IMPLEMENTASI BLACKBOX AUTOMATION TESTING PADA APLIKASI DONOR MENGGUNAKAN FRAMEWORK STLC DALAM LINGKUP PENGEMBANGAN AGILE SCRUM

Syafierra Yasmine Shalsabilla, Eka Dyar Wahyuni, Nur Cahyo Wibowo
Sistem Informasi, UPN "Veteran" Jawa Timur
Jl. Rungkut Madya No.1, Surabaya, Indonesia
syafira13yasmin@gmail.com

ABSTRAK

PT. XYZ sedang mengembangkan aplikasi yang disebut Donora, yang memungkinkan pengajuan darah darurat untuk orang-orang yang membutuhkan. Oleh karena itu, Donora membentuk tim pengembang yang terdiri dari *product owner*, *mobile*, dan *backend developer*. Namun, tim pengembang aplikasi Donora belum memiliki tugas pengujian kualitas untuk memastikan bahwa aplikasi tersebut memenuhi persyaratan yang ditetapkan. Oleh karena itu, perusahaan mengakui peran pemeriksaan kualitas selama proses pengembangan aplikasi. Pengembangan aplikasi Donora dengan *Agile Scrum* sering menyebabkan kesalahan karena perubahan kebutuhan yang sering terjadi. Oleh karena itu, selama masa pengembangan, verifikasi dan validasi diperlukan. Pengujian *black box* automasi dapat mendeteksi kesalahan atau *bug* selama proses pengembangan, serta lebih mudah untuk didokumentasikan dan mengurangi risiko kesalahan manusia. Pengujian mampu mendeteksi sepenuhnya bagaimana pengguna menggunakan software karena berfokus pada spesifikasi kebutuhan dan memiliki sudut pandang pengguna. Selain itu, *Software Testing Life Cycle* memungkinkan mereka untuk membuat pengujian menjadi lebih teratur, yang berarti bahwa fokus pengerjaan pengujian dapat lebih dikontrol. Tujuan penelitian ini adalah untuk melakukan pengujian pada aplikasi Donora menggunakan framework *Software Testing Life Cycle* dalam lingkup pengembangan *Agile Scrum*, selain itu juga untuk mengetahui apakah fungsionalitas dari aplikasi Donora sudah berjalan dengan baik dan bebas dari *error*. Hasil pengujian menunjukkan bahwa dari tujuh fitur, tujuh sudah dapat digunakan.

Kata kunci : *Blackbox Testing, Automation, STLC, Agile Scrum*

1. PENDAHULUAN

Pada era digital saat ini, banyak aplikasi baru yang berbasis mobile, web, dan desktop yang bermunculan guna mempermudah pekerjaan manusia, sebagai wadah hiburan, hingga wadah belajar bagi masyarakat. Banyak perangkat lunak tentunya melewati masa pengembangan perangkat lunak atau *Software Development Life Cycle* mulai dari tahap analisis, perancangan, desain, pengembangan, pengujian, dan pemeliharaan perangkat lunak [1] sebelum akhirnya digunakan oleh pengguna. Seluruh tahapan tersebut dilakukan untuk menghasilkan sebuah perangkat lunak yang sesuai dengan kebutuhan dan dapat digunakan secara maksimal dan efisien.

PT. XYZ adalah sebuah perusahaan yang terfokus pada industri kreatif yang berbasis di Kota Surabaya. Perusahaan ini menawarkan berbagai layanan spesialis dalam berbagai bidang, termasuk pengembangan web, pengembangan aplikasi mobile, desain dan multimedia, serta pemasaran digital. Dalam operasionalnya, PT. XYZ menangani suatu proyek pengembangan sistem informasi pengelolaan darah darurat (Aplikasi Donora) berbasis web untuk admin dan UTD PMI dan *mobile* untuk pengguna yang dilakukan serta bekerjasama oleh UTD PMI Kota Surabaya.

Berdasarkan identifikasi masalah dan observasi yang ada pada pegembangan aplikasi Donora di PT XYZ, aplikasi Donora yang dikembangkan menggunakan *framework Agile Scrum* dan tidak adanya pengujian aplikasi yang dilakukan oleh tim *Quality Assurance* pada tiap sprint dalam framework Scrum yang digunakan. Pengujian yang dilakukan hanya menggunakan dari sudut pandang developer dengan skenario positif secara manual, sehingga pengujian bersifat subjektif tanpa adanya *scenario* negatif yang dapat dilakukan oleh pengguna. Selain itu, pengujian juga tidak dilakukan secara cepat, sistematis, menyeluruh, dan ada pada tiap iterasi, dimana ketiga hal tersebut diperlukan dalam pengembangan aplikasi Donora yang menggunakan framework *Scrum* untuk menghasilkan aplikasi yang dapat digunakan bebas-*error* dengan performa yang baik yang sesuai dengan rancangan pada dokumen *Software Requirement Spefication*. Pentingnya pengujian perangkat lunak dilakukan agar dapat mendeteksi kesalahan/ *error* pada perangkat lunak sedini mungkin sehingga dapat meminimalisir atau bahkan menghilangkan kerugian dari kesalahan perangkat lunak tersebut [2].

Melihat permasalahan yang ada, solusi yang tepat menurut peneliti adalah untuk melakukan *automation blackbox testing* menggunakan *framework Software Testing Life Cycle* dalam lingkup *agile scrum* [3][4][5][6][7][8][9]pada aplikasi Donora, dengan adanya implementasi pengujian ini diharapkan QA dapat menemukan bug dan error sejak dini sehingga dapat menjadi acuan perbaikan dan bahan evaluasi

untuk pengembangan aplikasi Donora guna menghasilkan aplikasi yang berkualitas tinggi, bebas *error*, efisien, dan mudah digunakan, seperti pada penelitian-penelitian sebelumnya[10][11]. Tujuan penelitian ini adalah untuk melakukan pengujian pada aplikasi Donora menggunakan framework *Software Testing Life Cycle* dalam lingkup pengembangan *Agile Scrum*, selain itu juga untuk mengetahui apakah fungsionalitas dari aplikasi Donora sudah berjalan dengan baik dan bebas dari *error*.

2. TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Dalam penelitian dengan judul Pemanfaatan Katalon Studio Untuk Otomatisasi Pengujian *Black-Box* pada Aplikasi iPosyandu yang diteliti oleh Zulianto et al. Jurnal JEPIN, Volume 7, No. 3, halaman 370-378[11]. Dengan melakukan pengujian otomatis menggunakan *testing tools* yaitu Katalon Studio pada aplikasi iPosyandu yang berbasis web untuk memastikan aplikasi iPosyandu sudah berjalan sesuai yang diharapkan. Hasil dari pengujian secara manual dan otomatis dibandingkan dengan menggunakan parameter status eksekusi dan waku eksekusi dari tiap test case. Didapatkan status yang sama yaitu 10 pass dan 3 fail pada test case dengan *response time* yang berbeda-beda. Total waktu eksekusi pengujian otomatis yaitu 283,08 detik dan 719,27 detik untuk pengujian manual, sehingga terjadi peningkatan kecepatan sebanyak 2,54 kali dengan pengujian otomatis.

Pengujian menggunakan metode *blackbox automation testing* dengan *tools* katalon studio juga dilakukan oleh Tempomona & Hendry pada Jurnal Inovtek Polbeng Seri Informatika, Volume 7, No. 2, halaman 193–204 pada tahun 2022 [9]. Hasil dari pengujian *blackbox* secara otomatis menggunakan katalon studio menghasilkan *report* hasil pengujian dalam bentuk PDF dari *test suite* yang dijalankan. Terdapat 15 test case dengan 5 *test case* yang berhasil, 9 test case yang gagal, dan 1 *test case* yang tidak selesai.

Penelitian sebelumnya dengan judul Rancang Bangun *Automation Test Journey* pada *E-Commerce* (Studi Kasus : *Marketplace* PT. Tokopedia) pada jurnal AUTOMATA, Volume 3, No. 2 oleh Wicaksono & Rani pada tahun 2022 [12]. Didapatkan hasil penelitian yaitu :

a. Hasil pengujian *test suite* yang belum menerapkan metode *automation journey* membutuhkan waktu 6 menit 12 detik, sedangkan test suite yang menerapkan metode *automation journey* membutuhkan waktu 1 menit 27 detik.
b. Pengujian jauh lebih cepat, efektif, dan skalabel menggunakan metode *automation test journey* daripada pengujian otomatis tanpa metode journey.

2.2. *Software Testing Life Cycle*

Software Testing Life Cycle merupakan tahapan rinci dalam proses pengujian yang harus dilakukan [8]. *Software Testing Life Cycle* termasuk ke dalam bagian *Software Development Life Cycle*), namun pada fase pengujian saja. Tahapan pertama dimulai saat kebutuhan pengguna atau *Software Requirement Specification* dibuat.

Software Testing Life Cycle memiliki 6 tahapan utama untuk pengujian, tetapi seluruh tahapan ini tidak wajib untuk diikuti secara keluruhan. Tahapan yang digunakan bergantung pada jenis perangkat lunak yang dikembangkan, waktu, biaya, dan tenaga yang dialokasikan untuk melakukan pengujian. 6 tahapan tersebut yaitu *requirement analysis*, *test planning*, *test case development*, *test environment setup*, *test case execution*, dan *test cycle closure*[8].

Gambar 1. Tahapan STLC

2.3. *Blackbox Testing*

Blackbox testing atau dengan nama lain *behavioral testing*, *closed-box*, atau *specification-based testing* merupakan sebuah metode pengujian yang dimana seorang penguji perngkat lunak tanpa harus memperhatikan isi kode dari perangkat lunak tersebut. Jadi penguji melakukan pemeriksaan fungsionalitas perangkat lunak berdasarkan masukan oleh pengguna dan keluaran oleh sistem untuk mengetahui perangkat lunak tersebut sudah berjalan sesuai yang dibutuhkan dan perancangan awal [13].

Menurut [14] pengujian *blackbox* memiliki beberapa teknik yang dapat dilakukan untuk pengujian, antara lain *Equivalence Partitioning, Boundary Value Analysis, Decision Table Based Testing, State Transition*, dan *Error Guessing.*

2.4. *Automation Testing*

Automation testing merupakan sebuah pengujian yang bergantung pada script uji yang dibuat oleh penguji perangkat lunak dan pelaksanaan pengujiannya dilakukan secara otomatis menggunakan *automation testing tools* untuk membandingkan hasil yang didapatkan sesuai dengan yang diharapkan [12]. Kelebihan menggunakan pengujian otomatis dibanding dengan pengujian secara manual adalah dapat dilakukan secara berulang (iteratif), memberikan laporan pengujian secara cepat, rinci, dan akurat, serta dapat berjalan otomatis tanpa interaksi manusia. Maka dari itu, pengujian ini dapat meningkatkan frekuensi pengujian dan memungkinkan untuk melakukan pengujian regresi tanpa intervensi dari penguji manual. Pengujian otomatis juga dapat mengatasi masalah *human error* yang terjadi saat pengujian manual, sehingga laporan pengujian yang dihasilkan lebih kredibel.

2.5. *Bug*

Bug adalah sesuatu yang seharusnya tidak dilakukan oleh perangkat lunak atau kerusakan yang terjadi pada perangkat lunak yang tidak sesuai dengan kebutuhan perangkat lunak. Setiap *bug* memiliki atribut data yang digunakan untuk identifikasi, seperti nama bug, tipe bug, tanggal ditemukan, hingga cara bagaimana bug tersebut bisa muncul dan ditemukan [15].

Menurut [16] , terdapat beberapa jenis bug yang ada, antara lain :

a. *Logic bug* : terjadi saat perangkat lunak salah dalam mengeksekusi perintah, sehingga luaran yang dihasilkan tidak sesuai
b. *Functional error* : menggambarkan masalah pada fungsional software
c. *Usability defects* : membuat software berjalan tidak maksimal kaerna kesalahan kode atau kesalahan desain *user interface*
d. *Security error* : kesalahan yang berkaitan dengan keamanan perangkat lunak
e. *Performance defects* : bug yang berkaitan dengan kecepatan,stabilitas, ataupun *response time software*
f. *Syntax error* : bug yang terjadi karena kesalahan penulisan kode

2.6. Katalon Studio

Katalon Studio adalah aplikasi open source untuk pengujian otomatis yang dikembangkan oleh Katalon LLC, dapat dijalankan di berbagai sistem operasi seperti Windows, MAC OS, dan Linux. Tool ini menyediakan antarmuka IDE khusus untuk pengujian dan memiliki tiga fitur utama yang mendukung pengujian di beberapa *platform*, seperti *Web testing*, API *testing*, *Mobile testing*, dan *Desktop testing*. Selain itu, Katalon Studio terintegrasi dengan beberapa teknologi eksternal seperti GitHub, Microsoft Teams, dan Jira [4].

Katalon Studio menyediakan beragam tampilan UI grafis, menu, pohon tabel, dan lainnya untuk mengelola test case, objek, dan file data. Meskipun masih dalam tahap pengembangan, Katalon Studio sudah mendukung beberapa *environment* seperti browser dan sistem operasi. Hal ini sangat berguna bagi pengujian yang membutuhkan kemampuan drag-and-drop yang memiliki keterbatasan dalam pemrograman [4]. Menurut [9] katalon studio menyediakan fitur laporan analitik yang memungkinkan *Software Quality Assurance* menganalisa hasil pengujian berdasarkan skenario yang dijalankan dan dapat diekspor dalam bentuk file HTML, PDF, Excel, atau CSV.

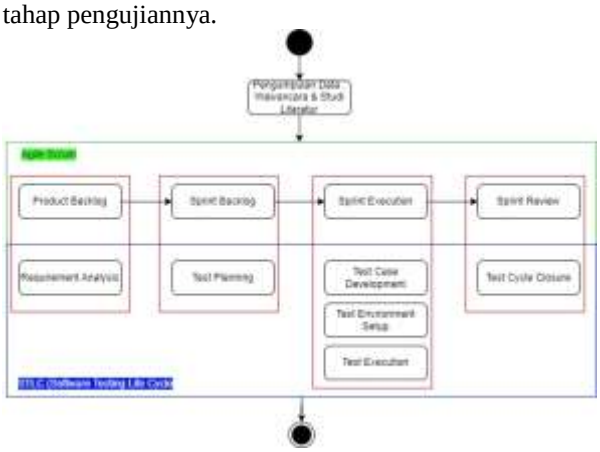
2.7. Agile Scrum

Metodologi - metodologi pengembangan perangkat lunak yang bersifat *agile* atau sering disebut sebagai metodologi *agile* adalah sekelompok pendekatan pengembangan yang berlandaskan pada iterasi. Dalam pendekatan ini, persyaratan dan solusi mengalami perkembangan melalui kerjasama antar tim yang terstruktur [17]. Pendekatan lainnya dijelaskan oleh Sommerville [18] yang menggambarkan metode agile sebagai pendekatan pengembangan secara bertahap yang menitikberatkan pada pertumbuhan yang cepat. Dalam metode ini, perangkat lunak dirilis secara bertahap untuk mengurangi proses yang berlebihan, serta menghasilkan kode berkualitas tinggi. Selama proses pengembangannya, pelanggan terlibat secara langsung.

Beberapa model pengembangan perangkat lunak yang termasuk dalam kategori agile mencakup *extreme programming*, *adaptive*, *software development*, *dynamic systems developments*, model scrum, dan *agile modelling*. Menurut [17] metode pengembangan perangkat lunaks secara cepat (*agile*) salah satunya adalah model scrum. Schwaber & Sutherland [19] berpendapat bahwa *Scrum* merupakan suatu struktur kerja yang mampu mengatasi tantangan dari situasi kompleks yang terus berubah. Selain itu, mereka menyimpulkan bahwa *Scrum* memiliki kemampuan untuk menciptakan produk berkualitas tinggi sesuai dengan preferensi pengguna dengan cara yang inovatif dan efisien. Dalam *scrum*, memiliki tim *scrum* yang memiliki peran masing-masing, *scrum event*, *scrum artifacts*, dan aturan main dalam scrum.

3. METODE PENELITIAN

Metodologi yang digunakan pada penelitian ini merupakan penggabungan metodologi *agile scrum* dalam pengembangan aplikasi Donora dan metodologi *Software Testing Life Cycle* yang digunakan untuk tahap pengujiannya.



Gambar 2. Metodologi Penelitian

3.1. *Product Backlog & Requirement Analysis*

Product backlog disusun berdasarkan kebutuhan dan spesifikasi yang telah ditetapkan oleh PT. XYZ melaui wawancara dan studi literatur pada dokumen *Software Requirement Specification*. Pada tahap ini juga tahapan STLC dilakukan, yaitu analisa kebutuhan (*requirement* analysis). Terdapat 3 tipe pengguna dalam aplikasi Donora, yaitu Admin, UTD PMI, dan User yang dirancang untuk menggunakan aplikasi Donora ini.

Terdapat beberapa modul yang ada dalam aplikasi Donora yaitu, modul autentikasi, ajuan darah

darurat, ajuan bukti donor, stok darah, reward, jadwal donor, profile user & UTD PMI, dashboard, dan *push notification*.

3.2. *Sprint Planning & Test Planning*

Sprint Planning adalah aktivitas yang berlangsung pada awal setiap *Sprint*. Tujuan utama dari *Sprint Planning* adalah untuk menentukan tugas-tugas yang akan dikerjakan selama Sprint berikutnya. Tim akan memilih item-item dengan skala prioritas yang tinggi dari *Product Backlog* dan merencanakan bagaimana tugas-tugas tersebut akan dikerjakan serta apa yang dapat diselesaikan dalam jangka waktu *Sprint* tersebut. Detail dari tahap sprint planning akan menjadi sebuah luaran yaitu *sprint backlog* yang berisi fitur/ *task* apa saja yang akan dikerjakan dalam jangka waktu sprint (*sprint backlog*).

Pada tahap ini juga tahapan *Software Testing Life Cycle* yang dilakukan yaitu *test planning*. Beberapa hal yang dilakukan saat *test planning* yaitu menentukan strategi pegujian, menetapkan sumber daya, lingkungan pengujian, batasan pengujian, menetapkan jenis pegujian yang dilakukan, dan jadwal pengujian. pengujian, menetapkan jenis pegujian yang dilakukan, dan jadwal pengujian.

3.3. *Sprint Execution*

Pada tahap *test execution scrum*, semua anggota *team scrum* mengerjakan task yang telah disepakati saat *sprint planning* dan disusun dalam *sprint backlog*. Pada tahap ini juga dilakukan beberapa tahapan dalam STLC antara lain *test case development*, *test environment setup*, *test execution* (*functional* atau *non functional*).

3.3.1. *Test Case Development*

Tahap *test case development* merupakan fase dimana tim QA membuat menyusun *test case/ script*, menyiapkan *data testing*, dan mengidentifikasi hasil yang diharapkan dari tiap *test case* berdasarkan *test plan*. Pembuatan *test case* akan dilakukan manual dan otomatis menggunakan katalon studio (*record and playback*). *Data testing* dibuat dan disesuaikan dengan case – case yang memiliki peluang terjadi saat user menggunakan aplikasi Donora. Luaran yang dihasilkan pada tahap ini yaitu *test case* dan *test data*.

3.3.2. *Test Environment Setup*

Tahap *environment setup* dilakukan setelah tahap *test case development*. Pada fase ini tim QA/ tim *developer* akan menyiapkan *software*, *hardware*, dan *tools* yang akan digunakan dalam pengujian. Tim pengujian juga harus melakukan *readiness check*/ *smoke test* dari *environment* yang telah diberikan, selain itu tim pengujian juga menyiapkan *environment* dan *test data* yang dimasukkan ke dalamnya.

3.3.3. *Test Execution*

Tahap *Software Testing Life Cycle* terakhir dan terpenting yang dilakukan pada *sprint execution* adalah *test execution*, dimana QA melakukan pengujian berdasarkan *test plan* dan *test case* yang telah disiapkan sebelumnya. Proses dalam test execution antara lain t*est script/case execution, test script maintenance/ updating*, dan *bug reporting*. Eksekusi pengujian dilakukan untuk fungsional fitur aplikasi Donora.

Jika ditemukan *bug* atau *error*, *quality assurance* akan melaporkan temuan bug tersebut kepada developer sehingga dapat segera diperbaiki dalam waktu *sprint* yang berjalan. *Bug* yang telah diperbaiki akan diajukan pengujian lagi ke *quality assurance* dan *quality assurance* akan menguji dan memastikan kembali apakah *bug* sudah teratasi. Luaran yang dihasilkan pada tahap ini yaitu dokumen *Requirement Traceability Metric* yang telah dilengkapi status eksekusi pengujian, *test case* yang telah diupdate dengan hasilnya, dan *defect/ bug report*.

3.4. *Sprint Review & Test Cycle Closure*

Sprint review adalah tahapan guna memeriksa hasil dari Sprint dan menentukan adaptasi yang akan datang. Tim *Scrum* mempresentasikan hasil kerjanya kepada *stakeholder*, dan kemajuan menuju *Product Goal* dibahas. Pada *sprint review*, tim *Scrum* dan pemangku kepentingan juga meninjau apa yang telah dicapai dalam *Sprint* dan apa yang telah berubah dalam lingkungan mereka. Berdasarkan informasi ini, tim bekerja sama untuk menentukan langkah selanjutnya. *Product Backlog* juga dapat disesuaikan untuk memenuhi peluang-peluang baru dalam sprint selanjutnya.

Pada *framework Software Testing Life Cycle*, terdapat tahap yang dilaksanakan pada *sprint review*, yaitu *test cycle closure*. Tahap *test cycle closure* merupakan tahap akhir dalam *Software Testing Life Cycle* di mana semua aktivitas terkait pengujian pada suatu iterasi diselesaikan, hasil pengujian didokumentasikan, dan pelajaran dari pengujian saat ini digunakan untuk meningkatkan proses pengujian di masa depan. Aktivitas utama yang dilakukan pada tahap ini yaitu membuat *test summary reporting*, *test environment clean-up*, *knowledge transfer*, serta saran dan perbaikan pengembangan. Luaran yang dihasilkan pada tahap *test cycle closure* adalah *test summary report* dan *test metrics*.

4. HASIL DAN PEMBAHASAN

4.1. *Product Backlog & Requirement Analysis*

Tahap awal *requirement analysis* untuk aplikasi Donora akan melakukan analisa dari kebutuhan funsgional yang ada dokumen *Software Requirement Specification*, terdapat 9 modul utama yaitu modul autentikasi, ajuan darah darurat, ajuan bukti donor, stok darah, reward, jadwal donor, profile user & UTD PMI, dashboard, dan *push notification*. Pada tahap ini juga QA mengetahui apa saja detail/ aktivitas apa saja yang dapat dilakukan oleh tiap role di aplikasi Donora. Berikut merupakan detail aktivitas untuk tiap role :

a. Admin
   - Autentikasi (login, logout, dan *reset password*)
   - Membuat, mengupdate, melihat, dan mengaktif/nonaktifkan akun dan profile UTD PMI
   - Melihat dan mengaktif/nonaktifkan akun user
   - Melihat, mengupdate, dan menghapus *push notification*
   - Melihat ajuan dan mengupdate status ajuan darah darurat
   - Melihat ajuan bukti donor dan mengupdate status ajuan bukti donor
   - Membuat, melihat, mengupdate, dan menghapus reward
   - Melihat dashboard, pendonor darurat, stok darah, dan jadwal donor

b. UTD PMI
   - Autentikasi (login, logout, dan *reset password*)
   - Mengupdate dan melihat akun dan profile UTD PMI
   - Melihat dashboard
   - Membuat, melihat, mengupdate, menghapus, dan mengaktif/non aktifkan jadwal donor
   - Melihat dan mengupdate status pendonor darurat
   - Mengupdate stok darah

c. User
   - Autentikasi (register, login, logout, dan *reset password*)
   - Melihat dan mengupdate profile
   - Membuat, melihat, dan menghapus ajuan darah darurat
   - Membuat, melihat, mengupdate, dan menghapus ajuan bukti donor
   - Mendaftar jadi pendonor darurat
   - Melihat dan mengklaim reward
   - Mengupdate status notifikasi
   - Melihat notifikasi, jadwal donor, stok darah, dan profile UTD PMI

4.2. Sprint Backlog & Test Planning

Setelah melalui tahap *product backlog* & *requirement analysis*, tahap selanjutnya yaitu menyusun sprint backlog yang akan diimplementasikan selama 2 *sprint* kedapan. Berikut ini adalah *sprint backlog* yang disusun dan akan diimplementasikan :

a. *Sprint* 1 (2 minggu)
   - Autentikasi admin & UTD PMI
   - Admin : Ajuan bukti donor
   - Admin : Ajuan darah darurat
   - Admin : push notifikasi
   - Admin : home dashboard
   - Admin : akun dan profile UTD PMI
   - Admin : profile user
   - Admin : reward

b. Sprint 2 (2 minggu)
   - Admin : reward, stok darah, jadwal donor
   - UTD PMI : akun & profile UTD PMI
   - UTD PMI : pendonor darurat
   - UTD PMI : stok darah dan jadwal donor
   - UTD PMI : home dashboard

Pengujian yang akan dilaksanakan selama sprint berjalan yaitu pengujian secara otomatis menggunakan metode blackbox dengan teknik pengujian menggunakan *Equivalence Paritioning* untuk mengetahui hasil/ output dari inputan yang dimasukkan oleh user. Hal ini didasari oleh pengujian otomatis dapat membuat pengujian menjadi lebih efisien dari segi waktu, biaya, dan usaha. Selain itu, dapat menghindari kesalahan dari human *error*. Pengujian otomatis dilakukan menggunakan alat pengujian otomatisasi Katalon Studio. Cakupan yang diambil adalah dari sudut pandang admin aplikasi Donora. Pengujian dilakukan setiap hari (Senin-Jum'at) selama 4 minggu (1 bulan) sesuai dengan lamanya sprint. Berikut merupakan rincian *test planning* yang telah dibuat

Tabel 1. Test Planning

Modul	Fitur
Reward	Pengujian fungsionalitas tambah, ubah, dan hapus data reward
Ajuan Darah Darurat	Pengujian fungsionalitas update status ajuan, call status, dan menghapus darah darurat

Tabel 1 tersebut dibuat menggunakan acuan yang ada pada dokumen *Software Requirement Specification* dan sudah didiskusikan dengan stakeholder, kemudian tersusunlah dokumen *test plan*.

4.3. Sprint Execution

Selama 2 sprint berjalan, *quality assurance* akan melaksanakan beberapa tahap pengujian yaitu *test case development*, *test environment setup*, dan *test execution*.

a. Test Case Development

   Test case dibuat berdasarkan sprint backlog yang sedang berjalan. Ketika berada di sprint 1, maka *quality assurance* akan membuat *test case* untuk modul yang ada di sprint 1, salah satunya yaitu reward dan ajuan darah darurat. Pembuatan test case dilakukan dengan 2 cara, yaitu manual dan otomatis. Untuk *test case* yang dibuat secara manual ini mengacu pada dokumen *test plan* yang telah dibuat dan dapat dilihat seperti pada tabel 2 di bawah ini

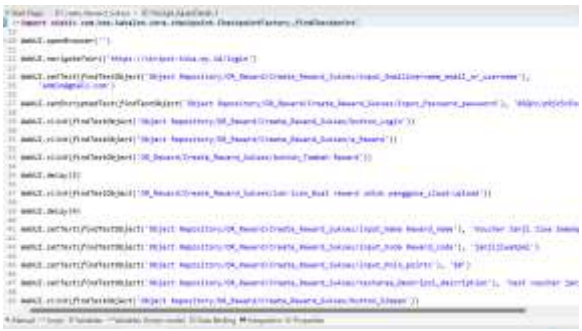
Tabel 2. Test Case Manual

Test Case ID	Nama Test Case	Expected Result
Add_Reward_Success	Membuat reward dengan inputan yang valid dan lengkap	Berhasil menambahkan reward baru
Add_Reward_Failed	Membuat reward dengan inputan yang tidak valid dan tidak lengkap	Gagal menambahkan reward baru
Update_Rewar_Success	Mengupdate reward dengan inputan yang valid dan lengkap	Berhasil mengupdate reward
Update_Reward_Failed	Mengupdate *reward* dengan inputan yang tidak valid dan tidak lengkap	Gagal mengupdate *reward*
Accept_AjuanDarah	Menerima ajuan darah darurat	Berhasil menerima ajuan darah darurat
Reject_Ajuan Darah_Success	Menolak ajuan darah darurat dengan mengisi alasan penolakan	Berhasil menolak ajuan darah darurat
Reject_Ajuan Darah_Failed	Menolak ajuan darah darurat dengan mengosongkan alasan penolakan	Gagal menolak ajuan darah darurat

Implementasi *test case* manual ke *test case* otomatis dengan menggunakan fitur *record and playback* pada Katalon Studio. Pada Gambar 3 dan 4 menggambarkan proses pembuatan *test script* dengan fitur *record and play* pada Katalon Studio. Skrip pengujian otomatis akan menjalankan *test case* yang telah dibuat sesuai dengan urutannya. Setelah melakukan *record and play*, jika ingin menambahkan skrip kode tertentu, maka dapat juga ditambahkan pada skrip manual seperti pada Gambar 5.

Gambar 3. Hasil *Record and Playback* Fitur Tambah *Reward*

Gambar 4. Hasil *Record and Playback* Fitur Ajuan Darah Darurat

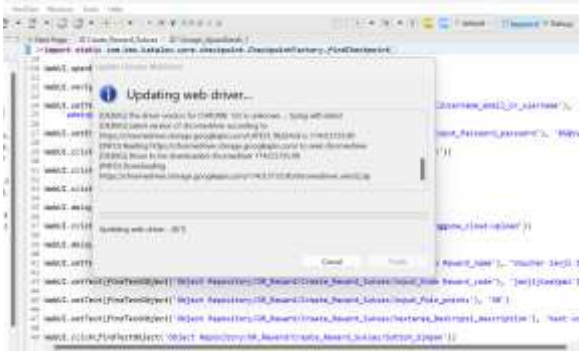


Gambar 5. Skrip Manual Fitur Reward

b. *Test Environment Setup*

Katalon studio memiliki beberapa persyaratan yang harus dipenuhi untuk menjalankan skrip pengujian yang telah dibuat, khususnya jika skrip pengujian akan diimplementasikan menggunakan web browser, sehingga perlu untuk mengupdate dan atau menginstall *web browser* dan *web driver browser* sesuai dengan yang dibutuhkan.

Mengupdate/ menginstall *web driver browser* pada Katalon Studio dapat dilakukan pada menu *Tools* → *Update WebDrivers* → Pilih browser yang dibutuhkan. Setelah diklik, akan menampilkan proses *updating webdrivers* seperti pada Gambar 6.

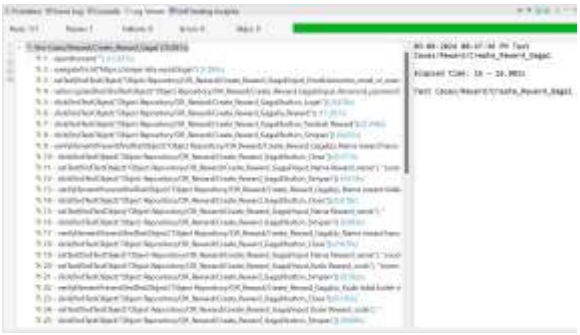


Gambar 6. *Updating* WebDrivers

c. *Test Execution*
Setelah selsai melakukan *test case development* dan *environtment setup*, *test case* yang telah dibuat siap untuk diuji. Centang berwarna hijau pada yang berada di sebelah *test case* menandakan bahwa *test case* tersebut berstatus *passed* atau sesuai dengan yang diharapkan/ fungsionalitasnya. Namun jika terdapat silang merah, hal tersebut menandakan bahwa *test case* berstatus *failed* atau gagal. Pada Gambar 7 hingga Gambar 13 merupakan hasil eksekusi *test case* yang sudah dibuat pada tahap sebelumnya.

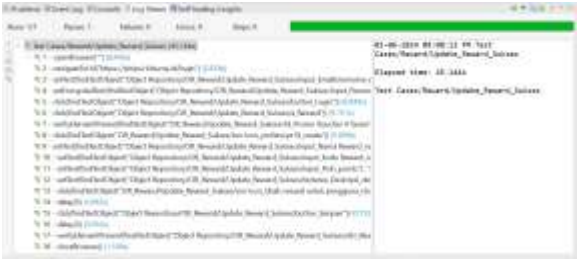
Gambar 7. Hasil *Test Case* Tambah *Reward* Sukses

Gambar 7 menunjukkan bahwa hasil eksekusi tambah data *reward* berhasil. Sebagai admin dapat menambahkan *reward*.



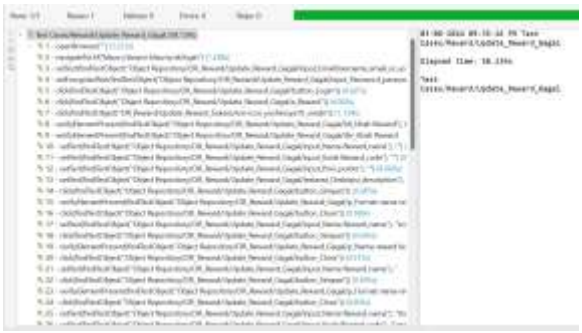
Gambar 8. Hasil *Test Case* Tambah *Reward* Gagal

Gambar 8 menunjukkan bahwa hasil eksekusi tambah data *reward* gagal. Hal ini menunjukkan adanya *handling* pada field jika diisi tidak sesuai dengan ketentuan input data.



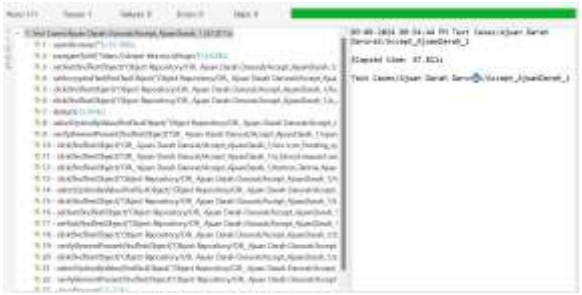
Gambar 9. Hasil *Test Case Update Reward* Sukses

Gambar 9 menunjukkan bahwa hasil eksekusi update data *reward* berhasil. Sebagai admin dapat mengupdate *reward*.



Gambar 10. Hasil *Test Case Update Reward* Gagal

Gambar 10 menunjukkan bahwa hasil eksekusi *update* data *reward* gagal. Hal ini menunjukkan adanya *handling* pada field jika diisi tidak sesuai dengan ketentuan input data.

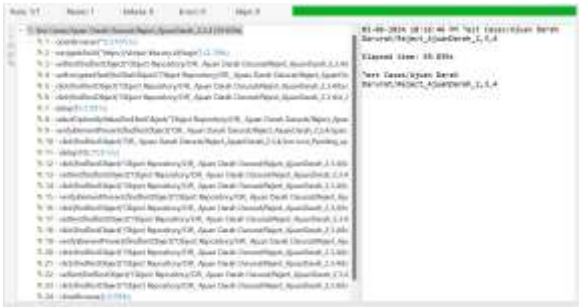


Gambar 11. Hasil *Test Case Accept* Ajuan Darah

Gambar 11 menunjukkan bahwa hasil eksekusi *accept*/ terima ajuan darah sukses.

Gambar 12. Hasil *Test Case Reject* Ajuan Darah Sukses

Gambar 12 menunjukkan bahwa hasil eksekusi *reject*/ tolak ajuan darah sukses.



Gambar 13. Hasil *Test Case Reject* Reward Gagal

Gambar 13 menunjukkan bahwa hasil eksekusi *reject* data reward gagal. Hal ini menunjukkan adanya

handling pada field jika diisi tidak sesuai dengan ketentuan input data.

4.4. Sprint Review & Test Cycle Closure

Seluruh rangakaian pengujian telah selesai dilakukan. Pada tahap ini *sprint review* dan *test cycle closure* digunakan untuk memaparkan hasil dari rangakaian pengujian yang sudah dilakukan mulai dari tahap perencanaan pengujian (*test planning*) hingga eksekusi pengujian (*test execution*).

Berdasarkan tahap *execution* dapat ditarik kesimpulan bahwa hasil dari 7 fungsionalitas yang diuji, telah sesuai dengan *expected result* atau hasil yang diharapkan, 100% serta tidak ada *bug* yang ditemukan selama pengujian berlangsung. Rata-rata waktu pengujian untuk menguji 7 fungsional adalah 51,28 detik untuk beberapa fitur yang cukup kompleks dan banyak pengecekan. Hasil pengujian dapat dilihat di Tabel 3 yang menampilkan hasil pengujian yang telah dilakukan.

Tabel 3. Hasil Pengujian

Test Case ID	Nama Test Case	Status
Add_Reward_Success	Membuat reward dengan inputan yang valid dan lengkap	*Passed*
Add_Reward_Success	Membuat reward dengan inputan yang tidak valid dan tidak lengkap	*Passed*
Update_Rewar_Success	Mengupdate reward dengan inputan yang valid dan lengkap	*Passed*
Update_Reward_Failed	Mengupdate reward dengan inputan yang tidak valid dan tidak lengkap	*Passed*
Accept_AjuanDarah	Menerima ajuan darah darurat	*Passed*
Reject_Ajuan Darah_Success	Menolak ajuan darah darurat dengan mengisi alasan penolakan	*Passed*
Reject_Ajuan Darah_Failed	Menolak ajuan darah darurat dengan mengosongkan alasan penolakan	*Passed*

5. KESIMPULAN DAN SARAN

Pengujian otomatis menggunakan Katalon Studio berhasil mendeteksi fungsionalitas aplikasi Donora dengan efektif dan efisien, meskipun terdapat kendala dalam pengenalan objek pada skrip karena pengaruh atribut aplikasi. Pengujian pada aplikasi Donora menghasilkan 7 dari 7 fungsional berstatus *Passed* atau sesuai dengan *expected result*, selain itu rata-rata waktu pengujian dari 7 fungsionalitas yaitu 51,28 detik yang dapat dikatakan tergolong cukup cepat. Meskipun demikian, tool ini memberikan kemudahan dalam menghasilkan laporan pengujian yang lengkap dan dapat diekspor dalam berbagai format, seperti PDF, CSV, HTML, dan Excel. Pengujian pada modul Ajuan Darah Darurat dan Reward menunjukkan bahwa *blackbox testing* efektif dalam menguji fungsionalitas fitur karena melihat dari input oleh user dan output dari sistem.

Dukungan dari Katalon Studio dalam otomatisasi pengujian juga membantu dalam dokumentasi dan mengurangi kesalahan manusia. Penerapan *Software Testing Life Cycle* memastikan alur pengujian yang terstruktur dan fokus pada setiap tahapan pengujian. Penelitian selanjutnya direkomendasikan untuk melakukan pengujian *regression* dan *integration* pada aplikasi Donora yang dikembangkan.

DAFTAR PUSTAKA

[1] Inggi. R, Sugiantoro, and Y. Prayudi, "PENERAPAN SYSTEM DEVELOPMENT LIFE CYCLE (SDLC) DALAM MENGEMBANGKAN FRAMEWORK AUDIO FORENSIK," *SemanTIK*, vol. 4, no. 2, pp. 2502–8928, 2018.

[2] W. Wibisono and F. Baskoro, "PENGUJIAN PERANGKAT LUNAK DENGAN MENGGUNAKAN MODEL BEHAVIOUR UML," *JUTI: Jurnal Ilmiah Teknologi Informasi*, vol. 1, no. 1, pp. 43–49, 2002.

[3] N. Rita Mustika, "Automated Black Box Testing using Selenium Python," *International Journal of Computer Science and Software Engineering (IJCSSE)*, vol. 7, no. 9, 2018, [Online]. Available: www.IJCSSE.org

[4] Y. Kosasih and A. B. Cahyono, "Automation Testing Tool Dalam Pengujian Aplikasi The Point Of Sale (Studi Kasus TPOS PT. JAVASIGNA INTERMEDIA)."

[5] A. Arfan, "Penerapan STLC dalam Pengujian Automation Aplikasi Mobile (Studi kasus: LMS Amikom Center PT.GIT Solution)."

[6] V. Nugraha, R. Yanwastika Ariyana, and E. K. Nurnawati, "UJI BLACK BOX TES APLIKASI SOFTWARE DEVELOPMENT SYSTEM INFORMATION (SODEVI) PT. DIMATA SORA JAYATE MENGGUNAKAN KATALON STUDIO," 2022.

[7] D. Katarina and E. Windia Ambarsari, "STRING (Satuan Tulisan Riset dan Inovasi Teknologi) AUTOMATION TESTING TOOL DALAM PENGUJIAN APLIKASI BELAJAR TAJWID PADA PLATFORM ANDROID."

[8] B. Huda and A. Lia Hananto, "Penerapan Software Testing Life Cycle Pada Pengujian Otomatisasi Platform Dzikra Application of Software Testing Life Cycle in Automated Testing of Dzikra Platform," vol. 15, no. 1, pp. 1–11, 2023, doi: 10.22303/csrid.15.1.2023.01-11.

[9] V. Tirza Tempomona, "Penerapan Metode Blackbox Pada Perangkat Lunak Menggunakan Katalon Studio (Studi Kasus: Aplikasi Absensi di PT Astra Sedaya Finance)," vol. 7, no. 2, p. 2022.

[10] I. Kusyadi, S. Mulyati, A. P. Setiany, D. Noviyanto, and S. Aisah, "Pengujian Aplikasi Kas Keuangan Menggunakan Katalon," *Jurnal Teknologi Sistem Informasi dan Aplikasi*, vol. 5, no. 2, p. 91, May 2022, doi: 10.32493/jtsi.v5i2.16958.

[11] A. Zulianto *et al.*, "JEPIN (Jurnal Edukasi dan Penelitian Informatika) Pemanfaatan Katalon Studio untuk Otomatisasi Pengujian Black-Box pada Aplikasi iPosyandu".

[12] F. Dhimas Wicaksono and S. Rani, "Rancang Bangun Automation Test Journey pada E-Commerce (Studi Kasus: Marketplace PT. Tokopedia)." [Online]. Available: https://infuse.it/quality-engineering/test-automation/

[13] T. Desyani, A. Syamsiana, E. Ukung Kobun, F. Tri Ananda, and S. Priaji, "Jurnal Teknologi Sistem Informasi dan Aplikasi Otomatisasi Pengujian Aplikasi Web Otten Coffee Menggunakan Katalon Studio," vol. 6, no. 2, pp. 186–190, 2023, doi: 10.32493/jtsi.v6i2.26902.

[14] A. Arfan and H. Hendrik, "Penerapan STLC dalam Pengujian Black Box dengan Automation Testing Tool (Studi kasus: PT.GIT Solution)," *AUTOMATA*, vol. 3, no. 2.

[15] W. U. TALITHA, "Sistem Informasi Pencatatan Bug Berbasis Web," Underrgaduate Thesis, Universitas Islam Indonesia, Yogyakarta, 2021.

[16] Administrator, "Software Bug: Pengertian, Penyebab, Jenis dan Cara Mengatasinya," Ivosight.

[17] R, S. Pressman, *Software Engineering: A Practitioner's Approach*, 7th ed. McGraw-Hill Science, Engineering & Mathematics, 2010.

[18] I. Sommerville, *Software engineering (9th ed.)*, 9th ed. Pearson Corp, 2011.

[19] "K" "Schwaber" and "J" "Sutherland," "The Scrum Guide," Scrum.org