesuai dengan *user story* [14]. Terdapat juga penelitian lain yang menerapkan metode *black box testing* pada aplikasi *web*. Pengujian menggunakan *test case* pada setiap menu untuk memperoleh hasil yang optimal. Hal ini diperkuat dengan tingkat kepuasan tinggi yang didapatkan dari hasil kuisioner uji beta kepada beberapa pihak terkait penggunaan aplikasi *web* tersebut [15].

Penelitian tentang pengujian perangkat lunak juga dilakukan terhadap aplikasi berbasis Android dengan metode *black box testing*. Adapun aplikasi yang diuji pada salah satu penelitian menggunakan teknologi *Phonegap*, suatu platform pengembangan teknologi berbasis aplikasi *web* untuk membuat *mobile native*. Pengujian yang dilakukan bertujuan untuk memastikan bahwa fungsionalitas sistem dan kebutuhan pengguna sudah berjalan dengan semestinya dan sesuai dengan literatur pendukung. Hasil yang diperoleh tidak terdapat kesalahan pada *interface* dan transaksi data [16]. Penelitian lainnya

menerapkan metode pengujian yang berbeda, yaitu dengan metode *white box testing* untuk menguji alur dari pengkodingan suatu perangkat lunak. Aplikasi berbasis Android diuji dengan teknik *basis path* untuk mengetahui kompleksitas logika dalam program. Pengujian membuahkan hasil dengan fungsi-fungsi yang sudah berjalan dengan semestinya, namun terdapat alur program dan logika yang belum optimal [17].

Pengujian perangkat lunak dapat dilakukan secara otomatis dengan *automation tool*. Salah satu penelitian membahas pengujian yang dilakukan dengan cara melakukan *record* dan *playback*. Adapun tujuan dari penelitian ini untuk menganalisis efektivitas *software* yang sudah dibangun dengan pengujian otomatis menggunakan Katalon Studio. Berdasarkan penelitian ini, dapat didapati bahwa secara garis besar Katalon Studio sudah memenuhi fungsinya sebagai *automation tool* yang bekerja dengan baik untuk melakukan pengujian otomatis meskipun masih terdapat beberapa kekurangan [18]. Dilanjutkan penelitian lain yang menguji aplikasi *platform mobile* menggunakan Katalon Studio, peneliti juga mendapati hasil yang memuaskan. Katalon Studio memiliki integrasi yang baik dengan Microsoft Excel untuk melakukan data binding yang digunakan selama pengujian. Selain itu, pengujian secara otomatis dapat dilakukan dengan waktu yang lebih efisien dibandingkan secara manual [19].

Pengujian perangkat lunak dengan metode *black box testing* memiliki teknik-teknik yang dapat diimplementasikan selama penelitian. Terdapat penelitian tentang pengujian perangkat lunak pada aplikasi *parking management* yang mana pengujian ini menggunakan metode *black box testing* dengan teknik *equivalence partitioning*. Teknik ini memecah masukan kedalam kelas-kelas data tertentu yang dapat menggambarkan masukan tersebut bernilai *valid* atau *invalid*. Pada rancangan *test case* mencakup *form* data kendaraan masuk dan *form* kendaraan keluar. Peneliti menyarankan untuk mempersiapkan penelitian dengan matang dan menggunakan teknik pengujian *black box testing* yang lain agar dapat menemukan lebih banyak variasi *error*, seperti *boundary value* atau *error guessing* sehingga hasilnya dapat sesuai ekspetasi. Dengan penemuan variasi *error* tersebut, diharapkan kualitas aplikasi dapat dikembangkan menjadi lebih baik [20].

Berbeda dengan penelitian berikutnya, pengujian pada sistem penilaian mahasiswa menggunakan teknik *boundary value analysis*. Teknik ini menerapkan pengujian terhadap nilai batas atas dan nilai batas bawah pada data yang akan menjadi masukan. Hasil yang didapatkan berdasarkan masukan dan keluaran aplikasi menggunakan teknik *boundary value analysis* sudah cukup baik. Namun, peneliti menyarankan untuk menggunakan teknik pengujian yang lain untuk memperoleh hasil yang lebih optimal [21].

2. Metode

Metode yang digunakan dalam penelitian yaitu studi literatur, analisis alur dan fungsi, perancangan *test case*, melakukan pengujian, dan merekap hasil pengujian. Studi literatur dilakukan untuk memperoleh sumber-sumber dalam penelitian yang dilakukan dari berbagai karya ilmiah, seperti buku, tugas akhir, internet, dan sumber yang memiliki kredibilitas lainnya. Analisis alur dan fungsi dilakukan untuk menemukan alur dari fungsi yang akan dilakukan pengujian yang dilakukan dengan mengakses aplikasi terlebih dahulu. Perancangan *test case* dilakukan untuk membuat kumpulan skenario pengujian berdasarkan analisis kebutuhan yang dihasilkan saat menemukan alur pada fungsi. Melakukan pengujian dilakukan secara langsung oleh manusia (manual) dan alat bantu khusus berupa *automation tools* (otomatis). Merekap hasil pengujian merupakan tahap menulis laporan berdasarkan hasil yang diperoleh. Pada Gambar 1 menjelaskan alur dari tahapan metode penelitian yang digunakan.

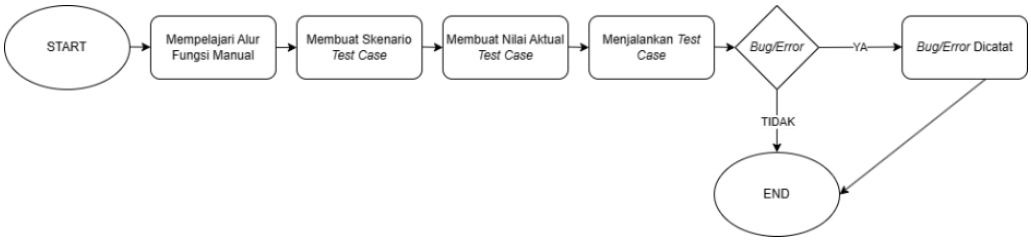
Gambar 1. Bagan Alur Metode Penelitian

3. Hasil dan Pembahasan

3.1. Deskripsi Pengujian

3.1.1. Analisis Alur Pengujian Manual

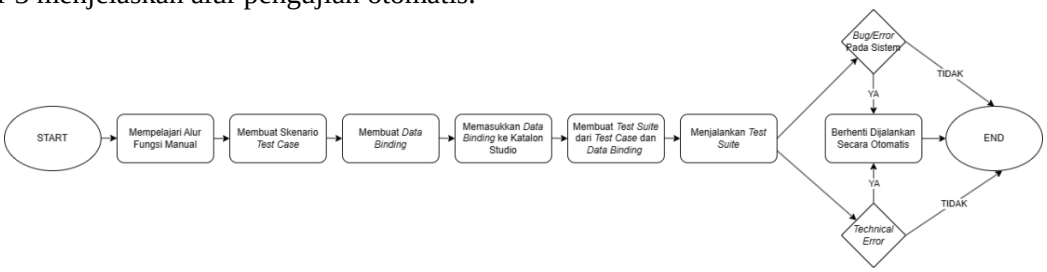
Analisis pengujian manual pada aplikasi TEA dilakukan secara langsung oleh manusia untuk merancang alur dan *test case* tanpa *tools* pendukung. Hasil keluaran dari pengujian secara manual akan dicatat dan dibandingkan dengan keluaran yang diharapkan sehingga akan didapatkan nilai aktual atau kesesuaian. Gambar 2 menjelaskan alur pengujian manual yang dilakukan.



Gambar 2. Bagan Alur Pengujian Manual

3.1.2. Analisis Alur Pengujian Otomatis

Analisis pengujian otomatis dilakukan dengan membuat *test case* berdasarkan alur yang sudah didapat saat akses aplikasi langsung. Alur dan *test case* yang telah diperoleh dirancang pada *automation tool* Katalon Studio dan membuat *data binding* yang berisi nilai aktual pada Microsoft Excel. *Data binding* tersebut kemudian dimasukkan ke dalam *test suite* untuk dilakukan pengujian. *Test suite* merupakan wadah yang digunakan sebagai penghubung antara data *binding* dan *test case* pada saat dijalankan. Gambar 3 menjelaskan alur pengujian otomatis.



Gambar 3. Bagan Alur Pengujian Otomatis

3.2. Hasil Pengujian

3.2.1. Hasil Pengujian Manual

Pada pengujian yang dilakukan secara manual diperoleh nilai kesesuaian antara keluaran (*output*) dengan keluaran yang diharapkan (*expected output*) berdasarkan *test case matrix* yang telah dibuat. Salah satu hasil dari pengujian manual yang dilakukan terhadap masing-masing *use case* adalah fungsi *TEA Management – Competency – Delete.* Tabel 1. merupakan nilai aktual untuk *Use Case* Fungsi *TEA Management – Competency – Delete* yang dirancang sesuai dengan matriks skenario. Pengujian dilakukan 3 kali dengan 3 variasi skenario yang berbeda untuk menemukan kemungkinan kesalahan yang dapat terjadi. Berdasarkan perbandingan antara *output* dengan *expected output* yang tercantum pada tabel, pengujian *Use Case* Fungsi *TEA Management – Competency – Delete* secara manual memiliki persentase kesesuaian fungsionalitas mencapai 100% dengan waktu uji 32 detik. Dengan kata lain, fungsionalitas telah berjalan sesuai dengan ekspetasi dan tidak ada kesalahan pada fungsi tersebut.

Tabel 1. *Actual Value* dari *Test Case*

Test Case ID	Skenario	*Competency Name*	*Delete Competency*	*Expected Output*	*Output*	Keterangan	Hasil Pengujian
TC1	S1: BF	outsystems	*OK*	Data berhasil dinonaktifkan.	Muncul pemberitahuan data berhasil dinonaktifkan dan status data menjadi *Inactive.*	Data berhasil dinonaktifkan.	Sesuai
TC2	S2: BF A1	english	N/A	Data tidak berhasil dinonaktifkan.	Data yang dicari tidak ditemukan pada *tab Active*.	Data tidak ada yang sama dengan masukan.	Sesuai
TC3	S4: BF A2	laravel	*Cancel*	Data tidak berhasil dinonaktifkan.	Data tidak berhasil dinonaktifkan.	*Actor* membatalkan penonaktifan data.	Sesuai

3.2.2. Hasil Pengujian Otomatis

Item	Object	Input
1 - Call Test Case	0. Start Browser	[:]
2 - Click	menu_TEA MANAGEMENT	
3 - Click	submenu_Competency	
4 - Set Text	input_SearchBar	varSearch
5 - Click	btn_Search	
6 - Switch Statement		varResult
6.1 - Case Statement		case "no":
6.1.1 - Verify Element Visible	alert_No items to show_Active	
6.1.2 - Break Statement		
6.2 - Case Statement		case "yes":
6.2.1 - Click	btn_DeleteCompetency	
6.2.2 - Switch Statement		varConfirm
6.2.2.1 - Case Statement		case "delete":
6.2.2.1.1 - Click	btn_DeleteCompetency_Delete	
6.2.2.1.2 - Click	tab_Inactive	
6.2.2.1.3 - Verify Element Visible	result_Inactive	
6.2.2.1.4 - Break Statement		
6.2.2.2 - Case Statement		case "cancel":
6.2.2.2.1 - Click	btn_DeleteCompetency_Cancel	
6.2.2.2.2 - Verify Element Visible	tab_Active	
6.2.2.2.3 - Break Statement		
6.2.3 - Break Statement		
7 - Delay		3

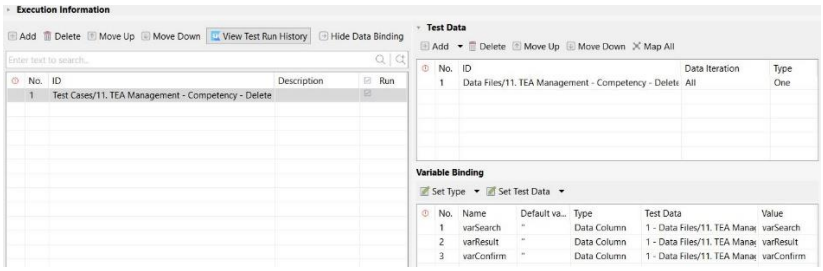
Gambar 4. *Test Case* Katalon Studio

No.	Name	Type	Default value
1	varSearch	String	""
2	varResult	String	""
3	varConfirm	String	""

Gambar 5. Variabel Katalon Studio

No.	varSearch	varResult	varConfirm
1	php module	yes	delete
2	mandarin	no	
3	katalon	yes	cancel

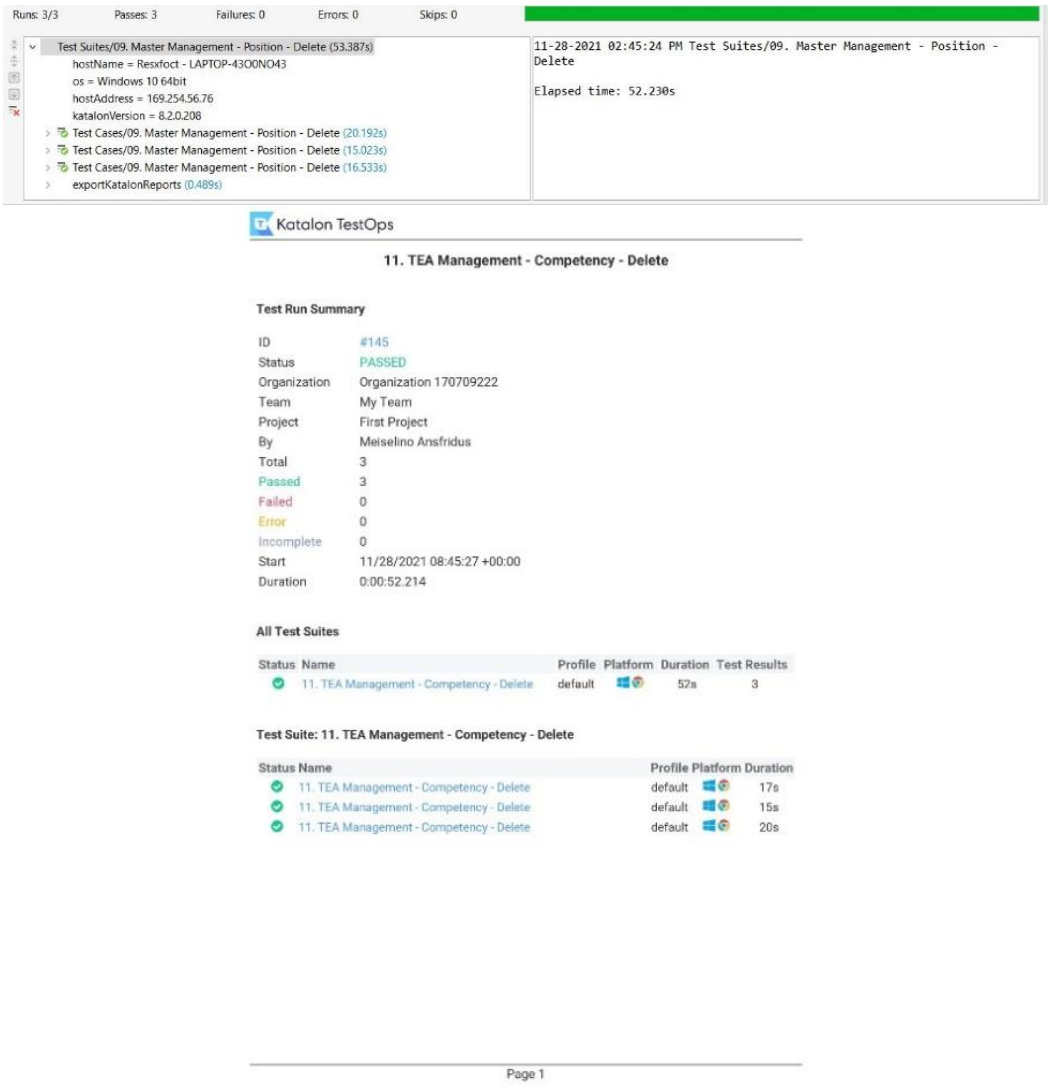
Gambar 6. *Data Binding* Katalon Studio



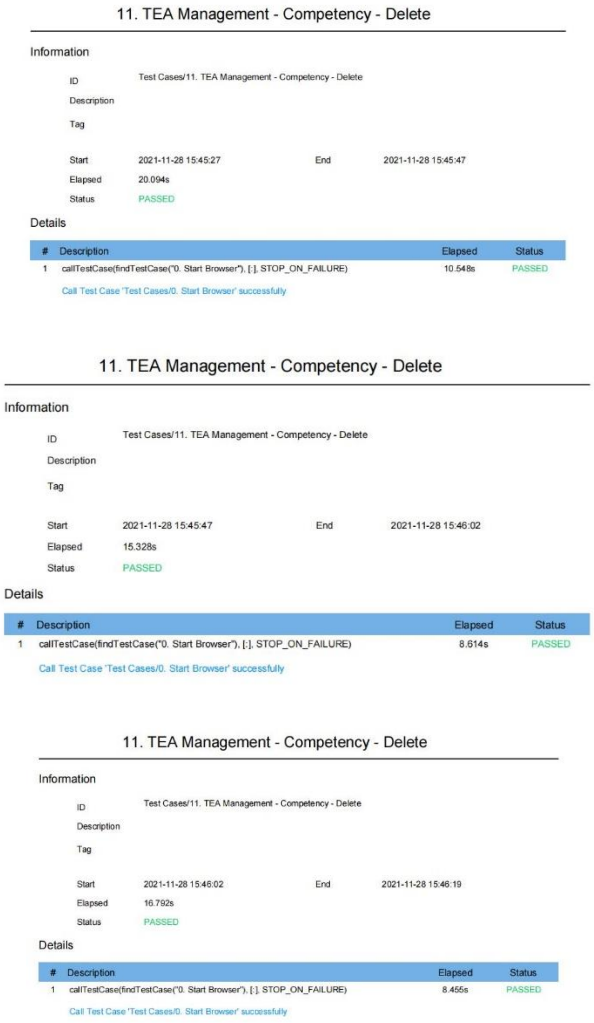
Gambar 7. *Test Suite* Katalon Studio

Gambar 4 dan 5 merupakan *test case* dan variabel yang digunakan untuk pengujian. Langkah awal saat menjalankan *test case* adalah membuka *browser*. Setelah *browser* terbuka, langkah berikutnya menjalankan *test case login* agar dapat mengakses sistem. Saat halaman utama sistem sudah terbuka, *test case* berjalan untuk memilih 'menu_TEA Management' dan 'submenu_Competency'. Pada tampilan utama *Competency*, *test case* berjalan untuk melakukan pencarian data dengan memasukkan nilai dari variabel 'varSearch' pada 'input_SearchBar' dan dieksekusi dengan menekan tombol 'btn_Search'. Hasil pencarian data akan dibandingkan dengan variabel 'varResult' yang menentukan data berhasil ditemukan atau tidak pada 'tab_Active'. Jika data berhasil ditemukan, *test case* berjalan untuk menekan tombol 'btn_DeleteCompetency' pada data tersebut. Setelah tombol ditekan, maka variabel 'varConfirm' berfungsi untuk mengonfirmasi penonaktifan data. Pada *case* 'delete' berarti mengeksekusi penonaktifan data yang berarti *positive flow*, sedangkan pada case 'cancel' berarti membatalkan penonaktifan data yang berarti *negative flow*.

Gambar 6 merupakan *data binding* yang berisi nilai aktual yang digunakan untuk menjalankan *test case* dan terdapat 3 *data binding* yang berarti akan dilakukan 3 kali pengujian secara berurutan. Data tersebut ditampung pada variabel-variabel yang telah diinisialisasi pada Gambar 5. Lalu pada Gambar 7 merupakan *test suite* sebagai penghubung antara *test case* dengan *data binding*.



Gambar 8. Laporan Katalon Studio terkait Hasil Pengujian Otomatis



Gambar 9. Durasi *Test Case* Katalon Studio saat *Start Browser*

Setelah *test suite* selesai dirancang, maka dilakukan proses eksekusi dengan menjalankan *test suite* tersebut. Hasil pengujian otomatis dapat terlihat pada Gambar 8 yang berisi laporan informasi total data dengan status *Passes* berjumlah 3 yang menandakan bahwa dari fungsional dapat 100% berjalan sesuai ekspetasi. Total waktu yang diperoleh adalah 52 detik seperti yang tercantum pada Gambar 9. Namun, total waktu tersebut akan dikurangi dengan durasi untuk menjalankan *test case* '*Start Browser*' pada Gambar 9 dengan waktu akumulatif 28 detik. Hasil pengurangan antara total waktu pengujian dengan waktu akumulatif untuk menjalankan *test case* '*Start Browser*' adalah 24 detik. Berdasarkan hasil yang diperoleh dari pengujian fungsi tersebut, dapat disimpulkan bahwa tidak ditemukan adanya kesalahan sehingga fungsionalitas telah berjalan sesuai ekspetasi.

3.3 Perbandingan Hasil Pengijian Manual dengan Otomatis

Berdasarkan hasil pengujian manual dan otomatis yang telah dilakukan, dapat diketahui bahwa fungsionalitas dari 20 fungsi dari menu *Master Management* dan *TEA Management* yang diuji pada sistem TEA diperoleh masing-masing 106 *passes dan* 0 *failures* dengan *application readiness* mencapai 100%. Dengan kata lain, pengujian otomatis dan manual memperoleh hasil fungsionalitas yang sama. Namun, terdapat perbedaan waktu pengujian yang didapatkan saat melakukan pengujian manual maupun otomatis. Hal ini terjadi saat melakukan pengujian otomatis menggunakan Katalon Studio, *test*

case akan menjalankan *Start Browser* dan *login* terlebih dahulu sebelum melakukan eksekusi pada fungsi. Akibatnya, total waktu pengujian yang dibutuhkan lebih banyak saat menjalankan *test case*.

Oleh karena itu, total waktu pengujian otomatis akan dikurangi dengan waktu akumulatif menjalankan *Start Browser* dan *login* pada setiap *test case*. Hal ini dilakukan untuk memperoleh waktu pengujian terhadap fungsi tanpa harus membuka *browser* dan *login* pada aplikasi terlebih dahulu. Sebab, pengujian manual tidak menghitung waktu yang dibutuhkan untuk membuka *browser* dan *login*. Waktu pengujian manual akan dibandingan dengan hasil pengurangan pengujian otomatis yang sudah diperoleh karena waktu pengujian tersebut memiliki kondisi yang setara. Perbandingan ini memperoleh persentase selisih dan rata-rata waktu pengujian manual dengan pengujian otomatis. Berikut merupakan rumus perhitungan persentase selisih waktu pengujian dan rumus perhitungan rata-rata persentase selisih waktu pengujian:

$$\text{Persentase Selisih Waktu} = \frac{\text{Waktu Pengujian Otomatis} - \text{Waktu Pengujian Manual}}{\text{Waktu Pengujian Otomatis}} \text{ x } 100\% \quad (1)$$

$$\text{Rata} - \text{Rata Persentase Selisih Waktu} = \frac{\text{Total Persentase Selisih Waktu}}{\text{Total Fungsi}} \quad (2)$$

Perhitungan perbandingan waktu berdasarkan persentase selisih dan rata-rata waktu yang dibutuhkan untuk melakukan pengujian manual dengan pengujian otomatis yang telah dilakukan tercantum pada Tabel 2.

Tabel 2. Perbandingan Waktu Pengujian Manual dengan Pengujian Otomatis

Fungsi	Waktu Pengujian Manual (Detik)	Waktu Pengujian Otomatis (Detik)	Selisih Waktu Pengujian	Persentase Selisih Waktu Pengujian (%)
Master Management – Project – Create	83	64	-19	-29,69
Master Management – Project – Edit	101	72	-29	-40,28
Master Management – Project – Delete	26	24	-2	-8,33
Master Management – Project – Restore	38	26	-12	-46,15
Master Management – Rank – Delete	31	26	-5	-19,23
Master Management – Rank – Restore	35	25	-10	-40,00
Master Management – Position – Create	76	63	-13	-20,63
Master Management – Position – Edit	91	72	-19	-26,39
Master Management – Position – Delete	30	25	-5	-20,00
Master Management – Position – Restore	35	25	-10	-40,00
TEA Management – Competency – Delete	32	24	-8	-33,33
TEA Management – Competency – Restore	38	25	-13	-52,00
TEA Management – Chapter – Create	128	76	-52	-68,42
TEA Management – Chapter – Edit	161	87	-74	-85,06
TEA Management – Chapter – Delete	38	25	-13	-52,00
TEA Management – Chapter – Restore	38	25	-13	-52,00
TEA Management – Final Test – Create	217	98	-119	-121,43
TEA Management – Final Test – Edit	304	120	-184	-153,33
TEA Management – Final Test – Delete	44	23	-21	-91,30
TEA Management – Final Test – Restore	45	25	-20	-80,00
Total	1591	950	-641	-64,47
Rata-Rata	79,55	47,50	-32,05	

Pada Tabel 2 memperlihatkan selisih total waktu pengujian manual dengan pengujian otomatis mencapai 641 detik. Selain itu, selisih rata-rata waktu pengujian manual dengan pengujian otomatis mencapai 32,05 detik. Hasil persentase selisih total dan rata-rata waktu pengujian adalah pengujian otomatis memangkas waktu 64,47% lebih cepat dibandingkan dengan pengujian manual. Hasil ini diperoleh dengan kondisi setara dengan tidak menyertakan durasi saat membuka *browser* dan *login* pada aplikasi.

Meskipun begitu, pengujian manual dan pengujian otomatis memiliki kelebihan dan kekurangan berdasarkan pengalaman dan hasil selama pengujian berlangsung. Berikut kelebihan dari pengujian manual: (1) Pengujian manual tidak perlu membuka dan menutup *browser* hingga *login* berulang-kali seperti pengujian otomatis dengan *automation tools*. (2) Pengujian manual tidak perlu melakukan banyak persiapan seperti membuat *script* dan aplikasi khusus untuk menjalankan pengujian. (3) Pengujian manual lebih mudah mengeksplorasi alur pengujian pada fungsi. (4) Pengujian manual tidak perlu memiliki keahlian khusus terkait pemrograman aplikasi yang akan diuji.

Selain kelebihan, pengujian manual juga memiliki kekurangan sebagai berikut: (1) Pengujian manual cenderung kurang konsisten dengan waktu pengujian sehingga dapat memperoleh hasil pengujian yang kurang stabil. (2) Pengujian manual perlu melakukan pencatatan terhadap hasil dan waktu pengujian secara manual. (3) Pengujian manual yang menjalankan *data binding* lebih banyak cenderung memakan waktu lebih lama. (4) Pengujian manual dapat merasa lelah dan bosan karena melakukan pengujian terus-menerus sehingga potensi terjadinya *human error* meningkat akibat kesalahan pengetikan maupun penurunan fokus.

Setelah membahas pengujian manual, berikut kelebihan dari pengujian otomatis menggunakan Katalon Studio dalam penelitian yang telah dilakukan, yaitu: (1) Pengujian otomatis akan mendapatkan laporan hasil dan waktu pengujian secara otomatis dari Katalon Studio. (2) Pengujian otomatis dapat melakukan pengujian yang lebih terstruktur dengan adanya *test case*. (3) Pengujian otomatis mengurangi tingkat terjadinya *human error*. (4) Pengujian otomatis dapat melakukan perubahan pada *test case* yang sudah dirancang untuk menyesuaikan perubahan pada fungsi yang diuji.

Namun, pengujian otomatis menggunakan Katalon Studio memiliki kekurangan sebagai berikut: (1) Pengujian otomatis membutuhkan waktu lebih lama karena menjalankan *case* berulang-kali seperti membuka dan menutup *browser* hingga *login*. (2) Pengujian otomatis membutuhkan waktu lebih lama untuk mendeteksi perangkat lunak dan objek uji sehingga terdapat adanya delay. (3) Pengujian otomatis membutuhkan koneksi internet yang stabil untuk memuat dan mendeteksi objek yang terdapat pada aplikasi. (4) Pengujian otomatis tidak dapat mendeteksi semua objek pada aplikasi sehingga harus dilakukan secara manual untuk mencari koordinat dari objek uji.

4. Kesimpulan

Penelitian yang dilakukan memperoleh hasil dari fungsional aplikasi *Techno Expertise Academy* (TEA) telah berjalan sesuai ekspetasi. Hasil dari pengujian manual dan otomatis yang menghasilkan kesiapan aplikasi dengan *score* 100% pada setiap fungsi. Hal ini juga memenuhi standar kualitas dari perusahaan Astra Credit Companies, dibuktikan dengan seluruh keluaran bersifat '*pass*' dan pengujian User Acceptance Test telah diterima. Selain itu, pengujian manual dan otomatis pada total 20 fungsi memperoleh hasil 106 *passes* dan 0 *failures*. Dengan kata lain, pengujian manual dan otomatis dapat memperoleh hasil yang sama. Namun, pengujian otomatis dengan *automation tools* Katalon Studio efisiensi durasi 64,47% dibandingkan pengujian manual saat kondisi setara.

Referensi

[1] N. Edwin Kiky Aprianto, "Peran Teknologi Informasi dan Komunikasi dalam Bisnis," *Int. J. Adm. Bus. Organ.* |, vol. 2, no. 1, pp. 1–7, 2021, [Online]. Available: https://ijabo.a3i.or.id.

[2] Abdul Kadir, "Peranan brainware dalam sistem informasi manajemen jurnal ekonomi dan manajemen sistem informasi," *Sist. Inf.*, vol. 1, no. September, pp. 60–69, 2018, doi: 10.31933/JEMSI.

[3] A. D. Galang, I. V. Ancho, U. Hindu, N. I. Gusti, and B. Sugriwa, "Jurnal Penjaminan Mutu Corpus Thematic Analysis : Trends in Quality and Quality Assurance Research," ... *Penjaminan Mutu*, vol. 6, pp. 124–133, 2020, [Online]. Available:

https://www.ejournal.ihdn.ac.id/index.php/JPM/article/view/1596.
[4] L. Marwick, “Cost of Software Errors,” *raygun.com*, 2023. https://raygun.com/blog/cost-of-software-errors/ (accessed Apr. 21, 2024).
[5] Berijalan, “Techno Center,” *berijalan.co.id*, 2021. https://berijalan.co.id/tab-techno (accessed Apr. 21, 2024).
[6] Astra, “ASTRA CREDIT COMPANIES,” *www.astra.co.id*. https://www.astra.co.id/Business/Financial-Services/Car-Financing/ASTRA-CREDIT-COMPANY (accessed Apr. 21, 2024).
[7] A. Garg, R. Kumar Kaliyar, and A. Goswami, “PDRSD-A systematic review on plan-driven SDLC models for software development,” *8th Int. Conf. Adv. Comput. Commun. Syst. ICACCS 2022*, vol. 1, pp. 739–744, 2022, doi: 10.1109/ICACCS54159.2022.9785261.
[8] J. Mcdowell and Y. Ki Kwong, “Managing Accessibility in Software Systems,” *PNSQC.ORG*, pp. 1–10, 2021.
[9] Hozairi, Buhari, S. Alim, and Rofiudin, *PANDUAN KOMPREHENSIF PENGUJIAN PERANGKAT LUNAK*. 2024.
[10] C. Henard, M. Papadakis, M. Harman, Y. Jia, and Y. Le Traon, “Comparing white-box and black-box test prioritization,” *Proc. - Int. Conf. Softw. Eng.*, vol. 14-22-May-, pp. 523–534, 2016, doi: 10.1145/2884781.2884791.
[11] G. Ken, S. Tresnavitane, P. Mudjihartono, and Y. Harjoseputro, “Pengujian Aplikasi Mobile untuk Lelang Mobil dengan Metode Black Box menggunakan Automation Testing Tool,” pp. 79–87, 2019.
[12] M. Sholeh, I. Gisfas, Cahiman, and 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, 2021, doi: 10.1088/1742-6596/1823/1/012029.
[13] M. A. Umar and C. Zhanfang, “A Study of Automated Software Testing : Automation Tools and Frameworks,” *Int. J. Comput. Sci. Eng.*, vol. 8, no. 06, pp. 217–225, 2019.
[14] A. Purnomo, “Software Testing Aplikasi Website PT Gramedia Menggunakan Metode Blackbox pada PT WGS Bandung,” *E-Journal Univ. Dianapura*, vol. 91, pp. 399–404, 2017.
[15] U. Salamah and F. Khasanah, “Pengujian Sistem Informasi Penjualan Undangan Pernikahan Online Berbasis Web Menggunakan Black Box Testing,” *Inf. Manag. Educ. Prof.*, vol. 2, no. 1, pp. 35–46, 2017.
[16] W. N. Cholifah, Y. Yulianingsih, and S. M. Sagita, “Pengujian Black Box Testing pada Aplikasi Action & Strategy Berbasis Android dengan Teknologi Phonegap,” *STRING (Satuan Tulisan Ris. dan Inov. Teknol.*, vol. 3, no. 2, p. 206, 2018, doi: 10.30998/string.v3i2.3048.
[17] C. T. Pratala, E. M. Asyer, I. Prayudi, and A. Saifudin, “Pengujian White Box pada Aplikasi Cash Flow Berbasis Android Menggunakan Teknik Basis Path,” *J. Inform. Univ. Pamulang*, vol. 5, no. 2, p. 111, 2020, doi: 10.32493/informatika.v5i2.4713.
[18] H. Herlinda, D. Katarina, and E. W. Ambarsari, M.Kom., “Automation Testing Tool dalam Pengujian Aplikasi Belajar Tajwid pada Platform Android,” *STRING (Satuan Tulisan Ris. dan Inov. Teknol.*, vol. 4, no. 2, p. 205, 2019, doi: 10.30998/string.v4i2.5285.
[19] F. Ardi and H. P. Putro, “Pengujian Black Box Aplikasi Mobile Menggunakan Katalon Studio (Studi Kasus: ACC Partner PT. Astra Sedaya Finance),” *Automata*, vol. 2, no. 1, 2021.
[20] F. A. Fauzi, G. E. Putra, S. Supriyanto, N. A. Saputra, and T. Desyani, “Pengujian Terhadap Aplikasi Parking Management Menggunakan Metode Black-Box Berbasis Equivalence Partitions,” *J. Teknol. Sist. Inf. dan Apl.*, vol. 3, no. 2, p. 64, 2020, doi: 10.32493/jtsi.v3i2.4685.
[21] D. Debiyanti, S. Sutrisna, B. Budrio, A. K. Kamal, and Y. Yulianti, “Pengujian Black Box pada Perangkat Lunak Sistem Penilaian Mahasiswa Menggunakan Teknik Boundary Value Analysis,” *J. Inform. Univ. Pamulang*, vol. 5, no. 2, p. 162, 2020, doi: 10.32493/informatika.v5i2.5446.