FRAMEWORK TESTING OTOMATIS BERBASIS SERENITY DAN JENKINS AUTOMATED BUILD

Umi Sa'adah, Jauari Akhmad Nur Hasim, Andhik Ampuh Yunanto, Desy Intan Permatasari, Fadilah Fahrul Hardiansyah, Irma Wulandari, Hazna At Thooriqoh
Departmen Teknik Informatika dan Komputer, Politeknik Elektronika Negeri Surabaya
Jalan Raya ITS, Kampus PENS, Sukolilo, Surabaya 60111, Indonesia
E-mail: umi@pens.ac.id, jauari@pens.ac.id, andhik@pens.ac.id, desy@pens.ac.id, fahrul@pens.ac.id, irma@pens.ac.id, haznaatthooriqoh@it.student.pens.ac.id

ABSTRAK

Pengujian aplikasi memainkan peran penting dalam menghasilkan produk dengan kualitas tinggi dan tepat waktu. Proses pengujian yang dilakukan secara manual sering kali tidak akurat, kurang bisa diandalkan dan menghabiskan lebih banyak waktu dibandingkan dengan pengujian otomatis. Penelitian ini mengajukan sebuah framework untuk automation testing. Framework ini akan membantu pengembang untuk membuat aplikasi yang berkualitas dan mempersingkat waktu testing. Framework ini menawarkan solusi bagi pengembang agar proses pengujian dilakukan dengan mudah dan cepat. Konsep yang diajukan memuat script automation testing berbasis Framework Serenity yang bisa dijalankan sebagai background proses menggunakan Jenkins. Masukan dalam sistem berupa skenario testing, lalu dipetakan ke dalam Bahasa Pemrograman Java. Hasil yang ditampilkan dari Framework Automation Testing berupa test report yang merepresentasikan skenario yang telah dijalankan. Hasil pengujian menunjukkan bahwa proses ujicoba aplikasi menjadi lebih mudah. Serta proses penemuan error atau bug pada editor juga menjadi lebih cepat dibandingkan dengan cara manual. Sehingga pada penelitian ini dapat disimpulkan bahwa framework testing otomatis yang telah dikembangan ini dapat meningkatkan kualitas produk aplikasi melalui cara kerja yang efektif dan efisien.

***Kata Kunci**: Framework Automation Testing, Multiplatform framework, Background Process Testing*

AUTOMATIC TESTING FRAMEWORK BASED ON SERENITY AND JENKINS AUTOMATED BUILD

Umi Sa'adah, Jauari Akhmad Nur Hasim, Andhik Ampuh Yunanto, Desy Intan Permatasari, Fadilah Fahrul Hardiansyah, Irma Wulandari, Hazna At Thooriqoh
Department of Information and Computer Engineering, Surabaya State Electronics Polytechnic
Jalan Raya ITS, Kampus PENS, Sukolilo, Surabaya 60111, Indonesia
E-mail: umi@pens.ac.id, jauari@pens.ac.id, andhik@pens.ac.id, desy@pens.ac.id, fahrul@pens.ac.id, irma@pens.ac.id, haznaatthooriqoh@it.student.pens.ac.id

ABSTRACT

Software Testing plays an important role in making high quality products and the right time. The process of testing done manually is often inaccurate, unreliable, and needed more than automatic testing. This research proposes a new framework for automation testing. This framework will help developers to create applications with better quality and shorten testing time. This framework offers a solution for developers so that the testing process is carried out easily and quickly. Our proposed concept consists of an automated test script based on Serenity Framework and can be done as a background process using Jenkins. Input of the system is a testing scenario, then mapped into Java Programming Language. Output of this system are test reports that represent the scenario that has been carried out. the results of implementation system prove that developers are helped by this framework in the software testing process. So that in this study it can be concluded that the automated testing framework that has been developed can improve the quality of application products through effective and efficient work methods.

***Keywords**: Framework Automation Testing, Multiplatform framework, Background Process Testing*

I. PENDAHULUAN

DALAM setiap perubahan atau penyempurnaan sistem, pengujian memainkan peran yang sangat penting dalam menghasilkan produk dengan kualitas tinggi dan tepat waktu. Pengujian perangkat lunak merupakan bagian integral dari sebuah pembangunan perangkat lunak (*software development*) [1]. Perangkat lunak yang berkualitas merupakan perangkat lunak yang memenuhi *user requirements* dan *business process*, sekaligus mudah untuk dimodifikasi guna pengembangan lebih lanjut serta minim adanya *bugs.*

Untuk menjaga kualitas suatu perangkat lunak, maka sebelum dirilis developer melakukan proses pengujian. Jika pengujian dilakukan secara manual, hasilnya sering kali tidak akurat karena keterbatasan manusia atau *human error*. Pengujian manual kurang bisa diandalkan apabila dibandingkan dengan pengujian otomatis yang dilakukan oleh *tools* dan *script*. Selain itu pengujian manual juga menghabiskan waktu lebih banyak dibandingkan dengan pengujian otomatis. Secara signifikan dapat dikatakan bahwa pengujian otomatis lebih cepat dibandingkan dengan pengujian manual.

Ada beberapa penelitian tentang *Automation Testing*. Di antaranya penelitian dari Vishawjyoti dan Sachin Sharma [2] yang membahas tentang masalah penting dari pengujian perangkat lunak pada web. Pengujian dapat dilakukan secara manual maupun otomatis. Dengan *tools* yang mereka bangun, Hasil ujicoba dicatat secara otomatis pada proses *background* saat penguji memasukkan data dalam aplikasi web. Selain itu, Milad Hanna dkk. [3] juga melakukan penelitian untuk membandingkan fitur-fitur utama teknik *scripting* yang digunakan dalam proses *Software Automation Testing*. Dengan teknik scripting dapat menghemat biaya untuk *automation testing* perangkat lunak secara keseluruhan, meningkatkan kecepatan pengujian, mempersingkat *software development process*. Chandraprabha, Ajeet Kumar dan Sajal Saxena [4] melakukan penelitian dengan Selenium WebDriver, peneliti telah mengembangkan *data driven framework* yang berarti memisahkan data ke code untuk tujuan *reusability*. Dalam kerangka ini peneliti mengabstraksikan data yang akan digunakan dalam file excel dan program untuk mengakses data dari file excel tersebut.

Dewasa ini, berkembang beberapa framework *automation testing* yang akan memudahkan developer untuk melakukan *Component testing dan Scenario testing*. Dengan menerapkan *Automation testing* akan mempermudah *developer* dalam segi efisiensi waktu untuk mendeteksi adanya *bugs.* Selain itu, *developer* dimudahkan dalam pengujian yang dapat mendeteksi adanya regresi ketika terjadi perubahan pada sebuah kode program.

Penelitian ini mengajukan suatu pendekatan baru dengan membuat *Framework Automation Testing* yang multiplatform menggunakan teknologi bernama Serenity. Framework yang dibangun juga diintegrasikan dengan *Automated Build Server*. Framework ini dapat diimplementasikan pada aplikasi berbasis web, android maupun API. Serenity akan membantu developer untuk menulis *script automation* dengan bersih dan mudah dipelihara. Serenity juga menyediakan laporan hasil *automation testing*, sehingga developer tidak perlu membuat dokumentasi hasil testing [1] . Serenity juga memungkinkan untuk di integrasikan dengan beberapa teknologi automated build server seperti Jenkins. Jenkins akan membantu dalam menjalankan *automation testing* di background proses.

Dalam penelitian ini, framework *automation testing* diimplementasikan pada aplikasi Kasirsaku. Kasirsaku adalah aplikasi berbasis web dan android sekaligus menyediakan API. Kasirsaku digunakan untuk aplikasi kasir (*Point of Sale*) dalam menghitung barang pembelian konsumen. Kasirsaku merupakan aplikasi manajemen perdagangan dengan memberikan laporan penjualan, keuangan dan hutang. Selain itu, Kasirsaku juga merupakan aplikasi perangkat lunak berbasis mobile dan web yang dirancang oleh mahasiswa Politeknik Elektronika Negeri Surabaya dengan tema manajemen toko kelontong.

II. TEORI PENUNJANG

Dalam melakukan penelitian diperlukan beberapa teori yang nantinya sebgai konsep pengerjaan penelitian yang akan dikerjakan. Terdapat beberapa teori yang akan menunjang pengerjaaan penelitian nantinya, yakni:

A. Software Testing

Software Testing merupakan proses untuk menguji suatu perangkat lunak apakah sudah menjalankan proses yang sesuai dan menemukan cacat program sebelum digunakan. Proses pengujian memiliki dua tujuan berbeda:

1. Untuk menunjukkan kepada *developer* dan *customer* bahwa perangkat lunak memenuhi requirements.
2. Untuk menemukan situasi di mana perilaku perangkat lunak tidak benar, tidak diinginkan, atau tidak sesuai dengan spesifikasinya. Ini adalah konsekuensi dari software *defect*. *Defect* Testing berkaitan dengan mengatasi perilaku sistem yang tidak diinginkan seperti sistem crashes, interaksi yang tidak diinginkan dengan sistem lain, perhitungan yang salah, dan data yang salah [5]. Pada penelitian ini, proses testing lebih ditujukan untuk menemukan defect testing.

B. Automated Testing

Automation-Testing adalah pengujian terhadap suatu sistem (aplikasi) secara otomatis menggunakan alat bantu untuk mengeksekusi skenario uji coba yang telah anda buat. Piranti yang dikembangkan untuk melakukan automation-testing juga bisa memproses data dalam sistem yang diuji coba, membandingkan hasil yang didapat dan menghasilkan laporan detail dari pengujian yang dilakukan.

Proses pengembangan piranti lunak umumnya memerlukan test atau uji coba sebelum piranti lunak tersebut dirilis di pasaran. Automation-Testing menggeser testing konvensional yang ada yaitu dengan melakukan manual testing. Pada automation testing berarti segala proses uji coba tidak lagi melibatkan intervensi dari manusia (user). Tujuan dari automation testing sendiri adalah untuk mengurangi pengeksekusian skenario uji coba secara manual atau bahkan menghilangkan manual testing, sehingga mempermudah dan mempercepat proses uji coba

pada piranti lunak sebelum akhirnya dirilis. Secara keseluruhan proses dari automation testing ditunjukkan pada Gambar 1. Pada Gambar 1, terdapat alur proses yang terdiri dari: (1) pemilihan alat bantu testing, (2) pendefinisian *scope* dari *automation*, (3) *planning*, *design*, dan *development*, (4) eksekusi test, dan (5) *maintenance* [6].

C. *Framework Serenity*

Serenity BDD adalah library open source yang dapat membantu untuk menulis lebih bersih dan lebih mudah. Serenity juga menggunakan hasil tes untuk menghasilkan ilustrasi, narasi laporan, serta menjelaskan apa yang aplikasi lakukan dan bagaimana cara kerjanya.

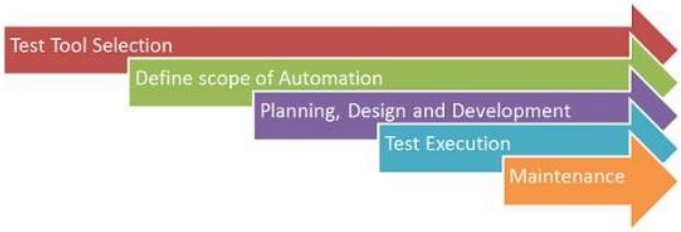
Serenity BDD memberikan dukungan yang kuat untuk tes web otomatis menggunakan Selenium 2, meskipun juga bekerja sangat efektif untuk tes non-web seperti tes pada web services atau bahkan memanggil kode aplikasi langsung. Tujuan dari Serenity adalah membuat penulisan secara baik dan terstruktur dengan cepat dan mudah, mudah untuk dilakukan maintain [7].

D. *Jenkins*

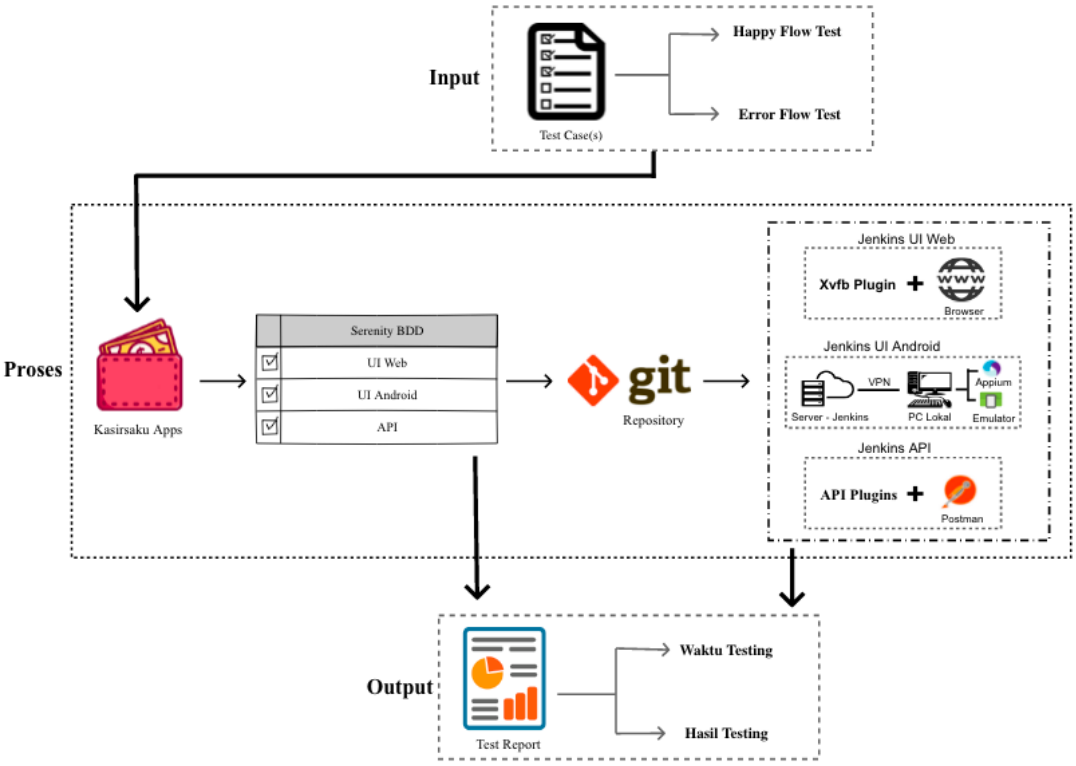
Jenkins adalah server otomatisasi *open source* yang ditulis dalam bahasa Java. Dengan integrasi berkelanjutan dan memfasilitasi aspek teknis pengiriman yang berkesinambungan, jenkins akan membantu developer dalam proses pengembangan perangkat lunak. Jenkins juga merupakan sistem berbasis server yang berjalan dalam kontainer servlet seperti Apache Tomcat. Build dapat dipicu dengan berbagai cara, misalnya dengan *commit* dalam *version control system*, dengan penjadwalan dan dengan meminta URL build spesifik [8].

III. DESAIN SISTEM

Pada penelitian ini, secara keseluruhan desain sistem yang diajukan terlihat pada Gambar 2.



Gambar 1. Bagan proses pada *automation testing*.



Gambar 2. Desain sistem aplikasi.

Pada Gambar 2, desain sistem aplikasi menunjukkan gambaran umum mengenai penelitian yang dilakukan. Terdapat bagian-bagian tertentu di dalam rancangan desain sistem diatas, dengan fokusan bahasan yang berbeda-beda. Alur umum keterkaitan proses yang terjadi dimulai dari adanya masukan data dari pilihan *testing* yang akan digunakan. Data tersebut akan diproses oleh apliaksi yang dibangun mengenai antarmuka dan APInya. Selanjutnya hasil testing tersebut akan otomatis terupload pada sistem Git agar dapat diintegrasikan dengan tools Jenkins. Sehingga hasil akhir atau *output mengenai automation testing* akan muncul. Untuk lebih detil, penjelasan mengenai bagian-bagian dijelaskan pada subbab selanjutnya.

A. Input Data Sistem

Input berupa *test-case(s)*, *test-case(s)* sendiri adalah komponen dan kriteria yang perlu diuji dalam suatu sistem. Test-case(s) adalah skenario yang dirancang dan disepakati sejak awal oleh semua pihak dalam pengembangan sistem yang akan diuji coba. Jenis test-case(s) yang akan digunakan yaitu *full-component test case* dan *scenario test case*. *Full-component test case* artinya semua komponen akan diuji satu persatu, dalam berbagai keadaan masukan. *Scenario test case* artinya test yang berdasarkan perilaku atau suatu aktivitas yang umunya dilakukan.

Menurut keadaan yang dimasukkan, biasa dibedakan menjadi dua yaitu *Happy-Flow* dan *Error-Scenario*. *Happy-Flow* adalah keadaan dimana suatu komponen diuji sesuai dengan tata cara atau aturan yang ada untuk mendapat-kan hasil yang diinginkan, sedangkan *Error-Scenario* adalah keadaan dimana suatu komponen diuji dengan tata cara yang salah, atau parameter yang tidak seharusnya, untuk mengetahui bagaimana sistem menangani kesalahan. *Test case* ini nantinya akan diterjemahkan kedalam skenario atau *story* dalam *main project*.

B. Automation Testing Project

Automation testing *project* merupakan project utama yang fungsinya adalah untuk menuliskan skenario *automation testing* yang dilakukan. Pada penelitian ini, *Automation Testing* akan diujikan pada Aplikasi Kasirsaku (API, web, dan Android). Pada Aplikasi Kasirsaku terdapat 6 fitur utama yang telah dibangun diantaranya sebagai berikut:

1. *Barcode scanner* dan pencarian barang
2. Daftar supplier dan pelanggan
3. Payment Point Online Banking
4. Manajemen Hutang
5. Laporan

C. Design Output

Output dari Aplikasi ini berupa *Test Report* yang ter-*generate* otomatis oleh *Serenity BDD* setelah menjalankan *Automation Testing*. *Framework Serenity* akan secara otomatis mengenerate *test report* beserta informasi skenario manakan yang *error*, *pending*, dan *passing*. *Serenity* memberikan laporan komprehensif tentang hasil tes & eksekusi termasuk:

1. Narasi untuk setiap tes
2. Tangkapan layar untuk setiap langkah dalam ujian
3. Tampilan agregat hasil pengujian berdasarkan persyaratan
4. Tampilan agregat dari hasil tes dengan rilis

Selain output yang dihasilkan dari menjalankan *Automation Testing*, penelitian ini juga menghasilkan output dari analisa dengan membandingkan apakah dengan Automation Testing proses pengujian akan lebih efisien dibandingkan dengan *manual testing*.

IV. PERANCANGAN SISTEM

A. Automation Testing API

Automation Testing API (*Application Programming Interface*) merupakan pengujian untuk mengidentifikasi adanya kecacatan atau *error* pada produk/*software* pada sisi *backend* atau API. *Tools* tambahan untuk menjalankan *Automation Testing API* adalah dengan menggunakan *Rest Assured.*Untuk membuat *Framework Automation Testing API* diperlukan beberapa perubahan, diantaranya adalah:

1. Konfigurasi POM.

Pada *Automation Testing API* dibutuhkan sebuah *Dependency* yaitu *Rest Assured*. *REST Assured* adalah sebuah java DSL (*Domain Specific Language*) untuk memudahkan testing untuk *REST services* yang dibangun pada HTTP *Builder*. *REST Assured* mendukung berbagai macam request seperti POST, GET, PUT, DELETE, OPTIONS, PATCH dan HEAD dan dapat digunakan untuk memvalidasi dan memverifikasi respon dari berbagai request tersebut.

2. Konfigurasi *class* serenity.properties

Pada *class* serenity.properties merupakan class untuk mengkonfigurasi *custom* webdriver, cucumber.option untuk *test* json API.

3. Konfigurasi *test report*

Dalam menggenerate test report, sistem perlu ditambahkan plugins *net.serenity-bdd.maven.plugins* dengan versi 2.0.11.

4. Menambahkan direktori APIController di package Main

Untuk membuat *Class Controller* pada *Automation API*, kita tidak bisa menggunakan class Page. Maka dibutuhkan direktori tambahan yakni direktori *controller*.

B. Automation Testing User Interface Web

Automation Testing UI (*User Interface Testing*) Web merupakan pengujian untuk mengidentifikasi adanya kecacatan atau *error* pada produk/*software* berbasis web. Untuk menjalankan *Automation Testing* pada UI Web diperlukan *tools* tambahan *browser* seperti chrome, firefox, atau safari.

C. Automation Testing User Interface Android

Automation Testing UI (*User Interface Testing*) Android merupakan pengujian untuk mengidentifikasi adanya kecacatan atau *error* pada produk/*software* berbasis android. Untuk menjalankan *Automation Testing* pada UI Web maka perlu *tools* tambahan untuk server android, dimana peneliti menggunakan *tools* bernama Appium. Selain Appium maka diperlukan juga Android device, bias menggunakan *real device* atau *virtual device*. Untuk membuat *Framework Automation Testing UI Android* diperlukan beberapa perubahan, diantaranya adalah:

1. Konfigurasi *class* serenity.properties

Class serenity.properties merupakan *class* untuk mengkonfigurasi *custom Android driver*, dan beberapa konfigurasi tambahan. *Class* ini berperan untuk menghubungkan kode *automated testing* dengan Appium. Informasi yang di butuhkan adalah *port* Appium, nama platform android, nama device android, nama activity dan app package, Appium hub dll.

2. Konfigurasi *class driver Android*

Untuk membuat Framework UI Android maka diperlukan modul tambahan sebagai Android *driver* yang akan menghubungkan antara *code Automation Testing* dengan *Device Android*.

D. Jenkins Automated Build

Jenkins merupakan *open-source automation server* yang ditulis dalam bahasa Java. Jenkins membantu menggotomatiskan bagian non-manusia dari proses pengembangan perangkat lunak, dengan integrasi berkelanjutan dan memfasilitasi aspek teknis pengiriman yang berkesinambungan. Developer atau QA akan meng-*commit script Automation Testing* yang akan terintegrasi dengan Jenkins. Maka jika ada *success* atau *fail* pada saat script di-*build*, user akan menerima notifikasi berupa *test report*.

V. UJI COBA DAN ANALISIS

Dalam melakukan evaluasi pengujian, penelitian ini menggunakan tiga parameter atau variabel penilaian sebagai berikut :

1. Perbandingan langkah-langkah yang dibutuhkan dalam menjalankan sistem antara secara manual dengan secara *automated testing*.
2. Perbandingan waktu respon yang dibutuhkan antara *manual testing* dengan *automation testing*.
3. Perbandingan kecepatan untuk menemukan *bugs* atau *error* dari aplikasi melalui *test report* antara *manual testing* dengan *automation testing*.

Pengujian ini menggunakan modul API, UI Web dan Android dari Aplikasi Kasirsaku. Selain Aplikasi Kasirsaku penguji juga membuat kode. Untuk skenario Automation Testing yang dilakukan peneliti dijabarkan melalui Tabel 1.

Tabel 1. Tabel Data Jumlah Fitur dan Skenario Pengujian.

No	Modul	Jumlah Fitur	Jumlah Skenario
1	API	14	40
2	UI Web	8	103
3	UI Android	6	37

Gambar 3. Diagram *automation testing* UI Android.

A. Implementasi Automated Testing pada User Interface Android

Pada implementasi *automation testing* untuk UI Android, terdapat beberapa class Java *automation testing* yang ditunjukkan pada Gambar 3.

1. Feature

Dalam Serenity Cucumber, skenario disimpan dalam 'File Feature, yang berisi deskripsi keseluruhan *feature* serta sejumlah skenario. Terdapat 2 macam testing pada skenario ini, yaitu *HappyFlow Testing* dan *ErrorFlow Testing.* Untuk skenario *HappyFlow Testing* dituliskan dengan cara pengguna menginputkan *username* dan *password* yang benar. Sedangkan untuk skenario *ErrorFlow Teting* dituliskan dengan cara pengguna menginputkan saat beberapa username dan password salah ataupun username dan password kosong.

2. Steps

Pada Serenity Cucumber, setiap baris skenario dipetakan di Java Class yakni *steps class*. Pada *class* ini digunakan anotasi *@Given, @When, dan @Then* yang disesuaikan dengan skenario yang ditulis. Pada *class steps* juga dapat mendefinisikan *regular expressions* yang mengindikasikan parameter yang akan diteruskan dalam *method*. *Class steps* pada Serenity cucumber digunaan untuk mengatur agar *code* step *definition* menjadi komponen yang lebih mudah digunakan kembali. Anotasi @Steps akan memberitahu Serenity bahwa variabel yang ada pada *class* ini merupakan *steps library*. Pada Serenity kita menggunakan *steps library* untuk menambahkan lapisan abstraksi antara “Apa” dan “Bagaimana” *acceptance testing*. Cucumber *step definition* akan menggambarkan “apa” yang dilakukan *acceptance testing*. Dalam istilah yang cukup netral, ramah bisnis. Pendekatan berlapis ini membuat tes lebih mudah dipahami dan dipertahankan, dan membantu membangun *library* yang besar, dapat digunakan kembali yang dapat kita gunakan dalam tes lain. Tanpa pendekatan berlapis semacam ini, definisi langkah cenderung menjadi sangat teknis dengan sangat cepat, yang membatasi penggunaan kembali dan membuat mereka lebih sulit untuk dipahami dan dipelihara.

3. Pages

Pada Serenity BDD, untuk membuat kode *automation testing* menggunakan *Design Pattern Page Objects*. *Design Pattern Page Object* merupakan cata untuk mengisolasi detail implementasi dari suatu halaman web dalam suatu *class*. Objek dari sebuah halaman hanya *class* biasa yang meng-*extend class PageObject*. Pada contoh *class HomePage* di atas penulis menggunakan anotasi @FindBy untuk menginisialisasi *element web* yang akan di test. Pada anotasi @FindBy maka menggunakan **xpath** atau **id** dari *element* tersebut.

B. Perbandingan Langkah-Langkah Manual dan Automated Testing

Untuk membandingkan langkah-langkah yang dibutuhkan dalam melakukan pengujian *manual dan automated*, peneliti membagi menjadi 2 tahap, yaitu tahap persiapan dan tahap uji coba. Tabel 2 merupakan tabel perbandingan untuk persiapan dalam melakukan manual dan *automated testing*. Tabel 3 berikut ini merupakan tabel perbandingan untuk menjalankan *manual* dan *automated testing*. Sedangkan Tabel 4 ini merupakan tabel perbandingan untuk melakukan *manual* dan *automated testing*.

Berdasarkan Tabel 2 dan Tabel 3, pengujian sistem pada penelitian ini menunjukkan bahwa dalam membuat *automated testing* dibutuhkan usaha yang lebih besar untuk mempersiapkan sistem. Sedangkan pengujian *manual* tidak membutuhkan usaha yang besar. Selanjutnya, proses pengujian menunjukkan bahwa penggunaan *automated testing* pada skenario transaksi hanya memiliki 3 tahap. Apabila dibandingkan dengan *manual testing*, persiapan ujicoba memang membutuhkan waktu yang sedikit, namun pada saat melakukan pengujian, *manual testing* memiliki 18 tahap dimana 6 kali lipatnya dari *automated testing*.

Di samping itu, pengujian dilakukan sebanyak 5 kali percobaan untuk menjamin keakuratan. Pada setiap percobaan, selisih perbedaan waktu yang dihasilkan sangat kecil sehingga hasil pengujian bisa dianggap valid. Sebagai tambahan, beberapa perbedaan waktu tersebut karena adanya beberapa faktor yang mungkin terjadi seperti contoh kecepatan mengetik, kelancaran perangkat, koneksi internet, dan pengalaman dari penguji saat melakukan percobaan.

Selanjutnya berdasarkan Tabel 4, langkah-langkah untuk *automated testing* pada skenario yang sama memiliki 7 tahap persiapan dan 4 tahap pengujian. Sedangkan langkah-langkah untuk *manual testing* memiliki 4 tahap persiapan dan 18 tahap pengujian. Sehingga total langkah-langkah yang dibutuhkan oleh *automated testing* lebih kecil dibandingkan dengan manual testing. Selain itu, waktu pengujian dihasilkan oleh *automated testing* memiliki waktu 12 detik cepat dibandingkan *manual testing*. Hal ini dipengaruhi oleh banyak faktor. Di antaranya adalah kualitas perangkat pengujian, SDM penguji, kualitas koneksi internet, dan lain sebagainya.

Tabel 2. Tabel perbandingan persiapan *automated* dan *manual testing*.

No	Manual Testing	No	Automated Testing
1	Install aplikasi kasirsaku pada *device* Android	1	Membuat skenario uji coba
2	Memahami aplikasi yang akan di uji coba	2	Membuat *project automated testing* menggunakan *framework* Serenity BDD
3	Membuat skenario uji coba	3	Konfigurasi *project*
4	Mempersiapkan dokumentasi laporan uji coba	4	Membuat dan menulis *class feature*
		5	Membuat dan menulis *class steps*
		6	Membuat dan menulis *class page*
		7	Install aplikasi kasirsaku pada *device* Android yang akan diuji

Tabel 3. Tabel perbandingan proses *automated testing* dibanding *manual*.

No	Manual Testing	No	Automated Testing
1	Membuka Aplikasi Kasirsaku pada perangkat Android	1	Menjalankan emulator android
2	User login terlebih dahulu	2	Menjalankan Appium
3	Memilih menu Pembelian	3	Membuka dan Menjalankan project Automated Testing
4	Mengecek tombol checkout sebelum menambahkan barang untuk memastikan bahwa terdapat validasi	4	Laporan hasil uji coba sudah terbuat secara otomatis
5	Klik tombol cari barang dengan menggunakan fitur pencarian berdasarkan nama barang apakah berfungsi		
6	Mulai pencarian barang berdasarkan nama		
7	Memastikan tampilan sesuai dengan barang yang dipilih		
8	Klik tombol add to cart untuk menambahkan barang pada keranjang belanja		
9	Klik tombol tambah dan kurang pada item yang terpilih dan pastikan tampilan total barang sesuai dengan item yang ada pada keranjang		
10	Klik tombol checkout dan paastikan halaman checkout menampilkan hasil yang sesuai		
12	Melakukan pencarian pelanggan berdasarkan nama		
13	Pastikan hasil dari total belanja sesuai		
14	Masukkan total bayar pelanggan		
15	Jika total belanja kurang dari total bayar maka akan muncul popup hutang		
16	Jika total belanja lebih dari total bayar maka muncul popup kembalian		
17	Aplikasi akan menampilkan hasil pembelian		
18	Masukkan setiap hasil uji pada laporan pengujian		

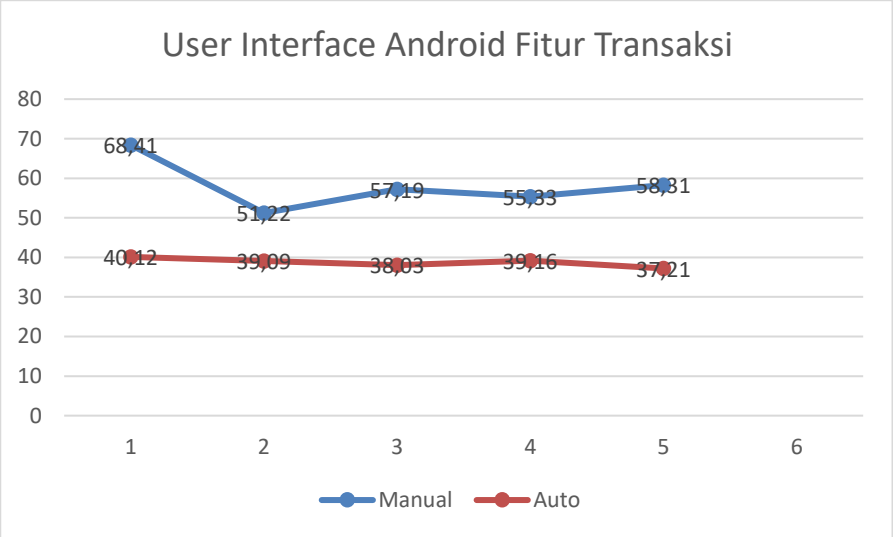
Tabel 4. Perbandingan langkah-langkah pengujian *manual* dan *automated*.

No	Komparasi	Langkah	Qty
1	Persiapan	Automated	7
		Manual	4
2	Uji Coba	Automated	4

		Manual	18

Tabel 5. Hasil waktu *manual testing* dan *automation user interface* pada *Android.*

No	Transaksi	Manual (s)	Automated (s)
1	user melakukan transaksi pembelian barang	68,41	40,12
2	user melakukan transaksi pembelian barang - Happyflow	51,22	39,09
3	user melakukan transaksi pembelian barang tanpa menginputkan barang - errorflow	57,19	38,03
4	user melakukan transaksi pembelian barang tanpa checkout - errorflow	55,33	39,16
5	user melakukan transaksi pembelian barang tanpa menginputkan jumlah bayar - errorflow	58,31	37,21



Gambar 4. Grafik perbandingan waktu pengujian *automated* dan *manual*.

Pada penelitian ini juga mengimplementasikan *Continous Integration and Continous Delivery (CI/CD)* menggunakan jenkins sehingga dalam melakukan *automated testing* tidak perlu dilakukan *trigger* kembali, karena setiap pengembang melakukan commit di git akan otomatis menjalankan automated testing nya pada background dan langsung menampilkan laporan di web server Jenkins.

C. Perbandingan Waktu Pengujian Manual dan Automated

Data hasil pengujian mengenai perbandingan waktu antara pengujian manual dan otomatis ditampilkan melalui Tabel 3. Grafik Serta waktu Hasil Pengujian User Interface untuk perangkat Android ditunjukkan pada Gambar 4.

Menurut hasil pengujian, dapat disimpulkan bahwa Automation Testing memerlukan waktu testing yang jauh lebih cepat dibandingkan dengan manual Testing.

VI. KESIMPULAN

Penelitian ini mengangkat masalah tentang implementasi automation testing pada aplikasi kasirsaku menggunakan *framework* Serenity dan Jenkins *Automated Build.* Framework Serenity yang digunakan pada paper ini menunjukkan bahwa kinerja testing akan lebih efektif dan efisien dibanding dengan *manual testing*. Pada serangkaian uji coba, penerapan *automation testing* menggunakan serenity menghasilkan waktu testing yang lebih cepat dibanding manual testing. Penelitian selanjutnya adalah mengembangkan sistem ini supaya dapat lebih reliabel dan fleksibel terhadap segala kasus tertentu sehingga terhindar dari faktor non-teknis.

DAFTAR PUSTAKA

[1] ISTQB Foundation. (N.D.). ISTQB Exam Certification. Retrieved May 18, 2018, From Istqb Exam Certification: http://istqbexamcertification.com/what-is-software-testing/

[2] John Ferguson Smart @Wakaleo. (N.D.). An Introduction to Bdd Test Automation with Serenity and Cucumber-Jvm. (Thycydides) Retrieved May 27, 2018, from http://thucydides.info/docs/articles/an-introduction-to-serenity-bdd-with-cucumber.html.

[3] Sommerville, I. (2019). Software Engineering (2011 - 9th Edition). in Software Engineering (2011 - 9th Edition) (P. 206). United States of America: Addison-Wesley.

[4] Guru 99. (N.D.). Guru 99. Retrieved July 9, 2018, from http://www.guru99.com/automation-testing.html.
[5] Serenity Bdd. (N.D.). Serebity Bdd. Retrieved July 9, 2018, from http://serenity-bdd.info/docs/serenity/.
[6] Wikipedia. (N.D.). Wikipedia. (Wikipedia) Retrieved July 8, 2018, from https://en.wikipedia.org/wiki/jenkins_(software).
[7] Wikipedia. (N.D.). Wikipedia. (Wikipedia) Retrieved July 9, 2018, from https://en.wikipedia.org/wiki/point_of_sale.
[8] Vishawjyoti, S. S. (2012). Study and Analysis of Automation Testing Techniques. Study and Analysis of Automation Testing Techniques, 3 (Automation Testing Techniques), 1.
[9] Milad Hanna, N. E.-H. (2014). A Review of Scripting Techniques Used in Automated Software Testing. A Review of Scripting Techniques Used in Automated Software Testing, 5 (Scripting Automated Software Testing), 1.
[10] Chandraprabha, A. K. (2015). Data Driven Testing Framework using Selenium. International Journal of Computer Applications, 118 (Software Testing), 18.