Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)DOI: 10.25126/jtiik.20196953
Vol. 6, No. 6, Desember 2019, hlm. 645-654 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 30/E/KPT/2018 e-ISSN: 2528-6579

STUDI PERBANDINGAN ALAT PENGUJIAN OTOMATIS UNTUK APLIKASI ANDROID

Arlinta Christy Barus[1], Leo Siburian[2]
[1,2]Institut Teknologi Del
Email: [1]arlinta@del.ac.id, [2]if314015@students.del.ac.id

(Naskah masuk: 25 Juli 2019, diterima untuk diterbitkan: 25 April 2019)

Abstrak

Pengujian adalah tahap yang penting dan harus dilalui dalam proses pengembangan perangkat lunak. Pengujian tersebut dilakukan untuk menghindari kesalahan yang mungkin terdapat pada perangkat lunak yang diuji. Ada banyak kasus uji (*test case*) yang harus dieksekusi dalam proses pengujian. Karena itu, pengujian yang dilakukan secara manual membutuhkan upaya yang besar. Oleh sebab itu pengujian otomatis (*automated testing*) menjadi hal yang penting untuk dipertimbangkan menggantikan pengujian manual. Pengujian otomatis adalah penggunaan kakas pengujian (*testing tools* atau *testing framework*) dalam melakukan pengujian suatu perangkat lunak yang secara signifikan mengurangi waktu yang dibutuhkan untuk melakukan pengujian. Ada banyak kakas yang dapat digunakan untuk melakukan pengujian otomatis, antara lain Selendroid, Calabash, dan UI Automator. Tulisan ini membahas tentang studi perbandingan kakas pengujian otomatis pada aplikasi *mobile* berbasis android dengan menggunakan Selendroid, Calabash, dan UI Automator. Eksperimen dilakukan untuk mengetahui kelebihan dan kekurangan masing-masing tools. Dari hasil analisis dan eksperimen, penulis merekomendasikan UI Automator sebagai kakas terbaik dalam hal kemudahan penginstalasian dan menjalankan kasus uji dalam sebuah kegiatan pengujian aplikasi *mobile* berbasis android.

Kata kunci: *Pengujian Otomatis; Android; Selendroid; Calabash; UI Automator;*

COMPARATIVE STUDY OF AUTOMATED TESTING TOOLS FOR ANDROID APPLICATIONS

Abstract

Testing is a must to do phase in software development process. It is perfomed to avoid any bugs that may exist in the software. There are many test cases to be executed in the testing process to make sure software is running according to its specification and without any bugs. Testing done manually takes a long time and extra work. Therefore, automated testing is important. Automated testing is the use of testing tools or testing framework in testing a software. Automated testing aims to test or significantly reduce the time required for testing. There are many tools that can be used to perform test automation of android mobile application, including Selendroid, Calabash, and UI Automator. This paper discusses about comparative studies of automated testing tools on android applications using Selendroid, Calabash, and UI Automator. Some experiments are conducted to know the strengths and weakness of each tool. Based on this study, we give recommendation to UI Automator as the handiest tool to use in term of installation and the execution of the test cases.

Keywords: ***Automated Testing; Android; Selendroid; Calabash; UI Automator;***

1. PENDAHULUAN

Sebelum merilis sebuah perangkat lunak (*software*) yang telah dikembangkan, perlu dilakukan pengujian (*testing*) terhadap perangkat lunak tersebut. Pengujian merupakan proses untuk menganalisis sebuah perangkat lunak untuk mendeteksi perbedaan antara kondisi yang ada dan kondisi yang diharapkan, mendeteksi kegagalan (*error, bugs*) dalam aplikasi dan untuk mengevaluasi fitur dari sebuah perangkat lunak (Interactive, 2016), (Sinaga, 2013).

Pengujian perangkat lunak dapat juga diartikan sebagai sebuah proses untuk melakukan verifikasi dan validasi sehingga sebuah aplikasi atau program memenuhi kebutuhan bisinis dan teknis yang dirancang oleh desainer dan dibangun oleh pengembangannya sehingga dapat bekerja sesuai dengan yang diharapkan (Bentley, Bank, & Nc, 2005), (Alaqail & Ahmed, 2018). Pengujian merupakan fase yang penting dan vital dalam *Software Development Life Cycle* (SDLC) yang dilakukan untuk menguji kode program dan

memastikan apakah sebuah perangkat lunak telah melakukan fungsi-fungsi sesuai dengan yang telah dirancangkan pada tahapan desain (Myers, 2004), (Alaqail & Ahmed, 2018), (Jindal, 2016). Dengan melakukan kegiatan pengujian maka perangkat lunak yang dihasilkan akan memiliki kualitas yang tinggi dan dapat menghemat biaya pemeliharaan.

Aplikasi *mobile* yaitu perangkat lunak yang dikembangkan untuk penggunaan layar yang kecil seperti *tablet* dan *smartphone* daripada penggunaan komputer. Android adalah sebuah *software stack* untuk *mobile* yang termasuk di dalamnya sistem operasi berbasis Linux, *middleware* dan *Core Application*. Karena itu, Android berfungsi sebagai sebuah *platform* yang dapat digunakan dan diadaptasikan untuk menghidupkan konfigurasi perangkat keras yang berbeda-beda. *Mobile phone* merupakan perangkat utama dari android namun android dapat juga digunakan untuk elektronik *book readers*, *netbooks*, *tablets* dan *set-top boxes* (Collins, Galpin, & Kaeppler, 2011).

Android adalah salah satu sistem operasi yang banyak digunakan pada perangkat *mobile* saat ini karena bersifat terbuka dan penggunaanya yang besar (Khan & Firdous, 2017). Pada tahun 2012, total pengguna *smartphone* android di beberapa negara mencapai 31%, persentase ini jauh lebih tinggi dibandingkan sistem operasi lain seperti IOs atau Symbian (Interactive, 2016). Berdasarkan informasi di statista.com (Statista, 2016), pada tahun 2015, total *smartphone* yang menggunakan sistem operasi android mencapai 1.808.000.000. Sifat android dengan jumlah pengguna yang menduduki peringkat pertama di dunia membuat platform ini menjadi *platform* menarik untuk diserang sehingga diperlukan pengujian yang benar-benar layak.

Ketika seorang pengembang perangkat lunak *(developer)* membuat dan meluncurkan sebuah aplikasi android maka aplikasi tersebut harus telah diuji dengan baik untuk menghindari kesalahan yang potensial merusak. Pengujian pada aplikasi android dibagi atas dua bagian yaitu (Collins, Galpin, & Kaeppler, 2011):

1 *Unit test*
   Unit test memfokuskan pada testing sebuah kode yang spesifik, biasanya sebuah class.
2 *Functional test*
   Functional test atau pengujian fungsional merupakan pengujian terhadap perilaku dari aplikasi anda terhadap beberapa potongan program atau kode. Pengujian ini sangat maksimal karena memperbolehkan penguji untuk menerjemahkan cara penggunaan aplikasi secara langsung ke sebuah kelompok uji kasus yang dapat memverifikasi bahwa aplikasi beroperasi sesuai dengan harapan.

Ada ber-ribu kasus uji *(test-case)* yang harus dieksekusi dalam proses pengujian sebelum meluncurkan sebuah aplikasi. Jika dilakukan secara manual, maka proses pengujian ini akan membutuhkan waktu yang lama dan pekerjaan ekstra (Smartbear, 2016). Itulah sebabnya digunakan pengujian otomatis *(automated testing)*. Pengujian otomatis dapat dilakukan dengan bantuan alat uji *(testing tool)* atau kerangka uji *(testing framework)*. Alat atau kerangka pengujian ini digunakan untuk menyimpan hasil atau laporan dari setiap kasus uji yang dieksekusi (Dary, 2015). Beberapa alat yang sering digunakan untuk pengujian sebuah aplikasi android adalah Selendroid, Calabash, dan UI *Automator*.

Selendroid merupakan salah satu kerangka pengujian untuk aplikasi android yang masih menggunakan infrastruktur Selenium untuk *web* (Dary, 2015). Calabash merupakan kerangka otomatisasi uji yang memungkinkan pengembang *mobile* untuk membuat dan melaksanakan pengujian secara otomatis (Helppi, 2016). UI *Automator* merupakan instrumen yang menggunakan *script* yang dapat disusun sebagai serangkaian pengujian yang akan dijalankan terhadap aplikasi (Developers, 2018).

Salah satu tantangan terbesar dalam pengujian perangkat lunak adalah memilih alat yang tepat dan sesuai dengan kebutuhan pengujian. Dalam memilih sebuah alat atau kerangka uji yang ingin digunakan, ada hal yang harus diperhatikan oleh pengembang *(developer)* atau penguji *(tester)* untuk menentukan alat mana yang memang benar-benar dibutuhkan.

Selendroid, Calabash, dan UI Automator dipilih sebagai alat yang akan dibandingkan karena ketiga alat tersebut merupakan sumber terbuka *(open source)* sehingga penulis tidak perlu untuk membayar dan alat tersebut sepadan untuk dibandingkan karena memiliki cara kerja yang sama dalam melakukan pengujian aplikasi android (Ghahrai, 2017). Selendroid, Calabash, dan UI Automator juga dapat dijalankan pada perangkat keras *mobile (real device)* maupun emulator.

Penelitian yang akan dilakukan penulis bertujuan untuk melakukan perbandingan alat Selendroid, Calabash, dan UI *Automator* dengan melakukan pengujian fungsional aplikasi android menggunakan masing-masing tiga alat tersebut. Pendekatan yang digunakan pada pengujian fungsionalitas yaitu dengan teknik pengujian *black box* yaitu pengujian yang dilakukan dengan mengamati hasil eksekusi atau fungsionalitas suatu perangkat lunak. Perbandingan dilakukan berdasarkan kriteria-kriteria yang ditetapkan sebagai parameter pembanding. Hasil penelitian ini dapat dijadikan sebagai referensi dalam mempertimbangkan pemilihan alat pengujian aplikasi android bagi pengembang maupun penguji.

2. METODE PENELITIAN

Pada bagian ini merupakan tahapan-tahapan yang dilakukan pada penelitian ini sehingga penelitian ini dapat diselesaikan secara sistematis, terarah dan jelas. Metode yang dilakukan selama proses pengerjaan penelitian ini adalah:

1. Eksplorasi Alat Pengujian
   Melakukan eksplorasi pada alat pengujian *mobile* Android, Calabash, dan UI Automator.
2. Analisis
   Setelah melakukan studi literatur dan eksplorasi alat maka penulis melakukan analisis untuk menentukan kriteria-kriteria yang digunakan sebagai pembanding. Menganalisis setiap perilaku pada kriteria pembanding, kekuatan atau kelemahan pada sebuah alat dibandingkan dengan pengimplementasiannya dengan alat lain.
3. Perancangan dan Eksperimen
   Mengimplementasikan hasil analisis yaitu setiap kriteria yang sudah dirancang. Aplikasi yang akan diuji selama eksperimen adalah aplikasi android Siantar *Smart City*, aplikasi *Employee Directory* dan Selendroid-test-app-0.17.0 (Dary, 2015) .

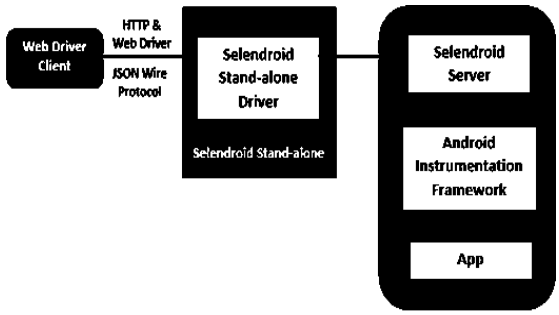
2.1. Eksplorasi Alat Pengujian

a. Selendroid

Selendroid adalah kerangka pengujian otomatis *(test automation framework)* untuk berbagai jenis aplikasi *mobile (multi type mobile application)*, yaitu *native*, *hybrid* android app dan *mobile web*. Pengujian ditulis dengan menggunakan API Selenium2 *client*. Selendroid dapat digunakan pada *real device* dan emulator juga dapat diintegrasikan dengan selenium-*grid*. Selendroid terdiri atas empat komponen utama (lihat Gambar 1), yaitu:

1. Selendroid Client yaitu java client library yang berbasis pada SeleniumWebBrowser
2. Selendroid Server yang dijalankan pada android device
3. AndroidDriver-App yaitu aplikasi khusus untuk menguji mobile web
4. Selendroid-Standalone yaitu *proxy* antara selendroid *client* dan selendroid *server*.

Komunikasi antara *web driver client* dan *selendroid standalone* berbasis pada JSON *wire protocol*. Fungsi utama dari selendroid *standalone* adalah menangani *request* dari *client* dan meneruskannya ke *device* atau *emulator* termasuk men-*transfer* aplikasi dari komputer ke android *device* (Dary, 2015).



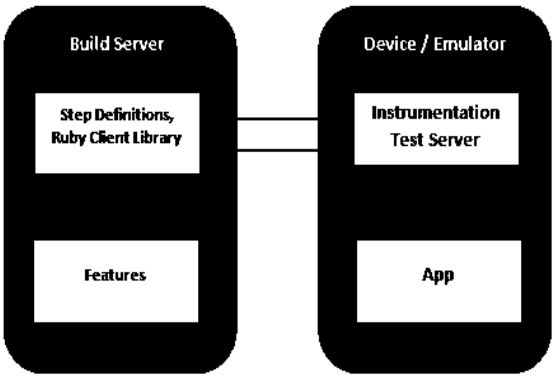
Gambar 1 Arsitektur Selendroid

b. Calabash

Calabash adalah kerangka pengujian otomatis yang memperbolehkan pengembang atau siapapun yang tidak mempunyai keahlian pemrograman untuk membuat dan mengeksekusi kasus uji untuk android maupun IoS. Calabash bekerja dengan memperbolehkan interaksi UI secara otomatis terhadap aplikasi seperti menekan tombol, memasukkan teks, memvalidasi respon dan lain-lain.

Kasus uji pada Calabash ditulis menggunakan bahasa pemrograman ruby. Untuk menjalankan atau mengeksekusi kasus uji dapat menggunakan *command prompt*. Perintah calabash-android akan membentuk sebuah *folder* baru bernama *features*.

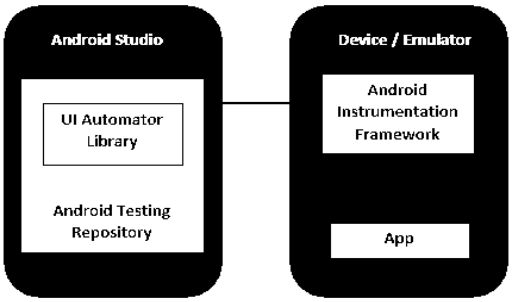
Di dalam *folder* tersebut terdapat **step_definition**, ***support*** dan **file my_first_feature**. *File* my_first_feature berisi skenario yang akan dijalankan, dan pada *folder* step_definition terdapat *file* Calabash_steps.rb, *file* ini adalah kode dari skenario yang akan dijalankan. Arsitektur Calabash dapat dilihat pada Gambar 2.



Gambar 2 Arsitektur Calabash

c. UI Automator

UI Automator adalah kerangka pengujian otomatis yang disediakan pada koleksi pengujian android *(testing library)*. UI Automator memperbolehkan pengguna berinteraksi dengan elemen pada perangkat secara langsung tanpa memperhatikan fokus aktivitas yang dilakukan. UI Automator merupakan sebuah *intrumentation-based* API dan bekerja dengan AndroidJUnitRunner *test runner* dengan aristektur yang dapat dilihat pada Gambar 3.

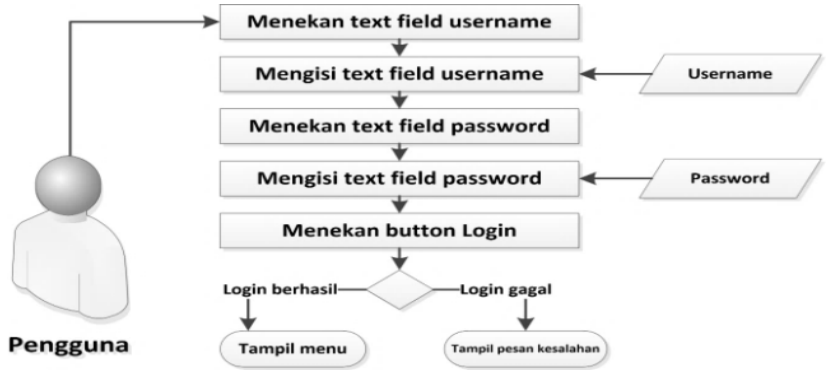


Gambar 3 Arsitektur UI Automator

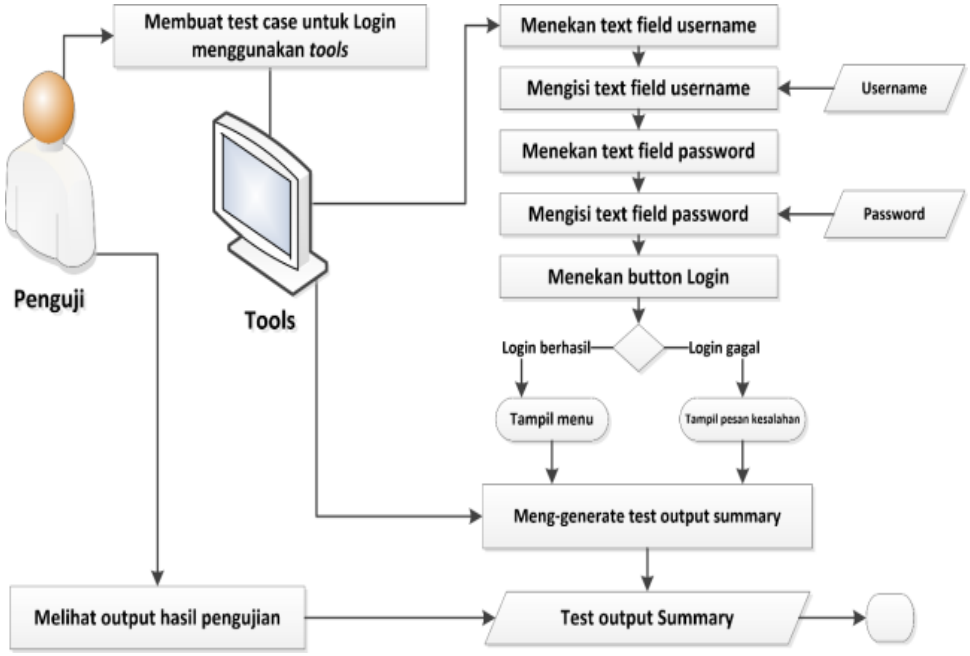
d. Cara Akses

Selendroid, Calabash, dan UI Automator merupakan alat yang diperuntukkan dalam melakukan pengujian fungsional terhadap aplikasi android. Pengujian ini dilakukan berdasarkan cara seorang pengguna dalam menggunakan sebuah fitur dari aplikasi *android*. Contoh kasus misalkan saat seorang pengguna mengakses fitur *login*, pengguna perlu memasukkan *username* pada *textbox username*, memasukkan *password* pada *textbox password*, kemudian menekan *button Login*.

Maka alur pengujian menggunakan ketiga alat pengujian sama seperti cara pengguna menggunakan fitur tersebut. Perbedaannya adalah proses tersebut dijalankan menggunakan sebuag alat. Langkah-langkah untuk mengakses fitur tersebut dibuat menjadi kasus uji menggunakan bahasa pemrograman dapat dilihat pada Gambar 4 dan Gambar 5.



Gambar 4 Alur seorang user saat mengakses fitur Login pada aplikasi



Gambar 5 Alur pengujian fitur login menggunakan alat pengujian

2.2. Analisi Kriteria Pembandingan

Agar dapat melakukan perbandingan tehadap ketiga alat yaitu Selendroid, Calabash dan UIAutomator maka telah ditetapkan kriteria-kriteria yang akan dijadikan sebagai parameter pembanding. Kriteria-kriteria ini ditentukan berdasarkan studi literatur dan berdasarkan hasil analisis.

Pengujian dilakukan terhadap tiga (3) aplikasi android yang berbeda yaitu: aplikasi android Siantar Smart City yang merupakan proyek akhir dari mahasiswa Institut Teknologi Del, aplikasi Employee Directory dan Selendroid-test-app-0.17.0 yang didapatkan dari: http://selendroid.io/.

Pengujian dilakukan pada fitur-fitur dari ke tiga aplikasi dengan memperhatikan kriteria-kriteria yang ditetapkan. Artinya pengujian terhadap fitur-fitur yang dimiliki aplikasi android dilakukan untuk

mendapatkan hasil yang sesuai dengan kriteria-kriteria pembanding yang telah ditetapkan.

Berdasarkan kriteria pengujian yang dibahas pada penelitian sebelumnya antara lain (Venkatasubramanian, 2015), (Bajaj, 2015), (Kaur, 2015), (Jagadale, Pagar, Deore, Raut, & Kumavat, 2015) ditetapkan beberapa kriteria pengujian sebagai berikut:

K.1. Apakah alat pengujian dapat digunakan untuk menguji aplikasi Native, Hybrid, dan Web Application (Venkatasubramanian, 2015).

Aplikasi pada android dibagi ke dalam tiga bagian, yaitu *Native*, *Hybrid* dan *Web Application*. Aplikasi *native* adalah aplikasi yang dikembangkan secara khusus untuk satu *platform*, aplikasi seperti ini dapat dipasang melalui *application store* seperti Google Play. Contoh aplikasi *native* adalah Whatsapp. *Web Application* merupakan *website* yang dapat diakses melalui *smartphone* dengan bantuan *web browser*. *Hybrid* adalah suatu cara untuk menampilkan konten dari sebuah *website* kedalam bentuk aplikasi, singkatnya *hybrid* adalah perpaduan antara *Web Application* dan *Native Application*. Untuk mendapatkan hasil perbandingan pada kriteria ini, maka akan dilakukan pengujian terhadap aplikasi *Native*, *Hybrid* dan *Web Application* dengan menggunakan ketiga alat pengujian. Kriteria ini penting untuk diuji karena merupakan salah satu faktor yang menentukan keberhasilan dan kesuksesan sebuah alat pengujian otomatis.

K.2. Waktu yang dibutuhkan untuk mengeksekusi kasus uji (Bajaj, 2015), (Venkatasubramanian, 2015).

Estimasi waktu yang dibutuhkan dalam pengujian merupakan hal yang sangat penting. Kriteria ini digunakan untuk membandingkan waktu yang dibutuhkan oleh setiap alat pengujian untuk mengeksekusi kasus uji dengan cara menguji sebuah aplikasi dengan ketiga alat pengujian dan dengan kasus uji yang sama dan kemudian membandingkan waktu yang dibutuhkan oleh setiap alat pengujian untuk mengeksekusi kasus uji.

K.3. Apakah alat pengujian tersedia untuk versi Android terbaru (Kaur, 2015).

Dari pertama kali android dirilis hingga saat ini, selalu ada peningkatan versi dari android tersebut. Versi android dimulai dari versi 1.5 yang diliris pada 27 april 2009 hingga versi terbaru 7.1.1 yang dirilis pada 22 agustus 2016. Lima versi terbaru dipilih untuk menguji apakah ketiga alat pengujian dapat melakukan pengujian terhadap ke-lima versi android tersebut.

K.4. Apakah alat pengujian dapat digunakan untuk melakukan pengujian baik pada *real device* dan emulator (Venkatasubramanian, 2015).

Emulator adalah perangkat lunak yang dapat menjalankan program-program android tanpa harus menggunakan perangkat aslinya. Kriteria ini digunakan untuk membuktikan apakah alat pengujian dapat melakukan pengujian baik pada emulator maupun pada perangkat nyata (*real device*). Kriteria ini merupakan salah satu dari 10 kunci penting dalam pemilihan alat pengujian yang akan digunakan dalam melakukan pengujian aplikasi android.

K.5. Sumber dalam mengeksekusi kasus uji pada masing-masing alat pengujian.

Penggunaan memori merupakan hal yang sangat penting untuk diperhatikan. Kriteria ini digunakan untuk mengetahui seberapa besar kapasitas memori yang digunakan oleh masing-masing alat pengujian dalam menjalankan atau mengeksekusi kasus uji saat melakukan pengujian perangkat lunak.

K.6. Bahasa pemrograman yang digunakan (Bajaj, 2015).

Setiap alat pengujian pasti menggunakan bahasa pemrograman dalam menuliskan kasus uji yang akan dieksekusi. Kriteria ini penting untuk mengetahui bahasa pemrograman apa yang digunakan oleh setiap alat pengujian dalam menuliskan kasus uji. Apakah alat pengujian hanya medukung satu bahasa pemrograman atau mendukung penulisan kode-kode yang dijadikan sebagai kasus uji dalam 2 atau lebih bahasa pemrograman.

K.7. Dokumentasi hasil eksekusi kasus uji dari setiap alat pengujian (Jagadale, Pagar, Deore, Raut, & Kumavat, 2015).

Setelah selesai mengeksekusi kasus uji, maka diperlukan laporan berupa dokumentasi hasil eksekusi dari setiap kasus uji, baik yang gagal maupun berhasil. Ketika terjadi kegagalan atau kesalahan, maka perlu diketahui letak permasalahan tersebut. Kriteria ini digunakan untuk membandingkan hasil dokumentasi dari setiap eksekusi kasus uji yang dilakukan saat pengujian aplikasi android menggunakan ketiga alat pengujian.

K.8. Apakah alat pengujian mendukung penundaan *(delay)* selama melakukan pengujian.

Selama melakukan pengujian, sangat perlu jika eksekusi kasus uji dapat ditunda beberapa waktu. Hal ini penting, sehingga proses pengujian dapat diperhatikan dengan lebih baik. Penundaan ini dapat diterapkan misalkan saat melakukan penekanan tombol dan menunggu sampai hasil yang diharapkan muncul. Kriteria ini dipilih dari dokumen yang dituliskan oleh AQUA (Aqua, 2013) yang merupakan organisasi yang membuat standarisasi untuk pengujian dan *quality assurance aplikasi mobile*.

K.9. Apakah alat pengujian mendukung pengangguhan atau mulai ulang *(Suspend*/*Resume)* aplikasi saat melakukan pengujian.

Kriteria ini perlu untuk mengetahui apakah saat melakukan pengujian menggunakan ketiga alat pengujian, kondisi terakhir *(state)* sebelum aplikasi ditangguhkan tersimpan atau tidak.

K.10. Apakah alat pengujian dapat menjalankan ulang *(re-launch)* aplikasi selama melakukan pengujian.

Perbedaan antara *suspend/resume* dan *re-launch* yaitu saat melakukan *resume* maka aplikasi harus menampilkan *state* terakhir dari aplikasi sebelum aplikasi di-*suspend*. Sementara saat melakukan *re-launch* maka aplikasi harus benar-benar memulai dari awal.

K.11. Apakah alat pengujian dapat menampilkan bagian-bagian yang tidak muat di layar *(scrolling)* saat melakukan pengujian.

Fitur *scrolling* sangat penting terutama saat mengakses informasi yang sangat panjang dan tidak dapat ditampilkan pada satu halaman. Fitur ini penting untuk mengetahui apakah ketiga alat pengujian dapat melakukan *scrolling* baik ke atas maupun ke bawah saat melakukan pengujian aplikasi.

K.12. Apakah alat pengujian mendukung *Simultaneous keypresses* atau *Multiple touch* saat melakukan pengujian.

Simultaneous keypresses sederhananya adalah menekan beberapa *button* atau *key* secara bertahap dalam periode yang singkat. Dan *multiple touch* adalah proses menekan dua atau lebih button atau *key* secara bersamaan. Kriteria ini penting untuk melihat apakah ketiga alat pengujian dapat melakukan penekanan *key* secara simultan maupun penekanan dua atau lebih *button* atau *key* secara bersamaan.

K.13. Apakah alat pengujian mendukung *Multiple Display* Format dalam melakukan pengujian.

Aplikasi android dapat ditampilkan secara *portrait* maupun *landscape*. Elemen pada aplikasi tentu harus konsisten sekalipun ditampilkan dalam format yang berbeda. Untuk itu, perlu dilakukan pengujian apakah ketiga alat pengujian dapat melakukan pengujian aplikasi dengan format tampilan yang berbeda, dan apakah alat pengujian dapat melakukan pengubahan format tampilan saat melakukan pengujian.

K.14. Apakah perubahan yang dilakukan selama melakukan pengujian menggunakan alat pengujian tersimpan.

Ada beberapa aplikasi yang konfigurasi atau pengaturannya dapat diubah. Penting untuk mengetahui apakah perubahan konfigurasi yang dilakukan selama melakukan pengujian menggunakan ketiga alat pengujian tersimpan atau tidak.

2.3. Eksplorasi Obyek Pengujian

a. Aplikasi Siantar City

Aplikasi Siantar *Smart City* adalah sebuah aplikasi *mobile* yang dapat memudahkan masyarakat dalam menyampaikan pengadauannya serta memudahkan pihak SKPD (SKPD) untuk mengetahui dan menindaklanjuti pengaduan yang disampaikan masyarakat.

Aplikasi ini digunakan oleh empat peran yang berbeda yaitu, Masyarakat, SKPD dan Administrator dan *guest* (tamu). Masing-masing peran memiliki hak akses yang berbeda. Ketika guest mengakases aplikasi web, guest memiliki hak akses untuk mendaftar kedalam sistem dan melihat daftar serta detail dinas SKPD. Masyarakat memiliki hak akses menyampaikan pengaduan kepada SKPD dan melihat informasi pengaduannya. Untuk dapat menggunakan aplikasi ini, masyarakat harus mendaftarkan diri terlebih dahulu. Setelah masyarakat mendaftar dan masuk ke dalam aplikasi, masyarakat dapat membuat pengaduannya.

b. Aplikasi Selendroid Test App

Selendroid test app merupakan aplikasi android yang disediakan oleh Selendroid.io yang diperuntukkan untuk melakukan pengujian aplikasi android. Pada Selendroid *test app* terdapat sebuah fitur registrasi sederhana, fitur untuk pengujian *scrolling*, *click*, *exception handling* hingga fitur untuk pengujian *multi-touch.*

c. Employee Directory

Employee directory adalah aplikasi *hybrid* yang disediakan oleh Selendroid.io untuk pengujian aplikasi *hybrid*. Aplikasi ini merupakan aplikasi sederhana dengan fitur pencarian karyawan dan menampilkan detail informasi dari seorang karyawan.

3. RANCANGAN DAN EKSPERIMEN

Tabel 1 dibawah ini menjelaskan rancangan eksperimen yang dilakukan terhadap kriteria pengujian pada penelitian ini.

Tabel 1 Rancangan pada Masing-Masing Kriteria

Kategori	Rancangan
K.1.	Rancangan eksperimen untuk menguji fitur ini adalah dengan melakukan pengujian terhadap tiga aplikasi yang berbeda. Aplikasi *Native* yang dipilih adalah aplikasi Siantar Smart City dan fitur yang akan diuji adalah fitur registrasi. Fitur ini dipilih karena dengan dapat melakukan pengujian untuk fitur registrasi maka alat pengujian juga dapat melakukan pengujian terhadap fitur lainnya. Untuk aplikasi *Hybrid* yang akan diuji adalah aplikasi Employee Directory yang didapatkan dari selendroid.io dan fitur yang akan diuji adalah fitur mencari karyawan yang disediakan oleh aplikasi ini. Untuk *Web Application* yang akan diuji adalah aplikasi *web* siantar smart city, dan fitur yang

Kategori	Rancangan
	diuji adalah fitur registrasi, jika alat pengujian dapat menjalankan pengujian untuk fitur registrasi, maka alat pengujian juga dapat melakukan pengujian untuk fitur-fitur lainnya.
K.2.	Rancangan pengujian pada kriteria ini adalah dengan membuat kasus uji untuk setiap fitur pada Aplikasi Siantar Smart City. Perlu dilakukan pengujian terhadap fitur yang lebih banyak daripada kriteria sebelumnya, untuk mendapatkan hasil yang lebih signifikan. Kemudian setiap kasus uji dijalankan menggunakan ketiga alat pengujian. Setelah setiap kasus uji berhasil dijalankan maka akan dibandingkan waktu yang dibutuhkan oleh masing-masing alat pengujian untuk menjalankan setiap kasus uji. Waktu dalam mengekseskusi kasus uji dapat dilihat dari laporan yang dihasilkan oleh alat pengujian setelah proses pengujian selesai dilakukan.
K.3.	Rancangan pengujian pada kriteria ini adalah dengan menjalankan pengujian fitur registrasi pada aplikasi Selendroid Test App. Kasus uji dijalankan menggunakan ketiga alat pengujian pada versi android yang berbeda-beda. Dari eksperimen ini akan didapatkan hasil apakah alat pengujian dapat melakukan pengujian pada versi Android yang berbeda-beda atau tidak. Jika alat pengujian dapat menjalankan pengujian terhadap fitur registrasi, maka alat pengujian juga dapat menjalankan pengujian terhadap fitur lainnya.
K.4.	Rancangan pengujian pada kriteria ini adalah dengan menjalankan pengujian terhadap fitur registrasi pada aplikasi Selendroid Test App. Kasus uji akan dijalankan pada kedua perangkat yang berbeda menggunakan ketiga alat pengujian yaitu *real device* (*smartphone*) dan emulator android. Dari eksperimen ini akan didapatkan hasil apakah alat pengujian dapat menjalankan kasus uji pada *real device* dan emulator atau tidak.
K.5.	Rancangan pengujian pada kriteria ini adalah dengan membuat kasus uji untuk setiap fitur pada Aplikasi Siantar Smart City. Setiap fitur dipilih untuk mendapatkan hasil yang lebih signifikan. Kemudian setiap kasus uji dijalankan menggunakan ketiga alat pengujian. Setelah setiap kasus uji berhasil dijalankan, selanjutnya adalah membandingkan berapa besar sumber memori yang digunakan oleh masing-masing alat pengujian untuk menjalankan setiap kasus uji. Fitur yang dipilih lebih banyak dibandingkan pada kriteria lainnya untuk mendapatkan hasil yang lebih signifikan.
K.6.	Rancangan pengujian pada kriteria ini adalah dengan membuat kasus uji berdasarkan Bahasa pemrograman yang didukung oleh alat pengujian. Pengujian dilakukan untuk setiap fitur pada Aplikasi Siantar Smart City dan fitur registrasi pada aplikasi Selendroid Test App. Kemudian setiap kasus uji dijalankan menggunakan ketiga alat pengujian. Dari eksperimen ini akan didapatkan bahasa pemrograman apa yang digunakan oleh setiap alat pengujian dalam pembuatan kasus uji.
K.7.	Rancangan pengujian pada kriteria ini adalah dengan memperhatikan dan membandingkan bagaimana dokumentasi atau luaran *(output)* dari hasil ekseksekusi kasus uji yang dijalankan menggunakan ketiga alat pengujian.
K.8.	Rancangan pengujian pada kriteria ini adalah dengan membuat kasus uji yang memungkinkan alat pengujian melakukan penundaan atau berhenti beberapa saat selama melakukan proses pengujian. Aplikasi yang diuji adalah aplikasi Selendroid Test App dan fitur yang diuji adalah fitur registrasi. Dari eksperimen ini akan didapatkan hasil apakah alat pengujian dapat melakukan penundaan selama melakukan pengujian atau tidak.
K.9.	Rancangan pengujian pada kriteria ini adalah dengan menjalankan kasus uji pada aplikasi Selendroid Test App dan fitur yang diuji adalah fitur registrasi. Pada saat proses pengujian, dilakukan *suspend* pada aplikasi dengan cara menekan *button Home* hingga *smartphone* kembali ke layar awal kemudian membuka kembali aplikasi yang sedang diuji. Perlu diperhatikan kondisi aplikasi sebelum ditangguhkan dan setelah dilanjutkan. Alat pengujian mendukung *suspend* dan *resume* jika kondisi aplikasi sebelum ditangguhkan sama dengan kondisi aplikasi saat dilanjutkan. Dari eksperimen ini akan didapatkan hasil apakah alat pengujian dapat melakukan *suspend* dan *resume* atau tidak.
K.10.	Rancangan pengujian pada kriteria ini adalah dengan menjalankan kasus uji pada setiap fitur dari aplikasi Siantar Smart City. Cara pengujian nya adalah dengan menjalankan kasus uji baru setelah sebuah kasus uji selesai dijalankan. Alat pengujian mendukung *re-launch* jika memang aplikasi benar-benar dijalankan dari awal. Dari eksperimen ini akan didapatkan apakah alat pengujian mendukung proses *re-launch* atau tidak.
K.11.	Rancangan pengujian pada kriteria ini adalah dengan menjalankan kasus uji yang memungkinkan alat pengujian melakukan scrolling selama proses pengujian. Fitur yang diuji pada kategori ini adalah fitur *scroll* pada aplikasi Selendroid Test App. Pada fitur ini terdapat halaman yang lebih dari satu halaman penuh android.
K.12.	Skenario pengujian untuk kriteria ini adalah dengan membuat kasus uji untuk menekan beberapa tombol yang berbeda secara terus-menerus atau simultan dan membuat kasus uji yang memungkinkan pengaksesan terhadap dua tombol atau lebih secara bersamaan. Pengujian pada kriteria ini menggunakan fitur registrasi pada Selendroid Test App yang memiliki banyak *form* dan tombol yang harus diisi secara simultan, dan digunakan juga fungsi *Touch Actions* yang terdapat pada Selendroid Test App yang memungkinkan pengguna untuk menyentuh lebih dari satu elemen dalam waktu yang bersamaan.
K.13.	Rancangan pengujian pada kriteria ini adalah dengan menjalankan kasus uji yang memungkinkan layar android diubah menjadi *landscape* atau *portrait* selama melakukan pengujian. Fitur yang diuji pada kategori ini adalah fitur registrasi pada aplikasi Selendroid Test App. Saat menjalankan kasus uji terhadap fitur registrasi, disisipkan kode yang memungkinkan alat pengujian untuk mengubah orientasi layar dari *portrait* menjadi *landscape* atau sebaliknya.
K.14.	Rancangan pengujian pada kriteria ini adalah dengan menjalankan kasus uji pada fitur yang berhubungan dengan penyimpanan data ke *database*. Fitur yang dipilih adalah fitur registrasi dan pengaduan pada aplikasi Siantar Smart City. Fitur ini dipilih karena berhubungan dengan pengaksesan data melalui basis data. Kriteria ini terpenuhi jika alat pengujian berhasil menyimpan data yang dimasukkan selama proses pengujian ke dalam *database.*

4. HASIL DAN PEMBAHASAN

Pada eksperimen yang dilakukan ditemukan hasil yang dijelaskan pada Tabel 2.

Dapat dilihat bahwa hasil dari eksperimen atas kriteria apakah tools dapat digunakan untuk menguji aplikasi Native, Hybrid, dan Web *Application* adalah selendroid mendukung pengujian web *application* karena *tools* ini menyediakan fitur Selendroid Web

Driver *view app* yang merupakan aplikasi yang terintegrasi dengan Selendroid yang dapat membuka aplikasi web secara langsung pada perangkat atau emulator tanpa harus menggunakan bantuan *web-browser* seperti Google Chrome ataupun *web-browser* bawaan Android.

Tabel 2 Hasil Eksperimen

Kategori	Selendroid	Calabash	UI Automator
K.1.	✓	-	-
K.2.	26.1003s	73.0759s	15.6555s
K.3.	✓	✓	18 ke atas
K.4.	✓	✓	✓
K.5.	325.843 K	3,245 K	887,658 K
K.6.	Java	Ruby	Java
K.7.	HTML *format*, *report* pada IDE	*Command line report*	*Report* pada IDE, HTML dan XML
K.8.	✓	✓	✓
K.9.	✓	-	✓
K.10.	✓	✓	✓
K.11.	✓	✓	✓
K.12.	✓	✓	✓
K.13.	-	-	✓
K.14.	✓	✓	✓

Hasil eksperimen atas kriteria waktu yang dibutuhkan alat pengujian untuk mengeksekusi kasus uji diperoleh bahwa Calabash adalah alat uji yang paling lama untuk mengeksekusi kasus uji. Hal ini disebabkan karena Calabash meng-*uninstall* dan menginstalasi kembali aplikasi setiap kali menjalankan sebuah kasus uji. Waktu yang dibutuhkan dalam meng-*uninstall* dan instalasi kembali aplikasi ini dihitung sebagai waktu yang dibutuhkan dalam menjalankan kasus uji. Penggunaan UI *Automator* lebih efisien dikarenakan hasil pengujian terdapat pada *command prompt*, tidak menggunakan memori yang besar untuk menjalankannya dan membuat waktu yang lebih efisien. Adapun waktu yang tertera pada tabel adalah rata-rata waktu yang dibutuhkan oleh masing-masing alat uji, didapatkan dengan menjumlahkan semua waktu yang dibutuhkan dibagi jumlah fitur.

Hasil dari eksperimen dari kriteria apakah alat uji tersedia untuk lima versi android terbaru yaitu Selendroid dan Calabash dapat digunakan untuk menguji aplikasi android pada android versi dibawah 16-18 hingga 24-25. Sementara UI Automator sudah ditetapkan hanya dapat digunakan pada android versi 18 ke atas.

Hasil dari eksperimen dari kriteria sumber dalam mengeksekusi kasus uji pada masing-masing alat uji yaitu memori yang digunakan oleh UI *Automator* sangat besar hal ini dikarenakan UI *Automator* dijalankan dengan Android Studio yang membutuhkan kapasitas memori yang besar. Selendroid membutuhkan memori yang lebih kecil dari UI *Automator* karena dijalankan dengan mengunakan program IDE Eclipse. Calabash adalah tool yang menggunakan memori paling kecil, hal ini dikarenakan Calabash dijalankan menggunakan *command prompt*. Adapun waktu yang tertera pada tabel adalah rata-rata sumber memori yang dibutuhkan oleh masing-masing alat uji, didapatkan dengan menjumlahkan semua sumber memori yang dibutuhkan dibagi jumlah fitur.

Hasil dari eksperimen atas kriteria bahasa pemrograman yang digunakan adalah bahasa pemrograman dari Selendroid adalah Java, Calabash adalah Ruby, dan UI Automator adalah Java. Bahasa pemrograman tersebut merupakan bahasa pemrograman default atau bawaan dari masing-masing alat uji.

Hasil dari eksperimen atas kriteria dokumentasi hasil eksekusi kasus uji dari setiap alat uji adalah masing-masing alat uji menampilkan dokumentasi atau report hasil pengujian dalam bentuk atau format yang berbeda-beda. Selendroid menampilkan hasil eksekusi kasus uji dalam format HTML dan laporan pada IDE, Calabash menampilkan hasil eksekusi kasus uji dalam format HTML dan laporan pada *command line*, dan UI *Automator* menampilkan hasil eksekusi kasus uji dalam format HTML, XML dan report pada Android Studio.

Hasil dari eksperimen dari kriteria apakah alat uji mendukung *delay* selama melakukan pengujian adalah ketiga alat uji mendukung *delay* selama melakukan pengujian. Hal ini terlihat ketika melakukan pengujian menggunakan ketiga alat uji Selendroid, UI *Automator* dan Calabash, proses pengujian dapat dihentikan sementara waktu kemudian eksekusi kasus uji dilanjutkan kembali setelah waktu yang ditetapkan.

Hasil dari eksperimen dari kriteria apakah alat uji mendukung *suspend/resume* aplikasi saat melakukan pengujian adalah selendroid mendukung *suspend/resume* aplikasi selama melakukan pengujian demikian halnya dengan UI *Automator* namun Calabash tidak. Ketika melakukan suspend aplikasi saat proses pengujian berjalan, kemudian aplikasi dibuka kembali atau di *resume*, aplikasi menampilkan kondisi atau keadaan yang sama, ketika aplikasi di *suspend*.

Hasil dari eksperimen dari kriteria apakah alat uji dapat melakukan *re-launch* aplikasi saat melakukan pengujian adalah ketiga alat uji mendukung proses *re-launch* aplikasi saat melakukan pengujian. Ketika kasus uji baru dijalankan menggunakan ketiga alat uji, aplikasi yang diuji benar-benar dibuka dan dimulai kembali dari awal.

Hasil dari eksperimen dari kriteria apakah alat uji dapat melakukan *scrolling* saat melakukan pengujian adalah Selendroid, Calabash dan UI *Automator* dapat melakukan *scrolling* selama melakukan pengujian. Saat melakukan *scrolling*, dijalankan kasus uji dimana alat uji harus melakukan *scrolling* hingga menemukan kata kunci yang ditetapkan, yang terdapat pada sebuah halaman yang lebih dari satu

halaman penuh android. Dan ketiga alat uji dapat menjalankan kasus uji tersebut dengan melakukan *scrolling*.

Hasil dari eksperimen dari kriteria apakah alat uji mendukung *simultaneous key presses* atau *multipletouch* saat melakukan pengujian adalah ketiga alat uji mendukung *simultaneous key presses*. Ketiga alat uji menjalankan pengujian terutama terhadap fitur yang memiliki banyak elemen seperti form secara simultan sesuai urutan yang diatur pada kode *test-case* satu per-satu. Dan dapat melakukan simulasi untuk pengaksesan dua elemen dalam waktu yang bersamaan.

Hasil dari eksperimen dari kriteria apakah alat uji mendukung *multiple display format* dalam melakukan pengujian adalah pada saat melakukan perubahan orientasi layar selama melakukan pengujian, Selendroid dapat mengubah orientasi layar menjadi *landscape* namun terjadi kegagalan selama proses pengujian. Calabash tidak dapat mengubah orientasi layar menjadi *landscape*. UI *A*utomator dapat mengubah orientasi layar menjadi *landscape* dan melakukan pengujian pada format *landscape* tersebut.

Hasil dari eksperimen dari kriteria apakah perubahan yang dilakukan selama melakukan pengujian menggunakan alat uji dapat tersimpan adalah ketiga alat uji menyimpan setiap perubahan yang dilakukan selama proses pengujian. Hal ini dapat dilihat dari masuk atau tidaknya data yang dimasukkan atau diubah ke dalam basis data. Dan ketiga alat uji dapat melakukan penyimpanan maupun perubahan terhadap data yang dilakukan selama proses pengujian.

Selama melakukan pengujian menggunakan ketiga alat uji kami menemukan bahwa dalam hal penanganan ketika terjadi kegagalan internal pada aplikasi yaitu saat menangani masalah *application not responding*, Selendroid bukanlah alat uji yang baik, karena akan terjadi kegagalan berupa *command line* yang terus-menerus dieksekusi oleh Selendroid dan tidak muncul laporan pengujian ketika terjadi kegagalan internal pada aplikasi selama proses pengujian menggunakan *tool* Selendroid. Tapi hal ini berbeda dengan Calabash dan UI *Automator*, proses pengujian akan terhenti ketika terjadi kegagalan internal pada aplikasi selama proses pengujian.

Dari hasil analisis maka alat uji yang paling disarankan untuk digunakan adalah UI *Automator*. Hal ini dikarenakan cara instalasi dan cara menjalankan kasus uji yang lebih mudah dan sederhana dibandingkan Selendroid dan Calabash. Penggunaan UI Automator cukup dengan membuat sebuah proyek android baru pada Android Studio kemudian, mengimpor sebuah koleksi program yaitu *library* UI Automator pada proyek tersebut. Selendroid memiliki banyak keterikatan *file* yang harus diintegrasikan ke dalam eclipse demikian halnya juga pada Calabash. Cara menjalankan kasus uji pada UI *Automator* juga paling cepat yaitu dengan mengkonfigurasikan *test-case* mana yang akan dijalankan, kemudian cukup dengan me-*run* proyek android tersebut.

Dari segi waktu UI *Automator* menjalankan kasus uji lebih cepat dibandingkan Selendroid dan Calabash. UI Automator juga mendukung *multiple display format* lebih baik dibandingkan Selendroid dan UI *Automator*. Pada kriteria *resource* memori yang digunakan UI *Automator* menggunakan memori jauh lebih banyak dibandingkan Selendroid dan Calabash. Hal ini dikarenakan UI *Automator* berjalan pada program Android Studio yang menggunakan banyak resource memori. Kekurangan lainnya adalah tidak available terhadap android dibawah versi 18 dan web *application*. Namun dari segi versi yang disupport tidak menjadi masalah, karena saat ini API android yang paling banyak digunakan adalah API android versi 18 ke atas.

5. KESIMPULAN DAN SARAN

Setiap alat uji (*tools)* memiliki kelebihan dan kekurangan masing-masing. Hal ini dapat dilihat dari banyaknya kriteria yang dipenuhi oleh masing-masing alat uji. Dan dari segi banyaknya kriteria yang terpenuhi, maka Selendroid adalah alat uji yang paling banyak memenuhi kriteria yang ditetapkan.

Dari dua kriteria utama, waktu dan penggunaan memori, penelitian ini menemukan UI Automator adalah alat uji yang membutuhkan waktu eksekusi paling cepat dan Calabash adalah alat uji yang membutuhkan memori paling sedikit.

Penelitian ini merekomendasikan UI Automator dibandingkan alat uji lainnya. Hal ini dikarenakan instalasi serta cara menjalankan *test case* yang lebih mudah dibandingkan Selendroid dan Calabash. Fitur ini yang terutama dibutuhkan oleh penguji dalam menguji sebuah aplikasi Android.

DAFTAR PUSTAKA

ALAQAIL, H., & AHMED, S. (2018). Overview of Software Testing Standard ISO/IEC/IEEE 29119. IJCSNS International Journal of Computer Science and Network Security, 18(2), 112-116.

AQUA. (2013). Testing Criteria for Android Applications, 1.4, dari App Quality Alliance: AQuA App Quality Alliance. Tersedia di: <https://www.appqualityalliance.org/files/AQuA_testing_criteria_for_Android_for_v1.4%20final%207_feb_2013.pdf> [Diakses 1 Oktober 2017]

BAJAJ, H. (2015). Selecting the Right Mobile Test Automation Strategy: Challenges and Principles. Infosys. Bangalore, India: Infosys.

BENTLEY, J. E., BANK, W., & NC, C. (2005). Software Testing Fundamentals—Concepts,

Roles, and Terminology Paper 141-30. SUGI 30 Proceedings (hal. 1-12). Philadelphia, Pennsylvania: SAS Institute Inc.

COLLINS, C., GALPIN, M. D., & KAEPPLER, M. (2011). Android in Practice. (C. Kane, Penyunt.) Shelter Island: Manning Publications Co.

DARY, D. (2015). Selendroid. Tersedia di: <http://selendroid.io> [Diakses 10 Nopember 2017]

DEVELOPERS, G. (2018, May 8). Automate user interface tests. (Google). Tersedia di: <https://developer.android.com/training/testing/ui-testing/index.html> [Diakses 20 Juni Oktober 2017]

GHAHRAI, A. (2017, Oktober 2017). 10+ Open Source Mobile Test Automation Tools. Tersedia di: <https://www.testingexcellence.com/open-source-mobile-test-automation-tools/> [Diakses 20 Januari 2018]

HELPPI, V.-V. (2016). Helppi. Tersedia di: <https://bitbar.com/calabash-tutorial-for-mobile-app-testing/> [Diakses 11 November 2016]

INTERACTIVE, C. (2016). Android leads mobile Osusage in Singapore and the Asia Pacific region. Asian Technology. Tersedia di: <http://www.techrepublic.com/blog/asian-technology/> [Diakses 11 November 2016]

JAGADALE, N. J., PAGAR, A. G., DEORE, M., RAUT, P. V., & KUMAVAT, P. K. (2015). GUI Testing On Android Application. International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC), 3(1), 67-71.

JINDAL, T. (2016). Importance of Testing in SDLC. International Journal of Engineering and Applied Computer Science (IJEACS), 01(02), 54-56.

KAUR, A. (2015). An Essential Guide to Automated GUI Testing Of Android Mobile Applications. International Journal of Computer Techniques, 2(6), 8-12.

KHAN, S. N., & FIRDOUS, I. U. (2017). Review on Android App Security. International Journal of Advanced Research in Computer Science and Software Engineering, 7(4), 225-228.

MYERS, G. J. (2004). The Art of Software Testing (2nd ed.). (B. a. Sandler, Penyunt.) Hoboken, New Jersey: John Wiley & Sons, Inc.

SINAGA, A. M. (2013). On Feedback-Based Software Testing. Univerity of Wollongong, Wollongong. Australia: Computer Science Department.

SMARTBEAR. (2016). What is Automated Testing?. Tersedia di: <https://smartbear.com/learn/automated-testing/what-is-automated-testing/> [Diakses 11 November 2016]

STATISTA. (2016). Smarthphone Worldwide Installed base Operating System. Tersedia di: <https://www.statista.com/statistics/385001/smartphone-worldwide-installed-base-operating-systems/> [Diakses 11 November 2016]

VENKATASUBRAMANIAN, R. (2015). Selecting the Right Mobile Test Automation Strategy: Challenges and Principles. Cognizant. New Jersey: Cognizant.