p-ISSN 2477-3506
e-ISSN 2549-1938

EVALUASI PENGGUNAAN *MANUAL* DAN *AUTOMATED SOFTWARE TESTING* PADA PELAKSANAAN *END-TO-END TESTING*

Joe Lian Min[1], Adila Istiqomah[2], Ani Rahmani[3]

[1,3] Politeknik Negeri Bandung
[2]Profesional Tester PT Padepokan Tujuh Sembilan-Bandung
Email: [1]joelianmin@jtk.polban.ac.id, [2]aadilaistiqomah@gmail.com, [3]anirahma@jtk.polban.ac.id

Abstrak

Abstrak-- Pada siklus pengembangan *software, testing* diperlukan untuk memastikan kualitas *software* sebelum dirilis. Pemilihan teknik testing yang digunakan dalam pelaksanaan *testing* bergantung pada karakteristik *software* yang akan dirilis. Aplikasi berbasis web sebagai contoh, akan cenderung menggunakan teknik *end-to-end* untuk memastikan halaman web berinteraksi sesuai dengan *requirement*. Berkaitan dengan masa *development cycle* dari sebuah *software* yang semakin singkat, maka dipandang perlu kajian untuk menentukan teknik pengerjaan *testing* (secara manual maupun *automated testing*) pada *end-to-end testing*. Penelitian yang dilakukan bertujuan untuk melihat karakteristik dan teknis pengujian end-to-end testing yang mungkin dapat dilakukan dalam pengembangan perangkat lunak, baik secara manual atau pun secara otomatis. Dari kajian yang dilakukan, diketahui bahwa jumlah iterasi pelaksanaan *end-to-end testing* sangat berpengaruh dalam pemilihan teknik pengerjaan yang digunakan. Hal ini dapat menjadi dasar untuk menentukan teknik *testing* yang akan diambil: manual atau otomatis.

Kata Kunci: *end-to-end testing, manual testing, automated testing, protactor*

Abstract

In software development life cycle, testing is needed to ensure the quality of the software before it is released. Choosing the testing technique depends on the characteristics of the software to be released. Web-based applications, for example, will tend to use end-to-end testing techniques to ensure web pages interact according to requirements. The cycle of software development recently is getting shorter. It is necessary to study to determine the testing implementation techniques (manually or automated) on the right end-to-end testing. The research objective is to look at the characteristics and technical testing of end-to-end testing that can be done in software development, both manually and automatically. The result showed that the number of iterations of end-to-end testing is very influential in the selection of quality techniques used. That can be a basis for determining testing techniques: manual or automatic.

Keywords: *end-to-end testing, manual testing, automated testing, protactor*

I. PENDAHULUAN

Software testing (biasa disingkat *testing*) merupakan aktifitas penting dalam *software engineering*. Testing akan mengekseksi *software* dengan tujuan untuk melihat kesesuaiannya dengan *requirement* yang didefinisikan (Bertolino dan Faedo, 2007:1). Secara sederhana aktivitas testing akan memproses dan mengeksekusi program dengan maksud menemukan *error*. *Software testing* akan melakukan serangkaian proses yang telah dirancang untuk memeriksa hal-hal yang harus dilakukan dan (sebaliknya) tidak boleh terjadi pada sebuah *software* (Mirza dan Khan, 2018: 46568; Myers, G.J 2011:2).

Ada berbagai macam teknik *testing* yang dapat digunakan untuk melakukan pengujian pada sebuah *software*. Salah satunya yaitu *end-to-end* testing yang biasa digunakan oleh perusahaan yang bergerak dalam pengembang aplikasi web (Manova

n.d, 2018:267).

End-to-end testing merupakan teknik testing yang memiliki cakupan besar dan dimungkinkan menghabis-kan banyak waktu dalam pelaksaannya, sementara kini masa *development cycle* dari sebuah *software* semakin singkat. Menyiasati waktu yang singkat tersebut, maka pemilihan teknik eksekusi testing (dengan manual atau menggunakan *automated software testing*) yang tepat harus terus diupayakan.

Paper ini menjelaskan hasil pengamatan pada pelaksanaan *end-to-end testing* yang dilaksanakan secara manual vs otomatis pada aplikasi web sederhana. Tujuan pengamatan adalah untuk melihat teknik terbaik dari kedua teknik tersebut pada pelaksanaan *end-to-end testing*. Pengetahuan mengenai teknik terbaik untuk proses pengujian sebuah *software* diperlukan agar pelaksanaan *testing* dapat efisien.

II. METODE

Studi yang dilakukan untuk memperoleh pengetahuan mengenai teknik terbaik pelaksanaan testing, dirancang dalam beberapa kegiatan.

2.1 Penentuan Responden / Tester

Pengambilan sampel *tester* menggunakan *random selection*, dengan teknik sampling non probabilitas yaitu Convenience sampling. Convenience sampling (biasa dikenal juga sebagai Haphazard sampling atau Accidental sampling) adalah salah satu tipe dari sampling non probabilitas atau non random dengan anggota populasi target yang memenuhi kriteria praktis tertentu, seperti aksebilitas mudah, kedekatan geografis, ketersedian pada waktu tertentu, atau kesedian untuk berpartisipasi dimasukkan untuk tujuan penelitian (Etikan, 2015: 2).

Berdasarkan hal tersebut, populasi *tester* yang terlibat hanya *tester* yang sedang atau telah menyelesaikan jenjang pendidikan pada program studi teknik informatika di Politeknik Negeri Bandung dengan batasan rentang waktu angkatan masuk 2012-2015. Tester yang terlibat berjumlah 16 orang, dibagi ke dalam dua kelompok, dengan masing-masing 8 orang. Kelompok pertama melaksanakan *end-to-end testing* secara manual, dan kelompok kedua melaksanakan *end-to-end testing* secara otomatis.

Dari sisi jumlah tester, pada penelitian yang dilakukan kurang menjadi variable penentu. Dalam konteks ini, tester dilibatkan untuk teknis pelaksanaan dalam upaya melihat karakteristik pengujian yang sedang diamati, serta metode yang dijalankan baik dari sisi rancangan maupun teknikal.

Sebelum eksperimen, diadakan sebuah tranining singkat, dan sebelumnya *tester* juga diberikan dokumen tatacara pelaksanaan end-to-end testing. Training singkat bertujuan untuk menyetarakan kemampuan *tester* (pada penelitian ini tester dianggap memiliki kemampuan setara setelah pelaksanaan training) dalam pelaksanaan *end-to-end testing* untuk eksperimen.

Kemampuan *tester* yang dianggap setara setelah diberikan training untuk pelaksanaan *end-to-end testing* adalah sebagai berikut:

a. Melakukan *end-to-end* testing baik secara manual maupun *automated* sesuai dengan prosedur yang telah ditetapkan.
b. Memahami cara mendokumentasikan hasil *end-to-end testing* yang dilakukan apabila dikerjakan secara manual.
c. Memahami bagaimana menggunakan *automation framework* yang digunakan apabila melakukan end-to-end testing secara *automated*.

3.2 Eksperimen

Persiapan *environment* termasuk installasi software under test (SUT) dilakukan pertama kali sebelum eksperimen. Secara paralel, kuisioner yang mengandung beberapa pertanyaan dibagikan kepada *tester*. Pertanyaan yang diajukan adalah untuk menggali pengalaman dan pembelajaran yang pernah didapatkan responden dalam hal pelaksanaan *software testing*. *Test session* survey yang dibuat mengacu pada (Itnoken, J, 2008:61) dengan beberapa penyesuaian. Jawaban dari responden, selanjutnya dijadikan dasar untuk penentuan kelompok *tester*.

Tools *automated software testing* yang digunakan adalah Protractor. Kelompok yang melaksanakan *end-to-end testing* dengan Protractor dikhususkan untuk *tester* yang telah bekerja atau yang memiliki pengalaman dalam bidang-bidang yang berkaitan dengan pemanfaatan beberapa teknologi yang dipandang "dekat" dengan Protactor (seperti angular JS atau Javascript). Informasi tersebut diperoleh dari hasil survey sebelum pelaksanaan eksperimen.

Eksperimen dilakukan setelah diberikan training mengenai pelaksanaan *end-to-end testing* secara manual dan dipersilakan melakukan uji coba pada objek penelitian (dipersiapkan satu screen). Sedangkan untuk kelompok yang melaksanakan *end-to-end testing* dengan Protractor diberikan training mengenai cara pelaksanaan end-to-end

testing dengan penulisan *test script* menggunakan Protractor, dan dipersilakan untuk melakukan uji coba terlebih dahulu.

Dokumentasi tata cara pelaksanaan *end-to-end testing* secara manual maupun menggunakan Protractor telah di distribusikan terlebih dahulu sebelum hari pelaksanaan eksperimen. Dokumentasi tersebut dilengkapi juga dengan cara menjalankan SUT, panduan installasi alat eskperimen, contoh *test report* hasil uji dengan manual testing, contoh test script yang dikerjakan dengan Protractor, dan alamat referensi pendukung dari Protractor.

Pelaksanaan eksperimen kelompok manual adalah: *tester* mencatat waktu awal mulai mengeksekusi pada setiap test item, melakukan testing pada SUT1 dengan mengikuti langkah-langkah dan test data (input) untuk setiap test item yang terdapat pada *test case.* Selanjutnya membandingkan hasil eksekusi SUT dengan *actual output* yang diharapkan pada *test case.* Test report yang dibuat diharuskan menyertakan tangkapan layar hasil eksekusi setiap test item. Setiap selesai mengeksekusi satu test item, *tester* diharuskan mencatat waktu selesai.

Prosedur pelaksanaan eksperimen kelompok *automated* dimulai dengan *tester* membuat *test script* dengan acuan *test cases.* Ketika memulai transformasi test item ke dalam bentuk *script, tester* diharuskan mencatat waktu mulai. Apabila *test script* telah selesai, selanjutnya dijalankan di Protractor, dan hasil eksekusi diperiksa dari *generate report* yang tersedia, dan terakhir mencatat waktu selesai.

Tester yang tergabung pada kelompok *automated* dibolehkan melakukan manual testing pada test item yang dipandang tidak dapat diautomasi dengan prosedur yang disamakan dengan eksperimen kelompok manual. Hal tersebut dilakukan, karena tidak semua item test dapat dijalankan secara automated.

Setelah tester di kedua kelompok selesai melaksanakan pengujian pada SUT1, selanjutnya dilakukan satu kali lagi eksperimen menggunakan SUT2 (*re-testing*). Pelaksanaan end-to-end testing kedua kelompok pada *re-testing* sama dengan pada pelaksanaan testing sebelumnya.

2.2 End-to-end testing

End-to-end testing adalah sebuah metodologi yang digunakan untuk menguji apakah flow aplikasi bekerja sebagaimana yang dirancang dari awal hingga selesai. End-to-end testing merupakan salah satu teknik testing yang harus dilakukan oleh perusahaan pengembang web, karena jika telah melewati end-to-end testing, secara umum dipandang telah menjamin interaksi *user* dengan halaman web, dan sudah sesuai dengan kebutuhan *user* (Palmér, T, H, dkk, 2015: 1). Pelaksanaan end-to-end testing terbagi menjadi dua metode, yaitu *horizontal end-to-end testin*g dan *vertical end –to-end testing*.

Horizontal end-to-end testing merupakan metode yang paling banyak digunakan. Jika akan melakukan *end-to-end testing* dengan metode horizontal, misalnya pada *web* penjualan buku, maka semua proses yang mencakup pengisian data pembeli, detail pembelian, termasuk detail pembelian yang dilakukan pembeli harus dilakukan testing dari awal sampai akhir.

Vertical end-to-end testing merupakan metode yang sangat kompleks sehingga jarang dilakukan, karena *end-to-end testing* dengan metode ini melakukan testing termasuk *application programming interface* (API) dan *structured query language* (SQL).

Pelaksanaan *end-to-end testing* dapat menggunakan salah satu metode ataupun keduanya untuk kebutuhan testing scenario yang kompleks. Pada pelaksanaannya, *end-to-end testing* merupakan salah satu teknik testing yang bisa dieksekusi dengan *manual testing* ataupun dengan *automated software testing.*

2.2.1 *Manual Testing*

Manual testing adalah sebuah teknik testing dimana tester menyiapkan *test cases* secara manual dan mengeksekusi *test cases* untuk mengidentifikasi *defect* di *software* (Sharma, 2014: 252; Dobles, I dkk, 2019: 7)

Secara umum penggunaan *manual testing* pada pelaksanaan *end-to-end testing* adalah melakukan eksekusi *test* dengan menjalankan *software* sesuai dengan skenario yang tertulis pada *test cases.* Selanjutnya membandingkan *output* yang keluar dari aplikasi dengan *output* yang diharapkan dari setiap *test cases.*

2.2.2 *Automated Software Testing*

Automated software testing melibatkan pengem- bangan test script menggunakan scripting languages seperti phyton, java script, atau tool command language (TCL), sehingga test case dapat dieksekusi oleh komputer dengan minimal campur tangan manusia. (Sharma,R,M, 2014: 252)**.** Pada (Dudekula, 2011: 7) juga dijelaskan bahwa *automated software testing* adalah proses membuat sebuah program (*test* script) yang mensimulasikan langkah-langkah *test case* manual dalam bahasa pemrograman apapun dengan bantuan *external automation helper tool* lainnya. Eksekusi *end-to-end testing* menggunakan *automated software*

testing membutuhkan konversi *test case* menjadi *test script*, yang setelah itu dilakukan *running test script* pada *automation tools* tersebut.

2.3 Protractor

Pemilihan *automation framework* yang digunakan bergantung pada teknik testing yang akan diterapkan untuk pengujian *system under test*. Pemilihan Protractor sebagai automation framework, karena teknik testing pada penelitian yang dilakukan adalah end-to-end testing. Selain itu juga dilatarbelakangi oleh object penelitian yang dibangun menggunakan angular js pada front-end, dimana angular js untuk pelaksanaan end-to-end testing telah menyediakan Protractor sebagai *automation framework*.

Protractor adalah sebuah *open source end-to-end test framework* untuk aplikasi AngularJS (Bustamante 2017: 1). Protractor dibangun di atas WebDriverJs, yaitu sebuah implementasi resmi Javascript Selenium. Protractor dapat dilihat sebagai sebuah *tool* yang berinteraksi dengan *website* seperti yang biasa dilakukan oleh *user*. Protractor sebagai *test-runner* masih menggunakan sebuah *test-framework*, yaitu Jasmine, Mocha dan Cucumber. Pada penelitian ini, *test-framework* yang digunakan adalah Jasmine.

Pemilihan Jasmine selain diperkuat karena Jasmine merupakan *test-framework default* ketika installasi Protractor, juga karena menggunakan Cucumber. Versi Protractor yang digunakan adalah protractor versi 5.1.1. Protactor pada versi tersebut tidak menyertakan Cucumber sebagai *test-framework* secara *default* yang dapat digunakan pada Protractor. Apabila menggunakan Mocha maka Chai harus disertakan sebagai *framework* tambahan. Hal ini karena pada Mocha, fungsi expect tidak diadaptasi untuk mengerti *promise*, sedangkan Protractor dibangun di atas WebDriverJs, dan WebDriverJS berjalan secara *asynchronous*, sehingga keseluruhan fungsi akan memiliki kembalian *promise*.

Selain menggunakan Protractor, untuk memudah-kan pembacaan hasil *running test script*, pada *console* digunakan juga Jasmine-spec-reporter. Jasmine-spec-reporter adalah sebuah *package* npm yang berupa *real time console spec reporter* untuk Jasmine. Selain Jasmine-spec-reporter juga digunakan Protractor-jasmine2-html-reporter untuk kebutuhan meng-*generate* laporan hasil ekseskusi *test* (telah di lengkapi tangkapan layar) pada eksperimen *automated.*

2.4 Software under Test (SUT)

Software yang digunakan untuk pengujian (*software under test* / SUT) adalah sebuah web online shop sederhana untuk penjualan kue.

SUT yang dibangun diasumsikan telah melalui level testing sebelum *end-to-end testing* (unit testing, dan integrasi testing). Terdapat tiga SUT yang disiapkan, yaitu SUT training, dan SUT untuk eksperimen yang dinamakan sebagai SUT1 dan SUT2.

SUT training digunakan untuk kebutuhan pelaksanaan training sebelum pelaksanaan eksperimen end-to-end *testing*. SUT1 merupakan SUT yang digunakan pada eksperimen end-to-end testing iterasi pertama (SUT1 dikondisikan sebagai SUT sebelum proses *bug fixing*). SUT 2 adalah SUT yang digunakan untuk eksperimen pelaksanaan *re-end-to-end testing* (SUT 2 di kondisikan sebagai SUT setelah bug fixing).

Pada SUT1 terdapat *bugs* yang ditanam secara sengaja sebanyak 20 buah. Bugs tersebut terdiri dari kesalahan penulisan (*typo*), kesalahan perpindahan halaman, fitur yang tidak berfungsi (contoh fungsi searching tidak berfungsi), dan kesalahan perhitungan.

End-to-end testing difokuskan pada simulasi *actual user* (skenario test atau *end-to-end flow* aplikasi per screen) tanpa pemeriksaan basis data, maka SUT hanya dibangun menggunakan fake back-end untuk pemrosesan data yaitu hanya dengan Javascript, sedangkan pada bagian *front-end* memanfaatkan Angular Js.

2.5 Test Case

Penelitian dilaksanakan untuk melihat waktu pengerjaan *end-to-end testing* secara manual mengguna-kan *automation framework.* Selain mengetahui waktu pengerjaan, juga luaran dari penelitian ini dapat merepresentasikan faktor-faktor yang berpengaruh dalam pelaksanaan *end-to-end testing.* Karena penelitian berfokus pada *test exexution,* maka perancangan *test cases* menjadi proses terpenting.

Software Testing Life Cycle (STLC) didefinisikan sebagai rangkaian kegiatan yang sistematis dalam testing. Menurut (Afzal, 2007:9), banyak sumber memiliki pendekatan berbeda mengenai aktivitas atau phase yang harus dilewati pada STLC. Pada penelitian ini phase yang diikuti adalah phase utama STLC, yaitu terdiri dari *test planning, test design, test execution*, dan *test review*.

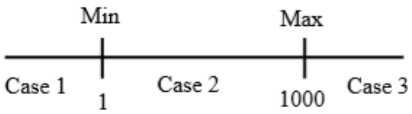
Dokumen test cases yang digunakan untuk eksperimen ini telah melalui proses test planning dan design. hHasil dari test planning adalah dokumen test plan yang disusun berdasarkan acuan

IEEE Std 829-1998 untuk Software Test Documentation, dan dokumen *test design* yang disusun berdasarkan sumber yang sama dengan *test plan*.

Pada pengembangan *test cases* untuk pelaksanaan *end-to-end testing* tidak ada *template* khusus (Palmér, T, H, dkk, 2015: 1). Pada penelitian yang dilakukan *test case* yang dikembangkan menerapkan teknik *equivalence partitioning, nature of the application (web-based software). Equivalance Partitioning* melibatkan pembagian domain input ke dalam koleksi subset (atau *equivalence classes*) berdasarkan pada kriteria atau relasi yang ditentukan (19759:2005ISO, 2005: 4-8). Contoh penerapan pada *test cases* misalnya *input qty* kue yang valid adalah angka 1 – 1000, maka ada tiga *case* yang terbentuk (lihat representasi Gambar 2).

a. *Case* 1 digunakan untuk testing *invalid* input, dimana *test data* dari nilai input *qty* berupa nilai yang lebih kecil dari nilai minimum input yang valid (lebih kecil dari 1).
b. *Case* 2 digunakan untuk testing *valid* input, dimana *test data* dari nilai input *qty* berupa nilai yang memiliki *range* sama atau diantara nilai minimum dan nilai maksimum dari input yang valid (contoh 10).
c. *Case* 3 digunakan untuk testing *invalid* input, dimana *test data* dari nilai input *qty* berupa nilai yang lebih besar dari nilai maskimum input yang valid (lebih kecil dari 1000).

Techniques based on the nature of the application yang diterapkan adalah *nature* dari *web-based software*. Pengembangan *test case* yang berdasasrkan teknik ini dan masih berkaitan dengan *end-to-end testing* diantaranya pengecekan *field* label, fungsional tombol pada *page,* dan pengecekan scenario (termasuk pengecekan data yang ditampilkan, dan hasil perhitungan).



Gambar 1. Contoh *Equivalence Partitioning*

Total *test case* yang terbentuk berdasarkan SUT yang disediakan dan dari penerapan teknik yang telah dipaparkan adalah lima buah *test cases,* dengan salah satu *test case* digunakan untuk keperluan *training.* Jumlah *test item* pada masing-masing *test cases* ditunjukkan pada Tabel 1.

Tabel 1. Jumlah *Test Item*

Test Cases Code	Jumlah Test Item
TC 1	40
TC 2	51
TC 3	37
TC 4	44
TC 5	27

III. HASIL DAN PEMBAHASAN

Hasil eksperimen dan pembahasan untuk mengetahui perbandingan lama pengerjaan *end-to-end testing* dengan manual dan dengan *automated software testing* dijelaskan sebagai berikut.

3.1 Hasil Eksperimen

Tabel 2 merupakan rekapitulasi dari lembar survei yang disebar pada *tester* yang terlibat pada penelitian. Dari lembar survei ini dapat diketahui lebih dalam latar belakang pengalaman responden terhadap hal-hal yang berkaitan dengan beberapa aspek yang dibutuhkan dari eksperimen.

Pada table 3 dan table 4 secara berurut merupakan data lama pengerjaan eksperimen secara manual dan *automated software testing.*

Pada prosedur pelaksanaan end-to-end testing secara otomatis, *tester* dibolehkan melakukan manual testing jika menurut tester, test item yang terdapat pada test case tidak dapat diotomasi. Mengenai hal tersebut dinyatakan bahwa memilih penggunaan automated software testing pada pelaksanaan testing tidak sepenuhnya menghilang-kan manual testing, karena ada beberapa hal pada test cases yang tidak dapat dioutomasi, (Sharma, 2014: 253), sehingga pada *test metric* dapat ditentukan berapa banyak *test case* yang dapat di-automated. Test metric tersebut adalah percent automable (persamaan 1).

$$\text{Percent automable} = \frac{\text{no. of test items automable}}{\text{no.of total test items}} \text{x100\%} \quad (1)$$

3.2 Pembahasan

Berdasarkan tabel 3 dan 4 dapat diketahui bahwa:

a. Pengerjaan *end-to-end testing* lebih cepat pada iterasi pertama kali apabila dikerjakan secara manual.
b. Pada saat *re-testing*, *end-to-end testing* akan lebih cepat apabila dikerjakan secara *automated.*

Dari kesimpulan yang diperoleh dianalisis lebih lanjut bahwa ada beberapa hal yang harus dipertim-bangkan untuk memutuskan pelaksanaan *end-to-end testing* secara manual ataupun otomatis.

Tabel 2. Rekapitulasi Data Lembar Survei

Tester	Matakuliah *Software Testing*	*Training Software*	Javascript	AngularJS	Protractor	Selenium	Keterangan
T1	Y	T	Y	T	T	T	Belum bekerja
T2	Y	T	Y	T	T	T	Bekerja
T3	T	T	T	T	T	T	Belum bekerja
T4	Y	T	Y	T	T	T	Bekerja
T5	T	T	T	T	T	T	Belum bekerja
T6	T	T	T	T	T	T	Belum bekerja
T7	T	T	T	T	T	T	Belum bekerja
T8	Y	T	Y	T	T	T	Bekerja
T9	T	T	T	T	T	T	Belum bekerja
T10	T	T	Y	T	T	T	Belum bekerja
T11	Y	T	Y	Y	T	T	Belum bekerja
T12	T	T	T	T	T	T	Belum bekerja
T13	Y	T	Y	T	T	T	Bekerja
T14	T	T	T	T	T	T	Belum bekerja
T15	T	T	Y	Y	T	T	Bekerja
T16	T	T	T	T	T	T	Belum bekerja

Tabel 3. Rekapitulasi Lama Pengerjaan - Eksperimen Manual

Test case	Tester	Lama pengerjaan (menit)	Lama pengerjaan re-testing (menit)
TC1	T6	69	17
	T7	98	75
TC2	T5	104	22
	T9	63	15
TC3	T3	142	51
	T14	54	30
TC4	T12	110	33
	T16	92	42

Tabel 4. Rekapitulasi Lama Pengerjaan - Eksperimen *Automated*

Test case	Tester	Lama pengerjaan (menit)	Lama pengerjaan re-testing (menit)
TC1	T10	298	5
	T11	126	92
TC2	T1	222	13
	T13	120	15
TC3	T8	252	26
	T15	220	26
TC4	T2	152	20
	T4	220	40

Tabel 5. Rekapitulasi *Automable* Eksperimen *Automated*

Test case	Jumlah *Test Item*	Tester	Total *Automated* (*Test Items*)	*Percent Automable* (%)
TC1	40	T10	36	90.0
		T11	17	42.5
TC2	51	T1	33	64.7
		T13	27	52.9
TC3	37	T8	26	70.3
		T15	26	70.3
TC4	44	T2	17	38.6
		T4	16	36.4

a. *Iterasi* pelaksanaan testing

End-to-end testing dengan otomatis sangat dianjurkan jika SUT yang dibangun selalu terdapat peningkatan fitur secara berkala, sehingga dimungkinkan dibutuhkan proses *re-testing* pada semua fitur untuk meyakinkan penambahan fitur baru tersebut tidak menganggu fungsi fitur yang sudah ada. Proses pelaksaan *end-to-end testing* akan dengan cepat dapat dilakukan pada tiap iterasi apabila telah diotomasi, hanya perlu melakukan *running* kembali *test script* yang sudah dibuat. (perbandingan dapat dilihat pada tabel 3 dan 4 di kolom lama pengerjaan re-testing).

Sedangkan *end-to-end testing* dengan manual juga dapat dilakukan, namun lebih dianjurkan pada SUT yang tidak memiliki banyak fitur dan langsung selesai (dalam artian tidak ada penambahan fitur secara berkala). Hal ini dapat terlihat dari table 3 dan 4, bahwa pada iterasi pertama penggunaan manual testing, lama pengerjaan lebih singkat dari pada pembuatan *test script.*

b. *Test report*

Pengerjaan *end-to-end testing* secara manual membutuhkan *evidence* tangkapan layar untuk melengkapi status *actual result* dari proses testing yang dilakukan. Hal ini berpengaruh pada lama pelaksanaan *end-to-end testing* secara manual ketika iterasi (lihat perbandingan re-*testing* pada tabel 3 dan 4), dengan *automated software testing* diuntungkan dengan *generate report.*

c. *Training / experience* dalam *software testing*

Training atau pengalaman dalam *software testing* sangat diperlukan terutama untuk pelaksanaan *end-to-end testing* dengan *automated software testing*. Hal tersebut dapat terlihat pada table 4. Pada iterasi pertama *tester* membutuhkan waktu lama untuk pembuatan *test script.*

Hasil penelitian dimungkinkan memberikan lama pengerjaan yang berbeda apabila eksperimen dilakukan oleh *tester* yang telah berpengalaman dalam pelaksanaan testing dengan *automated software testing,* khususnya protractor. Selain lama pengerjaan, tester yang telah berpengalaman dengan *automated software testing* dimungkinkan banyak *test item* yang bisa diotomasi. Hal ini juga akan menekan lama pelaksanaan *end-to-end testing,* karena jika dilihat hubungan antara tabel 4 dan tabel 5 semakin besar *percent automable* maka semakin singkat waku pelaksanaan *re-testing.*

d. *Environment*
Environment yang mumpuni sangat berpengaruh untuk menekan waktu pelaksanaan *end-to-end testing,* terutama apabila *end-to-end testing* menggunakan *automated software testing*. Hal ini merupakan hal sederhana, namun patut menjadi pertimbangan bagi tim developer ketika akan melaksanakan pengujian terhadap software yang sedang dikembangkan.

IV. PENUTUP

Kesimpulan

Berdasarkan eksperimen yang dilakukan dapat disimpulkan bahwa:

a. *End-to-end testing* dapat dikerjakan secara manual testing jika pada aplikasi yang diuji memiliki lingkup yang tidak terlalu besar dan tidak direncanakan adanya penambahan fitur secara berkala. Hal tersebut karena proses *re-testing* akan menambah waktu lama pelaksanaan *end-to-end testing* jika dilakukan dengan manual testing.
b. Dari hasil eksperimen dapat diketahui bahwa *end-to-end testing* menggunakan *automated software testing* membutuhkan lama pengerjaan yang tidak sebentar pada iterasi pertama (pembuatan *test script*), akan tetapi ketika ada proses *re-testing*, lama pelaksanaan *end-to-end testing* dapat dipangkas dengan hanya melakukan *running* kembali pada *test script* yang sudah ada .

Selain hal tersebut, dapat diketahui beberapa faktor yang harus dipertimbangkan apakah *end-to-end testing* dilaksanakan dengan manual testing atau *automated software testing.* Faktor tersebut antara lain dalam pembuatan *test report*, *training,* atau pengalaman dalam bidang *software testing,* dan ketersedian *environment.*

Saran

Untuk studi lebih lanjut, beberapa hal disarankan:

a. Berdasarkan pembahasan dapat diketahui bahaw faktor kompetensi *tester* yang terlibat selama proses *end-to-end testing* memiliki pengaruh terhadap lama pengerjaan *end-to-end testing,* maka agar mendapatkan kesimpulan yang lebih mendalam dibutuhkan penelitian lebih lanjut dengan keragaman subjek penelitian (tester) yang terlibat dan berfokus hanya pada satu *test cases.*.
b. Berdasarkan studi pustaka dapat diketahui juga bahwan *end-to-end testing* merupakan kombinasi metode secara horizontal dan vertical, walaupun pada pelaksanaannya *end-to-end testing* metode horizontal lebih sering dilakukan. Dari hal tersebut disarankan untuk melakukan penelitian serupa dengan cakupan pelaksanaan *end-to-end testing* diperluas misalnya dengan integrasi dengan basis data, agar kesimpulan yang diambil bisa menggambarkan pelaksanaan *end-to-end testing* secara keseluruhan.
c. SUT yang digunakan untuk eksperimen memiliki keterbatasan dalam sisi fungsional (aplikasi SUT masih sederhana). Mengenai hal tersebut dipandang perlu untuk melakukan penelitian pembanding untuk melihat lama pengerjaan *end-to-end testing* dengan menggunakan SUT yang melibatkan keberagaman fungsi serta konten data yang lebih bervariasi, seperti video, grafik, dan lain-lain.
d. Eksperimen lain dapat juga dilakukan menggunakan objek penelitian berupa website yang telah ada (sudah di-*launching),* dengan skala yang lebih besar.

V. DAFTAR PUSTAKA

19759:2005, ISO /IEC TR. 2005. “Guide to the Software Engineering Body of Knowladge ISO/IEC 19759:2005, ISO /IEC TR.”

Afzal, W. “Metrics in Software Test Planning and Test Design Processes.” Department of Systems and Software Engineering. Blekinge Institute of Technology. Karlskrona. Sweerden, 2007.

Bertolino, Antonia, and Informazione A Faedo. 2007. “Software Testing Research :

Achievements , Challenges , Dreams Software Testing Research : Achievements , Challenges , Dreams." (September 2007).

Bustamante, Justo. 2017. "Introduction of Protractor as Test Automation Framework for AngularJS Applications."

Dobles, I, et, al. 2019. "Comparing the Effort and Effectiveness of Automated and Manual Tests: An Industrial Case Study." *Iberian Conference on Information Systems and Technologies, CISTI* 2019-June(June): 19–22.

Dudekula, R.M. 2011. "Automated Software Testing: A Study of the State of Practice." (December).

JUHA ITKONEN. 2008. "Do Test Cases Really Matter? An Experiment Comparing Test Case Based and Exploratory Testing."

Manova, Denitsa. "TASSA Methodology : End-to-End Testing of Web Service Compositions." *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*: 264–67.

Mirza, Aamir Mehmood, and Muhammad Naeem Ahmed Khan. 2018. "An Automated Functional Testing Framework for Context-Aware Applications." *IEEE Access* 6: 46568–83.

Myers, G.J, et al. 2011. *The Art of Software Testing*.

Palmér, T, H, et, Al. 2015. "Automated End-to-End User Testing on Single Page Web Applications Examensarbete Utfört i Medieteknik Vid Tekniska Högskolan Vid Linköpings Universitet." http://www.ep.liu.se/.

Sharma, R M. 2014. "Quantitative Analysis of Automation and Manual Testing." *International Journal of Engineering and Innovative Technology (IJEIT)* 4(1): 252–57.