LOGIC: Jurnal Penelitian Informatika
Vol. 2, Issue 1, pp. 61–66 (2024)
doi: http:/doi.org/10.25124/logic.v2i1.7458

RESEARCH ARTICLE

Pengujian Fungsionalitas Aplikasi Menggunakan Keyword-Driven Testing pada Website TEKOS

Dzakwan Daffa Hidayatullah, Mira Kania Sabariah* and Rosa Reska Riskiana

Fakultas Informatika, Universitas Telkom, Bandung, 40257, Jawa Barat, Indonesia
*Corresponding author: mirakania@telkomuniversity.ac.id
Received on 18 April 2024; accepted on 19 May 2024

Abstrak

Perkembangan pesat teknologi informasi membawa perubahan signifikan dalam kehidupan manusia. Banyaknya website yang muncul juga menimbulkan kecemasan terhadap kualitas produk perangkat lunak tersebut. Demi menjaga kualitasnya dan menemukan kesalahan atau bug sejak dini, pengujian perangkat lunak tidak boleh dikesampingkan pada proses pengembangan produk tersebut, salah satunya pengujian fungsional antarmuka website yang dikembangkan. Penelitian ini bertujuan untuk menganalisis kesesuaian fungsional antarmuka pengguna website Tekos dengan persyaratan yang telah ditetapkan dan melakukan pengujian dengan pendekatan Keyword-Driven Testing untuk pengembangan kasus uji secara manual dan terotomasi. Metode Keyword-Driven Testing digunakan untuk menggambarkan aksi pada elemen antarmuka. Solusi ini didukung oleh teknik pengembangan kasus uji dengan metode blackbox testing. 80dari 35 kebutuhan fungsional pencari dan pemilik menjadi cakupan pengujian. Hasilnya 21 kata kunci dipakai pada proses pengembangan kasus uji. Dari 220 kasus uji yang direncanakan 100% berhasil tereksekusi, ditemukan 6 kecacatan terdokumentasi dan 4 diantaranya berhasil diperbaiki. Sehingga 26 fungsionalitas yang diuji dinilai sesuai. Namun, menyisakan dua fungsionalitas yang ditemukan kecacatan fungsional, yakni pada fitur edit kos dan kontrakan. Kedua fitur tersebut tidak mampu menampilkan data yang sebelumnya sudah dibuat, sehingga pengguna harus mengunggah ulang foto produk.

Key words: pengujian antarmuka pengguna, pengujian fungsionalitas, pengujian terotomasi, blackbox testing, keyworddriven testing

Pendahuluan

Perkembangan teknologi informasi yang semakin pesat saat ini membawa perubahan signifikan dalam kehidupan manusia, terutama dalam hal penggunaan aplikasi web. Website semakin penting dalam kehidupan sehari-hari karena akses dan kegunaannya yang relatif mudah. Salah satu yang menentukan kualitas sebuah aplikasi website adalah kualitas dari web user interface (UI). Banyaknya website yang muncul membawa pengaruh yang besar bagi perkembangan informasi, namun pada waktu yang sama banyak insiden kegagalan terhadap teknologi tersebut sehingga menimbulkan kecemasan terhadap kualitas produk perangkat lunak tersebut [1].

Dalam konteks ini, penentuan dan pengimplementasian tuntutan kebutuhan yang jelas pada Spesifikasi Kebutuhan Perangkat Lunak (SKPL) sangatlah penting. SKPL atau biasa disebut Software Requirement Specification (SRS) akan menjadi dokumen acuan kerangka kerja yang menguraikan secara sistematis semua kebutuhan yang diharapkan dari perangkat lunak yang sedang dikembangkan [2]. Dengan demikian, SKPL akan memastikan bahwa fungsionalitas dari perangkat lunak akan memenuhi ekspektasi para pengguna dan pemangku kepentingan.

Antarmuka pengguna merupakan komponen perangkat lunak yang sering berinteraksi dengan end-user, oleh karena itu tahap evaluasi atau pengujian perlu dilakukan [3]. Pengujian dari komponen antarmuka website menjadi salah satu tugas dari proses penjaminan mutu perangkat lunak, yang mana proses pengujian sangat penting dan peran pengujian software tidak boleh dikesampingkan pada lingkup pengembangan perangkat lunak [4]. (Graphical) User Interface test berarti memvalidasi objek dari yang ditampilkan pada antarmuka, memeriksa alur fungsional dari objek antarmuka, serta memverifikasi data yang dihasilkan dari backend dan tampil dalam halaman website [5].

Menurut Anand Nayyar [6], banyak organisasi berpindah haluan menerapkan automated testing, karena hanya memanfaatkan seminimal mungkin waktu dan sumber daya. Alasan tersebut selaras dengan pernyataan Rafiq yakni pengujian harus memastikan kualitas produk, juga harus seefisien mungkin mulai dari waktu, sumber daya hingga biaya [7]. Automated testing yang mana merupakan praktik dari pengujian dengan alat kerja terotomasi sangatlah membantu pengembangan aplikasi untuk lebih efisien dari segi waktu dan sumber daya [1], [6]. Kasus uji akan lebih baik menggunakan bahasa alami, akan tetapi akan

sulit jika akan diotomatiskan dengan mesin [5]. Pendekatan keyword-driven testing (KDT) berdasarkan framework-nya mendukung untuk pengujian otomatis dan memiliki manfaat lebih dalam hal pemeliharaan pengujian, kemudahan penggunaan, dan pemakaian ulang (tidak bergantung pada tools) [8], [9].

Pengembangan kasus uji dengan KDT dibantu dengan pemanfaatan teknik seperti equivalence class partitioning (EQP), decision table (DT), dan exploratory test yang merupakan bagian dari metode black box testing untuk menghasilkan kasus uji. Equivalence class partitioning digunakan untuk menentukan kasus uji yang menjalankan fungsi atau perilaku yang sama pada kelasnya [10]. Tabel keputusan digunakan pada fitur yang memerlukan kombinasi masukan yang dilakukan, pada penelitian milik Saeed, U. et al. [10] ditampilkan perbandingan antara BVA-EQP-DT yang mana tingkat kecanggihan DT dan EQP jauh lebih tinggi dibandingkan BVA. Metode blackbox testing ini berfokus pada perilaku eksternal sistem tanpa memperhatikan implementasi internalnya [10], [11].

Tel-U Kost atau Tekos merupakan sebuah startup bisnis rumah singgah atau indekos untuk mahasiswa dan orang sekitar Universitas Telkom. Dalam proses pengembangannya, tim pengembang Tekos memerlukan adanya pengujian yang menjamin fungsionalitas antarmuka sesuai dengan requirement-nya, salah satunya yakni pengujian fungsional UI website guna menemukan bug atau defect sejak dini. Oleh karena itu pada kegiatan ini akan dilakukan uji fungsionalitas UI pada website Tekos dari sisi pemilik (FR1-n Tekos) dan pencari (FR2-n Tekos), menggunakan pendekatan KDT pada pengembangan kasus uji serta Katalon Studio dalam membantu pengujian terotomasi.

Oleh karena itu, dirumuskan beberapa permasalahan yakni:

1. Bagaimana penerapan keyword-driven testing dalam pengujian fungsionalitas aplikasi website Tekos.
2. Apakah UI website TEKOS sesuai dengan requirements.

Batasan kegiatan dari rumusan masalah tersebut yakni proses desain kasus uji didasarkan pada teknik-teknik blackbox testing seperti equivalent class partitioning, decision table, dan exploratory testing. Fungsi yang akan diuji terbatas pada sisi pemilik yakni, daftar akun, masuk akun, lupa password, keluar akun, membuat data baru kos, memperbarui data kos, menghapus data kos, membuat data kamar, memperbarui data kamar, menghapus data kamar, membuat data baru kontrakan, memperbarui data kontrakan, dan menghapus data kontrakan. Pada sisi pencari terbatas pada fungsi daftar akun, masuk akun, lupa password, keluar akun, pencarian kos dan kontrakan, filter kos dan kontrakan, detail kos dan kontrakan, dan whatsapp pemilik.

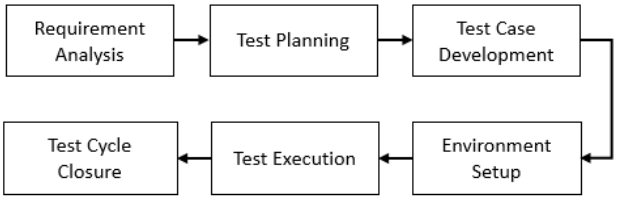
Penelitian ini memiliki tujuan utama yakni menganalisis kesesuaian antara antarmuka pengguna (UI) situs web dengan persyaratan yang ditetapkan untuk Tekos.

Tinjauan Pustaka

Software Testing

Software Testing bila merujuk pada IEEE Standard 1059-1993 mendefinisikan pengujian sebagai proses analisa perangkat lunak untuk memindai kondisi yang keluar untuk dibandingkan dengan requirement perangkat lunak [12]. Semua masalah perangkat lunak bisa didefinisikan sebagai bug, yang mana bug perangkat lunak adalah saat perangkat lunak tidak memberikan hasil yang sesuai dengan ekspektasi [13].

Pengujian perangkat lunak sangat penting untuk diterapkan pada fase pengembangan ataupun pembangunan perangkat lunak [5, 14]. Kegiatan pengujian perangkat lunak mengharuskan fokus pada pengadaan informasi tentang produk dan juga menemukan kemungkinan bug atau defect sebanyak mungkin [15]. Dalam jurnal yang menerangkan STLC [16], terdapat fase pengujian perangkat lunak atau *Software Testing Life Cycle (STLC)* yang merupakan kumpulan aktifitas yang dilakukan oleh seorang penguji, ini meliputi *requirement analysis*, *test planning*, *test case development*, *environment setup*, *test execution*, *test cycle closure*. Proses ini tergambar pada Gambar 1.



Gambar 1. Software Testing Life Cycle (STLC)

Blackbox Testing Method

Blackbox testing merupakan metode perancangan kasus uji berdasarkan analisa dari spesifikasi dari komponen atau sistem tanpa memperhatikan struktur internal aplikasi, dalam kata lain yakni berfokus pada perilaku eksternal sistem [10, 11, 15]. Penerapan strategi pengujian kotak hitam memiliki peran yang signifikan dalam mengidentifikasi potensi kegagalan sistem dan dapat mendukung penyelesaian sistem yang sukses sesuai dengan tujuannya [10]. Sebagai gantinya, pengujian dilakukan berdasarkan spesifikasi fungsi, persyaratan pengguna, dan perilaku yang diharapkan dari sistem. Contoh teknik dari *blackbox testing* yakni [10, 11, 17]:

- *Equivalent class partitioning*, merupakan perancangan kasus uji dengan memilah input dan output yang memiliki fungsi atau perilaku yang sama.
- *Decision table*, merupakan tabel yang memiliki kondisi serta aksi yang dijalankan pada kombinasi yang terjadi untuk membangun kasus uji.
- *Exploratory test*, pengujian yang berbasis pengalaman penguji dalam merancang dan menjalankan pengujian secara spontan.

Automated Testing

Dalam buku *"Instant Approach to Software Testing: Principles, Applications, Techniques, and Practices"*, menjelaskan bahwa pengujian otomatis adalah pemanfaatan alat uji terkomputerisasi guna mengotomatiskan proses pengujian [6]. Disebutkan pula bahwa tahap pengujian bisa mengonsumsi hingga 50% sumber daya proyek, dengan begitu proses pengujian terotomasi menjadi salah satu alasan untuk dilaksanakan (meminimalkan proses, waktu, biaya, dan sumber daya) [6]. Teknik pengujian otomatis menguji perangkat lunak sesuai dengan rencana pengujian yang telah ditentukan sebelumnya dengan meniru prosedur pengujian manual dan menjalankan skrip pengujian dalam bahasa tertentu [1]. Salah satu alat automasi yang dapat digunakan pada pengujian fungsional website yang terotomasi adalah Katalon Studio.

User Interface (UI)

Antar muka pengguna merupakan bagian atau elemen penting di sistem informasi website atau aplikasi yang berinteraksi dengan punggunanya [**?**], [**?**]. Kualitas dari UI biasanya memberikan kesan pertama pengguna (bahkan bisa menjadi yang terakhir). Secara grafis atau visual dan fungsional dari website haruslah memenuhi kebutuhan pengguna dan tidak memberikan kesan buruk. Dalam halnya pengujian web UI, dilaksanakan untuk memastikan bahwa setiap elemen dan fungsi yang tampil pada web tersebut sudah sesuai dengan rancangan desain oleh UI designer dalam arti lain adalah memastikan kebenarannya [18].

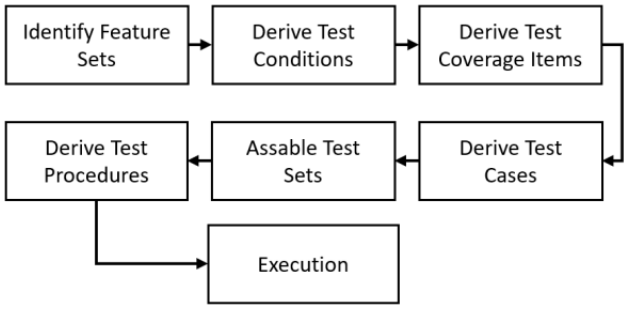
Pengujian antarmuka yang dimaksud yakni menguji alur atau fungsional dari komponen yang ditampilkan oleh aplikasi. Menurut artikel yang dimuat dalam website pengujian (BrowserStack), berikut merupakan beberapa pengujian antar muka yang bisa dilakukan [19], data type error memastikan bahwa hanya data yang valid saja yang bisa masuk kedalam field yang disediakan, field weight memastikan bahwa masukkan di field tidak lebih atau tidak kurang panjang karakternya, navigational elements memastikan semua tombol yang tersedia berjalan sesuai fungsinya, progress bars memastikan bilah kemajuan tersedia jika proses memuat halaman lama, type ahead jika UI web memiliki konten dropdown uji ini diperlukan untuk memastikan jika pengguna mengetik huruf pertama maka daftar akan muncul pertama, table scrolling memastikan pengguliran tabel bisa berjalan jika website menyediakan tabel, error logging memeriksa bahwa jika sistem terjadi kesalahan maka perangkat lunak mencatat detail eror yang terjadi, item menus memastikan apakah perangkat lunak hanya menampilkan menu yang tersedia di lokasi tertentu saja (hanya jika diterapkan), working shortcut jika sistem menerapkan pintasan ini berguna untuk memastikan pintasan berjalan diberbagai perangkat.

Keyword-Driven Testing

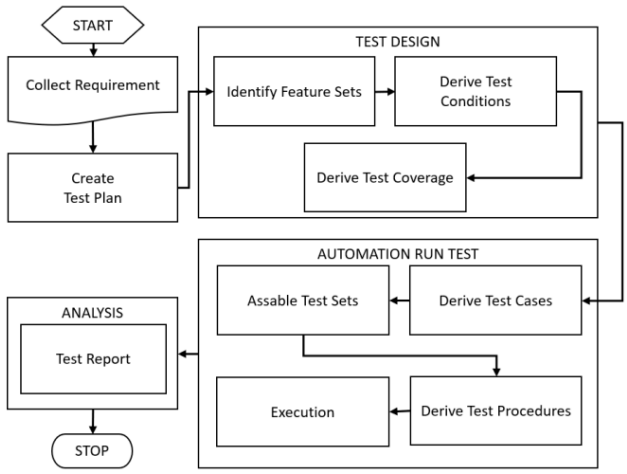
Kerangka kerja berbasis kata kunci (keyword) merupakan pendekatan yang dapat dilakukan untuk pengujian dengan tabel atau kata dalam melakukan aksi untuk mendukung pengujian otomatis [8], [20]. Kerangka kerja ini dikembangkan dalam pengujian terotomasi seperti dengan Selenium Webdriver, dibanding harus menyusun banyak fungsi secara satu per satu yang tentu kurang efisien [20]. Tabel yang dimaksud berisi kolom aksi, objek, input, dan deskripsi untuk setiap kasus ujinya, beberapa keuntungan yang dapat dimanfaatkan saat menggunakan pendekatan kerangka kerja ini yakni, lebih modular dibanding DDT, kata kunci dapat dipakai untuk tes lainnya, tidak bergantung pada bahasa ataupun alat, dan daftar keyword solid untuk perubahan-perubahan kecil pada perangkat lunak [21]. Berdasarkan ISO/IEC/IEEE 29119-5-2016, keyword dibagi menjadi dua tingkatan yakni pada tingkat bawah (kata kunci berhubungan dengan satu atau lebih tindakan yang harus dilaksanakan), dan tingkat tinggi (setiap nama yang bermakna digunakan untuk melacak kata kunci) [8]. ISO/IEC/IEEE 29119-2 proses desain pengujian dan implementasinya menjadi 7 proses yang mana digambarkan secara berurutan namun pada praktiknya bisa diimplementasikan secara iteratif dengan tinjauan pada beberapa prosesnya (lihat GAMBAR 2). Tahapan test design and implementation yang terdapat pada ISO/IEC/IEEE 29119-2 meliput, Identify Feature Set, yang mana pada proses ini adalah tahap memahami fitur yang akan di tes. Derive Test Conditions, tahap ini mendefinisikan setiap kondisi pada fitur yang akan diuji. Derive Test Coverage Items, tahap ini membantu untuk medefinisikan cakupan perilaku selama pengujian terhadap fitur yang diuji. Derive Test Cases, merupakan tahap dimana menyiapkan kasus uji dengan memasukkan seperti pre-kondisi, aksi, nilai/value, dan ekspektasi hasil. Assable Test Sets, mengumpulkan kasus uji menjadi sebuah set pengujian untuk manajemen pengujian lebih terstruktur. Derive Test Procedures, mengatur alur pengujian dari setiap set yang sudah dibuat. Execution, tahap eksekusi terhadap set yang sudah dibuat baik secara manual ataupun terotomasi.

Tel-U Kost (TEKOS)

Berdasarkan definisi pada dokumen software requirement specification (SRS) milik Tekos, Tekos adalah aplikasi berbasis website yang didesain untuk mempermudah pengguna dalam mencari kos disekitar Universitas Telkom. Fungsi utama aplikasi yakni, memberikan informasi tentang kos atau kontrakan, mencari kos dengan beberapa filter, dan melakukan penambahan atau penghapusan atau pengubahan data kos. Cakupan pengujian dalam pengembangan Tekos yakni



Gambar 2. Test design and implementation process



Gambar 3. Framework flow

seperti unit testing, integration testing, performance testing, accessibility testing, dan Usability testing, telah terakomodir oleh tim pengembang. Oleh karenanya, tim memerlukan penguji independen untuk menguji fungsionalitas dari UI yang sudah dibuat pada website Tekos.

Metodologi Penelitian

Metodologi pada penelitian ini menggunakan *keyword-driven testing* yang merupakan pengujian berbasis kata kunci yang mendukung untuk dilakukan pengujian secara terotomasi. KDT dipilih selain mendukung pengujian secara terotomasi juga untuk meningkatkan produktivitas pengujian serta alur desain pengujian yang sangat terstruktur untuk implementasinya. Kata kunci terbagi atas dua tingkatan yakni rendah dan tinggi. Contoh dari kata kunci pada tingkat rendah atau *low* yakni *click*, *setText*, *uploadFile*, dll., sedangkan pada tingkat yang tinggi seperti aksi yang cukup kompleks. Pembangkitan kasus uji dilakukan menggunakan teknik pada *blackbox testing* yakni *equivalent class partitioning*, *decision table*, dan *exploratory testing*.

Katalon Studio yang merupakan alat pengujian terotomasi mendukung penggunaan kata kunci atau aksi yang akan dilakukan terhadap objek penelitian. Berdasarkan metode yang dipakai yakni KDT, oleh karena itu kerangka kerja pengujian pada Gambar 3 mengadopsi proses pada *Software Testing Life Cycle* dan ISO/IEC/IEEE 29119-2 dan 5 bagian *Test Design & Implementation Process*.

Pengujian perangkat lunak yang dilakukan dalam penelitian ini mengikuti langkah-langkah berikut:

1. **Collect Requirement**, pada tahap ini dilakukan analisa dokumen SKPL website TEKOS. Tahap ini bertujuan untuk menggali lebih dalam pengetahuan penguji terhadap produk.
2. **Test Plan**, proses pembuatan dokumen rencana pengujian dilakukan untuk mengorganisasikan pengujian, konsistensi pengujian, rencana pengujian, kriteria pengujian, dan lingkungan pengujian. Hal ini akan menjadi acuan implementasi pengujian kedepannya.
3. **Test Design**, langkah selanjutnya meliputi *Identify Feature Sets*, proses ini bertujuan mengidentifikasi kumpulan fitur (*feature sets*) yang akan diuji berdasarkan batasan masalah dan hasil dari proses sebelumnya. Lalu *Derive Test Conditions*, bertujuan untuk menentukan kondisi fitur yang memungkinkan selama pengujian (*test conditions*). Proses ini berdasarkan kamus data dan prototipe dari desain antarmuka website Tekos itu sendiri. Terakhir pada proses *test design* yakni, *Derive Test Coverage*, untuk menentukan dan mengidentifikasi cakupan pengujian terhadap sistem yang akan diuji.
4. **Automation Run Test**, berdasarkan proses sebelumnya, pada proses ini merupakan implementasi pengujian secara terotomasi, yang memiliki 3 sub-proses yang meliputi *derive test cases*, *assemble test sets*, *derive test procedure*, dan *execution*.
5. **Analysis**, membandingkan hasil pengujian dengan persyaratan dan harapan yang telah ditetapkan. Mengevaluasi kesesuaian perangkat lunak dengan persyaratan yang ada. Mengidentifikasi masalah dan kekurangan yang ditemukan selama pengujian. Memberikan rekomendasi perbaikan atau pengembangan lebih lanjut. Matriks yang dipakai dalam analisis yakni [22].

   a. ***Test Coverage***
   Test coverage menunjukkan sejauh mana fungsionalitas aplikasi tercakup. Dalam pengujian ini sisi yang diuji yakni pencari dan pemilik.

   b. ***Test Tracking and Efficiency***
   Metrik ini menunjukkan nilai dari status pengujian gagal, berhasil, dan cacat yang ditangguhkan untuk rilis mendatang.

Hasil dan Pembahasan

Pengumpulan kebutuhan fungsional ini dilakukan dengan analisis dokumen, yang mana pengambilan data requirement tersebut berasalkan dari dokumen SKPL (Spesifikasi Kebutuhan Perangkat Lunak) milik Tel-U Kost (TEKOS). Functional requirement (FR) yang diambil meliputi FR untuk website pencari dan juga pemilik indekos. Hasil yang didapat yakni FR untuk pencari memiliki total 21 fungsi dan 14 FR pada pemilik. Kumpulan kebutuhan baik fungsional dan kamus data dapat dilihat pada Lampiran SKPL. Matriks keterkaitan antara FR dan set pengujian yang diimplementasikan terletak pada Lampiran 16.

Proses pembuatan rencana pengujian menghasilkan dokumen yang mencakup tujuan pengujian, cakupan pengujian, jadwal, kriteria, lingkungan, dan dokumentasi pasca pengujian. Pembuatan Test Plan didasarkan pada rencana kegiatan yang akan dilakukan, dalam hal ini topik tugas akhir. Kriteria pengujian ditentukan berdasarkan kesepakatan tim pengembangan TEKOS. Pada akhirnya dihasilkan dokumen rencana pengujian (Lampiran 2).

Berdasarkan hasil dari proses sebelumnya, pada proses desain pengujian ini sub-proses identifikasi fitur, kondisi fitur yang diuji, dan cakupan aksi atau items pada fitur yang diuji menghasilkan satu tabel yang memuat ketiga sub-proses tersebut. Lampiran 3 merupakan hasil dari kumpulan fitur-fitur yang diuji beserta kondisi dan cakupan fiturnya. Dari total 35 kebutuhan fungsional pencari dan pemilik yang telah didefinisikan pada proses sebelumnya, diambil fitur sebanyak 28 untuk diuji antar muka fungsionalitasnya. Sebagai penghubung untuk kemudahan penelusuran, digunakan *requirement traceability matrix* (RTM) pada Lampiran 16.

Selanjutnya dari hasil *test design* sebelumnya dilakukan proses menjalankan pengujian terotomasi. Proses ini dimulai dengan memperoleh kasus uji. *Blackbox testing* merupakan metode yang sesuai dengan jenis pengujian antarmuka website karena fokus pengujian adalah menguji tanpa mengetahui internal program. Pada Lampiran 4 dan Lampiran 5 merupakan pembangkitan kasus uji berdasarkan bidang ataupun fitur yang ada pada Tekos.

Berdasarkan pembangkitan tersebut didapatkan hasil *test case* atau kasus uji sebanyak 143 untuk `https://sisfo.telyukost.com`, dan 77 untuk `https://telyukost.com`. Untuk lebih terperinci kumpulan *test case* yang diujikan ada pada Lampiran 6. Setelah mendapatkan kumpulan *test case* beserta aksi yang harus dilakukan untuk setiap pengujian, maka diimplementasikan pada Katalon Studio.

Sub-proses selanjutnya yakni set pengujian berdasarkan modul fitur. Lampiran 8 merupakan hasil dari proses *Assemble Test Sets* sebelum menjalankan eksekusi terhadap kasus uji pada proses sebelumnya. Proses ini dilakukan pula pada fitur *test suite* milik Katalon Studio.

Pengujian berdasarkan kumpulan set pengujian dan urutan atau prosedur pengujian hingga siklus uji terakhir menghasilkan 218 kasus uji berhasil dan 2 kasus uji gagal pada dua fitur yang berbeda. Penentuan berhasil atau gagal yakni dengan verifikasi kesesuaian harapan atau ekspektasi pengguna terhadap aksi yang dilakukan. Tabel 1 dan Tabel 2 merupakan rekap hasil pengujian pada sisi pemilik dan pencari.

Dalam pengujian aplikasi Tekos sisi pencari dan pemilik indekos, dilakukan analisis terhadap sejumlah metrik dan cakupan pengujian yang telah dilakukan. Dari total 35 fungsionalitas yang tersedia, berhasil diuji sebanyak 28 fungsionalitas, mencapai tingkat cakupan sebesar 80%. Meskipun belum mencapai 100%, cakupan ini memberikan gambaran tentang sejauh mana pengujian telah dilakukan terhadap fungsi-fungsi antarmuka dari TEKOS. Penerapan KDT pada pengujian Tekos yakni terletak pada proses *derive test case* baik secara manual ataupun terotomasi melibatkan 21 jenis kata kunci.

Cakupan kasus uji yang berhasil dijalankan sebesar 100% merupakan hal yang positif. Dari total 220 kasus uji yang direncanakan, 218 di antaranya berhasil, sementara hanya 2 kasus uji yang mengalami kegagalan. Hal ini menunjukkan bahwa hampir seluruh kasus uji telah dijalankan secara efektif dan mencakup seluruh fungsionalitas aplikasi dengan baik. Hasilnya pada siklus pertama menunjukkan bahwa terdapat 6 kasus uji yang mengalami kegagalan. Namun, poin positif dalam analisis hasil pengujian adalah kemampuan tim pengembang dalam menangani temuan-temuan yang ditemukan. Dari 6 kasus uji yang gagal, tim berhasil menuntaskan perbaikan pada 4 temuan.

Selanjutnya, hasil persentase *test tracking and efficiency* menunjukkan bahwa sebanyak 99.09% kasus uji berhasil dilacak dan diselesaikan dengan baik, sementara hanya 0.9% yang mengalami kegagalan. Persentase kegagalan sebesar 0.9% merupakan 2 kasus uji yang ditemukan cacat, hasil nilai persentase *test tracking and efficiency* dari cacat yang ditangguhkan yakni sebesar 33.3%. Kasus uji yang gagal (*cannot be fixed*) hingga siklus pengujian terakhir termasuk pada FR2-06 *Edit Kos* dan FR2-09 *Edit Kontrakan*. Kegagalan tersebut merupakan jenis defect dari sistem `https://sisfo.telyukost.com` dalam menampilkan data sebelumnya, yakni pada data foto dimana setiap pemilik harus selalu mengunggah file foto kos dan kontrakan setiap kali akan memperbarui data.

Secara menyeluruh hasil pengujian berhasil menjawab permasalahan yang ada. Selain itu juga telah menghasilkan perbaikan yang positif, diantaranya menemukan defect lebih dini hingga memberikan rekomendasi perbaikan fungsi aplikasi. Dari 28 *functional requirement* yang diuji pada kebutuhan fungsional pemilik dan pencari, 26 fungsionalitas sudah sesuai. Terdapat 2 fungsionalitas yang tidak sesuai dengan kondisi fitur seharusnya yang disebabkan oleh tidak tersedianya data pada salah satu bidang di 2 fungsionalitas tersebut. Hal

Table 1. Hasil Owner

ID	Test Suites	Total of TC	Status	
			Passed	Failed
TS-TKS-1	Daftar Owner	17	17	0
TS-TKS-2	Login Owner	4	4	0
TS-TKS-3	Lupa Password Owner	10	10	0
TS-TKS-4	Keluar Owner	1	1	0
TS-TKS-5	Input Kos	18	18	0
TS-TKS-6	Edit Kos	19	18	1
TS-TKS-7	Hapus Kos	2	2	0
TS-TKS-8	Input Kamar	11	11	0
TS-TKS-9	Edit Kamar	12	12	0
TS-TKS-10	Hapus Kamar	2	2	0
TS-TKS-11	Input Kontrakan	22	22	0
TS-TKS-12	Edit Kontrakan	23	22	1
TS-TKS-13	Hapus Kontrakan	2	2	0
	Total	143	141	2

Table 2. Hasil Pencari

ID	Test Suites	Total	Status	
			Passed	Failed
TS-TKS-14	Daftar Seeker	16	16	0
TS-TKS-15	Login Seeker	4	4	0
TS-TKS-16	Lupa Password Seeker	10	10	0
TS-TKS-17	Keluar Seeker	1	1	0
TS-TKS-18	Search bar Kos	6	6	0
TS-TKS-19	Filter Kos	17	17	0
TS-TKS-20	Search bar Kontrakan	6	6	0
TS-TKS-21	Filter Kontrakan	11	11	0
TS-TKS-22	Detail Produk Kos	3	3	0
TS-TKS-23	Detail Produk Kontrakan	3	3	0
	Total	77	77	0

ini berdampak pada kondisi ekspektasi yang tidak sesuai dan mengharuskan pengguna mengisi data secara ulang pada bidang yang ditemukan kecacatan. Kesulitan-kesulitan yang ditemukan yakni dalam menjalankan beberapa kasus uji untuk memeriksa nilai dari objek seperti foto, *card produk* yang bersifat dinamis. Implementasi pada Katalon Studio belum mencakup hal tersebut sehingga diperlukan skrip tambahan baik untuk mengambil nilai objek ataupun skrip kondisi untuk data produk yang ditampilkan.

Kesimpulan

A. Kesimpulan

Pada pengujian fungsionalitas aplikasi dengan KDT pada pembuatan kasus uji yang dibantu blackbox testing menghasilkan 220 kasus uji. KDT dilakukan pada proses pembuatan aksi untuk setiap kasus uji serta implementasi pada katalon studio. Cakupan kasus uji yang berhasil dijalankan mencapai 100%, dengan 99.09% kasus uji berhasil dan 0.9% kasus uji gagal menurut persentase test tracking and efficiency. Hasil analisis pengujian ini menunjukkan bahwa cakupan fungsional yang diuji mencapai 80% dari total 35 fungsionalitas yang tersedia. Tim pengembang berhasil menuntaskan 4 dari 6 temuan terdokumentasi. Dengan mendapatkan hasil dari matriks tersebut, kemudian dikaitkan dengan jumlah FR yang diuji, maka 26 kebutuhan fungsional yang berstatus bebas cacat dinilai sesuai dan menyisakan defect pada 2 fungsionalitas. Penyebabnya yakni data tidak ditampilkan, sehingga berdampak pada pengguna yang diharuskan untuk mengunggah ulang data foto setiap kali mengedit. Pengujian terotomasi KDT telah berhasil mengurangi jumlah defect yang muncul sebelum aplikasi dirilis. Secara keseluruhan dapat disimpulkan bahwa upaya dalam mengatasi masalah ini telah menghasilkan perbaikan yang positif dalam stabilitas dan kinerja aplikasi.

B. Saran

Setelah melakukan implementasi dan mempertimbangkan batasan masalah, terdapat beberapa saran yang dapat diperhatikan. Pertama, tim pengembang perlu memperbaiki fitur edit kos dan kontrakan dengan menyediakan data foto sebelumnya secara otomatis, tanpa mengharuskan pengguna untuk mengunggah ulang. Perlu adanya domain pengujian (staging) sebelum production. Kedua, peneliti selanjutnya perlu menemukan cara yang lebih efektif untuk pengujian otomatis pada fitur pencarian dan filter. Terakhir, peneliti selanjutnya juga dapat mengimplementasikan pengujian pada fitur sorting secara terotomasi.

Daftar Pustaka

1. Liu X. Research on Graphic User Interface Automation Testing Technology Based on Cloud Platform. Journal of Physics: Conference Series. 2019 Nov.
2. IEEE Computer Society Software Engineering Standards Committee and IEEE-SA Standards Board. IEEE recommended practice for software requirements specifications. Institute of Electrical and Electronics Engineers; 1998.
3. Oguz RF, Erdem I, Olmezogullari E, Aktas MS. End-to-End Automated UI Testing Workflow for Web Sites with Intensive User–System Interactions. International Journal of Software Engineering and Knowledge Engineering. 2022 Oct;32(10):1477-97.
4. Anand A, Uddin A. Importance of Software Testing in the Process of Software Development Comperative study on realtime data processing systems View project Cloud Security View project Importance of Software Testing in the Process of Software Development. IJSRD-International Journal for Scientific Research & Development. 2019;6:2321-0613. Available from: www.ijsrd.com.
5. Pradhan L. User Interface Test Automation and its Challenges in an Industrial Scenario. 2012. Available from: www.mrtc.mdh.se/eureca.
6. Nayyar A. Instant Approach to Software Testing Principles, Applications, Techniques and Practices. BPB PUBLICATION; 2019.
7. Rafiq M, Ashraf R, Abid H. Automated VS. Manual Testing: A Scenario Based Approach Towards Application Development. Gyancity Journal of Electronics and Computer Science. 2020 Mar;5(1):47-55.
8. IEEE Computer Society Software Engineering Standards Committee and ISO/IEC JTC. ISO/IEC/IEEE 29119-5, Software and systems engineering-Software testing-Part 5: Keyword-Driven Testing; 2016. Available from: www.iso.org.
9. Mohammad SM. Automation Testing in Information Technology. 2015;3:2320-882.
10. Saeed U, Amjad AM. ISTQB: Black Box testing Strategies used in Financial Industry for Functional testing. Blekinge Institute of Technology; 2009. Available from: www.bth.se/tek.
11. Supriyono. Software Testing with the approach of Blackbox Testing on the Academic Information System. International Journal of Information System & Technology. 2020;3(2):227-33.
12. IEEE. IEEE Guide for Software Verification and Validation Plans Software Engineering Standards Committee of the IEEE Computer Society; 1993.
13. C P. Beginners Guide To Software Testing Beginners Guide To Software Testing-Padmini C.;.
14. Perry WE. Effective Methods for Software Testing Third Edition. 3rd ed. Indianapolis: Wiley Publishing, Inc.; 2006.
15. IEEE Computer Society and Software Engineering Standards Committee. Software and systems engineering-Software testing-Part 1: Concepts and definitions; 2013.
16. Arfan A, Hendrik. Penerapan STLC dalam Pengujian Black Box dengan Automation Testing Tool (Studi kasus: PT.GIT Solution). AUTOMATA. 2022 Aug;3(2).
17. IEEE Computer Society and Software Engineering Standards Committee. ISO/IEC/IEEE 29119-2:2013, Software and systems engineering-Software testing-Part 2: Test processes; 2013.
18. alle CP, Kondoju RS. Evidence and perceptions on GUI test automation An Explorative Multi-Case Study. 2017. Available from: www.bth.se.
19. Bose S. UI Testing: A Detailed Guide. BrowserStack. 2022. Available from: https://www.browserstack.com/guide/ui-testing-guide.
20. Jha N, Gulati R. Keyword Driven Testing Framework Using Selenium Webdriver. International Journal of Advanced Research in Engineering and Technology (IJARET). 2018;9(4):186. Available from: http://iaeme.com/Home/issue/IJARET?Volume=9&Issue=4http://iaeme.com.
21. Rashmi, Bajpai N. A Keyword Driven Framework for Testing Web Applications. IJACSA) International Journal of Advanced Computer Science and Applications. 2012;3(3). Available from: www.ijacsa.thesai.org.
22. ThinkSys. Top 34 Software Testing Metrics and KPIs; 2022. Available from: https://www.thinksys.com/qa-testing/software-testing-metrics-kpis/.
23. Ilyas QM, Ahmad M, Zaman N, Alshamari MA, Ahmed I. Localized Text-Free User Interfaces. IEEE Access. 2022;10:2357-71.