RANCANG BANGUN AUTOMATION TEST JOURNEY PADA E-COMMERCE

Disusun Oleh:

N a m a : Faiq Dhimas Wicaksono
NIM : 18523088

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

RANCANG BANGUN AUTOMATION TEST JOURNEY PADA E-COMMERCE

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Faiq Dhimas Wicaksono
NIM : 18523088

Yogyakarta, 3 Agustus 2022
Pembimbing,

(Septia Rani, S.T., M.Cs.)

HALAMAN PENGESAHAN DOSEN PENGUJI

RANCANG BANGUN AUTOMATION TEST JOURNEY PADA E-COMMERCE

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 3 Agustus 2022

Tim Penguji

Septia Rani, S.T., M.Cs. _____

Anggota 1

Aridhanyati Arifin, S.T., M.Cs. _____

Anggota 2

Elyza Gustri Wahyuni, S.T., M.Cs. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Faiq Dhimas Wicaksono

NIM : 18523088

Tugas akhir dengan judul:

RANCANG BANGUN AUTOMATION TEST JOURNEY PADA E-COMMERCE

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 3 Agustus 2022

(Faiq Dhimas Wicaksono)

HALAMAN PERSEMBAHAN

Saya persembahkan laporan akhir ini kepada kedua orang tua, keluarga, bapak/ibu dosen, teman-teman seperjuangan, rekan kerja, dan senior di tokopedia, serta seluruh pihak yang telah mendukung proses berjalannya program magang dan penyusunan laporan akhir ini.

HALAMAN MOTO

"*Resting at the end, not in the middle*" adalah kutipan inspiratif dari Kobe Bryant yang memberitahu kita untuk terus bekerja keras sepanjang waktu, tidak hanya saat ada kemauan. Kita harus konsisten setiap hari untuk mencapai tujuan kita, menerapkan standar yang tinggi, dan mempertahankannya.

Terkadang dalam hidup, pilihan yang mudah adalah dengan bersantai dan bermalas-malasan, tetapi dengan menjadi konsisten dan terus menerus berjuang untuk menjadi lebih baik setiap harinya, itulah tanda pertama kesuksesan.

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillah penulis haturkan kepada Allah Swt, yang telah melimpahkan rahmat dan taufiq serta hidayat-Nya sehingga penulis dapat menyelesaikan laporan tugas akhir dengan tepat waktu. Shalawat serta salam tak lupa penulis haturkan kepada Nabi Muhammad SAW yang selalu menjadi teladan bagi umatnya.

Program magang merupakan salah satu faktor penting yang dapat membantu kesiapan karir mahasiswa setelah lulus. Lingkungan perkuliahan mungkin melibatkan diskusi, debat, interaksi teman sebaya, dan pengalaman belajar bersama, tetapi penting bagi mahasiswa untuk mencari pengalaman baru dalam lingkungan yang lebih profesional dengan menerapkan dan mengembangkan konsep akademik yang sudah dipelajari sebelumnya.

Laporan tugas akhir ini merupakan bentuk pertanggungjawaban tertulis atas terlaksananya kegiatan magang yang telah dilaksanakan oleh penulis selama 6 bulan di PT Tokopedia. kelancaran penyusunan laporan tugas akhir tidak terlepas dari bantuan berbagai pihak yang telah mendukung proses ini baik secara langsung maupun tidak langsung. Oleh karena, itu penulis ingin mengucapkan terimakasih kepada segenap pihak yang telah membantu:

1. Bapak Dr. R. Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Program Studi Informatika UII.
2. Ibu Septia Rani, S.T., M.Cs., selaku Dosen Pembimbing selama program magang.
3. Kak Jaka Pitana, selaku *Leader* Tim yang telah memberikan pengarahan kepada penulis selama magang di PT Tokopedia.
4. Kak Wiena Marcelina, selaku *supervisor* dan pembimbing lapangan di PT Tokopedia.
5. Segenap karyawan dan senior PT Tokopedia yang telah membantu dan membimbing penulis selama proses pelaksanaan magang.
6. Seluruh dosen Informatika Universitas Islam Indonesia yang telah memberikan banyak ilmu selama masa perkuliahan.
7. Orang tua dan teman-teman penulis yang telah memberikan dukungan dan semangat kepada penulis.

Atas dukungan dan bantuan dari berbagai pihak yang telah penulis sebutkan, penulis dapat menyelesaikan laporan tugas akhir ini sesuai dengan jadwal yang telah ditentukan. Semoga laporan tugas akhir ini dapat memberikan manfaat bagi para pembaca. Penulis menyadari masih banyak kekurangan dalam penulisan laporan ini, maka dari itu penulis mengharapkan kritik dan saran dari pembaca demi kesempurnaan laporan ini. Terima kasih.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 3 Agustus 2022

Faiq Dhimas Wicaksono

SARI

Pengujian perangkat lunak adalah serangkaian kegiatan yang dilakukan untuk mencari cacat atau kejanggalan pada aplikasi serta memastikan aplikasi sudah berjalan dengan baik sesuai persyaratan yang diharapkan. Proses ini dahulu sering dilakukan secara manual, tetapi saat ini sebagian besar sudah beralih ke otomatis karena pengujian manual sangat memakan waktu dan biaya sehingga rawan terjadi kesalahan (*human error*). Pengujian otomatis sendiri saat ini semakin dibutuhkan oleh semua perusahaan pengembang perangkat lunak karena sifatnya yang otomatis sehingga dapat dilakukan kapan saja secara berulang dan mengurangi upaya yang perlu dilakukan dalam pengujian secara manual. Penerapan pengujian otomatis dapat mempercepat siklus pengujian dan meningkatkan produktivitas suatu tim. Namun demikian, pengujian otomatis secara menyeluruh dengan banyaknya kasus uji akan membutuhkan durasi waktu yang cukup lama dalam proses pengujiannya. Oleh karena itu, diperlukan metode yang tepat dalam membuat skrip pengujian otomatis supaya dapat mengurangi langkah aksi yang berlebih dan berulang-ulang antar kasus uji.

Laporan tugas akhir ini bertujuan untuk membahas implementasi metode *Automation Journey* dalam pengujian otomatis dengan cara menambah logika baru dalam skrip pengujian untuk mengurangi langkah aksi yang kurang efektif serta mengubah urutan kasus pengujian agar saling berkaitan satu sama lain. Dengan adanya implementasi ini diharapkan dapat mempercepat keseluruhan rangkaian proses pengujian atau *End-to-end testing* secara otomatis dari awal hingga akhir.

Hasil dari penerapan pengujian otomatis menggunakan metode *Automation Journey* menunjukkan bahwa penggunaan metode ini memiliki akurasi pengujian yang sama dengan pengujian otomatis tanpa menggunakan metode apapun, tetapi waktu yang dibutuhkan dalam melaksanakan seluruh rangkaian pengujian jauh menjadi lebih cepat. Selain itu, dengan penerapan *Automation Journey* membuat proses identifikasi bug menjadi lebih mudah dan perawatan kode menjadi lebih gampang karena dengan dihapusnya penulisan skrip kode yang sama pada langkah aksi yang berulang menyebabkan kode akan menjadi lebih ringkas. Dengan demikian, dapat ditarik kesimpulan bahwa penerapan metode *Automation Journey* dapat membuat pengujian otomatis menjadi lebih cepat, efektif, dan skalabel sehingga cocok diterapkan pada pengujian otomatis yang perlu dijalankan secara terus menerus setiap hari supaya jika terdapat *bug* atau *error* dapat diketahui sedini mungkin.

Kata kunci: Pengujian perangkat lunak, Pengujian otomatis, *Automation Journey*, *End-to-end testing*, Siklus pengujian.

GLOSARIUM

Test Case	suatu rancangan kasus pengujian.
Test Suite	kumpulan test case atau kasus pengujian.
Agile	metodologi pengembangan perangkat lunak yang cepat.
Scrum	kerangka kerja penerapan dari metode agile.
Sprint	batasan waktu yang berisi periode dan aktivitas kerja.
Debug	langkah untuk menelusuri kesalahan kode program.
Black box	jenis pengujian perangkat lunak.
End-to-end	pengujian secara menyeluruh dari awal hingga akhir.
Deployment	proses penyeberan perangkat lunak.

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR **Error! Bookmark not defined.**
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI…… ix
GLOSARIUM x
DAFTAR ISI xi
DAFTAR TABEL xiii
DAFTAR GAMBAR xiv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Ruang Lingkup 3
1.3 Tujuan 5
1.4 Manfaat 5
1.5 Sistematika Penulisan 6
BAB II LANDASAN TEORI DAN TINJAUAN PUSTAKA 7
2.1 Manual Test 7
2.2 Automation Test 8
2.3 UI Test 8
2.4 Backend Test 9
2.5 Load Test 10
2.6 Test Case 10
2.7 Test Suite 11
2.8 Automation Journey 12
2.9 TopAds 13
2.10 Agile dan Scrum 14
2.11. Tinjauan Pustaka 15
BAB III PELAKSANAAN MAGANG 17
3.1 Manajemen Proyek 17
3.1.1 Sprint Kickoff 17
3.1.2 Membuat Kasus Uji dan Rencana Pengujian 18
3.1.3 Review dan Revisi Kasus Pengujian 18
3.1.4 Melakukan Manual Test Pada Staging 19
3.1.5 Deployment ke Production 19
3.1.6 Smoke Test Production 19
3.1.7 Membuat Pengujian Otomatis UI 20
3.1.8 Membuat Pengujian Otomatis Backend 21
3.1.9 Monitoring Load Test 21
3.1.10 Deploy dan Code Review 22
3.1.11 Memelihara Pengujian Otomatis 24
3.1.12 Sprint Review dan Retrospective 25
3.1.13 Weekly Meeting 25
3.2 Manajemen Pengujian 26
3.2.1 Jira Task 26
3.2.2 Test Case dan Test Set 27

3.2.3 Test Execution....28
3.2.4 Test Plan....28
3.3 Implementasi Penerapan Automation Journey....28
3.3.1 Memilih Kasus Uji dan Mengatur Urutannya....29
3.3.2 Memberi Percabangan Bersyarat Pada Awal Kasus Uji....31
3.3.3 Menambahkan Variabel Baru Pada Test Case dan Test Suite....32
3.3.4 Menambah Connector Pada Akhir Kasus Uji....34
3.3.5 Perbandingan Hasil Penggunaan Metode *Automation Journey*....35
BAB IV REFLEKSI PELAKSANAAN MAGANG....39
4.1 Relevansi Akademik....39
4.2 Pembelajaran Magang....41
4.2.1 Manfaat....41
4.2.2 Kendala, Hambatan, dan Tantangan....43
BAB V PENUTUP....45
5.1 Kesimpulan....45
5.2 Saran....45
DAFTAR PUSTAKA....47
LAMPIRAN....48

DAFTAR TABEL

Tabel 1.1 Ringkasan Pelaksanaan Aktivitas Magang 1
Tabel 2.1 *Test Suite* tanpa *Automation Journey* 12
Tabel 2.2 *Test Suite* dengan *Automation Journey* 12
Tabel 3.1 *Test Suite* tanpa Susunan *Automation Journey* 29
Tabel 3.2 *Test Suite* dengan Susunan *Automation Journey* 30
Tabel 3.3 Hasil pengujian pada eksperimen pertama 36
Tabel 3.4 Hasil pengujian pada eksperimen kedua 37
Tabel 3.5 Perbandingan Test Suite dengan 2 Metode Pengujian Berbeda 38
Tabel 4.1 Tabel Analisis Gap 39

DAFTAR GAMBAR

Gambar 2.1 Siklus Pengujian Otomatis 8
Gambar 2.2 *Test Suite* dan *Test Case* 11
Gambar 2.3 *Dashboard* TopAds 14
Gambar 3.1 Tahap Pengujian 17
Gambar 3.2 Kasus Pengujian 18
Gambar 3.3 *Log Viewer* pada Pengujian Otomatis 20
Gambar 3.4 *Load Test Report* pada *Slack channel.* 22
Gambar 3.5 *Pull Request* pada Git *repository* 23
Gambar 3.6 *Daily Test Report* 24
Gambar 3.7 Kumpulan Jira *Task Ticket* dalam *Sprint* 26
Gambar 3.8 *Test Set* Jira Tiket 27
Gambar 3.9 Langkah Aksi Sebelum Penerapan *Automation Journey* 30
Gambar 3.10 *Setup Test Case individualRun* 31
Gambar 3.11 Langkah Aksi Setelah Penerapan *Automation Journey* 32
Gambar 3.12 Tab *Variables* pada *tools* katalon studio 33
Gambar 3.13 Menambah Variabel *individualRun* 33
Gambar 3.14 *Setup Automation Journey* pada *Test Suite* 33
Gambar 3.15 Pengaturan *Variable Binding* dengan *Script* Variable 34
Gambar 3.16 Langkah Aksi *Close Browser* pada *Test Case* 35
Gambar 3.17 Penambahan Langkah Aksi untuk *Connector* 35
Gambar 3.18 *Test Suite* yang dijalankan Tanpa Metode *Automation Journey* 36
Gambar 3.19 *Test Suite* yang dijalankan Dengan Metode *Automation Journey* 36
Gambar 3.20 *Test Suite* 2 yang dijalankan Tanpa Metode *Automation Journey* 37
Gambar 3.21 *Test Suite* 2 yang dijalankan Dengan Metode *Automation Journey* 38

BAB I
PENDAHULUAN

1.1 Latar Belakang

Program magang merupakan salah satu faktor penting yang dapat membantu kesiapan karir mahasiswa setelah lulus. Lingkungan perkuliahan mungkin melibatkan diskusi, debat, interaksi teman sebaya, dan pengalaman belajar bersama, tetapi penting bagi mahasiswa untuk mencari pengalaman dalam lingkungan yang lebih profesional. Melalui program magang, mahasiswa berkesempatan untuk mengembangkan konsep akademik yang sudah dipelajari pada masa perkualiahan dan mempraktekkannya secara langsung pada pekerjaan yang diberikan saat pelaksanaan magang. Penulis telah melaksanakan aktivitas magang di PT Tokopedia yang merupakan salah satu platform jual beli online terbesar di Indonesia. PT Tokopedia adalah perusahaan teknologi dengan misi untuk mendemokratisasikan dan memberdayakan perdagangan melalui teknologi untuk semua orang. Sejak didirikan pada tahun 2009, Tokopedia telah menjadi kekuatan yang mempelopori transformasi digital di Indonesia. Tokopedia kini memiliki lebih dari 100 juta pengunjung tiap bulannya dan terdapat 12 juta penjual terdaftar sebagai mitra tokopedia. Di samping menjadi platform untuk mempertemukan pembeli dan penjual secara online, Tokopedia juga terus mengembangkan produk lainnya, seperti produk *fintech*, kredit modal bisnis, dompet digital, dan lainnya. (Tokopedia, 2020).

Tabel 1.1 Ringkasan Pelaksanaan Aktivitas Magang

No	Aktivitas	Durasi Waktu
1	*Pre-onboarding* Tokopedia	23 Aug - 5 Sept 2021
2	*Onboarding* TopAds	6 Sept - 12 Sept 2021
3	Mempelajari *tools* dan *framework* pengujian	13 Sept - 19 Sept 2021
4	Backlog dan Sprint kick off TopAds	Setiap 2 minggu sekali
5	Membuat pengujian otomatis TopAds	20 Sept - 17 Okt 2021
6	*Weekly Meeting* TE x PM x Dev	Setiap minggu sekali
7	Membuat kasus pengujian TopAds	18 Okt - 31 Okt 2021
8	Melakukan pengujian manual TopAds	1 Nov - 22 Nov 2021
9	*Monitoring Load Test* TopAds	Setiap bulan sekali

10	Pengujian TopAds pada projek TCO	23 Nov - 31 Des 2021
11	Memperbaiki pengujian otomatis TopAds	1 Jan - 22 Feb 2022

Aktivitas magang telah dilaksanakan selama 6 bulan oleh penulis terhitung sejak 23 Agustus 2021 hingga 22 Februari 2022, seperti terlihat seluruh rangkaian aktivitas magang dan jangka waktunya pada Tabel 1.1 di atas. Selama magang penulis berada pada tim TopAds dan bergabung dengan 1 projek yaitu *Total Cost of Ownership* (TCO) pada bulan November hingga Desember. TopAds adalah salah satu fitur di Tokopedia yang bertujuan untuk promosi toko dan produk yang bisa digunakan oleh seluruh *merchant* di Tokopedia. Dengan fitur ini akan membantu produk dan toko menjadi lebih mudah ditemukan oleh pembeli, yang kemudian dapat meningkatkan potensi penjualan dari *merchant* tersebut. Sedangkan TCO adalah projek yang bertujuan untuk membantu penjual tokopedia yang memiliki toko/gudang dengan banyak lokasi supaya dapat menjangkau pembeli mereka dengan total harga lebih terjangkau dan proses pengiriman yang lebih cepat. Dengan algoritma *Cheaper Shipping Price* (CSP) akan membantu penjual untuk menentukan lokasi toko/gudang yang akan mengirimkan produk tersebut ke pembeli yang memiliki harga ongkir paling murah. Dalam proses pengujian projek TCO, penulis membantu pada bagian TopAds untuk memastikan bahwa harga dan lokasi produk pada iklan-iklan di Tokopedia sudah sesuai dengan ekspektasi yang diharapkan berdasarkan lokasi pengiriman pembeli.

Selama proses pelaksanaan magang ditemukan bahwa PT Tokopedia dan perusahaan teknologi lainnya saat ini menghadapi tuntutan kualitas produk yang semakin meningkat setiap harinya, tetapi dengan waktu pengembangan yang lebih singkat. Hal ini dapat mempengaruhi keseluruhan proses pengembangan aplikasi dari awal hingga akhir atau biasa disebut *Software Development Life Cycle* (SDLC). Salah satu tahapan krusial yang terpengaruh adalah tahap pengujian (*testing*). Pada tahap ini seorang *tester* akan melakukan pengujian secara menyeluruh pada sistem untuk menilai apakah *software* dapat bekerja sesuai fungsionalitas yang diharapkan dan memastikan bahwa sistem bebas dari cacat atau *error*. Proses ini dahulu sering dilakukan secara manual, tetapi sangat memakan waktu dan biaya sehingga rawan terjadi kesalahan (*human error*). Padahal untuk menghindari banyaknya *bug* dalam perangkat lunak diperlukan pengujian secara menyeluruh, sebelum perangkat lunak yang dikembangkan dirilis kepada publik atau pengguna umum (Martantoh, 2020). Jika hanya puluhan kasus uji (*test case*) mungkin bisa dilakukan secara manual, tetapi jika aplikasi sudah sangat berkembang dan

terdapat hingga ribuan kasus uji maka usaha *tester* akan habis jika harus melakukan semua pengujian tersebut secara manual.

Berdasarkan permasalahan di atas maka diusulkan pengujian otomatis menggunakan metode *Automation Journey* sebagai solusi untuk mengatasinya. Pengujian tersebut dapat berjalan otomatis secara singkat oleh sistem tanpa interaksi manusia sehingga frekuensi pengujian dapat meningkat dan memberikan umpan balik lebih cepat kepada pengembang. Saat ini fokus pengujian sudah bergeser yang sebelumnya semua pengujian dilakukan secara manual, saat ini sudah dilakukan secara otomatis oleh robot yang skripnya sudah dibuat sesuai kasus pengujian. Dalam membuat pengujian otomatis akan lebih banyak *setup* di awal. Hal ini dilakukan untuk memastikan bahwa implementasi kode dalam skrip itu sudah efektif sehingga dapat melakukan pengujian secara otomatis dengan hasil yang akurat, dijalankan secara cepat, dan tidak banyak proses perbaikan terhadap skrip seiring perkembangan waktu.

Dengan penerapan solusi tersebut maka diharapkan dapat membantu proses pengembangan perangkat lunak dalam proses SDLC khususnya dan menerapkan pengujian otomatis dalam *Continuous Integration* (CI) Pipeline. Dalam proses CI atau pengintegrasian kode ke dalam repositori terdapat proses verifikasi dengan cara menjalankan proses pengujian secara otomatis, cepat, dan sering yang dilakukan secara terus menerus setiap ada perubahan atau fitur baru pada aplikasi supaya jika terdapat *bug* atau *error* dapat diketahui sedini mungkin (NKD, 2020).

1.2 Ruang Lingkup

Penulis melakukan magang di Tokopedia pada divisi Teknologi sebagai *Software Engineer - Engineering Productivity* (SE-EP) yang bertujuan untuk mengoptimalkan proses *engineering* supaya lebih cepat dalam proses *deliver* kepada pengguna serta memastikan produk dan layanan sudah memenuhi kebutuhan yang diperlukan. Selama proses pelaksanaan magang penulis hanya tergabung dengan 1 tim yaitu TopAds, dengan atasan penulis adalah Jaka Pitana sebagai *leader team* dan Wiena Marcelina sebagai sebagai *senior SE-EP* sekaligus mentor penulis.

Program magang berlangsung selama enam bulan dengan posisi sebagai *engineering productivity* atau secara umum bisa disebut *Tester*. Selama magang terdapat beberapa tugas dan tanggung jawab yang menjadi kewajiban dari penulis, yaitu sebagai berikut:

a. Membuat kasus pengujian

Test case atau biasa disebut kasus uji adalah suatu rangkaian tindakan yang dapat dilakukan oleh *tester* untuk melakukan verifikasi terhadap fitur atau fungsi tertentu dari sebuah aplikasi. *Test case* bertujuan untuk memastikan bahwa sistem dapat dijalankan dengan baik sesuai dengan ekspektasi dokumen PRD. *Test case* memiliki beberapa komponen, seperti *test case id*, *test case title*, *precondition*, *expected result*, *actual result*, dan status. Dari *test case* ini dapat diketahui bahwa fitur atau fungsi baru tersebut sudah sesuai harapan atau belum.

b. Melakukan pengujian manual

Manual *test* adalah pengujian pada fitur tertentu yang dilakukan secara manual menggunakan kasus pengujian yang sudah dibuat sebelumnya. Penulis melakukan pengujian pada *environment staging* dan *link kratos*. Kratos adalah *website* untuk melakukan pada *staging* dan beta envinronment. Link halaman kratos akan diberikan oleh *frontend developer* saat fitur yang dikembangkan sudah siap di uji. Dalam melakukan *testing*, SEEP dan QA tidak hanya mengikuti kasus pengujian secara umum, tetapi juga dituntut untuk mencari *bug* atau kejanggalan pada fitur tersebut serta visual feedback untuk memberikan opini mengenai tampilan UI/UX.

Jika semua kasus uji sudah berhasil lulus, lalu *bug* yang ditemukan sudah diperbaiki, dan tidak ada *bug* lagi yang ditemukan, maka SEEP dan QA dapat memberi persetujuan terhadap fitur tersebut melalui tiket jira dengan cara melampirkan hasil pengujian serta mengubah status *task* dari *in testing* menjadi *ready to merge*d. Nantinya jika PM sudah memberikan persetujuan maka fitur tersebut sudah dapat masuk pada *deployment* berikutnya ke *production environment*.

c. Melakukan pengujian otomatis

Pengujian otomatis adalah pengujian yang bergantung pada *script test* untuk membandingkan hasil sebenarnya dengan hasil yang diharapkan. Pada pengujian ini semuanya dilakukan secara otomatis, sehingga dapat dilakukan secara berulang untuk *regression testing* setiap harinya.

Sebelum membuat naskah pengujian otomatis, *tester* perlu membaca dan memahami terlebih dahulu terkait fitur yang akan uji, serta menyiapkan *tools* pengujian yaitu Katalon Studio untuk pengujian antarmuka dan *Tokopedia Rest Executioner* (T-Rex) untuk pengujian *backend.*

d. Memelihara pengujian otomatis

Tokopedia menerapkan *agile* sebagai metodologi pengembangan aplikasi yang mana akan sering terdapat perubahan terhadap fitur-fitur yang sudah ada, sehingga menyebabkan beberapa kasus uji menjadi tidak relevan lagi. Oleh karena itu *tester* perlu memelihara pengujian otomasi yang sudah ada. Salah satu caranya adalah dengan memperbaiki *script automation* supaya mutakhir dan sesuai dengan alur kerja aplikasi saat ini. Pengujian otomatis yang sudah dibuat akan berjalan setiap hari pada *server* tokopedia. Setiap tim akan mendapat laporan dari *daily automation* tersebut. Jika dari pengujian otomatis terdapat kasus uji yang *error* karena *bug* atau tidak berjalan sesuai harapan, maka tim terkait harus segera memeriksa *script* pengujian tersebut.

1.3 Tujuan

Tujuan dari penerapan metode *automation journey* dalam pengujian otomatis yaitu:

1. Menganalisis teknik pengujian otomatisasi yang cepat dan akurat.
2. Bentuk pertanggung jawaban dari program magang di Tokopedia sebagai bagian penjaluran prodi informatika UII.
3. Melihat performa aplikasi dan *website* dalam setiap rilis *deployment.*
4. Membandingkan hasil pengujian otomatisasi dengan metode *automation journey* dan dengan metode lain ataupun tanpa metode sama sekali.
5. Sebagai langkah pengujian lebih lanjut yang lebih efektif dan skalabel.
6. Mencari *error* dan kelemahan sistem untuk meningkatkan keamanan perangkat lunak.
7. Menyediakan informasi untuk meningkatkan kualitas produk perangkat lunak.

1.4 Manfaat

Terdapat beberapa manfaat yang didapatkan dari hasil penerapan metode *automation test journey*, yaitu:

1. Mengetahui teknik dan skema pengujian otomatisasi yang skalabel, mudah dibaca, dan mudah dikembangkan (*reusable*).
2. Mengetahui fitur-fitur tokopedia yang harus dioptimalkan agar pengalaman aplikasi pengguna lebih memuaskan.
3. Menyediakan informasi untuk mendeteksi eror secara dini.
4. Menyediakan informasi yang dapat mencegah terjadinya eror yang lebih besar.
5. Mengurangi risiko kegagalan dalam perangkat lunak.
6. Memastikan fitur dalam perangkat lunak sudah sesuai dengan kebutuhan pelanggan.

7. Hasil dari penerapan teknik ini dapat dijadikan referensi untuk pengujian pada pengembangan perangkat lunak supaya lebih optimal.
8. Mendapatkan pengetahuan mengenai pentingnya pengujian otomatis dalam penerapan pengujian secara menyeluruh.
9. Mendapatkan ide-ide baru untuk mengerjakan proyek lain menggunakan pengembangan metode pengujian otomatis.
10. Mengetahui standarisasi yang diterapkan pada *e-commerce* dalam melakukan pengujian yang valid dan skalabel.

1.5 Sistematika Penulisan

Sistematika Penulisan adalah rangkaian urutan penulisan dari suatu dokumen. Adapun penulisan tugas akhir pada laporan ini terbagi menjadi lima bab, dengan sistematika penulisan sebagai berikut:

a. BAB I PENDAHULUAN

   Bab ini berisi latar belakang yang merupakan landasan dan dasar dari laporan ini serta penjelasan mengenai ruang lingkup magang, tujuan, manfaat, dan sistematika penulisan.

b. BAB II LANDASAN TEORI

   Bab ini membahas mengenai konsep, detail, teori, dan definisi yang digunakan sebagai landasan dari tugas yang dikerjakan dalam laporan ini, beserta tinjauan Pustaka.

c. BAB III PELAKSANAAN

   Bab ini menjelaskan pelaksanaan magang, mulai dari tahapan persiapan hingga implementasi dari teori-teori dan metode yang telah diajarkan, beserta bukti dan gambar proyek yang dikerjakan.

d. BAB IV REFLEKSI

   Bab ini membahas relevansi akademik terkait gap teori akademik dengan pelaksanaan di lapangan selama magang, beserta refleksi yang diperoleh selama magang baik dari sudut padang teknis maupun non teknis. Refleksi ini berfokus kepada kerangka kerja yang digunakan dalam mengerjakan tugas magang.

e. BAB V PENUTUP

   Bab ini berisi kesimpulan dari keseluruhan laporan yang telah disampaikan pada bab-bab sebelumnya, beserta dengan saran yang dapat digunakan oleh individu maupun tim untuk menjadi lebih baik lagi.

BAB II
LANDASAN TEORI DAN TINJAUAN PUSTAKA

2.1 Manual Test

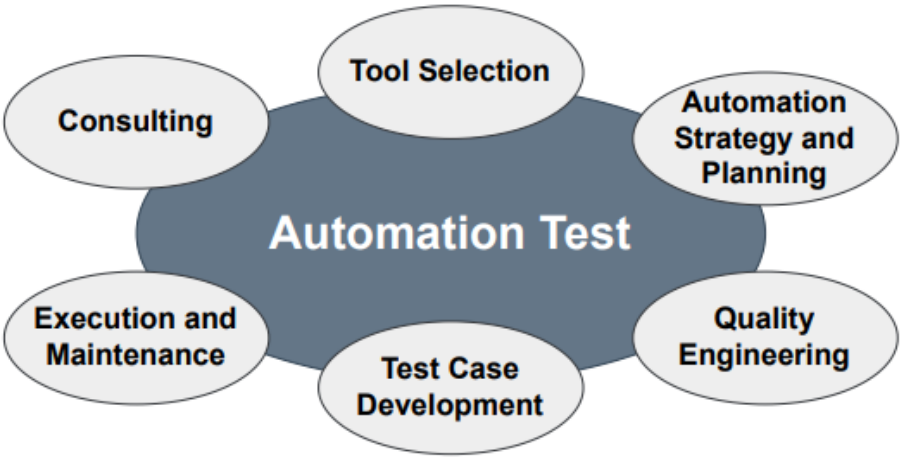
Manual *test* adalah pengujian yang dilakukan secara manual menggunakan kasus uji yang sudah dibuat untuk memastikan tidak adanya *bug* dalam aplikasi (Rizky, 2019). Manual *test* berperan penting sebagai tahap pengujian paling awal pada pengembangan fitur baru, fitur eksperimen, dan fitur dengan data dinamis. Untuk kasus-kasus tersebut maka pengujian manual yang akan dipakai oleh seorang *tester*. Pengujian manual pada *marketplace* secara umum terdapat 3 jenis, yaitu:

1. *Sanity Test*: merupakan pengujian yang berfokus pada fungsi baru yang dikembangkan saja. *Sanity testing* biasa dilakukan pada fitur yang benar-benar baru dan tidak memiliki kaitan dengan fitur lain sama sekali, sehingga seorang *tester* cukup memastikan fitur ini dapat berjalan dengan baik (T, 2020). *Sanity test* biasa dilakukan pada *staging* environment.
2. *Smoke Test*: pengujian yang berfokus kepada keseluruhan perangkat lunak, tetapi hanya pada fungsi-fungsi utama yang bersifat critical dan umum. Pengujian ini bertujuan untuk memastikan bahwa *flow* tetap berjalan dengan baik dan aman (T, 2020). *Smoke test* biasa dilakukan di *production* setelah ada unggahan fitur-fitur baru, yang bertujuan memastikan bahwa fitur-fitur utama aplikasi aman dan dapat berjalan dengan baik.
3. *Regression Test*: pengujian secara menyeluruh dan sedetail mungkin pada semua fitur dari yang utama hingga yang jarang digunakan, untuk memastikan bahwa fitur dapat berjalan 100% (T, 2020). *Regression test* biasa dilakukan setelah terdapat fitur baru yang berkaitan dengan fitur-fitur lainnya. Selain itu *regression test* merupakan pengujian yang rutin dilakukan oleh *tester* mulai dari seminggu sekali, seminggu dua kali, dan lain-lain. *Regression test* bertujuan memastikan perubahan yang dilakukan selama pengembangan tidak menyebabkan munculnya *bug* lama ataupun baru. *Regression test* juga untuk memastikan bahwa tidak ada *bug* tua yang muncul akibat deployment yang terus menerus terjadi.

2.2 Automation Test

Automation testing merupakan pengujian yang bergantung pada *script* uji yang telah dibuat oleh *tester* dan dilaksanakan secara otomatis untuk membandingkan hasil sebenarnya dengan hasil yang diharapkan. Kelebihan pengujian otomatis adalah dapat dilakukan secara berulang, memberikan umpan balik dengan lebih cepat, dan berjalan otomatis tanpa interaksi manusia (Rizky, 2019). Dengan demikian, pengujian ini dapat meningkatkan frekuensi pengujian dan memungkinkan untuk melakukan uji regresi tanpa intervensi dari manual *tester* sama sekali. Dahulu pengujian sering dilakukan secara manual, tetapi seiring berjalannya waktu proses tersebut sangat memakan waktu, banyak biaya, dan rawan terjadi kesalahan, sehingga dibutuhkan pengujian otomatis yang diusulkan sebagai solusi untuk ini.

Gambar 2.1 menunjukkan poin-poin penting dari pengujian otomatis untuk mencapai siklus SDLC yang produktif dan skalabel. Dengan memasukkan pengujian otomatis ke dalam SDLC akan memungkinkan suatu tim mencapai hal-hal, seperti *continuous integration* and *continuous delivery* (CI/CD), *DevOps*, *Quality Engineering*, *Risk Mitigation*, dan pengiriman lebih cepat kepada pengguna.



Gambar 2.1 Siklus Pengujian Otomatis

2.3 UI Test

User Interface (UI) merupakan tampilan program yang dilihat langsung oleh pengguna atau biasa disebut *front-end* sebagai media interaksi antara pengguna dengan perangkat lunak. Pengujian UI atau *front-end* adalah pengujian terhadap antarmuka dengan cara melakukan urutan peristiwa sesuai urutan user story dan kasus pengujian yang dibuat melalui sejumlah

komponen antarmuka, seperti klik tombol, membuka tab menu, memasukkan teks, dan lain-lain (Martantoh, 2020).

Pengujian antarmuka tidak memerlukan informasi apapun dari *back-end* seperti respon *web service*, *database*, dan sejenisnya, tetapi perlu memeriksa semua fungsionalitas aplikasi sudah berjalan sesuai yang diharapkan. Selain itu pada UI *testing*, *tester* perlu memeriksa waktu respon yang ditimbulkan dari kompleksitas rancangan antarmuka, karena kecepatan aplikasi dalam memproses suatu permintaan sangat berpengaruh terhadap pengalaman pengguna. Pengujian antarmuka bertujuan untuk memastikan aplikasi telah memberikan layanan terbaik kepada pengguna, supaya pengguna lebih mudah dan nyaman dalam menggunakan aplikasi.

Proses pengujian antarmuka dapat dilakukan secara manual maupun otomatis. Meskipun demikian, pengujian dengan cara manual memiliki lebih banyak kekurangan seperti cakupan pengujian yang kurang luas, karena *tester* harus melakukan hal yang berulang-ulang sehingga rawan terjadi kesalahan jika terdapat halaman antarmuka yang terlewat dalam pengujian. Selain itu, kekurangan lainnya saat pengujian manual yaitu *tester* tidak bisa melakukan pencatatan waktu respon secara otomatis, sehingga hasil pengujian menjadi kurang valid (Martantoh, 2020). Kelemahan-kelemahan tersebut dapat ditutupi dengan pengujian otomatis, karena dengan *automation testing* memungkinkan pengulangan urutan pengujian secara lebih cepat dan tepat. Selain itu, pengujian otomatis juga memungkinkan pencatatan waktu respon secara otomatis.

2.4 Backend Test

Back end adalah segala hal yang berhubungan dengan *server* dan *database*. *Back-end* tidak berinteraksi secara langsung dengan pengguna akhir karena sifatnya hanya menyampaikan informasi. *Back-end* lebih berfokus pada *database*, *scripting*, dan *server website* (*server-side*) (Damayanti, 2020). Tujuan pengujian *back-end* adalah untuk memastikan lapisan *web service* dan *database* dari perangkat lunak bebas dari cacat basis data seperti kebuntuan, kerusakan data, atau kehilangan data.

Pengujian *back-end* juga dikenal sebagai pengujian basis data. Data yang dimasukkan di front-end akan disimpan di *database back-end*. Basis data dapat berupa SQL *Server*, MySQL, Oracle, DB2, dan lain-lain. Data akan diatur dalam tabel sebagai catatan dan digunakan untuk mendukung konten halaman. Pengujian *database* atau *back-end* penting karena jika tidak dilakukan dengan benar dapat menyebabkan beberapa komplikasi serius seperti kebuntuan, korupsi data, dan kehilangan data.

Proses pengujian *back-end* dapat dilakukan secara manual ataupun otomatis. Namun, pengujian otomatis lebih sering dilakukan karena dapat mencakup banyak kasus uji sekaligus dalam satu waktu pengujian. *Back-end automation* diperlukan untuk deteksi awal pada kasus pengujian yang rawan terjadi *deadlock*, *data loss*, dan *data corruption*. Pengujian ini dapat digunakan untuk memastikan bahwa proses memasukkan data ataupun menampilkan data pada front-end akan berjalan aman dan sesuai dengan harapan.

2.5 Load Test

Load Test atau uji ketahanan *website* adalah pengujian terhadap pembangunan dan pengembangan *website* atau perangkat lunak yang bertujuan untuk memastikan bahwa sistem tersebut kuat saat dikunjungi oleh banyak pengguna dalam waktu yang bersamaan. Pengujian ini sebagai tolak ukur apakah *website* atau perangkat lunak tetap dapat bekerja secara maksimal walaupun digunakan oleh banyak pengguna.

Proses pengujian beban (*load test*) dilakukan secara otomatis menggunakan bantuan locust untuk menciptakan banyak *bot* yang akan mengakses *website* atau perangkat lunak secara bersamaan sebagai simulasi pengujian layaknya keadaan yang sebenarnya pada dunia nyata. Pengujian tipe ini lebih banyak dipilih oleh para *developer* dan perusahaan untuk menguji sistem yang telah dibuat dibandingkan dengan tipe pengujian lainnya.

Di Tokopedia *Test Engineer* bertugas melakukan *monitoring load test* bersama dengan tim *backend* pada waktu yang sudah ditetapkan. *Load test* biasanya terdiri dari 2-3 batch dan dilaksanakan pada dini hari dimana pengunjung / traffic tokopedia sedang tidak ramai, sehingga hasil pengujian bisa lebih akurat. *Load test* sendiri tidak dilakukan setiap hari seperti automated *testing*, untuk uji ketahanan *website* sebulan hanya 3-4 kali saja karena membutuhkan sumber daya yang sangat besar dalam melakukan pengujian tersebut dan biasanya *Big Load Test* dilakukan sebagai persiapan menghadapi *event* besar tokopedia seperti WIB (Waktu Indonesia Belanja) dan *event* double date seperti 11.11, 12.12 dimana pengunjung *website* dapat meningkat hingga 2 kali lipat dari hari biasa.

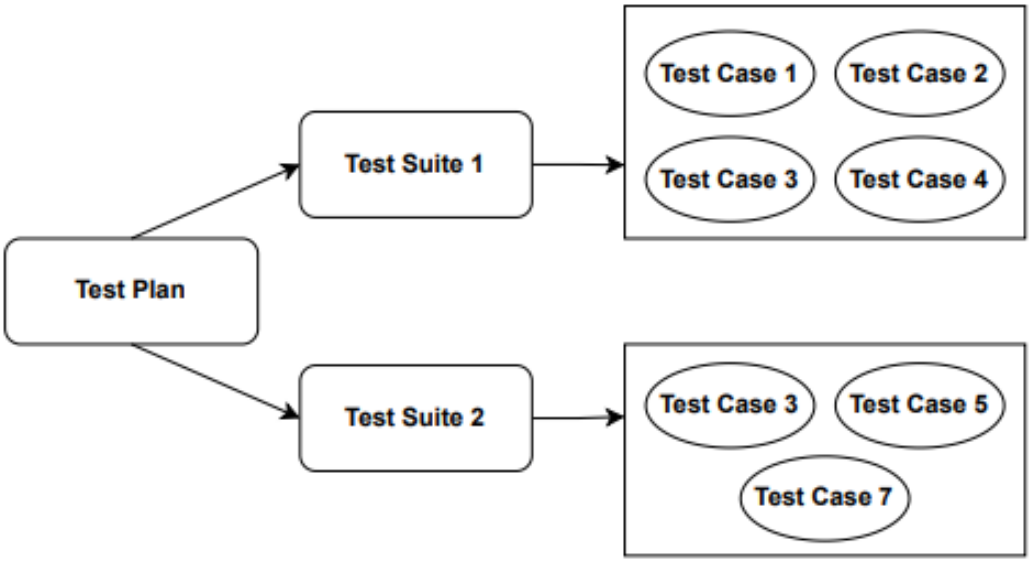
2.6 Test Case

Test case atau biasa disebut kasus uji adalah suatu dokumen dengan rangkaian tindakan yang dapat dilakukan oleh *tester* untuk melakukan verifikasi terhadap fitur atau fungsi tertentu dari sebuah aplikasi. *Test case* memiliki sekumpulan data uji, prasyarat, hasil yang diharapkan, dan kondisi pasca yang dikembangkan sesuai skenario kasus uji tertentu guna melakukan

verifikasi apakah perangkat lunak sudah berjalan sesuai dengan yang diharapkan atau belum (Damayanti, 2020).

Dalam membuat kasus uji, skenario *testing* dibagi menjadi dua yaitu skenario positif dan negatif. Skenario positif adalah kasus pengujian menggunakan alur yang sesuai dengan harapan sistem pada marketplace sehingga tidak ada notifikasi atau *warning error* yang muncul saat sedang dijalankan, sedangkan skenario negatif adalah kasus pengujian dengan alur yang menyalahi aturan sistem seperti mengisi harga barang menjadi 0 atau minus dan sebagainya. Skenario negatif membutuhkan sistem untuk menolak permintaan tersebut dan mengirim notifikasi pesan peringatan.

2.7 Test Suite



Gambar 2.2 *Test Suite* dan *Test Case*

Test suite adalah sebuah wadah yang memiliki serangkaian *test case* seperti terlihat pada Gambar 2.2 yang dapat dijalankan secara bersamaan (*pararel*) ataupun satu persatu (*sequence*). Suatu kasus uji dapat ditambahkan pada beberapa *test suite* secara bersamaan, karena *Test suite* dibuat berdasarkan siklus atau lingkup dari suatu rencana pengujian (*test plan*) (Damayanti, 2020).

Keuntungan dari *test suite* adalah memungkinkan *tester* menggabungkan beberapa *test case* untuk menciptakan kondisi pengujian yang unik dan bervariasi antar *test suite* satu dan

lainnya. Dengan *test suite* ini pengujian dapat dilakukan dengan lebih lengkap seperti E2E (*end-to-end testing*) yang tidak dapat diproduksi oleh suatu *test case* itu sendiri.

2.8 Automation Journey

Automation journey adalah sebuah teknik yang dapat dilakukan saat membuat skrip pengujian otomatis yang bertujuan untuk mengurangi waktu eksekusi dan meningkatkan tingkat keberhasilan. Penerapan *automation journey* dapat dilakukan dengan cara mengubah susunan kasus uji, serta menambah percabangan kondisi di awal dan akhir kasus uji. *Automation journey* diterapkan pada *test suite* yang memiliki kasus pengujian dengan langkah aksi yang berulang dan serupa, seperti contoh kasus pengujian yang terlihat pada Tabel 2.1 berikut.

Tabel 2.1 *Test Suite* tanpa *Automation Journey*

Test Suite		
Test Case A	Test Case B	Test Case C
1.) Open *browser* 2.) *Login* 3.) Onboarding X 4.) Do checking A	1.) Open *browser* 2.) *Login* 3.) Onboarding X 4.) Do checking B	1.) Open *browser* 2.) *Login* 3.) Onboarding X 4.) Do checking C

Dalam contoh tersebut *tester* dapat melewati langkah berulang yaitu "Open browser A" lalu "*Login*" dan melakukan "Onboarding X" untuk mengurangi waktu eksekusi secara keseluruhan dan meningkatkan tingkat keberhasilan. Oleh karena itu, akan ada penambahan kecil pada langkah akhir dari contoh pertama dan kedua untuk mengembalikan keadaan di mana pengecekan B dan C dimulai. Dalam penerapannya tidak semua *test case* dan *test suite* dapat dengan mudah diubah menjadi *automation journey*, karena perlu rencana untuk membuat *test suite* supaya menjadi satu aliran tes.

Tabel 2.2 *Test Suite* dengan *Automation Journey*

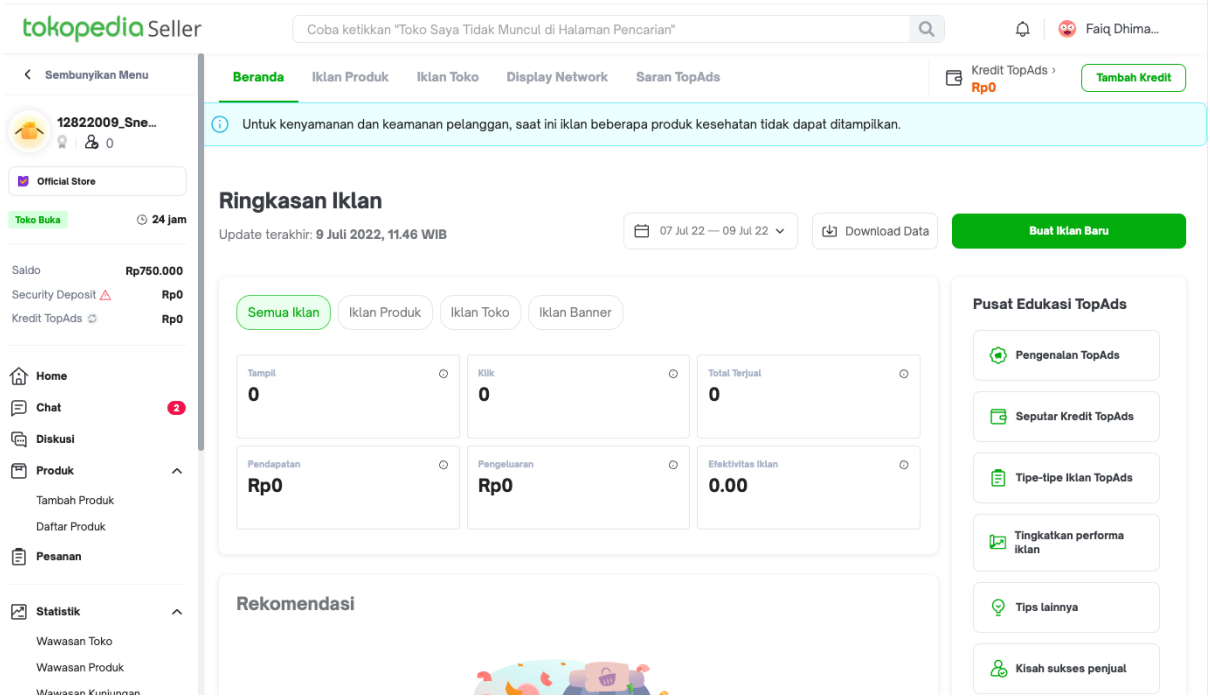
Test Suite with Automated *Journey*		
Test Case A	Test Case B	Test Case C
1.) Open *browser* 2.) *Login* 3.) Onboarding X 4.) Do checking A 5.) Press back	1.) If *Individual*Run 2.) Do checking B 3.) Press back	1.) If *individual*Run 2.) Do checking C

Dengan penerapan *automation journey* seperti terlihat pada Tabel 2.2, maka tahap perulangan yang sama dan duplikat hanya dilakukan sekali saja yaitu pada *test case* yang pertama. Dengan demikian *test case* selanjutnya dapat langsung melakukan pengecekan pada langkah aksi *test case* selanjutnya. Namun, jika *test case* sebelumnya memiliki hasil pengujian yang gagal, maka *test case* berikutnya harus melakukan tahap dari awal lagi.

Kelebihan penggunaan *automation journey* adalah rangkaian pengujian akan berjalan lebih cepat karena tidak perlu repot membuka dan menutup *browser* ataupun melakukan langkah-langkah yang sama berulang, karena sudah ada kondisi percabangan untuk menangani kasus-kasus tersebut. Namun, kelemahan penggunaan metode tersebut adalah *tester* perlu meluangkan lebih banyak waktu untuk menemukan dan memikirkan kasus uji yang dapat dijalankan melalui alur perjalanan logis *automation journey*.

2.9 TopAds

TopAds adalah fitur promosi toko dan produk yang bisa digunakan oleh seluruh merchant di Tokopedia. Fitur TopAds memungkinkan produk toko bisa tampil teratas di halaman strategis, serta lebih mudah dikunjungi. TopAds terbagi menjadi 2, yaitu dashboard dan display. Dashboard adalah tampilan yang menampilkan informasi dalam bentuk matriks, angka, ataupun visualisasi data yang bertujuan membantu penjual dalam membuat TopAds yang sesuai dan tepat berdasarkan dari informasi yang ada. Pada halaman dashboard TopAds, penjual dapat memilih fitur-fitur TopAds seperti terlihat pada Gambar 2.3 terdapat Iklan produk, iklan toko, iklan banner, iklan tanpa modal, dan iklan google.



Gambar 2.3 *Dashboard* TopAds

Sedangkan display adalah tampilan TopAds yang muncul pada suatu halaman sesuai dengan pilihan penjual saat membuat TopAds melalui dashboard. Dengan fitur ini akan membantu produk dan toko menjadi lebih mudah ditemukan oleh pembeli, yang kemudian dapat meningkatkan potensi penjualan dari *merchant* tersebut.

2.10 Agile dan Scrum

Agile adalah metodologi pengembangan aplikasi yang didasari dengan prinsip kerja yang memerlukan adaptasi dengan cepat, karena agile mengharuskan anggota tim untuk selalu siap terhadap segala perubahan yang akan terjadi, dan bersifat fleksibel ketika menghadapi suatu permasalahan (Binar, 2022). Agile hanyalah sebuah yang berisikan prinsip atau sifat untuk menyelesaikan masalah secara adaptif. Maka dari itu penerapan Scrum dibutuhkan untuk mewujudukan agile tersebut menjadi sebuah langkah-langkah solusi. Scrum merupakan metode yang mengatur manajemen dan pelaksanaan pengembangan aplikasi. Scrum membantu koordinasi antar tim supaya lebih mudah, terstruktur, dan kolaboratif antar anggota tim. Scrum berguna untuk mempercepat rilis produk kepada pengguna dengan produktivitas dan kualitas yang tinggi.

Komponen penting dalam Scrum adalah *sprint*. *Sprint* merupakan sejumlah rencana pekerjaan yang harus diselesaikan oleh tim dalam periode waktu yang telah ditentukan. Tujuan

dari *sprint* adalah untuk memecah *task* pekerjaan menjadi beberapa *task* yang berukuran lebih kecil, sehingga memungkinkan tim untuk merencanakan satu *sprint* dalam satu waktu dan menyesuaikan *sprint* ke depannya berdasarkan hasil *sprint* sebelumnya.

2.11 Tinjauan Pustaka

Terdapat beberapa penelitian yang berkaitan dengan pengujian otomatis antarmuka menggunakan Katalon Studio. Penelitian yang pertama adalah penelitian yang dilakukan oleh (Kosasih & Budi Cahyono, 2020) yang berjudul "*Automation Testing* Tool Dalam Pengujian Aplikasi The Point Of Sale". Penelitian tersebut menjelaskan metode pengujian otomatis menggunakan *automation testing tool* yaitu katalon studio, mulai dari tahap pembuatan *test case* secara manual hingga akhirnya menjadi pengujian yang dapat berjalan secara otomatis. Penelitian tersebut berfokus menganalisis efektivitas penerapan pengujian otomatis pada aplikasi *point of sale* dengan cara membandingkan dengan pengujian secara manual. Berdasarkan hasil penelitian tersebut ditemukan bahwa pengujian otomatis sangat membantu terutama dalam pengujian yang memiliki data statis dan perlu dilakukan secara berulang-ulang.

Penelitian kedua adalah penelitian yang berjudul "Pengujian Black Box Aplikasi *Mobile* Menggunakan Katalon Studio" oleh (Ardi & Putro, 2021). Penelitian ini membahas tentang penerapan pengujian otomatis dalam pengembangan aplikasi sebagai solusi dari pengujian manual yang kurang maksimal dan rawan terjadi kesalahan manusia. Penelitian tersebut juga menggunakan katalon studio sebagai *tools* yang digunakan untuk melakukan pengujian otomatis. Hasil penelitian tersebut memberi kesimpulan bahwa pengujian otomatis lebih efektif dibanding dengan pengujian secara manual, terutama dari sudut pandang *Tester* karena pengujian dapat dikukan secara lebih cepat, secara berulang, data hasil pengujian lebih detail, dan akurat.

Penelitian mengenai pengujian otomatis juga dilakukan oleh (Muhtadi dkk., 2019) dengan judul "Analisis GUI *Testing* pada Aplikasi *E-Commerce* menggunakan Katalon". Penelitian ini membahas analisis *response time* yang dibutuhkan dari halaman-halaman utama *website* pada 3 aplikasi *e-commerce* yaitu JD.ID, Tokopedia, dan Bukalapak. Halaman *website* yang dilakukan pengujian adalah halaman utama (*landing page*), halaman masuk (*login*), halaman pencarian *(search*), halaman detail produk (*product detail page*), dan halaman transaksi. Hasil dari penelitian tersebut memperlihatkan bahwa *response time* dari antarmuka halaman *website* dipengaruhi oleh banyak faktor tidak hanya dari kompleksitas GUI *website*,

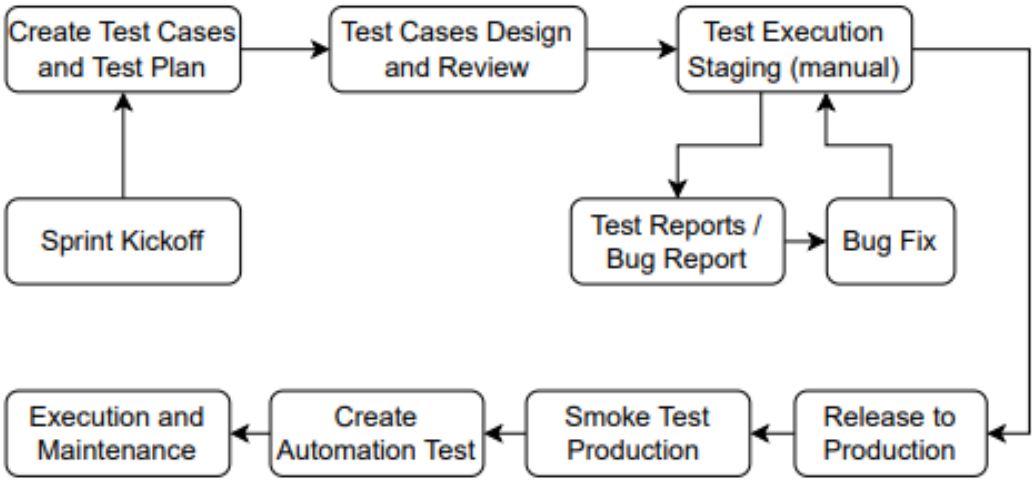
sehingga penelitian tersebut masih perlu dilanjutkan untuk mengetahui faktor-faktor lain yang dapat mempengaruhi *response time* pada halaman *website*.

Berdasarkan beberapa tinjauan pustaka di atas dapat disimpulkan bahwa pengujian otomatisasi antarmuka adalah jenis *black box testing*, karena dilakukan untuk menguji fungsionalitas aplikasi tanpa melakukan pengecekan kode *intern*al program. Selain itu katalon studio merupakan *tools* pengujian otomasi yang paling populer saat ini karena sifatnya yang mudah digunakan dan sudah mendukung semua platform pengujian. Meskipun demikian, ternyata masih belum ada penelitian yang mengkaji terkait metode yang digunakan dalam membuat *script* pengujian otomatis, khususnya metode *automation journey*.

BAB III
PELAKSANAAN MAGANG

3.1 Manajemen Proyek

Metodologi yang digunakan dalam pengujian otomatis adalah blackbox *testing* dengan prinsip kerja scrum. Pengujian blackbox berfokus menguji tampilan aplikasi, fungsi-fungsi aplikasi, dan kesesuaian alur fungsi sesuai dari sudut pandang pengguna, tanpa menguji struktur *intern*al atau *source code* program secara langsung (Kosasih & Budi Cahyono, 2020). Gambar 3.1 mengilustrasikan proses pengujian perangkat lunak yang dilakukan oleh *tester* dari awal hingga akhir pada *marketplace* khususnya Tokopedia.



Gambar 3.1 Tahap Pengujian

3.1.1 Sprint Kickoff

Tahap paling awal dimulainya *sprint* adalah *sprint kickoff* yang dilaksanakan setiap 2 minggu sekali. Pada saat *sprint* semua anggota tim berkumpul, dipimpin oleh *Project Manager* untuk membahas detail *task* pekerjaan yang akan diambil dalam 2 minggu kedepan. Pada tahap ini juga dilakukan pembagian tugas, estimasi waktu pekerjaan, dan urutan prioritas pekerjaan. Selain itu, *developer* dan *tester* juga perlu berdiskusi untuk memilih *task* mana saja yang sekiranya butuh pengujian secara manual, karena terdapat beberapa *task* yang dapat dilakukan pengujian secara otomatis dan *task* yang tidak membutuhkan *testing* sama sekali seperti menambah *log* pada program untuk kebutuhan *debugging*.

3.1.2 Membuat Kasus Uji dan Rencana Pengujian

Setelah *sprint* dimulai maka *tester* akan mulai membaca dan memahami product requirements document (PRD) yang berisi deskripsi, data-data yang diperlukan, serta persyaratan-persyaratan lainnya terkait *task* tersebut. PRD tersebut nanti akan digunakan oleh *tester* sebagai referensi dalam membuat sebuah *test case* atau kasus pengujian.

Dalam membuat kasus uji terdapat beberapa komponen yang wajib disiapkan, yaitu judul kasus uji, *platforms*, langkah aksi, dan *expected result* seperti terlihat pada Gambar 3.2. Jika terdapat langkah yang membutuhkan penjelasan lebih lanjut dapat ditanyakan kepada senior *tester* jika berkaitan dengan kasus pengujian atau dapat bertanya pada *project manager* jika berkaitan dengan detail fitur yang akan di uji.

Test Summary	Test Priority	Platforms	Action	Data	Expected Result
as a whitelist user design 3A, i can see shopads on pdp desktop	P1 - High	desktop	login tokopedia	faiq.dhms@gmail.com uid: 18523088	successful login and redirect to homepage
			click product on homepage		redirect to pdp
			scroll to see shopads		shop ads should be shown on **last line of product recommendation** with attribute **design 3A**: shop name, wording Ad, PM/OS badge (if any), 1 product image, name, and rating, button cek sekarang - src=pdp - tracking impression should be recorded
as a whitelist user design 3B, i can see shopads on pdp desktop	P1 - High	desktop	login tokopedia	arif.rahman@gmail.com uid: 18523092	successful login and redirect to homepage
			click product on homepage		redirect to pdp
			scroll to see shopads		shop ads should be shown on **last line of product recommendation** with attribute **design 3B**: shop image, name, and rating, wording Ad, PM/OS badge (if any), 1 product card and rating - src=pdp - tracking impression should be recorded
as a non whitelist user, i can not see shopads on pdp desktop	P1 - High	desktop	login tokopedia		successful login and redirect to homepage
			click product on homepage		redirect to pdp
			scroll to see shop ads		shop ads should be not appear
as a non login user, i can not see shopads on pdp desktop	P1 - High	desktop	go to tokopedia		redirect to homepage
			click product on homepage		redirect to pdp
			scroll to see shop ads		shop ads should be not appear
as a whitelist user design 3A, i can click product image/ product name on shopads pdp desktop	P1 - High	desktop	login tokopedia	faiq.dhms@gmail.com uid: 18523088	successful login and redirect to homepage
			click product on homepage		redirect to pdp
			scroll to see shop ads	after recom widget	shop ads **design 3A** should be appear
			click product image/name on shopads		- redirect to PDP - tracking click should be recorded
as a whitelist user design 3A, i can click product price/ product rating on shopads pdp desktop	P1 - High	desktop	login tokopedia	faiq.dhms@gmail.com uid: 18523088	successful login and redirect to homepage
			click product on homepage		redirect to pdp
			scroll to see shop ads	after recom widget	shop ads **design 3A** should be appear
			click product price/rating on shopads		- redirect to PDP - tracking click should be recorded

Gambar 3.2 Kasus Pengujian

3.1.3 Review dan Revisi Kasus Pengujian

Kasus pengujian yang sudah selesai dibuat oleh *tester* akan dilakukan *review* terlebih dahulu oleh *product manager* (PM) dan *developer* terkait. Setelah kasus pengujian selesai di-*review,* maka *tester* bertugas melakukan revisi terhadap kasus uji yang diberikan komentar oleh *reviewer.* Setelah itu *tester* dapat meminta *review* ulang kepada *reviewer* untuk memastikan

bahwa kasus uji sudah sesuai dengan semua kemungkinan yang mungkin bisa terjadi serta tidak ada kesalahpahaman yang diperoleh. Dengan demikian *test case* pengujian tersebut sudah dapat digunakan sebagai validasi apakah sebuah *task* sudah sesuai dengan persyaratan yang diharapkan atau belum sesuai.

3.1.4 Melakukan Manual Test Pada Staging

Kasus pengujian yang sudah mendapatkan persetujuan dari PM dan *developer* terkait dapat segera dilakukan pengujian secara mnaual terhadap fitur tersebut pada environment *staging* terlebih dahulu. *Staging* adalah lingkungan tiruan dari software utama yang sudah rilis kepada pengguna, tetapi tidak terhubung ke *production* secara langsung sehingga cocok digunakan sebagai tempat pengujian awal di mana *tester* dapat menguji secara menyeluruh tanpa perlu mengawatirkan risiko-risiko yang dapat terjadi. Dalam melakukan pengujian, *tester* perlu mencatat kasus uji mana saja yang belum terpenuhi serta mengisi daftar *bug* yang ditemukan untuk diberikan kepada *developer* supaya diperbaiki terlebih dahulu. *Bug* yang sudah diperbaiki akan dilakukan pengujian oleh *tester* menggunakan semua kasus uji yang sama seperti sebelumnya. Apabila semua kasus uji sudah terpenuhi dan sudah tidak terdapat *bug* yang ditemukan, maka *tester* dapat memberi persetujuan kepada fitur tersebut untuk deploy pada lingkungan *production* atau rilis kepada pengguna.

3.1.5 Deployment ke Production

Deployment adalah proses rilis dan menyebarkan *task* yang sudah dikerjakan oleh *developer* ke *environment production,* sehingga sudah dapat diakses oleh pengguna secara publik/umum. Cara penyebaran antara *website* dan aplikasi *mobile* juga berbeda, untuk *website* perubahan yang dilakukan oleh *developer* akan dihosting pada *server* menggunakan pipeline yang sesuai dengan fitur dan *task* tersebut dalam jangka waktu 1 minggu 2 kali. Sedangkan pada aplikasi *mobile* seperti ios dan android, *developer* akan melakukan deployment pada rilis versi tertentu, yang mana setiap versi akan dilakukan deployment ke playstore dan appstore secara bersamaan dalam waktu seminggu sekali.

3.1.6 Smoke Test Production

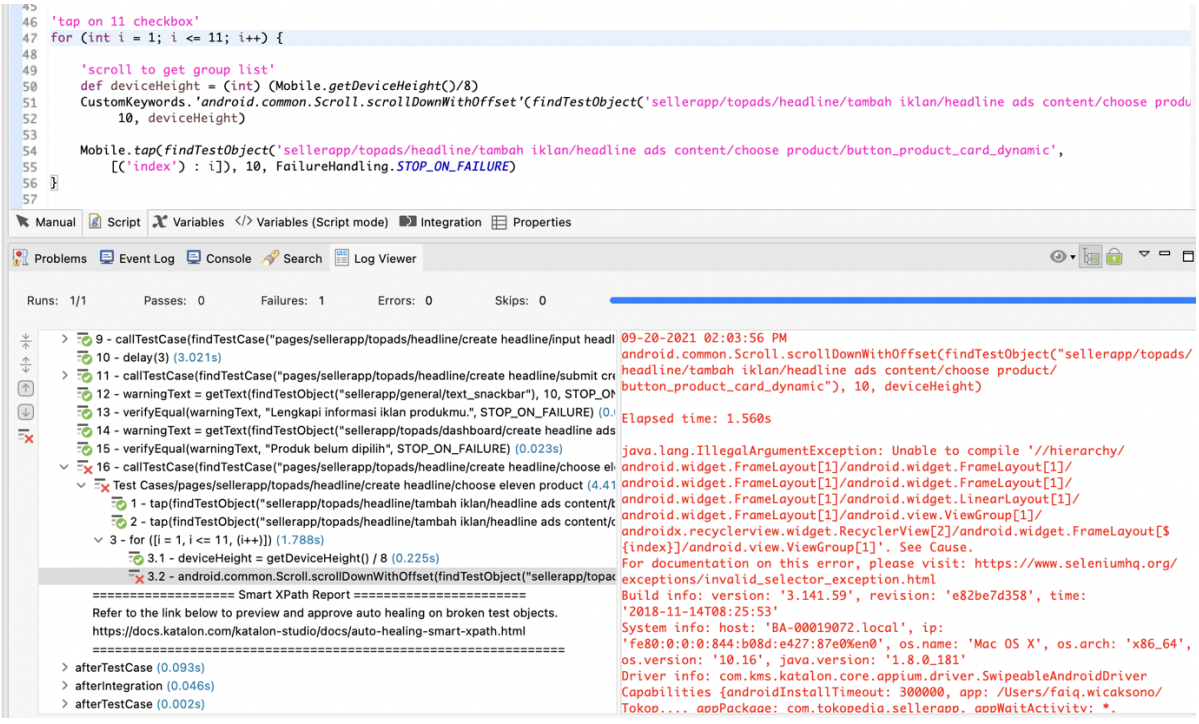
Smoke Test Production adalah tahap pengujian terakhir yang dilakukan setelah fitur atau *task* tersebut sudah sepenuhnya rilis kepada pengguna. Pengujian ini bertujuan untuk memastikan bahwa fitur tersebut sudah terimplementasi dengan baik dan sudah dapat dibuat

pengujian otomatisnya menggunakan kasus uji dari pengujian manual yang digunakan sebelumnya

3.1.7 Membuat Pengujian Otomatis UI

Katalon studio digunakan dalam membuat skrip pengujian otomatisasi antarmuka karena menawarkan solusi pengujian otomasi yang komprehensif dan dapat digunakan pada semua *platforms* (*website*, *mobile browser*, Android, dan iOS). Selain itu, Katalon studio juga sudah menggunakan mesin selenium dan appium yang merupakan standar industri dalam hal *tools* pengujian perangkat lunak.

Dalam membuat pengujian otomatis pada katalon studio terdapat 3 jenis yang berbeda, yaitu record and replay, manual mode, dan *script* mode. Namun, penulis dan sebagian besar *tester* saat ini menggunakan jenis *script* mode dalam membuat pengujian otomatisnya. Dengan *script* mode alur pengujian akan menjadi lebih mudah dibaca dan dipahami, sehingga dapat memudahkan jika ingin mengubah *script* kasus uji atau akan digunakan kembali oleh orang lain (*reusable*), begitu juga untuk *troubleshooting* saat terdapat kasus uji yang gagal, jenis *script* mode lebih mudah dilakukan, karena isi *log viewer* akan sama persis sesuai dengan alur pengujian pada jenis *script* mode seperti terlihat pada Gambar 3.3 berikut.



Gambar 3.3 *Log Viewer* pada Pengujian Otomatis

3.1.8 Membuat Pengujian Otomatis Backend

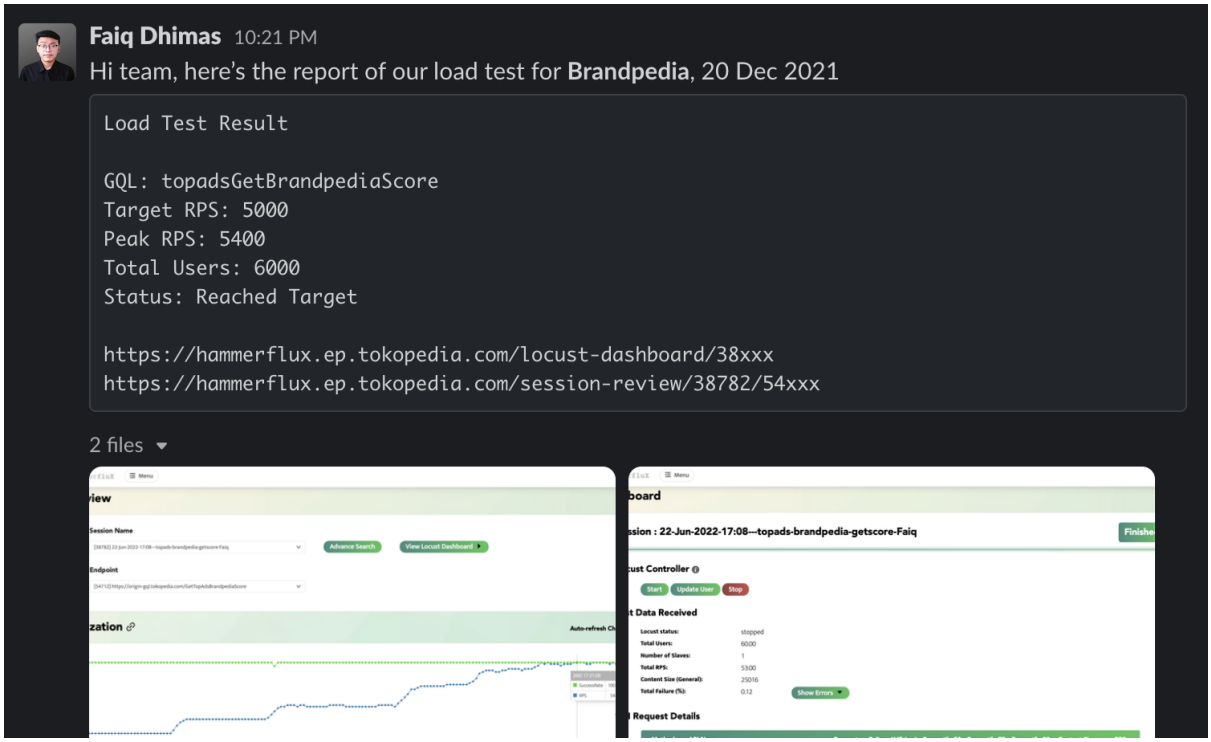
Berbeda dengan pengujian antarmuka, pengujian *backend* dilakukan pada *web service* seperti API, GQL, dan GRPC. *Web service* digunakan sebagai komunikasi antara sisi *backend* dan frontend, oleh karena itu *tester* wajib memastikan bahwa *web service* sudah memenuhi fungsionalitas yang diharapkan dengan cara menguji apakah response yang diberikan sudah sesuai dengan parameter *input* (*request*) atau belum. Selain memeriksa *body response*, *tester* juga menguji *code response* status HTTP dari *request* yang dikirim. Jika respon kode mengembalikan angka 200 menunjukan semua berjalan dengan baik. Sementara kode response 4XX atau 5XX menunjukan terdapat kesalahan, bisa karena request yang tidak benar ataupun karena *server webservice* yang sedang bermasalah.

Pengujian otomatis *backend* dilakukan dengan bantuan *intern*al tokopedia yaitu *Tokopedia Rest Executioner* (T-Rex) dengan beberapa komponen penting yang wajib disiapkan yaitu: host name, http method, environment, variable param, response string, dan response code. Dalam mempersiapkan komponen-komponen tersebut untuk pengujian *backend*, *tester* perlu berdiskusi dengan *backend developer*, supaya *script* pengujian otomatis yang dibuat sudah sesuai dengan kebutuhan dan permintaan pengujian.

3.1.9 Monitoring Load Test

Penulis sebagai SEEP juga bertugas membuat *script load test* menggunakan bernama *locust,* serta melakukan *monitoring* terhadap *load test* tersebut. *Monitoring* dilakukan bersama dengan tim *backend* dan *principal engineer* yang sedang bertugas. *Tester* akan *share screen* dan memantau jalannya *load test* pada semua kasus pengujian yang diperiksa. *Load test* memiliki beberapa komponen penting yang wajib diperhatikan oleh *tester*, seperti *job template name, total user, peak rps,* dan *target rps.*

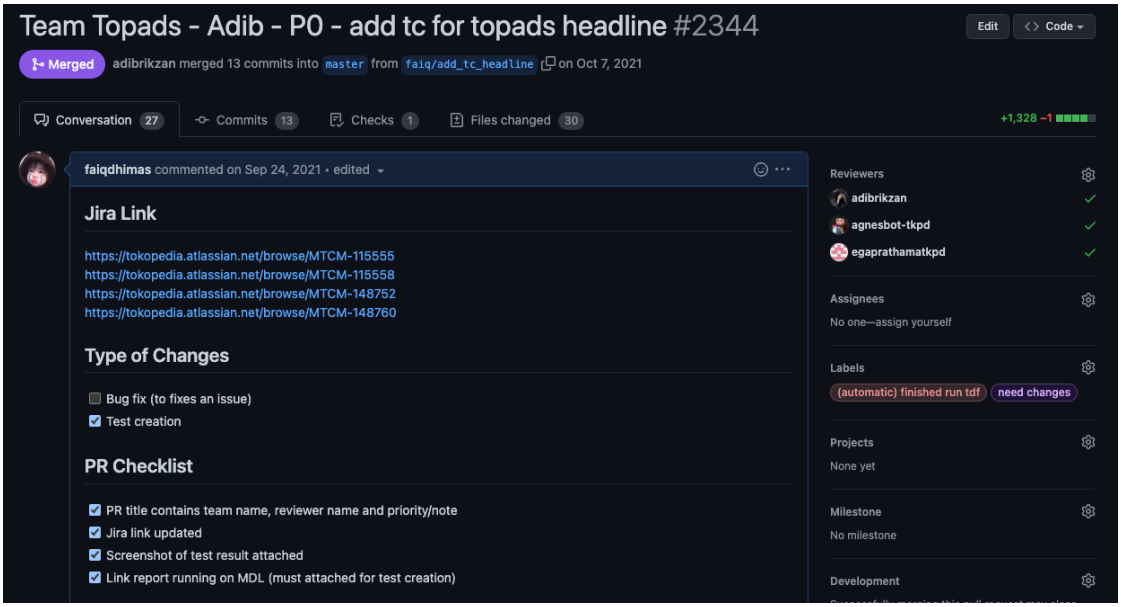
- *Job Template Name* merupakan judul kasus uji dari suatu *load test* pengujian.
- *Total User* merupakan jumlah maksimum pengguna yang digunakan untuk melakukan pengujian pada kasus uji *load test*. Total user ditandai dengan tulisan *Last user* pada locust.
- *Target RPS* merupakan komponen yang menjadi sasaran permintaan / *request per second* (RPS) yang dilakukan dalam satu detik pada kasus pengujian.
- *Peak RPS* merupakan nilai maksimul dari *request per second* (RPS) yang terjadi selama *load test* berlangsung.



Gambar 3.4 *Load Test Report* pada *Slack channel.*

Hasil pengujian diperoleh dengan cara membandingkan *peak rps* dengan *target rps,* serta melakukan verifikasi bahwa *total* user sudah sesuai dengan *total user* yang tercatat pada *script load test*. Jika peak rps dapat melebihi target rps maka target pengujian terpenuhi, sedangkan jika peak rps hampir menyentuh target rps maka dapat ditulis target hampir terpenuhi (*almost reached*), dan yang terakhir jika peak rps jauh dari target rps yang ditetapkan maka target tidak terpenuhi. *Tester* bertugas mengirim laporan tersebut pada channel slack seperti yang ditunjukkan Gambar 3.4. Jika terdapat kendala atau hasil *load test* tidak mencapai target RPS, *tester* bertugas memeriksa *script* pengujian, memeriksa *failure* yang diperoleh selama *load test* berlangsung, dan memberi kabar kepada tim *backend* supaya dilakukan pengecekan lebih lanjut pada sisi *server*.

3.1.10 Deploy dan Code Review



Gambar 3.5 *Pull Request* pada Git *repository*

Pengujian otomatis yang sudah selesai dibuat perlu dilakukan *pull request* seperti terlihat pada Gambar 3.5 supaya *script* pengujian dapat di-*review* oleh anggota tim dan *developer* terkait. Terdapat beberapa langkah dan persyaratan yang harus dilakukan oleh *tester* setelah melakukan *pull request*, yaitu sebagai berikut:

1. *Runtest* kasus uji di *server* tokopedia / *tokopedia device farm* (TDF).

   Kasus uji yang telah berjalan sukses tanpa hambatan di lokal *device* penulis, belum tentu juga akan berjalan lancar pada *device* lain. Oleh karena itu kasus pengujian tersebut harus dijalankan juga pada *server* tokopedia atau tokopedia *device* farm. Jika kasus uji sudah berjalan lancar juga pada TDF, maka *pull request* bisa diajukan beserta melampirkan screenshot dan link TDF sebagai bukti bahwa kasus uji tersebut sudah aman.
2. Meminta ulasan dan *review* dari senior pada anggota tim.

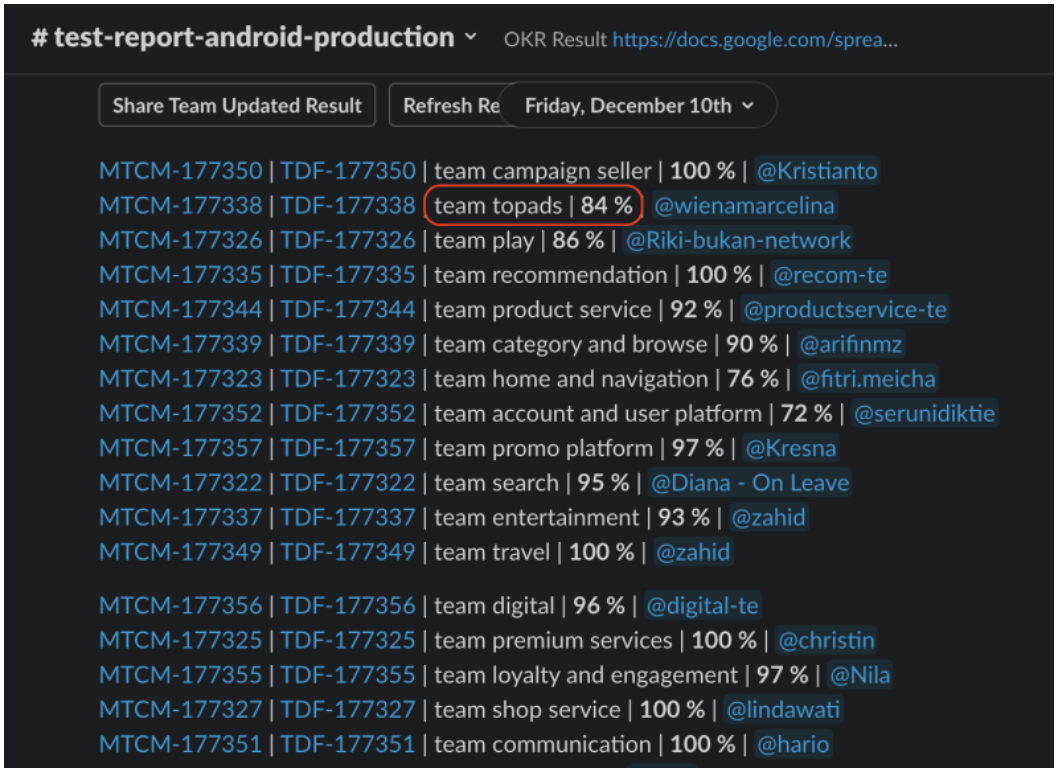
   Pull request yang sudah dibuat perlu dicek oleh senior terlebih dahulu, untuk memastikan bahwa *script* pengujian yang dibuat sudah benar dan sesuai dengan standarisasi tokopedia.
3. Meminta ulasan dan *approval* dari PIC Platform Tokopedia.

   Tahap terakhir yaitu meminta ulasan dari *person in charge* (PIC) dari pengujian otomatis platform yang sedang dilakukan *pull request*. Jika sedang membuat pengujian otomasi pada Android, maka *tester* perlu meminta ulasan dan approval dari PIC *Automation* Android, begitu juga dengan platform lain seperti desktop dan ios.

Saat *pull request* sudah mendapat approval dari PIC platform dan tidak ada revisi lagi, maka merekalah yang akan melakukan merged branch tersebut ke repositori utama tokopedia.

3.1.11 Memelihara Pengujian Otomatis

Pengujian otomatisasi yang sudah dibuat tidak dibiarkan begitu saja, tetapi perlu dilakukan *monitoring* setiap hari melalui *daily test report* seperti terlihat pada Gambar 3.6 yang diberikan oleh sistem setelah pengujian dilaksanakan secara otomatis setiap pagi hari.



Gambar 3.6 *Daily Test Report*

Jika terdapat kasus uji yang gagal maka *tester* bertanggung jawab untuk mencari tahu penyebabnya. Apabila penyebab kegagalan bukan karena *bug*, melainkan terdapat pembaruan sistem atau perubahan *flow*, maka *tester* perlu memperbaiki *automation script* supaya mutakhir dan sesuai dengan kondisi perangkat lunak saat ini. Secara umum terdapat 3 penyebab kasus uji mendapat hasil *failure* saat dijalankan, yaitu sebagai berikut:

1. Perubahan *flow* pada alur pengguna Tokopedia

   Terkadang tokopedia menambah fitur-fitur baru atau pop up baru yang membuat *flow* user menjadi berubah, sehingga *script* pengujian terkait fitur dan halaman tersebut akan selalu *failure* karena sudah tidak relevan lagi, sehingga perlu dilakukan

pembaruan (*fixing*) pada *script* tersebut supaya dapat dijalankan sebagai pengujian otomatis lagi.

2. Terdapat *bug* atau *error* pada sistem Tokopedia

   Jika saat dilakukan pemeriksaan ternyata tidak ada perubahan *flow* pada halaman tersebut, maka *tester* perlu melakukan pengujian secara manual menggunakan Langkah aksiyang sama persis dengan yang dijalankan pada *automation test*. Hal ini dilakukan untuk memastikan apakah ada fungsional tertentu yang memang bermasalah pada sistem Tokopedia. Jika *tester* menemukan hal tersebut, maka *tester* dapat membuat tiket *bug* pada jira board untuk segera diberi ke tim *developer* pada platform dimana kasus uji tersebut bermasalah.

3. Perubahan objek detail dari *developer*

   Pada *script* pengujian otomatis, *tester* membuat objek seperti button, text field, dan sebagainya menggunakan resource-id yang unik pada objek tersebut supaya komputer mengerti langkah aksi yang dimaksud. Namun, jika terdapat perubahan objek ataupun penggantian nama resource-id dari *developer*, maka hal tersebut akan menyebabkan komputer melakukan Langkah aksi pada objek yang salah atau objek yang tidak ada. Oleh karena itu, jika terdapat perubahan objek maka *tester* juga perlu memperbarui objek tersebut sesuai dengan perubahan yang terbaru.

3.1.12 Sprint Review dan Retrospective

Pada akhir *sprint* terdapat *sprint review* dan *sprint retrospective*. *Sprint review* membahas perkembangan *task* sebelumnya dari masing-masing anggota dan menanyakan alasan apabila *task* tersebut mundur atau belum selesai, dengan demikian *task* tersebut harus dilanjutkan lagi pada *sprint* berikutnya. Adapun *sprint retrospective* adalah kegiatan yang dilakukan untuk menilai kinerja setiap anggota tim dengan memberikan umpan balik dari pekerjaan *sprint* sebelumnya baik secara keseluruhan ataupun perorangan yang bertujuan untuk membuat perencanaan peningkatan pada *sprint-sprint* kedepan.

3.1.13 Weekly Meeting

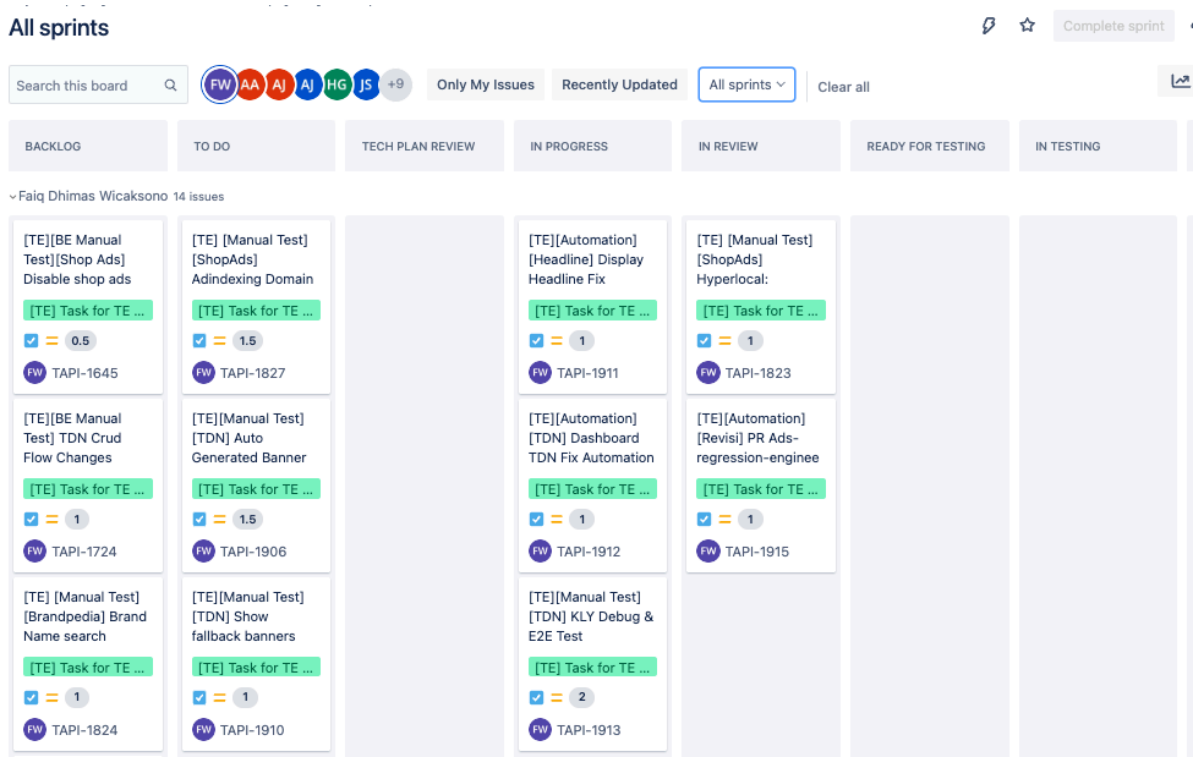
Komunikasi merupakan hal yang penting dalam sebuah tim supaya dapat memahami secara detail tugas yang diberikan dan mengetahui *update* serta *progress* yang dialami saat pengerjaan tugas tersebut. Kegiatan yang dilaksanakan seminggu sekali ini diadakan untuk memberikan perkembangan dari pekerjaan masing-masing anggota tim.

Weekly PM-EP adalah kegiatan meeting antara *Project Manager* dan *Engineer*ing Productivity (*tester*) yang diadakan setiap hari senin untuk membahas prioritas pekerjaan seminggu kedepan, serta estimasi tanggal dan batas waktunya, sedangkan *Weekly* BE-EP adalah kegiatan meeting antara tim *Backend* dan *Engineer*ing Productivity yang diadakan setiap hari jumat untuk memberikan informasi dari *backend task* apa saja yang sudah *deploy* pada lingkungan *production* dan siap untuk dilakukan pengujian sehingga tim SE EP dapat merencanakan pengujian secepatnya. Dua kegiatan weekly tersebut dilakukan untuk memastikan bahwa tugas-tugas yang ambil oleh anggota tim dapat selesai sesuai timeline yang sudah ditetapkan.

3.2 Manajemen Pengujian

Dalam melakukan pengujian *tester* menggunakan jira sebagai alat bantu melakukan pengujian suatu *task*/*project* supaya tim dapat memantau perkembangannya dan juga sebagai dokumentasi dari segala bentuk rencana dan hasil pengujian. Dalam melakukan manajemen pengujian terdapat 4 komponen penting yang perlu dipersiapkan oleh *tester* yaitu:

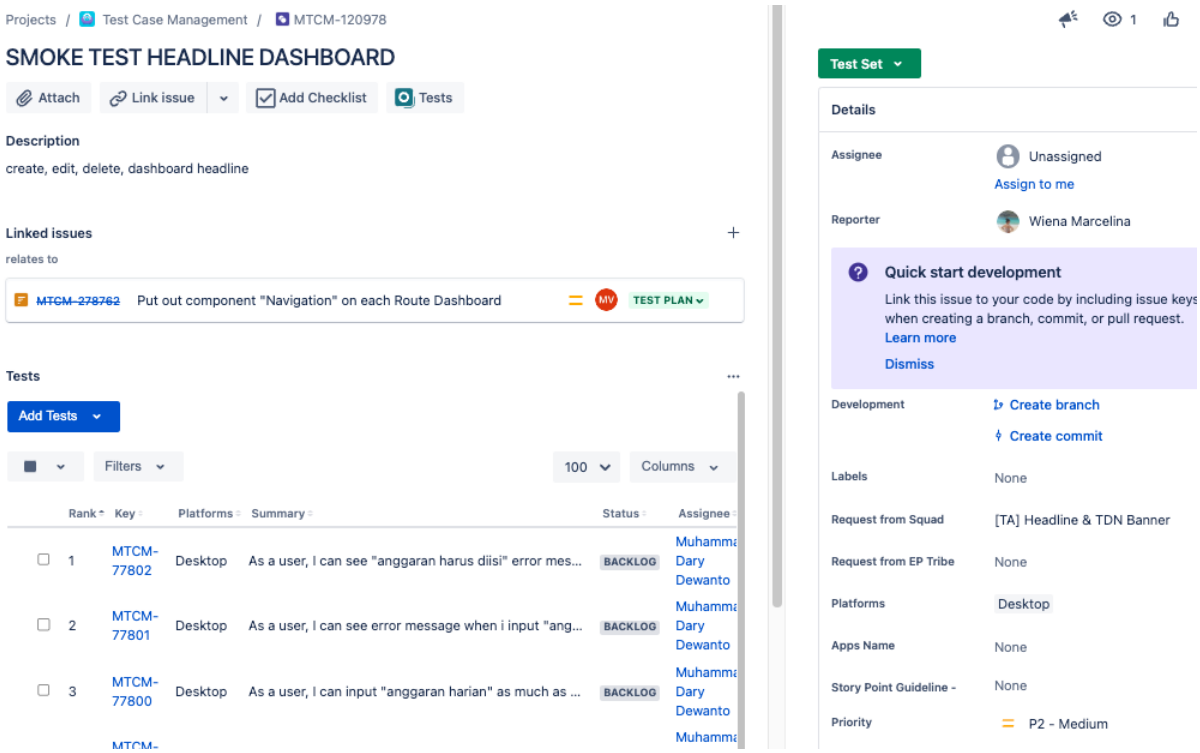
3.2.1 Jira Task



Gambar 3.7 Kumpulan Jira *Task Ticket* dalam *Sprint*

Tim menggunakan jira sebagai *tools* projek manajemen. Setiap tugas yang akan dilaksanakan dalam *sprint* akan terdapat tiket jira beserta *task assignment* nya yang didalamnya terdapat deskripsi *task*, link PRD, *priority task*, *story point*, dan lain-lain yang dibutuhkan sebagai acuan dalam melakukan pengujian. Dengan jira *task* membuat anggota tim dan atasan dapat mengetahui sejauh mana progress pekerjaan dari tugas tersebut seperti terlihat pada Gambar 3.7 dimana perkembangan tugas dapat dipantau, apakah masih di *backlog*, *to do*, *in progress*, *in review*, dan yang terakhir *done*.

3.2.2 Test Case dan Test Set



Gambar 3.8 *Test Set* Jira Tiket

Test case adalah suatu dokumen dengan rangkaian tindakan yang dapat dilakukan oleh *tester* untuk melakukan verifikasi terhadap satu fungsionalitas dari perangkat lunak, sedangkan *test set* adalah kumpulan *test case* yang dapat digunakan untuk memvalidasi seluruh rangkaian fitur apakah sudah berjalan sesuai yang diharapkan atau belum. Pada Gambar 3.8 diperlihatkan tiket jira dari sebuah *Test Set* yang didalamnya berisi kumpulan *test set*, serta terkait juga (*relates to*) pada tiket jira *Test Plan*.

3.2.3 Test Execution

Test execution adalah tiket jira yang berisi status eksekusi pengujian dari suatu *test set* yang tercatat dalam jira, apakah sebuah kasus uji sudah lulus pengujian atau terdapat kasus uji yang gagal, semua tercatat dalam jira *test execution*. *Test Execution* berisi kasus-kasus uji pilihan dari *test set* yang akan digunakan dalam menguji suatu fitur. karena pada setiap fitur baru yang dikembangkan terkadang *tester* tidak perlu menguji semua *test set* yang ada, cukup memilih beberapa *test set* pilihan yang sudah mencakup semua case dari fitur baru yang dikembangkan tersebut. Didalam *test execution* terdapat detail berapa kali pengujian dilakukan, *environment* yang digunakan, dan status pengujian.

3.2.4 Test Plan

Test plan adalah sebuah dokumen yang berisi semua rencana pengujian untuk melakukan validasi suatu fitur. Didalam *test plan* terdapat tujuan *task*, sasaran pengujian, data-data yang dibutuhkan, dan poin-poin pengujian. *Test plan* bertujuan untuk memudahkan *developer* dan PM dalam memantau pengujian yang dilakukan oleh *tester* supaya semua rencana pengujian menjadi lebih jelas dan tidak ada kesalahpahaman, sehingga hasil pengujian nantinya lebih akurat dan sesuai dengan harapan semua *stakeholder*. Pengujian yang sudah dilaksanakan akan dilampirkan hasil pengujian oleh *tester* pada *Test Plan* tersebut beserta bukti-bukti dari pengujian seperti *screenshot* ataupun *screen recorder* untuk dokumentasi dan validasi bahwa fitur tersebut sudah berjalan sesuai yang diharapkan.

3.3 Implementasi Penerapan Automation Journey

Automation journey adalah sebuah metode yang digunakan dalam membuat skrip pengujian otomasi supaya lebih optimal dan skalabel untuk dijalankan setiap hari. Pengujian ini dilakukan untuk menguji tampilan aplikasi, fungsi-fungsi aplikasi, dan kesesuaian alur perjalanan pembeli saat melakukan transaksi (*buyer journey*).

Pada penerapan *automation journey*, secara umum terdapat 4 poin utama yang perlu diperhatikan, yaitu pemilihan kasus uji yang mendukung alur perjalanan logis, memberi percabangan bersyarat pada langkah awal setiap kasus uji, menambahkan variabel baru yang dibutuhkan untuk alur *journey* pada *test case* dan *test suite*, serta menambahkan *connector* pada bagian akhir setiap *test case*.

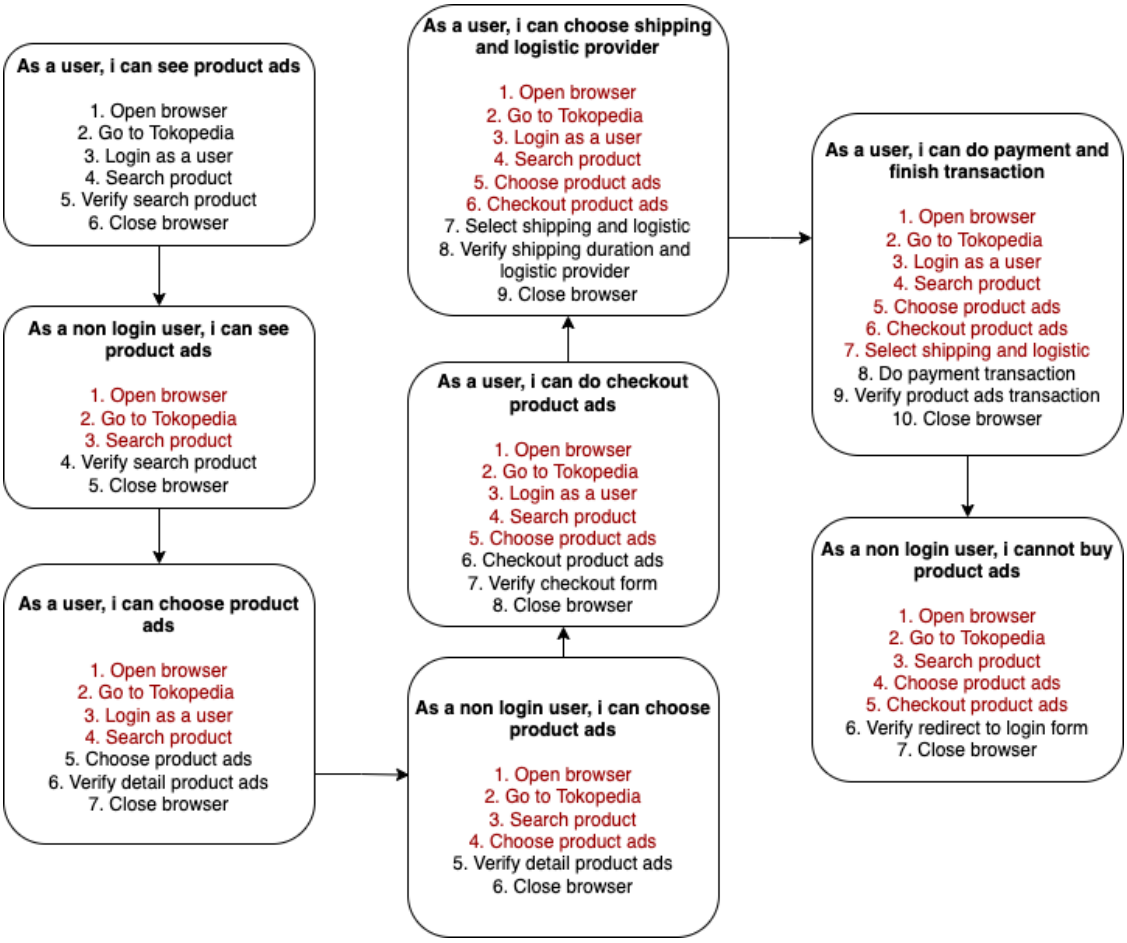
3.3.1 Memilih Kasus Uji dan Mengatur Urutannya

Pemilihan *test case* untuk dijalankan dalam *test suite* harus disesuaikan dengan alur perjalanan logis pengguna mulai dari awal hingga akhir. Dengan demikian langkah aksi dari satu kasus uji dengan kasus uji berikutnya tidak boleh jauh berbeda dan harus saling berkaitan, karena akan membentuk alur perjalanan yang sama untuk mensimulasikan perilaku pengguna.

Tabel 3.1 *Test Suite* tanpa Susunan *Automation Journey*

Test Suite without *Automation Journey*	
No	*Test Case*
1	As a user, I can see product ads
2	As a non-login user, I can see product ads
3	As a user, I can choose product ads
4	As a non-login user, I can choose product ads
5	As a user, I can choose shipping and logistic
6	As a user, I can do payment and finish transaction
7	As a user, I can do checkout product ads
8	As a non-login user, I cannot buy product ads

Susunan *test case* pada *test suite* pada Tabel 3.1 terlihat tidak saling berkaitan satu sama lain. Selain itu, kasus uji untuk pengguna yang *login* dan *non-login* juga saling terpisah tidak berurutan sehingga perlu disusun ulang supaya lebih optimal dan dapat diterapkan *automation journey* pada *test suite* tersebut. Langkah aksi pada *test suite* yang belum menerapkan *automation journey* dapat dilihat pada Gambar 3.9 terdapat beberapa langkah aksi berulang ditandai dengan warna merah yang harus dieksekusi berkali-kali sehingga jika pengujian tersebut dijalankan akan membutuhkan waktu yang cukup lama.



Gambar 3.9 Langkah Aksi Sebelum Penerapan *Automation Journey*

Oleh karena itu supaya *test suite* di atas lebih optimal dan dapat menerapkan *automation journey*, perlu diatur ulang susunannya serta memisahkan antara kasus uji untuk pengguna *non-login* dan pengguna *login*. Pada Tabel 3.2 terlihat *test suite* yang sudah diatur ulang memiliki urutan yang saling berkaitan satu sama lain dan terdapat pemisahan antara kasus uji *login* dan *non-login*. Dengan demikian tidak perlu melakukan aksi *login* dan *logout* lagi setiap kali berpindah kasus uji seperti sebelumnya yang membutuhkan tiga kali *login-logout*. Setelah diatur ulang hanya membutuhkan satu kali *login-logout* saja.

Tabel 3.2 *Test Suite* dengan Susunan *Automation Journey*

Test Suite with *Automation Journey*	
No	Test Case

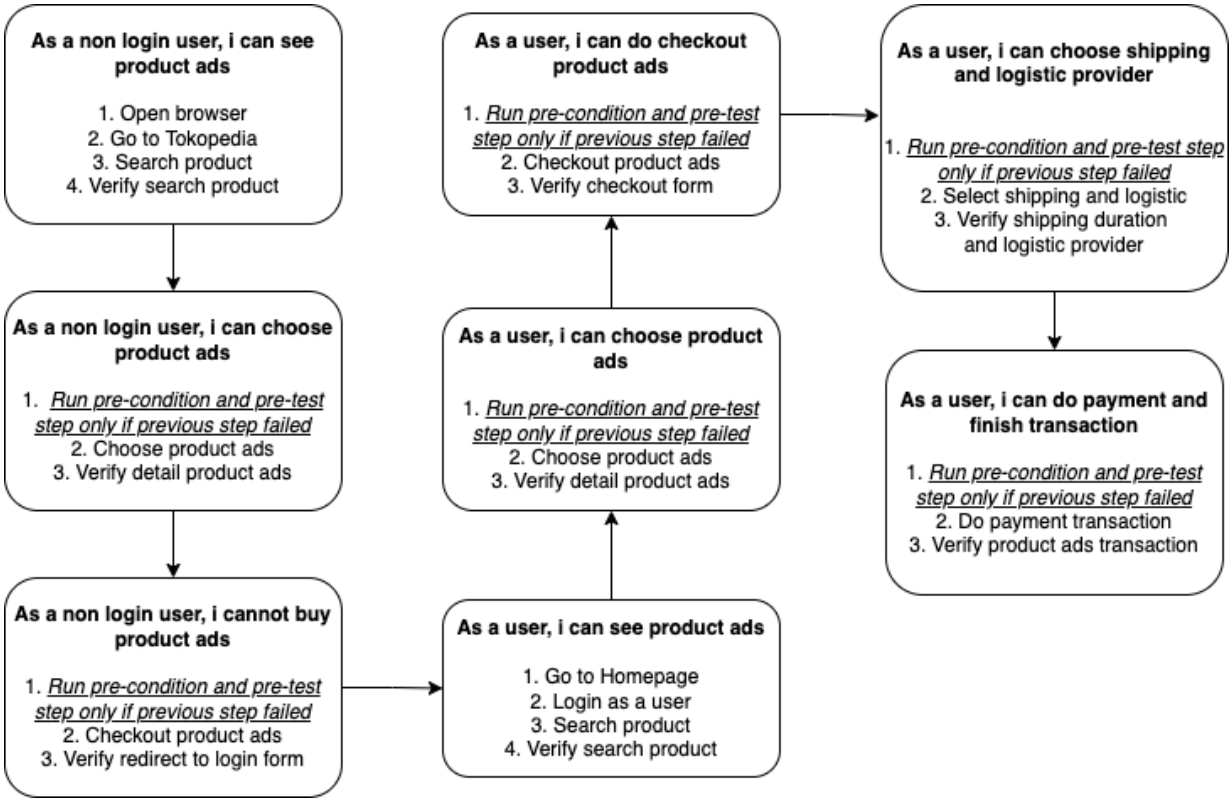
1	As a non-login user, I can see product ads
2	As a non-login user, I can choose product ads
3	As a non-login user, I cannot buy product ads
4	As a user, I can see product ads
5	As a user, I can choose product ads
6	As a user, I can do checkout product ads
7	As a user, I can choose shipping and logistic
8	As a user, I can do payment and finish transaction

3.3.2 Memberi Percabangan Bersyarat Pada Awal Kasus Uji

```
if (individualRun) {
// do login steps here
// also change product status and quantity here
}
```

Gambar 3.10 *Setup Test Case individualRun*

Kasus pengujian yang sudah diatur ulang sebelumnya jika dijalankan masih berjalan cukup lama karena masih terdapat langkah-langkah yang berlebih atau berulang pada setiap kasus uji. Hal ini dapat dimigrasi supaya hanya dijalankan satu kali saja dengan cara menambahkan percabangan bersyarat seperti terlihat pada Gambar 3.10 terjadi penambahan logika disetiap awal kasus pengujian kecuali kasus uji pertama sehingga lebih efisien dan optimal pada saat dijalankan.

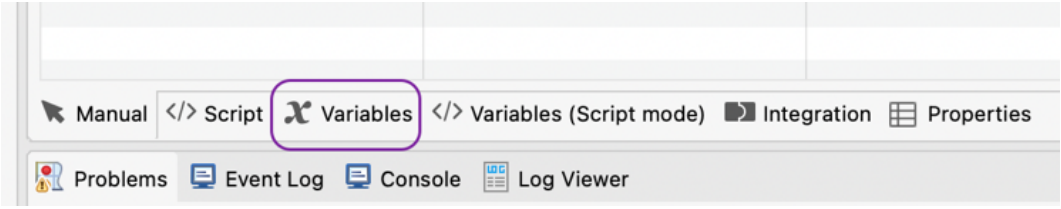


Gambar 3.11 Langkah Aksi Setelah Penerapan *Automation Journey*

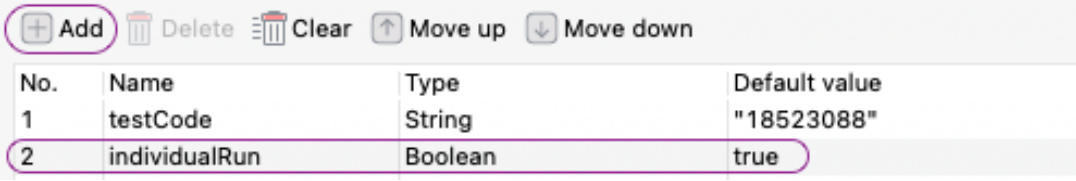
Pada Gambar 3.11 terlihat langkah aksi yang dibutuhkan setelah menerapkan *automation journey* menjadi lebih sedikit. Hal ini dikarenakan sudah ditambahkan logika percabangan untuk tidak menjalankan langkah aksi berulang yang sudah dijalankan sebelummnya. Kecuali jika kasus uji sebelumnya gagal atau terdapat *bug* atau cacat baru, maka langkah aksi berulang tersebut dijalankan. Hal ini akan menghemat banyak waktu pada saat dijalankan dalam pengujian otomasi.

3.3.3 Menambahkan Variabel Baru Pada Test Case dan Test Suite

Pada step sebelumnya telah ditambahkan percabangan bersyarat pada skrip pengujian menggunakan variabel *individual*Run. Namun, jika dijalankan pada saat ini akan terdapat *error* yang disebabkan oleh *test case* yang belum mengenali variabel tersebut. Oleh karena itu perlu ditambahkan variabel *individualRun* beserta nilainya pada setiap kasus uji dengan cara memilih tab *Variables* pada kasus uji di sebelah kanan *script* seperti terlihat pada Gambar 3.12 dan 3.13.



Gambar 3.12 Tab *Variables* pada *tools* katalon studio



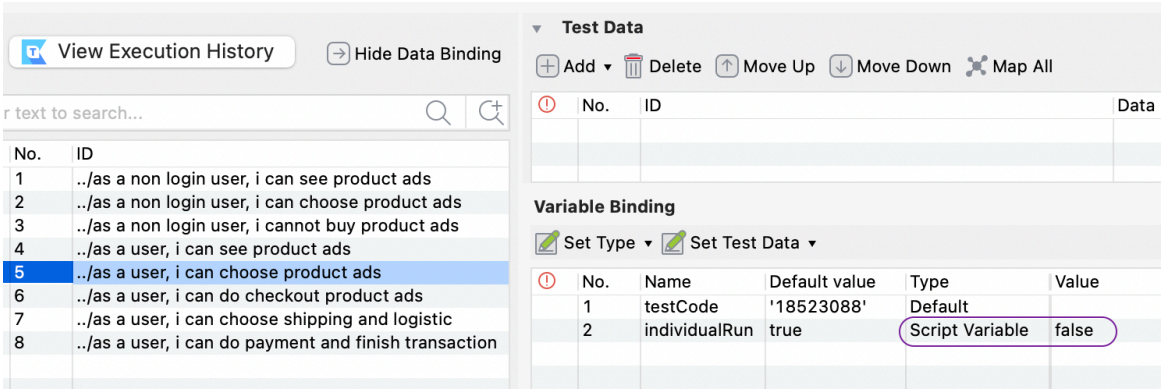
Gambar 3.13 Menambah Variabel *individualRun*

Individual run merupakan percabangan yang digunakan sebagai langkah mundur untuk menjalankan ulang langkah aksi *precondition* seperti *login* dan navigasi awal yang sebelumnya berulang (*redundant*) pada *test suite* tersebut. Aksi ini hanya dijalankan pada saat *test case* sebelumnya terdapat kegagalan atau *error* pada saat dijalankan yang mengakibatkan nilai *individualRun* menjadi 'true' dan masuk logika percabangan bersyarat tersebut. Hal ini dilakukan supaya kasus uji berikutnya terbebas dari kegagalan atau *error* yang terjadi pada *test case* sebelumnya. Dengan demikian kasus uji yang sudah di-*reset* dari awal dapat membuat pengujian otomasi berjalan dengan valid dan meningkatkan kepercayaan diri *tester* terhadap hasil pengujian otomatisnya.

```
/**
* Setup test suite environment.
*/
@SetUp(skipped = false)
def setUp() {
 GlobalVariable.isJourney = true
}
```

Gambar 3.14 *Setup Automation Journey* pada *Test Suite*

Pada *test suite* yang akan dijadikan sebagai automated *journey* juga perlu menambahkan variabel baru, tetapi berbeda dengan *test case* yang menambahkannya pada tab *variables*. Pada *test suite*, variabel ditambahkan pada skripnya seperti terlihat Gambar 3.14 dengan cara mengubah nilai *setupskipped* menjadi *false* dan menambahkan variabel *isjourney* dengan nilai *true* di dalam *function def setup.*

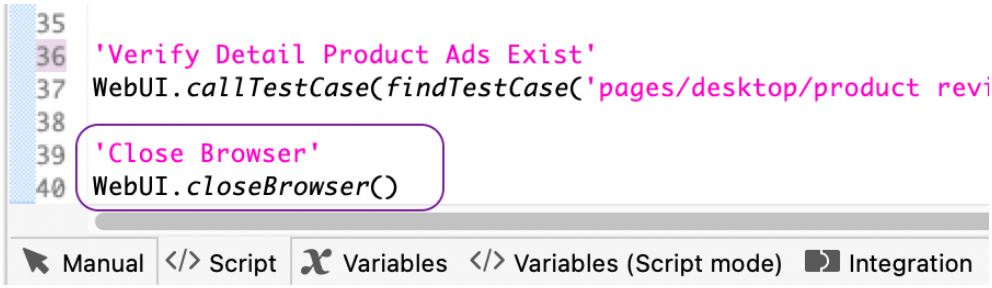


Gambar 3.15 Pengaturan *Variable Binding* dengan *Script* Variable

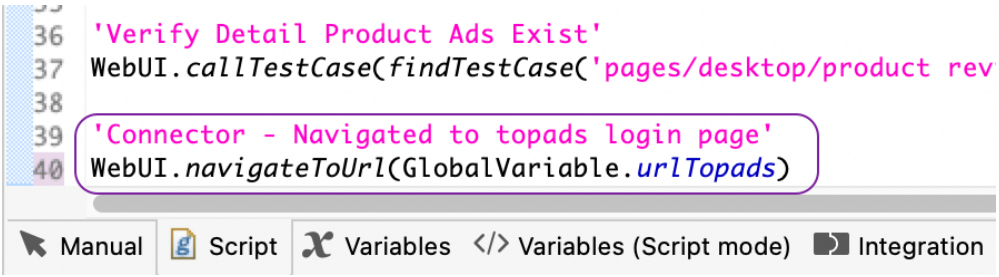
Semua *test case* sebelumnya sudah diberi nilai *true* untuk variabel *individualRun* pada nilai *default*-nya. Namun, untuk menerapkan *automation journey* perlu dilakukan binding dari *individualRun* tersebut pada *test suite* dengan cara menambahkan tipe *script* variable dengan nilai false pada semua *test case* yang memiliki *individualRun* seperti terlihat pada Gambar 3.15. Pemberian nilai false bertujuan agar *test case* tidak menjalankan langkah aksi *setup* dan *precondition* yang berada di dalam percabangan bersyarat (*individualRun*) jika *test suite* tersebut masih berjalan lancar dan belum terdapat *test case* yang gagal atau *error* sebelumnya.

3.3.4 Menambah Connector Pada Akhir Kasus Uji

Connector digunakan untuk menghubungkan satu *test case* dan *test case* lainnya pada *test suite* yang sama supaya langkah paling akhir pada suatu *test case* dapat terhubung langsung ke langkah aksi pertama pada *test case* berikutnya. Pada langkah akhir dari *test case* yang belum menerapkan *automation journey* akan terdapat aksi 'Close *Browser*' atau menutup *browser*. Awalnya aksi tersebut digunakan untuk mengakhiri *individual test case* dengan cara menutup *browser* yang digunakan untuk pengujian. Namun, jika *test case* tersebut menerapkan *automation journey* maka aksi tersebut tidak dibutuhkan lagi dan dapat diubah menjadi *Connector* untuk mendukung *test case* berikutnya seperti terlihat pada Gambar 3.16 dan 3.17.



Gambar 3.16 Langkah Aksi *Close Browser* pada *Test Case*



Gambar 3.17 Penambahan Langkah Aksi untuk *Connector*

Dengan menerapkan *connector*, alih-alih menutup *browser*, maka pengujian dapat berjalan lebih cepat karena *test case* berikutnya tidak perlu melakukan *setup* ulang dan hanya perlu melanjutkan langkah aksi baru yang belum dijalankan dari langkah-langkah *test case* sebelumnya.

3.3.5 Perbandingan Hasil Penggunaan Metode *Automation Journey*

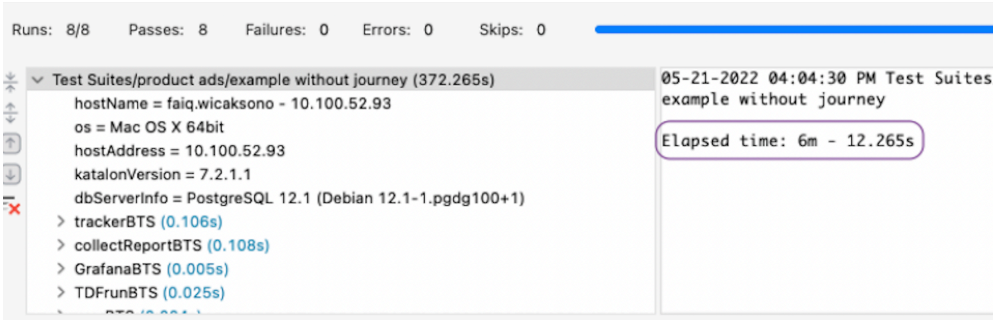
Dalam rangka membandingkan hasil pengujian antara *test suite* yang menerapkan metode *automation journey* dan *test suite* yang tidak menerapkan metode *automation journey*, maka dibuat 2 eksperimen yaitu: menjalankan pengujian UI otomasi pada perangkat lunak tanpa *bug* dan menjalankan pengujian pada perangkat lunak yang terdapat *bug* atau cacat. Pada eksperimen pertama, pengujian akan dilakukan pada perangkat lunak yang valid dan tidak terdapat *bug* atau *error* pada tampilan antarmuka dan fungsi-fungsinya, sehingga diharapkan semua *test case* dapat berjalan dengan lancar.

Pada setiap eksperimen dilakukan masing-masing 5 kali, jenis pengujian yaitu pada *test suite* yang belum menerapkan metode *automation journey* dan pengujian pada *test suite* yang sudah menerapkan metode *automation journey*.

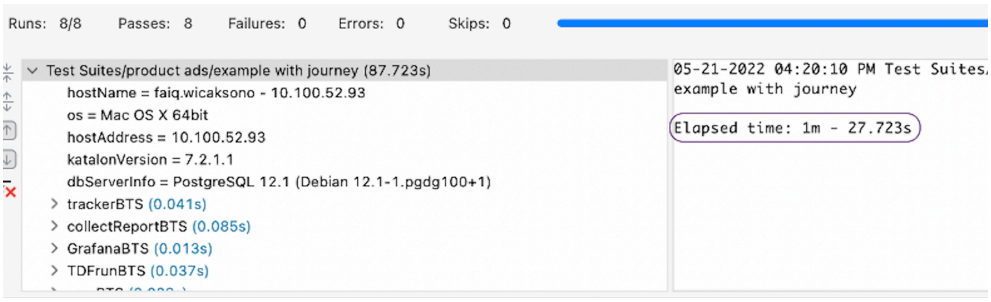
Tabel 3.3 Hasil pengujian pada eksperimen pertama

Eksperimen 1: Pengujian pada perangkat lunak tanpa bug		
Percobaan ke	**Test Suite tanpa journey**	**Test Suite dengan journey**
1	6 menit 36 detik	1 menit 48 detik
2	6 menit 27 detik	1 menit 27 detik
3	6 menit 12 detik	1 menit 36 detik
4	6 menit 32 detik	2 menit 1 detik
5	6 menit 19 detik	1 menit 33 detik
Rerata	**6 menit 25 detik**	**1 menit 41 detik**

Hasil pengujian pada *test suite* yang belum menerapkan metode *automation journey* membutuhkan rata-rata waktu 6 menit 25 detik untuk menyelesaikan seluruh rangkaian pengujian, sedangkan pada *test suite* yang sudah menerapkan metode tersebut hanya membutuhkan rata-rata waktu 1 menit 41 detik seperti terlihat pada Tabel 3.3.



Gambar 3.18 *Test Suite Report* tercepat dari eksperimen 1 tanpa journey



Gambar 3.19 *Test Suite Report* tercepat dari eksperimen 1 dengan journey

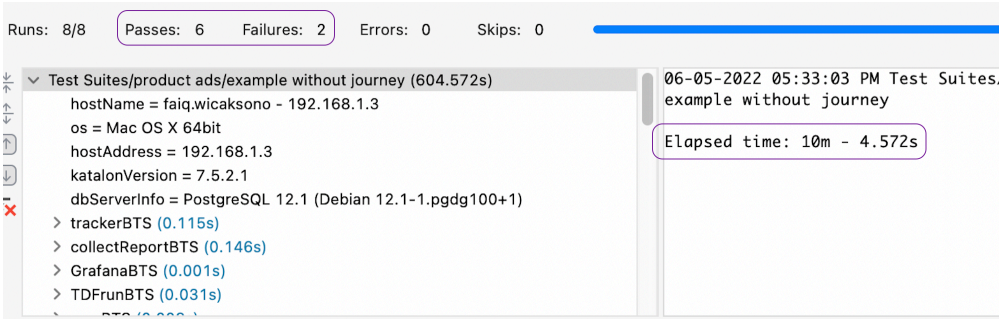
Semua pengujian mendapatkan hasil uji sesuai yang diharapkan (*8 passes, 0 failures*), semua kasus uji lulus dengan waktu tercepatnya adalah 6 menit 12 detik pada *test suite* tanpa *journey* dan 1 menit 27 detik pada *test suite* dengan *journey* seperti terlihat pada Gambar 3.18. dan 3.19.

Pada eksperimen kedua, pengujian akan dilakukan pada sistem yang memiliki 2 *bug* atau cacat pada fungsi-fungsinya sehingga laporan pengujian yang diharapkan yaitu terdapat 2 *test case* yang gagal atau *failure*.

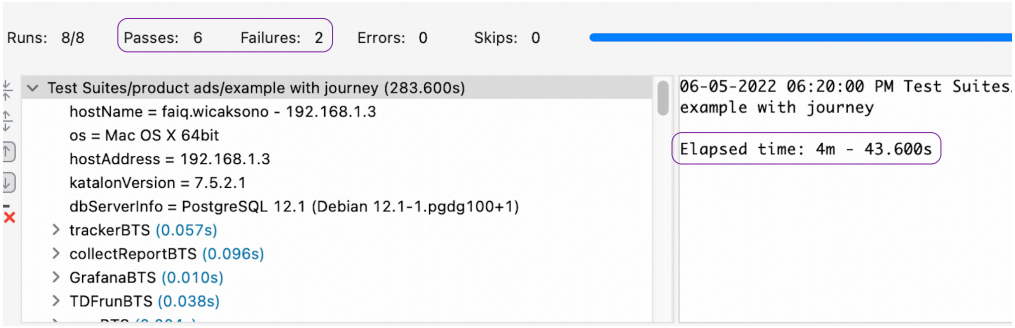
Tabel 3.4 Hasil pengujian pada eksperimen kedua

Eksperimen 2: Pengujian pada perangkat lunak dengan cacat / *bug*		
Percobaan ke	**Test Suite tanpa journey**	**Test Suite dengan journey**
1	10 menit 15 detik	4 menit 47 detik
2	10 menit 4 detik	4 menit 56 detik
3	10 menit 17 detik	5 menit 18 detik
4	10 menit 41 detik	5 menit 13 detik
5	10 menit 23 detik	4 menit 43 detik
Rerata	**10 menit 20 detik**	**4 menit 59 detik**

Hasil pengujian terlihat pada Tabel 3.4 dimana *test suite* tanpa *journey* membutuhkan waktu dengan rata-rata waktu 10 menit 20 detik. Sedangkan pada *test suite* dengan metode *journey* hanya membutuhkan waktu dengan rata-rata 4 menit 59 detik.



Gambar 3.20 *Test Suite* 2 yang dijalankan Tanpa Metode *Automation Journey*



Gambar 3.21 *Test Suite* 2 yang dijalankan Dengan Metode *Automation Journey*

Pada *test suite* yang belum menerapkan metode *automation journey* terlihat pada Gambar 3.20. Terdapat 6 kasus uji yang lulus dan 2 kasus uji yang gagal sehingga hasil pengujian sudah sesuai harapan, tetapi dengan catatan waktu pengujian tercepatnya adalah 10 menit 4 detik. Sedangkan pada *test suite* yang sudah menerapkan metode *automation journey* seperti terlihat pada Gambar 3.21 diperoleh hasil pengujian yang sama yaitu 6 kasus uji lulus dan 2 kasus uji gagal dengan catatan waktu tercepat hanya 4 menit 43 detik.

Dari kedua eksperimen tersebut dapat disimpulkan bahwa pengujian otomatis dengan metode *automation journey* memiliki hasil pengujian yang sama dengan pengujian otomatis tanpa *automation journey*. Meskipun demikian, waktu yang dibutuhkan dalam melaksanakan seluruh rangkaian pengujian jauh lebih cepat diselesaikan pada *test suite* yang menerapkan *automation journey*, seperti terlihat pada Tabel 3.5 walaupun kasus uji yang digunakan sama persis, hanya berbeda urutan *test case* pada *test suite* serta metode yang diterapkan.

Tabel 3.5 Perbandingan Test Suite dengan 2 Metode Pengujian Berbeda

***Test Suite* Eksperimen**	**Eksperimen tanpa *bug***	**Eksperimen dengan *bug***
Test suite* tanpa *Automation Journey	6 menit 25 detik	10 menit 20 detik
Test suite* dengan *Automation Journey	1 menit 41 detik	4 menit 59 detik

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Tabel 4.1 Tabel Analisis Gap

Perbandingan	Kondisi dilapangan	Teori akademik	Analisis GAP
Kasus Pengujian	Menggunakan *Test Case* sebagai acuan eksekusi.	Menggunakan *Use Case* sebagai acuan eksekusi.	Tidak ditemukan perbedaan, hanya berbeda istilah saja.
Kumpulan kasus pengujian	Menggunakan *Test Set* sebagai wadah yang memiliki serangkaian test case sesuai dengan rencana pengujian.	Sudah diterapkan, tetapi hanya mencakup pada suatu fitur tertentu saja.	Pada teori akademik masih berfokus pada verifikasi suatu fitur saja. Belum berdasarkan siklus atau lingkup dari suatu rencana pengujian.
Rencana Pengujian	Dijelaskan secara detail pada dokumen *Test Plan* mulai dari definisi, tujuan, sasaran pengguna, jadwal, prioritas, kebutuhan data, prasyarat, dan lain-lain.	Hanya berisi tujuan dan skenario pengujian.	Pada teori akademik belum terdapat dokumen yang lengkap dan rinci untuk dijadikan arahan dalam melaksanakan proses pengujian.
Pengujian otomatis UI antarmuka (frontend)	Jenis pengujian yaitu *blackbox* dengan perkakas katalon studio (web, api, seluler, dan desktop) dan metode *automation journey*.	Jenis pengujian yaitu *blackbox* dengan perkakas selenium yang mendukung pengujian pada platform *website* saja.	Selain terdapat perbedaan perkakas, pada teori akademik juga belum terdapat metode spesifik dalam membuat skrip pengujian otomatis.
Pengujian otomatis internal (backend)	Jenis pengujian *blackbox* dengan menguji *request* dan *response web service* dengan menggunakan TRex (framework internal tokopedia)	Jenis pengujian *whitebox* dengan metode TDD (test driven development) dan menguji unit/bagian terkecil dari sebuah kode menggunakan perkakas PHP unit.	Terdapat perbedaan jenis pengujian yaitu whitebox dan blackbox sehingga objek pengujian dan perkakas yang digunakan juga akan berbeda sesuai dengan kebutuhan jenis masing-masing.

Dalam membuat kasus pengujian pada pelaksanaannya dilapangan sudah menerapkan teori-teori akademik yang diajarkan pada masa perkuliahan, seperti *test case*, *test set*, hingga *test plan.* Namun, masih terdapat sedikit perbedaan sesuai dengan fungsi dan tujuannya masing-masing. Begitu juga dalam melakukan pengujian otomatis baik pada antarmuka (*frontend)* maupun internal (*backend*) masih banyak perbedaan yang cukup signifikan terutama yang berkaitan dengan jenis pengujian, perkakas, dan metode yang digunakan, seperti terlihat perbandingan analisis gap pada Tabel 4.1.

Pada kasus pengujian tidak ada perbedaan antara teori akademik dan pelaksanaan langsung dilapangan, keduanya memiliki alur dan skenario pengujjian yang sama mulai dari mendeskripsikan kasus uji, menambah langkah-langkah aksi, menyiapkan data uji, melampirkan hasil yang diharapkan, serta mencatat hasil pengujian yang sebenarnya. Begitu juga dengan kumpulan kasus pengujian (*test suite*) tidak ditemukan perbedaan yang spesifik karena keduanya mirip antara teori akademik dan pelaksanaan dilapangan, hanya saja cakupan yang diajarkan pada masa perkuliahan terlalu kecil sehingga memungkinkan setiap *test case* hanya dipakai satu kali saja pada *test suite* yang sesuai. Sedangkan dalam hal rencana pengujian (*test plan*) terdapat perbedaan, yaitu pada teori akademik belum terdapat dokumen rencana pengujian yang lengkap dan rinci untuk dijadikan arahan dalam melaksanakan proses pengujian, seperti strategi pengujian, jadwal pengujian, prioritas antar kasus uji, kondisi prasyarat, dan pengulas kasus pengujian.

Pada penerapan dasar dalam membuat pengujian otomatis pada UI antarmuka (*front end*) dan internal (*back end*) ditemukan perbedaan yang cukup signifikan. Saat ini sebagian besar perusahaan termasuk Tokopedia sudah beralih ke katalon studio sebagai perkakas yang digunakan dalam membuat dan melaksanakan pengujian otomatis antarmuka. Katalon studio merupakan software pengujian otomatis yang dibangun di atas kerangka kerja selenium dan appium sehingga dapat digunakan untuk semua platform (web, seluler, dan desktop). Sedangkan pada teori akademik, pengujian dilakukan menggunakan selenium yang merupakan perkakas individual untuk pengujian *website*. Saat ini pada teori akademik juga belum terdapat kajian yang mempelajari metode yang digunakan dalam membuat skrip pengujian otomatis, seperti *Automation Journey*. Pada teori akademik, skrip pengujian dibuat berdasarkan *test case* dan *test suite* yang ingin dicapai, tetapi belum menerapkan metode-metode yang bertujuan untuk mencapai hasil pengujian menyeluruh (*end-to-end)* yang lebih maksimal.

Dalam pengujian otomatis internal (*back end*) juga terdapat perbedaan yang cukup signifikan, yaitu pada teori akademik pengujian internal dilakukan dengan cara menguji unit/bagian terkecil dari sebuah algoritma kode atau biasa disebut *Unit Test.* Pengujian unit merupakan jenis pengujian whitebox karena berinteraksi dengan program secara langsung menggunakan perkakas PHP unit dan metode *Test Driven Development* (TDD). TDD adalah metode pengembangan yang mengacu kepada pengujian sebelum melakukan proses menulis kode (*coding*). Hal ini jauh berbeda dengan pelaksanaan langsung dilapangan karena dengan penerapan TDD akan membutuhkan waktu yang cukup lama untuk mengembangkan suatu sistem atau produk sehingga jarang diimplementasikan. Begitu juga dengan jenis pengujian yang digunakan, pada pelaksanaan dilapangan pengujian otomatis internal (*backend*) dilakukan oleh *tester* menggunakan jenis pengujian *blackbox* dengan cara menguji menguji *request* dan *response web service* menggunakan *Tokopedia Rest Executioner* (T-Rex) sebagai framework internal tokopedia. Pada pelaksanaan dilapangan, unit test tidak dilakukan oleh *tester,* melainkan oleh developernya itu sendiri karena akan berinteraksi langsung dengan kode dan algoritma secara langsung (whitebox), sedangkan tester hanya bertanggung jawab melakukan verifikasi kesesuaian antara data *request* dengan *response web service* / API.

4.2 Pembelajaran Magang

Terdapat banyak sekali pembelajaran dan hal-hal baru yang diperoleh penulis setelah melaksanakan program magang selama 6 bulan di Tokopedia. Terutama pengalaman baru dalam lingkungan kerja yang lebih profesional sekaligus dengan menerapkan dan mengembangkan konsep akademik yang sudah dipelajari selama masa perkuliahan. Hal-hal tersebut adalah sebagai berikut:

4.2.1 Manfaat

Penulis memperoleh banyak ilmu baru khususnya terkait pengujian dan manajemen pengembangan aplikasi secara berkelanjutan. Hal utama yang penulis pelajari dan dapatkan adalah keahlian dan *skill-skill* yang dibutuhkan untuk menjadi seorang *test engineer professional,* mulai dari bagaimana standarisasi dalam membuat kasus pengujian yang valid dan konsisten, bagaimana membuat otomatisasi yang dapat digunakan kembali, mudah dipelihara, dan skalabe. Serta pentingnya penerapan pengujian otomatis di dunia teknologi yang serba cepat seperti saat ini dimana terdapat kebutuhan konstan dalam membangun perangkat lunak secara efektif, yaitu dengan cara memaksimalkan produktivitas tim dan

meminimalkan pengerjaan yang berulang. Pengalaman magang ini akan sangat penting sebagai rencana karir penulis di masa depan.

Penulis juga menjadi tahu bagaimana struktur dan pembagian kerja pada perusahaan yang besar terus berjalan dan berkembang setiap hari, serta bagaimana upaya perusahaan untuk selalu mencoba mencari dan mendapatkan ide-ide baru untuk pengembangan perusahaan lebih lanjut supaya tetap dapat mengikuti proses perkembangan jaman dan persaingan yang ketat di masa ini. Sehingga setelah mengikuti program magang penulis menjadi lebih terbiasa dengan kondisi dan tekanan dalam pekerjaan.

Selain itu, masih banyak manfaat yang penulis dapatkan melalui program magang ini sebelum menapaki karir ke jenjang selanjutnya, yaitu sebagai berikut:

Perencanaan dan Organisasi:

- Belajar untuk menyelesaikan tugas dalam tenggat waktu yang diberikan.
- Belajar untuk memantau dan mengevaluasi kemajuan tugas sendiri.
- Memantau tugas sendiri secara efektif dengan mengikuti proses *check-in*, *daily syncup meeting*, dan rilis *deployment*.

Komunikasi dan Kolaborasi*:*

- Berkolaborasi lintas tim untuk memastikan kualitas produk.
- Belajar cara membangun kredibilitas dan hubungan diri sendiri dengan orang lain.
- Belajar berkolaborasi dengan berbagai tim, tidak terbatas pada fungsinya sendiri.

Pemecahan masalah dan Pengambilan keputusan:

- Belajar cara menganalisis dan meningkatkan pengujian sistem perangkat lunak Tokopedia.
- Berkontribusi kepada tim teknik masing-masing dalam memecahkan tantangan dilapangan secara langsung untuk produk yang lebih skalabel.

Kemampuan Teknis dan Fokus fungsional:

- Belajar cara mengembangkan pengujian otomatisasi yang *reusable*, *maintainable*, dan *scalable*.

- Merancang, mengembangkan, meningkatkan, dan memelihara sistem pengujian otomatis, *tools/framework*, dan skrip pengujian menggunakan praktik terbaik.

Selama magang penulis juga mendapat banyak relasi baru mulai dari rekan mahasiswa sesama magang, rekan anggota tim *tester* Tokopedia, rekan anggota TopAds secara keseluruhan (*PM, Data Scientist, Data Analyst, Data Protection, Software Engineer, Principal Engineer,* dan *Engineering Manager*). Mereka semua banyak membantu dan membimbing penulis selama melaksankan program magang. Semakin banyak relasi yang diperoleh sangat membantu penulis untuk mendapatkan lebih banyak kesempatan untuk belajar dan mengembangkan skill kedepannya.

4.2.2 Kendala, Hambatan, dan Tantangan

Hambatan selama penulis melaksanakan magang di Tokopedia adalah bagaimana menerapkan manajemen waktu yang baik mengingat penulis masih memiliki beberapa mata kuliah di kampus dan juga memiliki jam kerja magang yang full time yaitu 9 jam setiap hari. Selain itu proses magang yang dilaksanakan secara *work from home* (WFH) akibat dampak pandemi juga menjadi salah satu hambatan bagi penulis, sehingga komunikasi harus selalu terjaga khususnya dengan supervisor penulis dan juga anggota tim TopAds.

Tantangan yang dihadapi selama magang antara lain adalah sebagai *Engineering Productivity*, penulis dituntut untuk memiliki kemampuan aktif bahasa inggris yang baik karena banyak pengembang Tokopedia khususnya *backend developer* berasal dari berbagai negara. Oleh karena itu kemampuan bahasa inggris dibutuhkan untuk berdiskusi dan komunikasi dalam menyelesaikan segala jenis permasalahan yang ditemui.

Tantangan lain yang dirasakan oleh penulis adalah saat mempelajari *tools* dan yang akan digunakan dalam pekerjaan. Sebagai mahasiswa magang, penulis belum memiliki banyak pengalaman sebelumnya, lalu menemukan banyak *tools* baru yang belum penulis dikuasai atau bahkan belum pernah digunakan sama sekali. Penulis perlu beberapa hari untuk mengenal dan melakukan adaptasi terlebih dahulu sebelum akhirnya mulai mengerjakan tugas-tugas di Tokopedia. Untungnya *supervisor* penulis dan senior di Tokopedia sangat suportif mendukung karyawan baru tanpa membedakan-bedakan karyawan *intern* dan *associate* sehingga penulis sangat terbantu dengan bantuan mereka.

Selain itu hambatan dan tantangan saat melakukan magang yaitu koneksi internet yang kurang stabil. Internet saat ini menjadi kebutuhan dasar bagi penulis dalam melakukan magang

secara WFH, sehingga saat koneksi internet buruk dapat mengganggu komunikasi penulis dengan tim terutama saat menggunakan platform pertemuan online seperti Google Meet dan Zoom Meeting. Internet yang buruk dapat menghambat *progress* pekerjaan penulis.

Hambatan dan tantangan yang terakhir adalah saat melakukan *debugging* pada *bug* yang sifatnya *intermittent*. *Bug intermittent* adalah *bug* yang kadang terjadi dan kadang tidak ataupun *bug* yang hanya terjadi pada user dan toko tertentu. Hal ini membutuhkan cukup banyak waktu dalam melakukan proses *debugging*. Jika proses *debug*ging mengalami kendala dan belum mendapat solusi, biasanya senior dari tim *backend developer* akan ikut membantu untuk mencari penyebab permasalahan tersebut.

BAB V
PENUTUP

5.1 Kesimpulan

Setelah menjalani magang selama 6 bulan, penulis telah melakukan berbagai metode pengujian yang ada di Tokopedia, mulai dari *manual test* antarmuka, *manual test backend*, *automation test* antarmuka, *automation test backend*, hingga *load test*. Selain itu penulis juga ikut serta dalam membuat berbagai kasus uji untuk semua platform dan menjalankan skenario pengujian tersebut. Kemudian penulis juga menerapkan *script automation* yang sudah dibuat sebelumnya ke dalam *testing environment* dan *tools* supaya dapat terintegrasi ke dalam *CI/CD Pipeline*. Selama magang di Tokopedia penulis juga melakukan *maintenance* dan perbaikan terhadap *script automation* yang sudah ada, supaya lebih optimal dan relevan dengan kondisi saat ini.

Berdasarkan penerapan rancang bangun metode *automation journey* pada pengujian otomatis antarmuka yang perlu dijalankan setiap hari, dapat disimpulkan bahwa penerapan metode ini dapat membuat pengujian otomatis yang dijalankan supaya menjadi lebih cepat, efektif, dan skalabel. Siklus pengujian yang berjalan lebih cepat dapat meningkatkan persentase keberhasilan secara umum dan meningkatkan produktivitas suatu tim. Selain itu penerapan *automation journey* akan memudahkan *tester* dalam identifikasi bug yang ditemukan karena dengan dihapusnya langkah aksi yang berulang menyebabkan proses pencarian bug menjadi lebih mudah. Dalam hal perawatan dan penggunaan kembali kode sebagai skrip pengujian juga menjadi lebih mudah (*maintenable and reusable*) karena sudah tidak terdapat skrip yang *boilerplate* lagi, yaitu penulisan kode yang sama berulang-ulang (*redundant*) pada langkah aksi yang sama.

5.2 Saran

Setelah menganalisa dan menerapkan berbagai jenis pengujian otomatis pada saat melaksanakan program magang, terdapat beberapa saran yang mudah-mudahan dapat bermanfaat bagi pembaca, lembaga, maupun peneliti-peneliti selanjutnya diantara lain:

1. Bagi lembaga, diharapkan pada mata perkuliahan yang berkaitan dengan pengujian perangkat lunak dapat menggunakan katalon studio sebagai alat pengujian otomatisasinya. Dengan begitu mahasiswa dapat lebih mengenal terhadap

perkembangan-perkembangan *software* terbaru yang sudah sesuai dengan standar industri saat ini khususnya dalam hal pengujian otomatisasi.

2. Bagi pembaca, hasil laporan tugas akhir ini diharapkan dapat menambah pengetahuan dan wawasan terkait dengan pengujian otomatis antarmuka serta mengetahui metode-metode yang dapat digunakan untuk mendukung proses pengujian tersebut.
3. Bagi peneliti selanjutnya, diharapkan dapat lebih mengembangkan ruang lingkup dan metode yang akan digunakan dalam membuat skrip dan meningkatkan kualitas pengujian otomatis, khususnya pada antarmuka. Mengingat saat ini masih sedikit penelitian yang membahas terkait hal tersebut.

DAFTAR PUSTAKA

Ardi, F., & Putro, H. P. (2021). *Pengujian Black Box Aplikasi Mobile Menggunakan Katalon Studio (Studi Kasus: ACC Partner PT. Astra Sedaya Finance)*.

Binar, A. (2022, January). *Mengenali Konsep Agile, Scrum, dan Sprint ala Perusahaan IT*. https://www.binaracademy.com/blog/mengenali-konsep-agile-scrum-dan-*sprint*

Damayanti, A. (2020). *Frontend Testing vs Backend Testing | Medium*. Medium. https://medium.com/@amaliadmyt/frontend-testing-vs-backend-testing-529d6a940835

Kosasih, Y., & Budi Cahyono, A. (2020). Perancangan Sistem Dalam Pengujian Aplikasi The Point Of Sale (Studi Kasus TPOS PT. JAVASIGNA INTERMEDIA). *Teknik Informatika*, *3*(2), 24–30.

Martantoh, E. (2020, June). *Pengujian Otomasi Vs. Pengujian Manual: Apa Perbedaannya*. https://ekomartantoh.net/artikel/2020/06/29/pengujian-otomasi-vs-pengujian-manual-apa-perbedaannya/

Muhtadi, M. M., Friyadi, M. D., & Rahmani, A. (2019). Analisis GUI Testing pada Aplikasi E-Commerce menggunakan Katalon. *Prosiding Industrial Research Workshop and National Seminar*, *10*(1), 1387–1393. https://jurnal.polban.ac.id/proceeding/article/view/1443%0Apolban.ac.id

NKD, F. (2020). *Agile vs DevOps vs CI/CD: Apa Saja Perbedaanya?* Logique. https://www.logique.co.id/blog/2020/12/16/agile-vs-devops-vs-cicd/

Rizky, D. (2019). *Manual Testing VS Automated Testing | DOT Intern | Medium*. Medium. https://medium.com/dot-*intern*/manual-testing-vs-automated-testing-9244aaed1ed5

T, Z. (2020, May). *Sanity, Smoke, Regression Testing*. Linkedin. https://www.linkedin.com/pulse/*sanity-smoke-regression*-testing-zepri-togatorop/?trk=portfolio_article-card_title

Tokopedia. (2020). *Cerita Tokopedia: Lebih Banyak Tentang Perjalanan Kami*. https://www.tokopedia.com/about/our-story/

LAMPIRAN