Pengujian Tingkat Komponen dan Implementasi Pengujian Mutasi pada Aplikasi Berbasis Web: Studi Kasus Aplikasi General Reporting Universitas Telkom

Muhammad Rifqi Thomi Faiz Hawari[1], Dana Sulistyo K[2], Yanuar Firdaus[3]

[123]Fakultas Informatika, Universitas Telkom, Bandung

[1]rifqithomi@students.telkomuniversity.ac.id, [2]danakusumo@telkomuniversity.ac.id, [3]yanuar@telkomuniversity.ac.id

Abstrak

Pengujian Tingkat Komponen memastikan komponen yang terdapat pada fungsionalitas aplikasi General Reporting pada Universitas Telkom bekerja sesuai dengan spesifikasi. Tingkat komponen pada aplikasi berbasis web merupakan gabungan dari fungsi yang memanipulasi konten, komputasi, pemrosesan data untuk disajikan kepada end-user [8]. Untuk mencapai pengujian yang adequate, maka dilakukan pengujian Mutasi. Pengujian Mutasi memiliki ukuran adequacy suatu pengujian dengan rasio antara program mutant terdeteksi dengan total mutant yang tertanam pada aplikasi uji, hal ini dapat meningkatkan efektifitas dalam menemukan defect pada kegiatan pengujian.

Fungsionalitas Add Komponen Judul, Edit Komponen Judul, dan Delete Komponen Judul pada Aplikasi General Reporting diuji dengan menggunakan Selenium IDE. Pada pengujian Tingkat Komponen ditemukan defect pada fitur Logout. Test Case yang diturunkan pada pengujian Tingkat Komponen tidak mendeteksi program mutant yang ditanam oleh penguji. Maka penambahan test case baru terjadi pada penelitian ini.

Kata kunci: pengujian tingkat komponen, *adequacy*, pengujian mutasi, program *mutant*

Abstract

Component-Level Testing ensure that component contained in General Reporting Application should work properly in accordance with the specification. A web application component is combination of functions that manipulates content, computation, and data processing to be presented to the end-users [8].To achieve an adequate testing, Mutation testing will beconducted in this study.Mutation Testing will increase an effectiveness of finding defect in test activity. Mutation Testing also called mutation analysis has an adequacy metric of a test byratio from comparing the number of detected mutant with a total of mutants exist.

The functionality of Add Komponen Judul, Edit Komponen Judul, and Delete Komponen Judul in General Reporting application will be tested in this study. Defect found in Logout feature when Component-Level testing was conducted. Test case for Component-Level testing cannot detect the mutated program which is planted by the tester. So, the addition of a new test case will occur in this study.

Keywords: component-level testing, *adequacy*, mutation testing, mutated program

1. Pendahuluan

1.1. Latar Belakang

Kualitas perangkat lunak jika dilihat dari sisi user adalah perangkat lunak yang sesuai dengan spesifikasi kebutuhan yang diinginkan oleh user [8]. Pengujian merupakan salah satu elemen yang dilakukan dalam kegiatan penjaminan mutu perangkat lunak untuk menjaga kualitas perangkat lunak yang dikembangkan [8]. Kegiatan pengujian mencari defect yang mungkin terdapat pada perangkat lunak. Dalam hal pencarian defect, tidak semua defect dapat ditemukan. Hal ini dapat mempengaruhi efektifitas sebuah pengujian dalam menemukan defect [5].

Sistem Informasi Universitas Telkom sedang melakukan pengembangan perangkat lunak bernama General Reporting. Aplikasi ini merupakan aplikasi berbasis web yang dibangun dengan bahasa PHP: Hypertext Preprocessor (PHP). Aplikasi ini mendukung pengguna agar mendapatkan informasi

dari data – data kegiatan akademik di Universitas Telkom. General Reporting menghasilkan informasi yang dinamis, pengguna dapat menentukan data yang ingin dipakai untuk menghasilkan informasi tersebut. Aktor yang menggunakan aplikasi ini adalah administrator aplikasi dan pengguna laporan sebagai end-user. Aktor administrator aplikasi dapat melakukan kelola query laporan, kelola kolom tabel laporan, dan kelola hak akses laporan. Sedangkan, aktor pengguna laporan dapat melakukan view laporan, kostumisasi kolom tabel laporan, dan mencetak laporan.

Pihak pengembang aplikasi General Reporting belum pernah melakukan kegiatan pengujian, maka belum diketahui juga aplikasi ini terdapat defect atau tidak. Oleh karena itu, perlu diadakan kegiatan pengujian agar spesifikasi yang dibutuhkan oleh pengguna tercapai. Aplikasi ini dilakukan pengujian Tingkat Komponen atau disebut juga Function Test. Target pada pengujian ini memastikan fitur dan behavior yang disediakan oleh aplikasi sesuai dengan spesifikasi kebutuhan, seperti menguji validasi pada form, HTML/CSS, konsistensi data di database ketika melakukan edit, delete, create, dan kegiatan lainnya yang berhubungan dengan database [7]. Untuk menjalankan kegiatan Function Test dapat menggunakan teknik uji black-box [8]. Teknik uji black-box memastikan output sesuai dengan requirements tanpa mengetahui struktur kode internal pada aplikasi.

Selain melakukan Functional Test, dilakukan pengujian Mutasi atau Mutation Test. Merujuk kepada efektifitas pada kegiatan pengujian perangkat lunak, pengujian mutasi dapat meningkatkan efektifitas pada test suite dengan menggunakan metriks adequate, memasukkan *defect* pada kode program (mutasi) untuk menguji kemampuan test suite saat melakukan pengujian [9].

Ada beberapa alat uji otomatis yang dapat menguji perangkat lunak berbasis web, salah satunya Selenium IDE. Selenium IDE merupakan alat uji otomatis untuk melakukan Functional Test pada aplikasi web dengan cara merekam aktifitas fungsionalitas web untuk dijadikan test case lalu hasil rekaman tersebut dapat disimpan ke dalam beberapa bahasa, seperti HTML, Ruby scripts, C#, dan lainnya. Pada penelitian ini, Selenium IDE digunakan untuk membuat test case dengan cara merekam aktifitas yang dapat dilakukan oleh fitur – fitur yang disediakan aplikasi General Reporting untuk diuji.

1.2. Perumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, beberapa rumusan masalah dapat simpulkan sebagai berikut:
RQ 1. Bagaimana pengujian Tingkat Komponen menemukan defect menggunakan automation tools?
RQ 2. Bagaimana pengujian Mutasi dapat meningkatkan efektifitas kegiatan pengujian dalam menemukan defect dengan melihat pertambahan jumlah test case saat melakukan pengujian Mutasi?

1.3. Tujuan

Berdasarkan rumusan masalah yang telah dirumuskan, tujuan yang ingin dicapai dari penelitian ini adalah:

1. Mengimplementasikan pengujian Tingkat Komponen pada aplikasi berbasis Web sebagai bahan analisis dalam menemukan defect.
2. Menganalisis hasil pengujian Tingkat Komponen untuk melihat fungsionalitas aplikasi sesuai dengan spesifikasi.
3. Mengimplementasikan pengujian Mutasi sebagai peningkatan efektifitas *test suite* dalam menemukan *defect* pada aplikasi General Reporting.
4. Menganalisis hasil pengujian Mutasi dengan melihat jumlah *test case* saat melakukan pengujian Mutasi.

1.4. Batasan Masalah

Batasan – batasan yang diteliti agar tidak meluasnya pembahasan, sebagai berikut:

1. Aplikasi yang diuji adalah aplikasi web General Reporting Universitas Telkom.
2. Fungsionalitas yang diuji hanya fungsionalitas Add Komponen Judul, Edit Komponen Judul, dan Delete Komponen Judul yang terdapat pada dokumen *requirement* yang diberikan oleh pihak pengembang.
3. Penelitian tidak memperhitungkan penggunaan rangka kerja / *framework* yang digunakan pada aplikasi General Reporting

1.5. Metode Penelitian

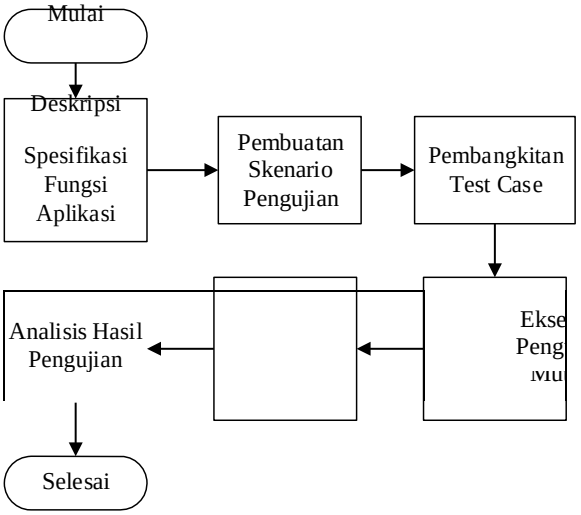
Metode yang dilakukan untuk dapat menyelesaikan penelitian, memiliki beberapa langkah sebagai berikut:

1. Melakukan studi literatur
2. Mempersiapkan dokumen *requirement* aplikasi General Reporting.

3. Mendeskripsikan beberapa fungsionalitas dari aplikasi General Rerporting.
4. Menurunkan *test case* menggunakan metode Boundary Value Analysis.
5. Implementasi *test case* menggunakan Selenium IDE untuk Functional Test.
6. Melakukan eksekusi Functional Test menggunakan Selenium IDE pada setiap fungsionalitas yang sudah dideskripsikan.
7. Jika eksekusi Functional Test selesai, maka eksekusi Mutation Test bisa dilakukan.
8. Melakukan analisa terhadap tiap hasil pengujian.

2. Metode dan Rancangan Pengujian

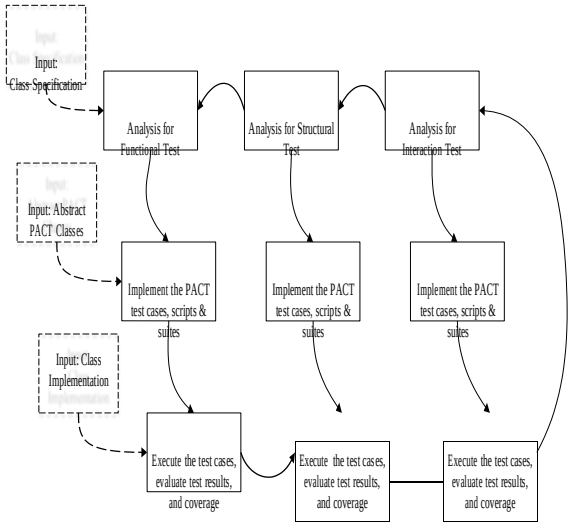
2.1. Metode Pengujian



Gambar 1: Diagram Blok Metode Pengujian

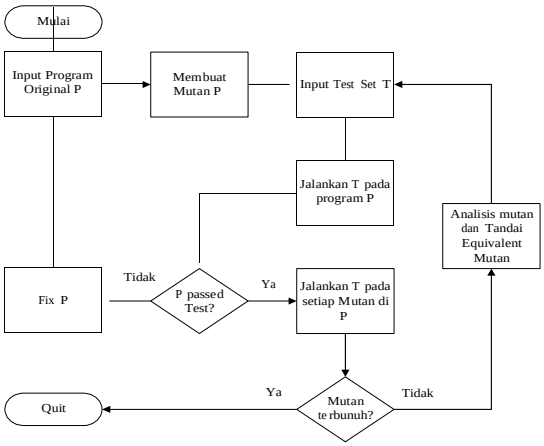
Penelitian menggunakan kombinasi dua metode pengujian, yakni metode Pengujian Tingkat Komponen dan metode Pengujian Mutasi. Pengujian Tingkat Komponen terlebih dahulu dieksekusi lalu dilanjutkan dengan pengujian Mutasi.

Pada pengujian Tingkat Komponen, pengujian memiliki beberapa tahapan atau fase sesuai dengan yang dikemukakan oleh John D. McGregor seperti pada gambar 2, yaitu pertama Analysis for Functional Test yang berisi tentang review dokumen spesifikasi kebutuhan perangkat lunak dalam penelitian ini ialah aplikasi General Reporting. Kedua, Implement the PACT test cases, scripts & suites. Parallel Architecture for Component Testing (PACT) merupakan sebuah arsitektur pengujian yang dapat membantu dalam pengaturan pembuatan *test case* pada pengujian Tingkat Komponen yang berisi pembuatan skenario pengujian dan pembangkitan *test case* yang berfokus komponen suatu aplikasi. Ketiga, execute the test cases, evaluate test results, and coverage yang berisi eksekusi pengujian, dan analisis hasil pengujian [4]. Pada penelitian ini, fase pengujian tidak meliputi kegiatan Structural Test dan Interaction Test.



Gambar 2: Fase Pengujian Tingkat Komponen [4]

Pada pengujian Mutasi, diagram blok pada proses ekseskusi pengujian Mutasi pada gambar 1 dipeluas sesuai dengan penelitian yang dilakukan oleh Yui Jia dan Mark Harmann [9] seperti pada gambar 3. Test set yang dipakai dalam pengujian Mutasi merupakan test set yang telah dibuat saat melakukan pengujian Tingkat Komponen.



Gambar 3: Flow Diagram Eksekusi Pengujian Mutasi [9]

2.2. Review Dokumen Spesifikasi Kebutuhan Perangkat Lunak Aplikasi General Reporting

Pada tahap ini, dilakukan wawancara, mempelajari dokumen, dan diskusi dengan pihak pengembang terkait dengan penentuan fungsionalitas yang diuji dan kegiatan pengujian yang dilakukan pada penilitian ini.

Tahap ini menghasilkan beberapa hal sebagai berikut:

1. Mendapatkan dokumen berupa SRS aplikasi General Reporting.
2. Mendapatkan dokumen presentasi print screen beberapa fungsionalitas aplikasi General Reporting.
3. Fungsionalitas yang baru dibangun hanya Add Komponen Judul, Edit Komponen Judul, Delete Komponen Judul.

Dari hasil wawancara diatas, maka pengujian dilakukan hanya pada fungsionalitas yang sudah dibangun, seperti Add Komponen Judul, Edit Komponen Judul, dan Delete Komponen Judul.

2.3. Skenario Pengujian

Skenario pengujian yang digunakan untuk pengujian Tingkat Komponen mengikuti langkah – langkah yang dilakukan oleh pengguna dalam menggunakan fungsionalitas aplikasi [2].

Dalam hal ini skenario pengujian sesuai dengan fungsionalitas yang diberikan aplikasi seperti Add Komponen Judul dengan memastikan Komponen Judul yang baru dibuat mengakibatkan daftar tabel laporan bertambah, Edit Komponen Judul dengan memastikan Komponen Judul yang baru di-edit mengakibatkan title laporan berubah., dan Delete Komponen Judul dengan Memastikan Komponen Judul yang dihapus tidak terdapat pada daftar tabel laporan.

Test case dibawah ini digunakan pada pengujian Tingkat Komponen dan pengujian Mutasi. Dalam pembangkitan *test case* untuk pengujian Tingkat Komponen, digunakan metode *Bundary Value Analysis*. Metode *Bundary Value Analysis* membagi dua kelas *input*, yaitu *input* yang bernilai *valid* dan *input* yang bernilai *invalid*. Masing – masing dari kelas tersebut diambil dari batas *value input* untuk dijadikan *test case*. Terdapat 4 *test case* yang dibangkitkan pada pengujian ini dengan nama sesuai dengan *input data valid* atau *invalid*, yaitu Test Add Komponen Judul – valid, Test Add Komponen Judul – invalid, Test Edit Komponen Judul – valid, Test Delete Komponen Judul – valid. *Test case* yang dibangkitkan sesuai dengan alur penggunaan aplikasi dengan memastikan tiap *input* yang menjadi *input data* dari tiap test case dapat ditangani oleh aplikasi dan tiap komponen aplikasi (link, form, HTML/CSS, text) sesuai dengan spesifikasi.

2.4. Test Environment

Karena tidak diizinkan untuk memirrorkan source code aplikasi General Reporting, maka pengujian dilakukan di komputer pengembang. Test environtment terdiri dari komputer yang digunakan untuk melakukan pengujian dan aplikasi General Reporting sebagai objek yang ingin diuji.

2.5. Script Otomasi Pengujian

Script pengujian merupakan hasil rekaman dari aplikasi Selenium IDE yang sesuai dengan perilaku user dalam menggunakan fungsionalitas aplikasi General Reporting. Hasil rekaman tersebut disimpan berupa tag HTML yang berisi command pengujian yang disebut *selenese*.

2.6. Rencana Eksekusi Pengujian

Pengujian Tingkat Komponen dilakukan sesuai dengan skenario pengujian, namun tiap skenario dilakukan pengujian Mutasi. Misal, dilakukan pengujian pada skenario "Add Komponen Judul", setelah pengujian Tingkat Komponen selesai dilakukan, selanjutnya langsung dilakukan pengujian Mutasi terhadap skenario uji tersebut, begitu selanjutnya terhadap skenario uji yang lain.

Pertambahan jumlah *test case* terjadi ketika *test case* yang digunakan untuk pengujian mutasi tidak dapat mendeteksi *program mutant* yang ditanam, yang bertujuan untuk membantu efektifitas menemukan *defect* dalam kegiatan pengujian.

2.7. Persiapan Alat Uji

Alat uji yang digunakan merupakan sebuah komputer (PC) yang memiliki aplikasi General Reporting dan perangkat lunak yang mendukung otomasi pengujian, yaitu Selenium IDE

Komputer yang digunakan sebagai alat uji memiliki spesifikasi sebagai berikut:

- Processor : Intel Core i3-2100 @3.10GHz (4CPUs).
- Memory : 3072 MB RAM.
- OS : Windows Technical Preview x64.
- Display : ATI Radeon HD 4600 Series 1786MB.

Perangkat lunak yang digunakan untuk melakukan pengujian otomatis, sebagai berikut:

- Browser Mozilla Firefox v. 33.1
- Selenium IDE v. 2.8.0

2.8. Persiapan Aplikasi Uji

Aplikasi yang dijadikan bahan uji ialah aplikasi General Reporting. Aplikasi dibagi menjadi 2 kategori, yaitu aplikasi original atau P dan aplikasi yang telah diberikan operator mutasi (P'). Pengujian Tingkat Komponen menguji aplikasi original P, sedangkan pengujian Mutasi menguji aplikasi yang telah diberikan operator mutasi (P').

3. Analisis Hasil Pengujian

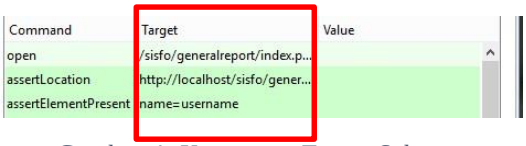
RQ 1. Bagaimana pengujian Tingkat Komponen menemukan defect menggunakan automation tools?

Pada pengujian Tingkat Komponen, penguji membangkitkan *test case* mengikuti behavior atau fungsionalitas yang dimiliki oleh aplikasi General Reporting. Tiap *test case* tersebut memastikan tiap komponen aplikasi (link, form, HTML/CSS, text) sesuai dengan spesifikasi, dengan Selenium IDE, komponen tersebut dapat dipastikan dengan menambahkan attribute komponen pada *command selense* sebagai target, seperti pada gambar 4.

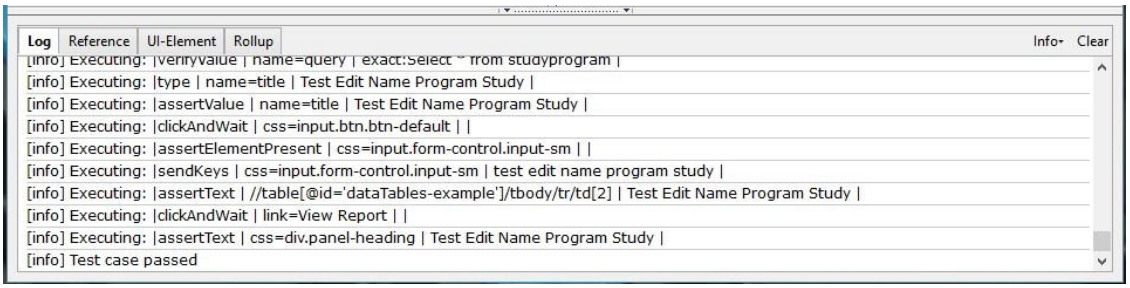
Test case yang dibangkitkan dijalankan secara otomatis dengan Selenium IDE, *test case* yang dieksekusi terlihat pada log seperti gambar 5. Ada 2 warna indicator yang menandakan tiap *command selenese* bekerja atau tidak, yaitu hijau dan merah. Warna hijau jika *command selenese* bekerja, sedangkan merah jika terjadi *error* pada *command selenese* yang menandakan ada *defect* pada aplikasi, seperti pada gambar 6 dan gambar 7.

Test case yang digunakan dalam pengujian Tingkat Komponen yaitu, Test Add Komponen Judul – valid, Test Add Komponen Judul – invalid, Test Edit Komponen Judul valid, Test Delete Komponen Judul – valid. Pada *test case* Test Add Komponen Judul – invalid terjadi kekeliruan pada penguji karena tidak terdapat *error* handle pada fungsionalitas Add Komponen Judul, oleh karena itu penguji mengalihkan pengujian dengan menguji keluaran hasil fungsionalitas Add Komponen Judul (bukan komponen form).

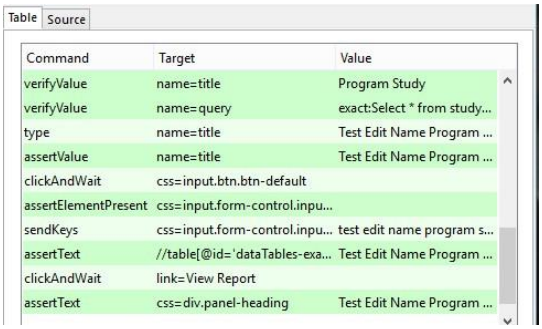
Pada pengujian Tingkat Komponen juga ditemukan *defect* pada fungsionalitas Logout, bahwa fungsionalitas ini hanya berfungsi pada halaman muka aplikasi saja, pada halaman saat menggunakan fitur Add Komponen, Edit Komponen, dan View Komponen fungsionalitas Logout tidak bekerja dengan baik. Tabel 1 merupakan ringkasan *test case* dan hasil pengujian Tingkat Komponen.



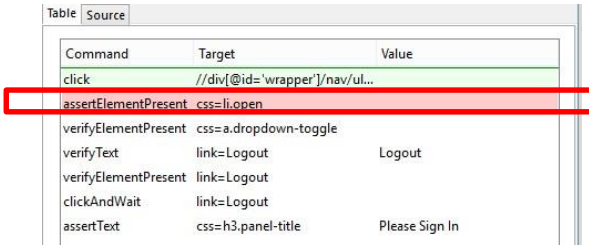
Gambar 4: Komponen Target Selenese



Gambar 5: Log Eksekusi Test Case



Gambar 6: Command Selenese Bekerja



Gambar 7: Command Error

Tabel 1: Ringkasan Test Case dan Hasil Pengujian Tingkat Komponen.

Nama	Valid Input	Invalid Input	Expected Result	Actual Result	Test
Test Add Komponen Judul – Valid	Form Filled	-	New Komponen added	New Komponen added	Pass
Test Add Komponen Judul – Invalid	-	Empty Form	New Komponen Error To View	New Komponen Error To View	Pass
Test Edit Komponen Judul – Valid	New Title	-	Title Changed	Title Changed	Pass
Test Delete Komponen Judul – Valid	Click Delete Menu	-	Komponen Deleted	Komoponen Deleted	Pass
Logout	Click Logout Menu		Return to Login Form	Return to Login Form but not at all Feature	Not Pass

RQ 2. Bagaimana pengujian Mutasi dapat meningkatkan efektifitas kegiatan pengujian dalam menemukan *defect* dengan melihat pertambahan jumlah *test case* saat melakukan pengujian Mutasi?

Pada pengujian Mutasi, *test case* yang digunakan pada pengujian Tingkat Komponen belum mampu mendeteksi program *mutasi* yang ditanam, maka dari itu penguji menambahkan *test case* yang memastikan program *mutasi* yang ditanam dapat terdeteksi sebagai *defect*. *Test case* yang ditambahkan pada pengujian Mutasi yaitu Link add check destination (WLR), Action insert destination (WFR), Transfer method add check (WTR), link edit check destination (WLR), action edit destination (WFR), transfer method edit check (WTR), dan link delete check destination (WLR).

Pada pengujian ini, semua program *mutant* yang ditanam oleh penguji dapat terdeteksi, artinya dari total 7 program *mutant* terdeteksi 7 *mutant*, maka pengujian ini termasuk pengujian yang *adequate* dengan memiliki rasio antara program *mutant* terdeteksi dan total program *mutant* yang ditanam sama dengan 100%. Tabel 2 merupakan ringkasan dari program *mutant* yang ditanam.

Tabel 2: Ringkasan Program Mutant

File PHP	Mutant			Total	Live	Killed	Score
	WLR	WFR	WTR				
index	3	0	0	3	0	3	1
insert	0	1	1	2	0	2	1

update	0	1	1	2	0	2	1
Total	3	2	2	7	0	7	100%

Pada penelitian ini, pengujian Mutasi memungkinkan pertambahan *test case* untuk meningkatkan efektifitas pengujian dalam mendeteksi *defect*, ketika *test case* yang digunakan pada pengujian Tingkat Komponen tidak mampu mendeteksi program *mutant* yang ada, sehingga dibutuhkan *test case* yang mampu mendeteksi program *mutant* tersebut. Seperti pada tabel 3 terlihat bahwa pengujian terjadi pertambahan *test case* pada pengujian Mutasi sebanyak 7 *test case*, sehingga pada pengujian ini total *test case* menjadi sebanyak 12 *test case.*

Tabel 3: Jumlah Test Case Pengujian

Nama Pengujian	Jumlah Test Case
Pengujian Tingkat Komponen	5
Pengujian Mutasi	7
Total	12

4. Kesimpulan dan Saran

Kesimpulan yang dapat ditarik dari penilitian ini adalah sebagai berikut:

1. Pengujian Tingkat Komponen menggunakan Selenium IDE memastikan komponen (link, form, HTML/CSS, text) yang terdapat pada aplikasi sesuai dengan spesifikasi dengan cara menjadikan komponen tersebut sebagai target *command selense*. *Command selenese* dieksekusi secara otomatis oleh Selenium IDE, jika berhasil, maka Selenium IDE memberikan indikator warna hijau, sedangkan jika tidak, maka Selenium IDE memberikan indikator warna merah dan Selenium IDE berhenti mengeksekusi *command selenese* berikutnya.
2. Peningkatan efektifitas dalam kegiatan pengujian dapat dilakukan dengan melakukan pengujian Mutasi, dengan cara menambahkan varian test case baru yang dapat mendeteksi program mutant yang ditanam saat melakukan pengujian Mutasi.

Berikut saran yang mungkin diperlukan saat melakukan penelitian lebih lanjut mengenai pengujian Tingkat Komponen dan pengujian Mutasi:

1. Pastikan tidak terjadi kekeliruan dalam membangkitkan *test case* untuk menguji validasi form. Karena validasi form merupakan salah satu komponen yang memiliki *defect* yang sangat mempengaruhi hasil data pada *database*.
2. Untuk melakukan pengujian Mutasi, pada penelitian ini program *mutant* ditanam dengan cara manual. Untuk aplikasi dengan skala *enterprise* membutuhkan waktu yang cukup lama jika dilakukan secara manual.

5. Daftar Pustaka

[1] A. J. Offut, "A Practical System for Mutation Testing: Help for the Common Programmer," IEEE Test Conference, 1994. Proceedings., International, 1994.

[2] C. Kaner, "An Introduction to Scenario Testing," Florida Institute of Technology, 2013.

[3] G. A. Di Lucca and A. R. Fasolino, "Testing Web-based applications: The state of the art and future trends," 2006.

[4] J. D. McGregor, "A Component Testing Method," [Online]. Available: http://people.cs.clemson.edu/~johnmc/joop/col5/column5.html. [Accessed 2014].

[5] J. Offut and U. Praphamontripong, "Applying Mutation Testing to Web Applications," Third International Conference on Software Testing, Verification, and Validation Workshops, p. 1, 2010.

[6] M. Pezzand and M. Young, "Fault-Based Testing," in Software Testing and Analysis Process, Principles, and Techniques, 2008.

[7] P. T. Test, "A Guide To Web Application Testing," Push To Test, [Online]. Available: http://www.pushtotest.com/guide-to-web-applications-testing. [Accessed 6 December 2014].

[8] R. Pressman, Software Engineering A Practitioner's Approach 7th Edition, New York: The McGraw-Hill Companies, Inc, 2010.

[9] Y. Jia and M. Harman, "An Analysis and Survey of the Development of Mutation Testing," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 2010.