TESIS

ANALISIS DAN IMPLEMENTASI POLA DESAIN PENGUJIAN OTOMATIS PADA PROSES END-TO-END TESTING

Disusun oleh:

Nama : **Mohamad Dhicy Ramdhani**
NIM : **22.55.1217**
Konsentrasi : **Digital Transformation Intelligence**

PROGRAM STUDI S2 TEKNIK INFORMATIKA
PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA
YOGYAKARTA
2024

TESIS

ANALISIS DAN IMPLEMENTASI POLA DESAIN PENGUJIAN OTOMATIS PADA PROSES END-TO-END TESTING

ANALYSIS AND IMPLEMENTATION OF AUTOMATION TESTING DESIGN PATTERNS IN THE END-TO-END TESTING PROCESS

Diajukan untuk memenuhi salah satu syarat memperoleh derajat Magister

Disusun oleh:

Nama : **Mohamad Dhicy Ramdhani**
NIM : **22.55.1217**
Konsentrasi : **Digital Transformation Intelligence**

PROGRAM STUDI S2 TEKNIK INFORMATIKA
PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA
YOGYAKARTA
2024

HALAMAN PENGESAHAN

ANALISIS DAN IMPLEMENTASI POLA DESAIN PENGUJIAN OTOMATIS PADA PROSES END-TO-END TESTING

ANALYSIS AND IMPLEMENTATION OF AUTOMATION TESTING DESIGN PATTERNS IN THE END-TO-END TESTING PROCESS

Dipersiapkan dan Disusun oleh

Mohamad Dhicy Ramdhani

22.55.1217

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Teknik Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Senin, 3 Juni 2024

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 3 Juni 2024
Rektor

Prof. Dr. M. Suyanto, M.M.
NIK. 190302001

HALAMAN PERSETUJUAN

ANALISIS DAN IMPLEMENTASI POLA DESAIN PENGUJIAN OTOMATIS PADA PROSES END-TO-END TESTING

ANALYSIS AND IMPLEMENTATION OF AUTOMATION TESTING DESIGN PATTERNS IN THE END-TO-END TESTING PROCESS

Dipersiapkan dan Disusun oleh

Mohamad Dhicy Ramdhani

22.55.1217

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Teknik Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Senin, 3 Juni 2024

Pembimbing Utama.

Dr. Arief Setyanto, S.SI., M.T.
NIK. 190302036

Anggota Tim Penguji

Dr. Andi Sunyoto, M.Kom
NIK. 190302052

Alva Hendi Muhammad, S.T., M.Eng., Ph.D
NIK. 190302493

Pembimbing Pendamping.

Dhani Ariatmanto, M.Kom., Ph.D.
NIK. 190302197

Dr. Arief Setyanto, S.SI., M.T.
NIK. 190302036

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 3 Juni 2024
Direktur Program Pascasarjana

Prof. Kusrini, M.Kom.
NIK. 190302106

HALAMAN PERNYATAAN KEASLIAN TESIS

Yang bertandatangan di bawah ini,

Nama mahasiswa : Mohamad Dhicy Ramdhani
NIM : 22.55.1217
Konsentrasi : Digital Transformation Intelligence

Menyatakan bahwa Tesis dengan judul berikut:
Analisis Dan Implementasi Pola Desain Pengujian Otomatis Pada Proses End-To-End Testing

Dosen Pembimbing Utama : Dr. Arief Setyanto, S.Si., M.T.
Dosen Pembimbing Pendamping : Dhani Ariatmanto, M.Kom., Ph.D

1. Karya tulis ini adalah benar-benar ASLI dan BELUM PERNAH diajukan untuk mendapatkan gelar akademik, baik di Universitas AMIKOM Yogyakarta maupun di Perguruan Tinggi lainnya
2. Karya tulis ini merupakan gagasan, rumusan dan penelitian SAYA sendiri, tanpa bantuan pihak lain kecuali arahan dari Tim Dosen Pembimbing
3. Dalam karya tulis ini tidak terdapat karya atau pendapat orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan disebutkan dalam Daftar Pustaka pada karya tulis ini
4. Perangkat lunak yang digunakan dalam penelitian ini sepenuhnya menjadi tanggung jawab SAYA, bukan tanggung jawab Universitas AMIKOM Yogyakarta
5. Pernyataan ini SAYA buat dengan sesungguhnya, apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka SAYA bersedia menerima SANKSI AKADEMIK dengan pencabutan gelar yang sudah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di Perguruan Tinggi

Yogyakarta, 3 Juni 2024
Yang Menyatakan,

Mohamad Dhicy Ramdhani

HALAMAN PERSEMBAHAN

Segala puji bagi Allah SWT yang telah memberikan kemudahan dan kelancaran kepada saya dalam menyelesaikan Tesis ini. Karya ini merupakan titik awal saya dalam mengarungi kehidupan baru dalam dunia pendidikan dengan jenjang yang lebih tinggi serta memotivasi diri untuk terus mencari ilmu yang semakin berkembang. Karya Tesis ini saya persembahkan kepada

1. Kedua orang tua kandung saya yang telah memberikan doa dan restu sehingga tesis ini bisa ditakdirkan untuk selesai pada waktunya
2. Orang yang paling saya cintai istri saya yang selalu berada di samping saya saat masih 0 dan saat masih menimba ilmu mulai dari S1 hingga S2. Dukungan doa dan semangat yang terus diberikan sehingga meskipun di tengah kesibukan pekerjaan sehari hari, saya selalu dipacu untuk menyelesaikan pendidikan S2 di AMIKOM Yogyakarta. Love you so much ..... My love
3. Anakku tersayang yang selalu mencari penyemangat dan pelepas penat dalam menyelesaikan Tesis ini. Doa papih semoga kelak engkau bisa menimba ilmu lebih dari papih dan berguna bagi agama, bangsa dan negara. Amiin
4. Seluruh pihak yang tidak bisa disebutkan satu persatu mulai dari dosen, saudara dan pihak lainnya baik secara langsung atau tidak langsung ikut andil dalam terselesaikannya pendidikan S2 ini.

HALAMAN MOTTO

Dengan ilmu orang bisa mulia, dengan ilmu seorang bisa terhina. Maka manfaatkanlah ilmu untuk kebaikan manusia dan jadikanlah ilmu sebagai bekal perjalanan panjang di akhirat. Jangan jadikan ilmu sebagai alat untuk berbuat kerusakan dan kejahatan.

Allah SWT berfirman didalam Al-quran surat Mujadalah ayat 11 yang artinya

"Hai orang-orang beriman apabila dikatakan kepadamu: "Berlapang-lapanglah dalam majelis", maka lapangkanlah niscaya Allah akan memberi kelapangan untukmu. Dan apabila dikatakan: "Berdirilah kamu", maka berdirilah, niscaya Allah akan meninggikan orang-orang yang beriman di antaramu dan orang-orang yang diberi ilmu pengetahuan beberapa derajat. Dan Allah Maha Mengetahui apa yang kamu kerjakan."

KATA PENGANTAR

Assalamualaikum Wr Wb.

Alhamdulillah Tesis ini dapat terselesaikan berkat doa dan dukungan dari berbagai kalangan khususnya dari civitas AMIKOM Yogyakarta. Pertama saya ingin menyampaikan terima kasih kepada pembimbing 1 yakni bapak Dr. Arief Setyanto, S.Si., M.T. yang selalu sabar dan memberikan ilmu yang cukup banyak selama proses bimbingan tesis maupun dalam kegiatan perkuliahan. Selanjutnya saya juga berterima kasih kepada pembimbing 2 yakni bapak Dhani Ariatmanto, M.Kom., Ph.D yang begitu detail dalam proses bimbingan tesis dan selama proses perkuliahan khususnya dalam bidang *engineering*. Semoga semua jasa dibalas Allah SWT sebagai pahala yang terbaik selama saya menuntut ilmu di AMIKOM Yogyakarta. Tidak lupa bapak Prof Suyanto selaku rektor AMIKOM Yogyakarta dan Prof Kusrini selaku Direktur pasca sarjana yang telah memberikan kesempatan kepada penulis untuk bisa menuntut ilmu di AMIKOM Yogyakarta.

Semoga Allah SWT membalas jasa kebaikan kalian semua dan saya doakan AMIKOM terus maju sampai ke level yang tertinggi khususnya dalam bidang *entrepreneurship*.

Yogyakarta, 3 Juni 2024

Penulis

DAFTAR ISI

HALAMAN JUDUL ii
HALAMAN PENGESAHAN iii
HALAMAN PERSETUJUAN iv
HALAMAN PERNYATAAN KEASLIAN TESIS v
HALAMAN PERSEMBAHAN vi
HALAMAN MOTTO vii
KATA PENGANTAR viii
DAFTAR ISI ix
DAFTAR TABEL xii
DAFTAR GAMBAR xiii
INTISARI xv
ABSTRACT xvi
BAB I PENDAHULUAN 1
1.1. Latar Belakang Masalah 1
1.2. Rumusan Masalah 10
1.3. Batasan Masalah 10
1.4. Tujuan Penelitian 11
1.5. Manfaat Penelitian 12
BAB II TINJAUAN PUSTAKA 14
2.1. Tinjauan Pustaka 14
2.2. Keaslian Penelitian 17

2.3. Landasan Teori 24
2.3.1 Agility dan Biaya Perubahan 25
2.3.3 Metode Waterfall 28
2.3.4 Pengertian Selenium 31
2.3.5 Infrastruktur Selenium 32
2.3.6 Object Repository 34
2.3.7 Datafiles 34
2.3.8 Locator 36
2.3.9 Test Case 36
2.3.10 Test Suite 37
2.4 Pengujian Otomatis Berbasis Kecerdasan Buatan 38
BAB III METODE PENELITIAN 41
3.1. Jenis, Sifat, dan Pendekatan Penelitian 41
3.2. Metode Pengumpulan Data 41
3.3. Metode Analisis Data 42
3.3.1 Studi Pustaka 42
3.3.2 Metode Observasi 42
3.3.3 Metode Eksperimen 43
3.4. Alur Penelitian 43
3.5 Kerangka Kerja Hybrid Sebelumnya (*Existing Hybrid Framework*) 45
3.6 Pembentukan Kerangka Kerja *Hybrid* Usulan 47
3.6 Spesifikasi Kode Otomatis 48
BAB IV HASIL PENELITIAN DAN PEMBAHASAN 49

4.1 Persiapan Pengujian.....50
4.3 Analisis Hasil Pengujian.....56
4.4 Hasil Pengujian terhadap Spesifikasi Kode Otomatis.....68
4.5 Strategi Pengujian.....69
4.6 Perbandingan *Existing Framework* dan *Proposed Framework*.....71
BAB V PENUTUP.....74
5.1. Kesimpulan.....74
5.2. Saran.....75
DAFTAR PUSTAKA.....77

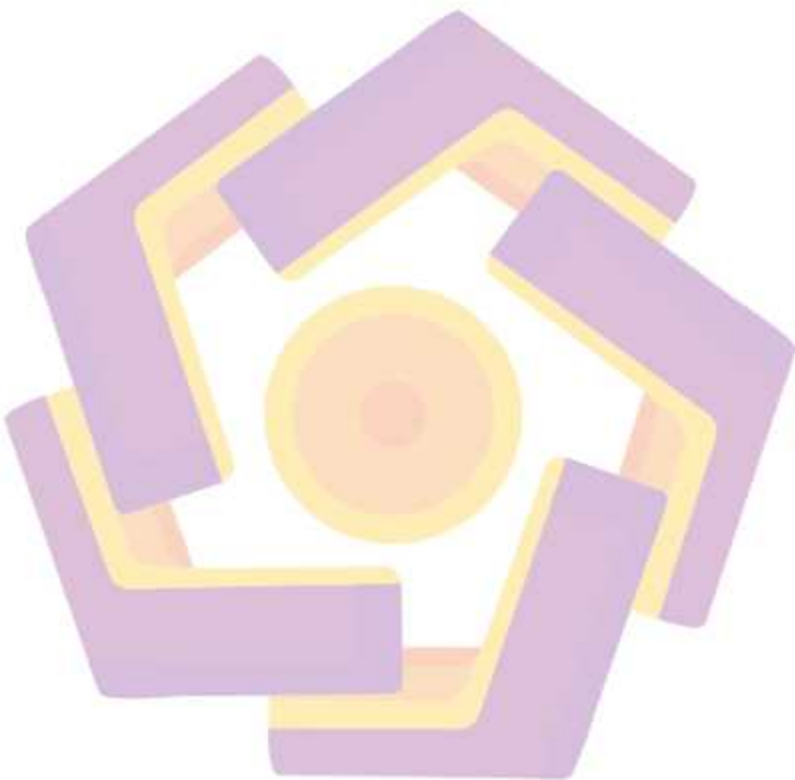
DAFTAR TABEL

Tabel 2.1. Matriks literatur review dan posisi penelitian 17
Tabel 2.1. (Lanjutan) 18
Tabel 2.1. (Lanjutan) 19
Tabel 2.1. (Lanjutan) 20
Tabel 2.1. (Lanjutan) 21
Tabel 2.1. (Lanjutan) 22
Tabel 2.1. (Lanjutan) 23
Tabel 3.1. Detail Alur Penelitian 45
Tabel 4.1. Total Locator dan masukkan sistem 51
Tabel 4.2. Hasil pengujian skenario proses *login* 57
Tabel 4.2. (Lanjutan) 58
Tabel 4.3. Hasil pengujian skenario proses pencarian tiket 58
Tabel 4.3. (Lanjutan) 59
Tabel 4.4. Hasil pengujian skenario proses *create account* 60
Tabel 4.5. Hasil pengujian skenario proses *checkout shopping bag* 61
Tabel 4.6. Hasil Pengujian Berdasarkan nilai EMTE 67
Tabel 4.7. Nilai Time Saved 68
Tabel 4.8. Implementasi Langkah Strategi Pengujian 70

DAFTAR GAMBAR

Gambar 2.1. Biaya Pengujian Perangkat Lunak 26
Gambar 2.2. Tahapan Metode Waterfall 29
Gambar 2.3. Arsitektur Selenium WebDriver 33
Gambar 2.4. Konsep Repositori Objek 34
Gambar 2.5. Konsep datafiles 35
Gambar 2.6. Konsep Test Case 37
Gambar 2.7. Konsep Test Suite 37
Gambar 2.8. Implementasi AI dalam Proses Pengujian Perangkat Lunak 39
Gambar 3.1. Alur Penelitian 44
Gambar 3.2. Hybrid Framework (Bozdemir, 2023) 46
Gambar 3.3. Hybrid Framework (Hanna, 2018) 47
Gambar 3.4. Langkah Pembentukan Kerangka Kerja 47
Gambar 4.1. Desain Framework Skenario 1 53
Gambar 4.2. Desain Framework Skenario 2 54
Gambar 4.3. Desain Framework Skenario 3 55
Gambar 4.4. Desain Framework Skenario 4 56
Gambar 4.5. Grafik hasil pengujian proses login 62
Gambar 4.6. Grafik hasil pengujian proses pencarian tiket 63
Gambar 4.7. Grafik hasil pengujian proses *create account* 64
Gambar 4.8. Grafik hasil pengujian proses *checkout shopping cart* 65
Gambar 4.9. Grafik Equivalent Manual Test Effort 66
Gambar 4.10. Screenshoot Mekanisme Kode Otomatis 68

Gambar 4.11 Hybrid Framework Usulan.....72



INTISARI

Penelitian yang berjudul "Analisis dan Implementasi Pola Desain Pengujian Otomatis pada Proses End-to-End Testing" merupakan penelitian yang berfokus pada analisis pengujian otomatis pada sistem aplikasi website dari sisi struktur, nilai efektifitas dan efektifitas langkah pengujian. Penelitian ini bertujuan untuk menganalisis bagaimana struktur kerangka kerja pengujian otomatis yang efektif pada 4 sistem yang berbeda. Nilai efektifitas dari pengujian otomatis dianalisis sehingga mampu menentukan kerangka kerja paling tepat untuk diimplementasikan. Penyusunan kerangka kerja dilakukan dengan membandingkan penelitian sebelumnya untuk membuat kerangka kerja yang baru. Pengujian otomatis menggunakan metode vertikal dan horizontal. Metode horizontal berfokus pada interaksi visual aplikasi dan metode vertikal pada proses *backend*.

Proses analisis pengujian otomatis dilakukan dengan mengimplementasikan kombinasi kerangka kerja usulan ke dalam 4 skenario pada 4 sistem. Percobaan dilakukan sebanyak 30 kali sehingga menghasilkan waktu pengujian untuk diolah menjadi nilai EMTE dan TS (*time saved*). kedua nilai tersebut kemudian dibandingkan sehingga didapatkan nilai paling efektif atau paling rendah. Kemudian 4 skenario diuji dari sisi respon sistem terhadap perubahan sehingga mendapatkan nilai total langkah paling efektif.

Hasil penelitian menunjukkan perbedaan waktu pengujian antara manual dan otomatis sebesar 48% - 50%. Penggunaan kerangka kerja pada skenario 1 menghasilkan nilai EMTE sebesar 1951.78 detik atau 0.5421 dibandingkan skenario yang lainnya pada 4 proses sistem yang diuji.. Ketika parameter pengujian manual dan nilai kisaran *fragility* ikut serta dalam menentukan waktu yang berhasil dioptimalisasi, maka skenario yang paling efektif juga adalah skenario 1. Hal ini berbanding lurus dengan posisi skenario yang memiliki nilai EMTE terbesar pada masing masing proses sistem.

Kata kunci: pengujian otomatis, waktu pengujian, kerangka kerja, nilai efektifitas, skenario

ABSTRACT

The research entitled 'Analysis and Implementation of Automated Testing Design Patterns in the End-to-End Testing Process' is a study that focuses on the analysis of automated testing on website application systems from the perspective of structure, effectiveness value, and testing step effectiveness. This research aims to analyze how an effective automated testing framework can be implemented on four different systems. The effectiveness value of automated testing is analyzed to determine the most appropriate framework for implementation. The development of the framework is carried out by comparing previous research to create a new framework. Automated testing uses vertical and horizontal methods. The horizontal method focuses on the visual interaction of the application, and the vertical method focuses on the backend process.

The automated testing analysis process is carried out by implementing a combination of the proposed framework into four scenarios on four systems. The experiment is conducted 30 times, resulting in testing times that are processed into EMTE and TS (time saved) values. These two values are then compared to obtain the most effective or lowest value. Then, the four scenarios are tested from the perspective of the system's response to changes to obtain the most effective total number of steps.

The results of the study showed a difference in testing time between manual and automated testing, with a reduction of 48% to 50%. The use of the framework in first scenario resulted in an EMTE value of 1951.78 seconds or 0.5421 compared to the other scenarios in the four system processes tested. When the parameters of manual testing and the range of fragility are also involved in determining the time that can be optimized, then scenario 1 is also the most effective scenario. This is directly proportional to the position of the scenario that has the largest EMTE value in each system process.

Keyword: automation testing, testing time, framework, effectiveness value, scenario,

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Sebagian besar sistem komputer saat ini telah diimplementasikan dalam bentuk aplikasi berbasis website yang semakin rumit. Hal ini membuat pengujian menjadi sangat sulit. Oleh karena itu, dibutuhkan pendekatan manajemen proyek yang tepat dengan menerapkan metode yang sesuai dalam proses pengembangan perangkat lunak mulai dari perencanaan hingga pengujian. Salah satu metode pengembangan perangkat lunak yang dapat digunakan adalah *Agile Methodology*. Sutherland (2014) menyatakan bahwa *Agile* adalah metode pengembangan perangkat lunak yang memiliki karakteristik dinamis dan fleksibel dimana setiap fitur bisa berubah dalam rentang waktu 2 minggu. Metode *Agile* tidak menekankan kepada perencaaan awal dan dokumentasi yang terstruktur secara formal, namun lebih menekankan bagaimana merespon kebutuhan dari customer melalui masukan yang diberikan (Ouriques et al., 2023). Fase development yang dimiliki *Agile* tidak dilakukan secara bertahap, bersifat fleksibel dan berkembang secara terus menerus.

Šimičková (2021) menyatakan bahwa proses *agile* dilakukan secara paralel dengan membagi tujuan ke dalam beberapa bagian terkecil serta memberikan skala prioritas dari setiap *task project* untuk masing-masing individu yang terlibat dalam *development* sesuai dengan fungsinya. lebih lanjut dinyatakan bahwa proses analisis *requirement* harus dilakukan oleh semua bagian dari tim *project* termasuk manager, stakeholder dan user untuk bisa mengidentifikasikan *business*

requirement. Bisnis requirement harus memiliki syarat kuantitatif, relevan dan detail. Hal ini dilakukan agar seluruh anggota tim bisa dengan jelas mengetahui tujuan yang akan dicapai di dalam satu fase pengembangan perangkat lunak. Proses *deployment* bukan merupakan akhir dari *project*, melainkan proses awal bagaimana customer bisa memberikan masukan sehingga tim project bisa memberikan solusi sesuai dengan kebutuhan customer.

Pengujian perangkat lunak merupakan aktivitas eksperimental pada sebuah *development cycle* untuk mencari galat dalam sebuah aplikasi (Sharma, 2014). Lebih lanjut (Lian Min et al., 2020)menyatakan bahwa proses testing digunakan untuk menyesuaikan antara requirement dengan sistem yang diimplementasi . Proses ini juga digunakan untuk mengukur kualitas dari sebuah *software* sebelum di-*release* ke konsumen dan sebagai cara menemukan anomali dari *software* secara dini. Proses testing mengambil peranan yang sangat penting dalam sebuah proses pengembangan perangkat lunak dalam sebuah sistem dengan porsi 30% - 60% dari sebuah project (Gojare et al., 2014). Pada proses testing *software* verifikasi dan validasi diuji secara bersamaan dan harus memenuhi kriteria seperti sudah sesuai dengan design serta mampu diimplementasi dengan karakteristik yang sama dan diselesaikan sesuai standar yang diterapkan oleh *stakeholder*. Verifikasi dan validasi harus bisa disederhanakan dan dibuat lebih cepat dengan cara mengimplementasikan *automatic testing* secara tepat sesuai dengan *requirement* Proses testing biasa dilakukan secara manual oleh *Quality Assurance Engineer* yang memerlukan waktu tidak sedikit dan dipengaruhi oleh faktor jumlah dari *application feature* beserta komponen yang harus di-*testing* .

Salah satu teknik testing yang biasa digunakan dalam pengembangan perangkat lunak adalah *end-to-end testing*. *End-to-end testing* adalah teknik testing yang menghabiskan waktu yang banyak serta cakupan sistem yang luas sehingga upaya untuk optimalisasi testing manual atau proses konversi pengujian otomatis harus terus dilakukan secara tepat dan berkesinambungan (Lian Min et al., 2020). Ketika *End-to-end testing* telah dilakukan, maka dianggap telah memenuhi standar requirement dan menjamin proses yang akan dilakukan oleh user pada halaman web berjalan dengan baik.Terdapat 2 metode pada pelaksanaan *end-to-end testing* yakni horizontal dan vertikal. Horizontal testing memusatkan pengujian fungsionalitas suatu web, sedangkan vertikal menguji sistem yang melibatkan komponen lainnya seperti *application programming interface* (API) (Lian Min et al., 2020).

Manual testing dan automation testing merupakan 2 cara dalam menguji sebuah sistem atau software. Manual Testing merupakan proses pengujian yang dilakukan oleh manusia tanpa bantuan dari kode otomatis. Sistem yang diuji sesuai dengan *requirement* yang telah disepakati pada dokumen perangkat lunak mulai dari desain hingga fungsionalitas (Mahalakshmi et al., 2020). Proses tersebut dilakukan terus berulang kali selama masa proses *development* atau selama galat belum diperbaiki. Proses pengujian berulang menjadi sebuah kekurangan dari pengujian manual karena pengujian akan terus dilakukan selama sistem mengalami perubahan. Masalah lain muncul ketika proses perulangan tersebut dijalankan pada *browser* yang berbeda (Hanna et al., 2018). Oleh karena itu pengujian otomatis sangat diperlukan untuk mengatasi keterbatasan manual testing khususnya dalam hal waktu testing.

Pengujian dapat dilakukan dengan menggunakan konsep *Automated Testing*. Malik dan Ashima (2022) menyebutkan bahwa *automated testing* merupakan testing yang dilakukan tanpa adanya intervensi dari manusia untuk menggantikan proses testing secara manual. sedangkan menurut Gojare et al. (2015) menyatakan bahwa *Automation testing* memiliki arti proses testing yang dilakukan secara otomatis yang terdiri dari proses membuat script kemudian melakukan eksekusi script dengan menggunakan software automation. Kode ini akan menggantikan beberapa langkah testing manual sehingga tidak perlu lagi dilakukan test ulang secara manual. Hal ini membantu ketika sistem berkembang menjadi lebih kompleks. Sistem yang dapat dikonversi kedalam bentuk *automation script* antara lain dalam bentuk *web application, mobile apps*, dan API (*Application Programming Interface*). Automatic testing berjalan lebih cepat dibandingkan manual testing dan memiliki sifat *repeatable*. Banyak perangkat lunak yang tersedia dalam mengimplementasikan testing secara automatisasi. Pemilihan perangkat lunak didasarkan atas kebutuhan dari sebuah perusahaan dan tingkat pengetahuan dari *Software Quality Assurance*.

Proses konversi automatisasi diharapkan bisa meningkatkan efisiensi testing, akurasi, realibilitas, *test coverage*, tingkat deteksi awal galat dan *return of investment (ROI)*. Sebab tidak semua perusahaan memerlukan pengujian otomatis dikarenakan pada sistem yang diimplementasi testing otomatis terdapat proses yang valid hanya jika diintervensi oleh manusia seperti metode *captcha*. Selain itu dibutuhkan proses analisis terlebih dahulu sebelum melakukan konversi kode

pengujian otomatis mulai dari sisi dana yang tersedia, sistem yang akan diautomatisasi dan tingkat pengetahuan *Software Quality Assurance* (SQA).

Strategi yang tepat dalam penyusunan *automatic testing script* harus diterapkan sehingga membuat keseimbangan antara perubahan fitur dengan perubahan yang ada di dalam skrip pengujian tanpa membutuhkan proses yang rumit. Selain itu faktor belum adanya pedoman dan struktur yang tepat untuk diikuti ketika menyusun *Automation script* menyulitkan QA *engineer* ketika menghadapi proses pengembangan yang menggunakan *Data Driven Testing* atau *Behaviour Driven Testing* Hal ini berbeda dengan pengembangan software menggunakan metode waterfall yang memiliki waktu yang cukup banyak dan jadwal yang jelas sehingga proses automasi sistem tidak terganggu dengan kemunculan fitur baru dikarenakan semua telah terjadwal dengan baik.

Salah satu cara mengimplementasikan testing otomasi adalah dengan menggunakan *native selenium*. *Native selenium* mampu melakukan komunikasi langsung terhadap browser untuk melakukan action dengan memanfaatkan metode *native* yakni dengan bantuan dari *webdriver* (Jain & Rajesh, 2015). *Selenium library* menjadi standar berbagai tools dalam menerapkan konsep automation. Hal ini dikarenakan selenium memiliki fungsi yang sangat lengkap dalam melakukan action pada automation web dibandingkan framework lainnya. Namun ada kekurangan yang dimiliki oleh selenium framework yakni tidak bisa menghasilkan report secara otomatis ketika terjadi *failure test case* saat proses dijalankan atau menyajikan detail dari seluruh tahapan proses testing mulai dari awal proses *running* hingga selesai (Gojare et al., 2015).Untuk mengatasi hal tersebut maka

dibutuhkan framework tambahan atau library pihak ketiga untuk menghasilkan report dengan import pada script selenium menggunakan testNG reporter. Namun cara import ke dalam script selenium dinilai memiliki inisialisasi awal yang rumit.

Proses penyusunan script automatisasi yang dilakukan secara konvensional cukup membuat sulit *quality assurance engineer* yang tidak memiliki pengetahuan script yang baik sehingga bisa menghambat proses automatisasi. Untuk mengatasi kesenjangan pengetahuan dari *quality assurance engineer* tersebut banyak muncul *tools automation* yang memiliki kemampuan lebih baik yakni dengan menyederhanakan native selenium dan report dari testNG dengan mengimplementasi *hybrid automation framework*. (Bhondokar et al., 2015) menyebutkan bahwa *hybrid framework* adalah framework yang mengkombinasikan *data driven framework* dan *keyword driven framework* dimana kedua framework tersebut menggambarkan bagaimana aliran data diproses yang terjadi selama proses testing.

Desain framework yang baik memiliki beberapa komponen dasar antara lain *test case, object repository, datafiles* dan *test suite*. Komponen tersebut harus ada dikarenakan untuk bisa menghasilkan test report harus melakukan running script pada *test* suite. Hal ini bernilai valid dikarenakan pada test suite terdiri dari beberapa test case yang mengakses HTML (*Hyper Text Markup Language*) object properties untuk melakukan aksi. Aksi yang dilakukan di dalam sebuah sistem akan mengakses database input untuk dimasukkan ke dalam object atau pemilihan action. Konsep ini mengaktualisasikan lebih dari satu test case ke dalam sebuah test suite atau lebih dengan alur proses *data driven* yang lebih baik.

Framework yang efektif dibutuhkan dalam proses testing agar mempermudah implementasi automatisasi sistem yang rumit dan bisa mengakomodir kesenjangan pengetahuan QA Engineer. Perubahan dari manual QA ke *automation* QA bertujuan agar proses manual testing yang melibatkan banyak *test case* bisa dikurangi sehingga proses testing bisa menjadi lebih cepat dan efisien. *Automation framework design* harus dibuat modular sehingga dalam implementasinya dapat menyesuaikan dengan metode testing menggunakan metode *test data driven* (TDD). Dengan menggunakan *automation script* yang disusun didalam *automation framework design* maka proses testing akan lebih efektif dengan mengurangi waktu testing hingga 50% dibanding testing manual. Sehingga proses bisnis perusahaan menjadi lebih cepat dalam melakukan deliver *new product* ke *customer* lebih cepat dan responsif. Terlebih lagi proses *development* menggunakan *agile development* yang mengedepankan fleksibilitas dalam proses pembuatan sebuah produk membutuhkan desain *automation script* yang baik agar bisa berjalan lancar proses testing.

Sistem harus melalui proses pengujian sebelum digunakan oleh pengguna. Jumlah waktu yang dibutuhkan dalam proses pengujian aplikasi dalam metode agile pada umumnya adalah 14 jam yang terbagi dalam 2 hari dari total waktu *project* selama 10 hari. Namun, lambatnya proses testing *end-to-end* menyebabkan *feature* aplikasi tidak bisa ter-*deploy* lebih cepat dan harus menunggu sampai proses testing selesai. Selain itu, perubahan dan penambahan *feature* atau *application flow* yang intensitasnya semakin banyak menyulitkan QA Engineer dalam melakukan testing antara *current feature* dan *past feature* secara bersamaan dalam 1 sprint (2 minggu)

dan hanya diberikan waktu yang singkat untuk menguji sistem yang sederhana sama yang paling rumit.

Fenomena lambatnya proses testing dalam sistem baik yang sederhana maupun kompleks membuat pengujian secara otomatis dibutuhkan. Pengujian otomatis bertujuan untuk memastikan sistem sudah berjalan sesuai dengan *requirement* dan menyelesaikan proses pengujian secara berulang dengan lebih cepat dan akurat (Margaret Teacher Banjarnahor & Istiyowati, 2022). Pengujian dilakukan dengan menggunakan perangkat lunak automatisasi yang didesain khusus untuk melakukan proses konversi pengujian manual menjadi pengujian otomatis. Sistem yang diuji pada penelitian sebelumnya (Lian Min et al., 2020) adalah sistem sederhana yang dikonversi ke dalam pengujian otomatis dengan menggunakan metode horizontal. Metode horizontal menitikberatkan pada *testing* fungsional dari sistem *website* yang mencakup input data yang ada. Selain metode horizontal juga terdapat metode vertikal yang melibatkan proses lain diluar proses utama fungsional dari sebuah sistem website.

Pada penelitian ini metode testing akan dikombinasikan dengan metode vertikal dimana melibatkan test selain fungsional yakni API (*Application Programming Interface*). API digunakan sebagai penghubung antara aplikasi website dengan server data melalui proses request data baik dengan metode SOAP dan REST (Kore et al., 2022). Metode vertikal ditambahkan agar bisa terlihat bagaimana pengaruh testing otomatis terhadap sistem yang bervariasi dalam sebuah SUT pada proses testing. Sistem yang akan terdiri atas 4 proses tipe sistem yang berbeda yakni proses login pada platform linkedin, proses pencarian tiket kereta api

pada platform KAI, proses create account pada platform hotel myHorison dan proses checkout cart pada platform automation website example.

Konversi pengujian manual menjadi pengujian otomatis dilakukan berdasarkan deskripsi detail dari test case dimana test case didefinisikan berdasarkan nama halaman web dan terintegrasi dengan data input yang disimpan di dalam dataset sehingga struktur test case bisa terdefinisikan dengan baik. Jika dibandingkan dengan framework standar, Implementasi desain framework rekomendasi *automation* diharapkan dapat memberikan kontribusi yang maksimal pada proses *end-to-end testing* yang fleksibel, berbagi data secara *realtime* serta mampu beradaptasi dalam menghadapi perubahan sistem. Waktu yang diperoleh dari proses pengujian otomatis diukur melalui proses pengukuran dengan menghitung nilai EMTE (*Equivalent Manual Test Effort*) dimana nilai EMTE menunjukkan keuntungan dari pengujian otomatis dibandingkan dengan pengujian manual. Selanjutnya efektifitas waktu pengujian otomatis jika dibandingkan dengan pengujian manual diukur dari nilai TS (*Time Saved*) yang harus benilai lebih kecil dari nilai ME (*Manual Effort*) atau lebih besar dari 1 yang dinyatakan dalam satuan jam atau detik (Dorothy, 2010). Ketika nilai TS lebih kecil dari 1 maka bisa dikatakan proses konversi testing otomatis tidak berjalan secara optimal dan dibutuhkan analisa ulang apakah sebuah sistem harus dikonversi ke dalam bentuk testing otomatis atau tidak. Peneliti akan menerapkan beberapa skenario untuk menghasilkan framework yang terbaik dari beberapa framework yang dikembangkan. Nilai rata-rata EMTE menunjukkan jumlah waktu yang dibutuhkan dalam menjalankan *testing automation* dan harus lebih cepat dibandingkan *manual*

testing. Jika nilai tersebut bernilai negative maka proses analisis harus dilakukan ulang supaya menghasilkan proses konversi yang optimal.

1.2. Rumusan Masalah

1. Bagaimana menentukan kerangka kerja pengujian otomatis yang paling optimal ?
2. Bagaimana tingkat efektifitas dari masing-masing skenario pengujian otomatis yang dinyatakan dalam nilai TS (*time saved)* dengan menggunakan keterlibatan nilai EMTE ?

1.3. Batasan Masalah

Ruang lingkup dari penelitian yang dilakukan antara lain ;

1. *Automation framework design* dengan library selenium framework
2. Metode pengembangan software yang akan dijadikan alat pengujian adalah metodologi *Agile*
3. Sistem yang dijadikan alat uji adalah proses pencarian tiket PT Kereta APi Indonesia, proses login Linkedin, proses create account myHorison dan proses checkout aplikasi simulasi pada website automation exercise.
4. Sistem yang diuji berbasis website
5. Penelitian terbatas hanya pada proses menjalankan kode otomatis dan tidak secara komprehensif membahas bagaimana waktu yang dibutuhkan dalam proses pembuatan kode.
6. Waktu pengujian dihitung ketika browser berhasil terbuka kemudian test case dijalankan sampai test case selesai dijalankan.

7. Pengukuran waktu menggunakan fungsi timer. Dimana waktu mulai dan waktu selesai akan dihitung secara otomatis. Hasil dari perhitungan kemudian disimpan pada tabel analisis.
8. Kecepatan internet yang digunakan adalah 13 Mbps (download), 16.1 (upload) dengan latensi 23 mili detik.
9. Browser yang digunakan adalah google chrome versi 58 keatas.
10. Spesifikasi hardware penelitian yang digunakan antara lain processor 1.4 GHz Quad-Core Intel Core i5, memory 8 GB 2133 MHz LPDDR3.
11. Software yang digunakan adalah Katalon Studio dengan spesifikasi minimal adalah CPU (2 GHz, 64 bit processor), memory (2 GB untuk 32 bit dan 4 GB untuk 64 bit) dan hard drive minimal menyisakan 1 GB untuk file project.
12. Data masukkan menggunakan *Test Data Driven* dengan penyimpanan secara *hard code* dan *realtime* pada menggunakan google spreadsheet
13. Teknik pengujian dilakukan dengan proses eliminasi dan deliminasi test case
14. Teknik pengukuran menggunakan nilai EMTE, *fragility,* waktu pengujian manual hingga menghasilkan nilai TS.

1.4. Tujuan Penelitian

1. Menganalisis bagaimana pengaruh kode pengujian otomatis dengan mengimplementasikan metode horizontal dan vertikal pada pengembangan perangkat lunak.

2. Menganalisis pengaruh kecepatan pengujian SUT (*System Under Test*) pada sebuah proses pengembangan sistem dalam satu *sprint system development life cycle* (SDLC)
3. Menganalisis efektifitas penerapan testing secara otomatis setelah dilakukan konversi manual test case menjadi test case dengan kode otomatis.
4. Menganalisis struktur kerangka kerja yang tepat dalam melakukan pengujian otomatis untuk dilakukan oleh QA manual dan lebih mudah dikelola ketika sistem menjadi besar

1.5. Manfaat Penelitian

Manfaat dari penelitian tentang "Analisis dan Implemenetasi Pola Desain Pengujian Otomatis pada Proses End-to-End Testing" adalah :

1. Memberikan pedoman dasar dalam membuat automation script yang baik sehingga mempermudah *Quality Assurance Engineer* dalam mengimplementasi *automation testing* secara lebih efisien.
2. Implementasi desain *framework* rekomendasi pada pengujian otomatis sehingga dapat meningkatkan efektivitas testing dari sisi *flexibility, product delivery* dan *cost* .
3. Membantu proses uji lebih cepat dengan kombinasi yang lebih banyak dalam satu sprint pada proses pengembangan perangkat lunak.
4. Mempermudah proses *script maintenance* ketika terjadi perubahan rencana pengembangan fitur dari sisi project dalam satu sprint sehingga *running script* tetap bisa dengan melakukan penyesuaian script secara cepat.

5. Mempermudah *Quality Assurance Engineer* dalam melakukan konversi test case manual ke dalam bentuk *automation script*.

BAB II

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

(Hanna et al., 2018) pada penelitiannya mengulas bagaimana kerangka kerja pengujian perangkat lunak diimplementasikan dengan menggunakan 2 metode yakni menggunakan *playback/record* dan menggunakan konsep pemrograman. Penelitian tersebut membandingkan antara *keyword driven framework, hybrid testing framework* dan *semi-automated driven automation framework*. Hasil dari penelitian adalah berhasil mengurangi 88% jika dibandingkan menggunakan kerangka kerja semi otomatis dalam hal total waktu yang dibutuhkan dalam melakukan pengujian secara otomatis. Hal ini bisa tercapau dengan melakukan injeksi ekstensi selenium yang disebut dengan SeleniumDB sehingga mengurangi waktu pemrosesan ketika proses pengujian berlangsung.

perbedaan dengan penelitian (Hanna et al., 2018) adalah pada jumlah *Sistem Under Test* yang digunakan berjumlah 4 sistem dengan menghasilkan nilai yang lebih komprehensif yakni perbandingan kerangka kerja yang diimplementasikan terhadap 4 sistem yang diuji. Dari hasil uji tersebut akan didapatkan nilai pengujian yang paling efektif dari masing masing kerangka kerja sehingga bisa menjadi referensi dalam melakukan penyusunan pengujian otomatis. Selain itu penelitian yang dilakukan peneliti juga menghasilkan kerangka pengujian yang baik ketika terjadi perubahan sistem baik dari sisi data, flow sistem dan *locator*.

(Lian Min et al., 2020) dalam penelitiannya menyebutkan terdapat perbedaan waktu pengerjaan antara pembuatan test case manual dan test case otomatis, dimana test case otomatis memerlukan waktu yang lebih lama dibandingkan test case manual. Hal ini disebakan karena faktor pengetahuan dari tiap tester yang terlibat dalam proses konversi kode otomatis. Selain itu disebutkan jika semakin besar presentasi tingkat konversi kode otomatis maka akan semakin singkat waktu pelaksanaan testing ulang.

Perbedaan dengan penelitian (Lian Min et al., 2020) adalah pada keterlibatan test case yang lebih besar pada *System Under Test* (SUT) yang sudah *launching* dengan menggunakan metode horizontal dan vertikal secara bersamaan. Pada metode vertikal akan digunakan API testing sebagai tambahan pengujian sehingga akan bisa terlihat bagaimana proses *end-to-end testing* bisa menggambarkan proses testing secara keseluruhan. Selain itu akan diteliti juga bagaimana cara menyusun kerangka kerja untuk membuat kode otomatis lebih terstruktur sehingga keterbatasan pengetahuan dari QA bisa diteratasi dengan baik. Selain itu akan diuji tingkat efektifitas dari test case yang berhasil dikonversi dengan waktu yang digunakan untuk test case manual.

(Aniwange et al., 2021) dalam penelitiannya melakukan desain automation framework dengan menggunakan *Unified Modelling Language* (UML) dan bahasa pemrograman java. Parameter yang digunakan untuk menilai value dari pengujian otomatis adalah *Test Time Performance*(TTP), *Performance Test Efficiency* (PTE), dan *Automation Scripting Productivity* (ASP). Kesimpulan yang didapatkan dari penelitian tersebut adalah peningkatan nilai PTE sebesar 80% dan TTP sebesar 4%.

(Aniwange et al., 2021 menyebutkan secara singkat bagaimana automation framework dibentuk berdasarkan UML. Sedangkan penelitian yang akan dilakukan adalah bagaimana automation framework dibentuk disertai dengan details dari komponen yang terbaru yakni object repository, datafiles, test case dan test suites. Selain itu akan dijelaskan bagaimana struktur framework yang diteliti diuji dengan perubahan fitur baik itu penambahan atau pengurangan fitur namun efektifitas testing tetap akan bertambah.

(Bozdemir et al., 2023) melakukan analisis bagaimana cara membuat *hybrid framework* dengan mengimplementasikan *page object modelling* dan *test config*. Masing masing digunakan untk menyimpak objek locator dan konfigurasi awal pengujian seperti browser dan test case yang dijalankan. Pada penelitian tersebut menghasilkan langkah untuk membentuk kerangka kerja yang baru.

Perbedaan dengan penelitian yang dilakukan oleh (Bozdemir et al., 2023) yaitu dari sisi desain framework, cara menyusun test suite yang benar sesuai dengan kaidah modular dan pengujian automation pada *environment* yang berbeda. Automation yang baik adalah yang bisa melakukan test fitur yang sama di *environment* yang berbeda. Selain itu akan dihitung bagaimana tingkat efektivitas automation testing terhadap manual testing dengan rumus EMTE dan TS.

2.2. Keaslian Penelitian

Tabel 2.1. Matriks literatur review dan posisi penelitian

Analisis Dan Implementasi Pola Desain Pengujian Otomatis Pada Proses End-To-End Testing

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
1	A New Hybrid Software Testing Automation Framework	Bozdemir, M., Bilgin, T., T., Oylum K., N. A , The European Journal of Research and Development, 2023	Mendeskripsikan pendekatan *hybrid framework* pada pengujian otomatis dengan mendefiniskan objek ke dalam *page factory*.	*Hybrid framework* memiliki keuntungan dari sisi ketersediaan data sehingga proses repetisi pengujian data bisa dilakukan secara maksimal. Selain itu waktu yang dibutuhkan oleh pengujian otomatis juga bisa terlihat pada bagian pelaporan hasil pengujain.	Penelitian tidak menjelaskan bagaimana implementasi pada sistem yang sedang berjalan dan bagaimana waktu yang dihasilkan ketika proses pengujian dilakukan secara manual dan otomatis	**Penelitian sebelumnya** - Metode berfokus pada pemisahan objek,konfigurasi dan. test layer - Hasil pengujian waktu tidak ditampilkan **Penelitian yang dilakukan** - Metode berfokus pada objek, test case dan test suite pada 4 sistem yang berbeda - Hasil pengujian otomatis menampilkan perbandingan waktu antar sistem dalam bentuk nilai EMTE dan T.S. (*time saved*).

Tabel 2.1. (Lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
2	Evaluasi Penggunaan Manual Dan Automated Software Testing Pada Pelaksanaan End-To-End Testing	Lian Min J,Istiqomah A,Rahmani A, Jurnal Teknologi Terapan, 2020	Melihat karakteristik dan teknis pengujian *end-to-end testing* yang dilakukan dalam pengembangan perangkat lunak, baik secara manual atau pun secara otomatis.	- Proses end to end testing bisa dilakukan pada lingkup yang tidak terlalu besar - Pembuatan kode otomatis membutuhkan waktu paling lama pada iterasi pertama. Sedangkan untuk iterasi selanjutnya bisa mempercepat proses testing dengan melakukan *re-run* kode otomatis.	- SUT yang digunakan masih sederhana sehingga bisa dikembangkan dengan melakukan perbandingan sistem - menggabungkan metode vertikaldan horizontal dalam pengujian otomatis - Pendalaman faktor kompetensi QA dalam melakukan proses konversi pengujian manual ke pengujian otomatis	**Penelitian Sebelumnya** - Menggunakan tools protactor - Menghitung nilai tingkat konversi automation - Test Case yang diuji tidak dijelaskan secara spesifik berdasarkan sistem - Metode horizontal dalam proses *end-to-end testing*. **Penelitian yang dilakukan** - Melakukan *end-to-end testing* dengan mengkombinasikan antara *horizontal method* dengan *vertical method*

Tabel 2.1. (Lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
						- Sistem yang terlibat dalam penelitian berjumlah 4 sistem sehingga bisa dilakukan perbandingan hasil pengujian otomatis - Menghitung nilai efektifitas test otomatis yang dilakukan terhadap test manual - Tools yang digunakan adalah katalon studio dengan harapan bisa membantu proses konversi test otomatis bagi *tester* yang belum terbiasa melakukan proses *scripting*

Tabel 2.1. (Lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
3	Web Application Automation Using Selenium	A.Mahalakshmi, N.Sowmya, J.Suvethaa, S.M.Swaroop, International Journal of Innovative Technology and Exploring Engineering, 2020	Menguji aplikasi berbasis web secara otomatis dengan menggunakan bahasa pemrograman c# berdasarkan dokumen uji yang tersedia.	Pengujian otomatis dapat mengurangi proses pengujian yang dilakukan oleh *tester*. Penguji melakukan inisialisasi terhadap test case sebelum dilakukan proses konversi ke dalam kode otomatis yang mampu menghasilkan laporan kesalah jika terjadi malfungsi sistem.	Pada penelitian ini data yang digunakan tidak dijelaskan bagaimana bentuknya. Hanya dijelaskan secara deksriptif. Selain itu proses yang ditampilkan masih memungkinkan terjadinya peningkatan proses pengujian otomatis pada lingkungan dinamis karena struktur framework yang belum tersentralisasi.	**Penelitian sebelumnya** - Menggunakan metode locator dengan tipe xpath yang rentan berubah ketika sebuah halaman mengalami perubahan struktur html - Waktu dari pengujian otomatis tidak dibahas secara mendalam **Penelitian yang dilakukan** - Menggunakan metode locator dengan tipe name atau id yang lebih resisten terhadap perubahan - Hasil pengujian otomatis dianalisis dan dibandingkan secara komprehensif agar mendapatkan hasil yang paling optimal

Tabel 2.1. (Lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
4	Automated Software Testing Frameworks : A Review	Hanna M, A. Elsayed, dan M. Mostafa, International Journal of Computer Applications, 2018	Membandingkan fitur utama dari kerangkan kerja pengujian otomatis.	Pengujian otomatis perlu dilakukan ketika proses regresi dilakukan. Hal ini karena proses regresi berperan 80% dari total waktu pembuatan sistem aplikasi. Oleh sebab itu keterlibatan pengujian otomatis harus dilakukan agar proses testing bisa berjalan secara cepat dan optimal pada setiap fase development software	Hanya menjelaskan *framework* pengujian otomatis tanpa memberikan contoh konkrit pada setiap kerangka kerja dalam melakukan proses pengujian otomatis. Oleh karena itu dibutuhkan contoh nyata bagaimana implementasi dari masing masing kerangka kerja.	**Penelitian sebelumnya** - Hanya menjelaskan kerangka kerja pengujian otomatis - Menjelaskan ulasan kerangka kerja dari sisi struktur **Penelitian yang dilakukan** - Menjelaskan pembentukan kerangka kerja pengujian otomatis disertai dengan implementasi pengujian otomatis dengan menggunakan katalon studio - Menjelaskan kerangka kerja dari sisi struktur dan variable waktu terhadap sistem yang berbeda.

Tabel 2.1. (Lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
5	A Hybrid Software Test Automation For Educational Portals	Aniwange A., Nyishar dan Afolabi, International Journal of Innovations in Engineering research and Technology , 2021.	Meneliti bagaimana Pengujian otomatis bisa berpengaruh terhadap kualitas dari website pada objek penelitian. Kerangka kerja disusun menggunakan UML dan diimplementasikan dengan bahasa pemrograman java.	Produktifitas dari pengujian otomatis menghasilkan nilai efisiensi test sebesar 80 % dan waktu pengujian meningkat sebesar 4%	Tidak menjelaskan bagaimana manajemen test case dan objek locator yang terlibat didalam sistem.	**Penelitian sebelumnya** - Pembentukan framework berdasarkan UML - Hybrid framework yang terbentuk bersifat general dan tidak membahas secara spesifik terlibatan test case dan test objek. **Penelitian yang dilakukan** - Pembentukan framework berdasarkan objek locator dan test case dengan metode Page Object Modelling - Pembentukan framework

Tabel 2.1. (Lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
6	Critical Analysis of Manual Versus Automation Testing	Halani K., Kavita dan R. Saxena, 2021 International Conference on Computational Performance Evaluation (ComPE) ,2021	Menguji waktu yang dibutuhkan antara pengujian secara manual dan otomatis	Pengujian otomatis memiliki waktu lebih cepat dibandingkan dengan pengujian secara manual.	Pengujian otomatis memiliki keterbatasan pada beberapa aspek seperti uji captcha, uji warna, uji ukuran dan lainnya. Selain itu pengujian otomatis lebih baik ketika dihadapkan pada lingkungan sistem yang sudah stabil.	**Penelitian sebelumnya** - Perbandingan antara pengujian manual dan otomatis pada satu sistem **Penelitian yang dilakukan** - Perbandingan waktu pengujian manual dan otomatis pada 4 sistem yang berbeda. - Hasil dari pengujian otomatis dikeluarkan dalam bentuk nilai EMTE dan TS. - Menghasilkan kerangka kerja yang paling optimal sehingga penyusunan kode pengujian otomatis bisa lebih efektif.

2.3. Landasan Teori

Agile merupakan kata kunci yang menggambarkan pengembangan software saat ini atau bisa diartikan sebagai sebuah kelincahan dalam menghadapi perubahan. *Agile Team* harus bisa mengimplementasikan sifat cepat tanggap untuk merespon perubahan yang terjadi ketika proses pengembangan software. Perubahan yang dimaksud antara lain perubahan proses *build* software, perubahan anggota tim, perubahan yang diakibatkan adanya teknologi baru dan perubahan lainnya yang mempengaruhi proses pengembangan *software* dalam menghasilkan sebuah produk. *Agile team* harus mampu mengikuti perubahan dari sebuah *software* karena perubahan merupakan suatu hal yang tidak bisa dihindari. Seluruh tim bertanggung jawab dalam mengidentifikasi dan memberikan solusi yang terbaik dalam menghadapi perubahan yang terjadi dengan melakukan kolaborasi baik secara tim maupun individual sehingga proses inti dari pengembangan *project* bisa berjalan dengan baik. Pemahaman atas objek yang berubah merupakan faktor kunci dalam mengakomodasi sistem *Agile*.

Agility lebih dari sekedar merespons suatu perubahan secara efektif, namun filosofi yang menjadi dasar adalah bagaimana semua tim bisa melakukan komunikasi yang baik antar sesama anggota tim dalam satu *project*, antara teknologi dan bisnis analis, antara *software engineer* dan *managers*. Pada metode ini, pelanggan menjadi salah satu komponen yang terlibat dalam tim development karena *feedback* yang bisa didapatkan dengan cepat sesaat produk ter-*deliver*. selain itu tidak ada istilah "kami" dan "mereka" dalam pengembangan *software* sebagai

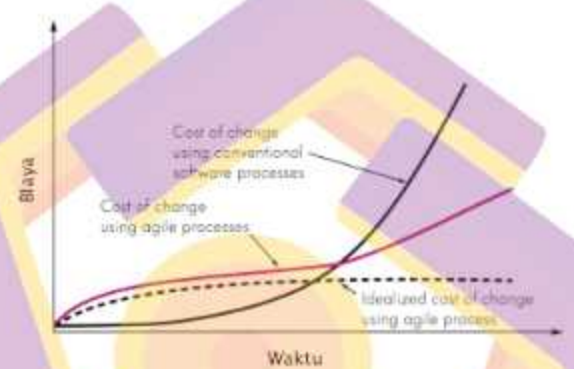
pemahaman dasar dalam melakukan perencanaan *project* ditengah ketidakpastian proyek yang bersifat dinamis.

Agile dapat diterapkan pada semua tipe pengembangan perangkat lunak. Untuk menyelesaikan permasalahan atau skenario baru yang ditimbulkan, maka tim project harus bisa beradaptasi dengaan product. Kemudian melakukan perencanaan yang sesuai dengan pendekatan *agile development*. Semua proses yang tidak memiliki level kepentingan yang ditinggi akan dieliminasi untuk dikerjakan nanti. Namun proses yang memiliki level *high priority* dikerjakan terlebih dahulu sehingga proses incremental *delivery* tetap berjalan dan diharapkan software diberikan kepada pelanggan untuk dipakai secepat mungkin sesuai dengan tipe produk yang diinginkan.

2.3.1 Agility dan Biaya Perubahan

Metode pengembangan perangkat lunak konvensional yang dipakai lebih dari 1 dekade yang lalu menunjukkan peningkatan biaya secara nonlinier ketika proses development berjalan seperti yang terlihat pada gambar 2.1 pada bagian kurva tebal hitam. Teknik konvensial relatif memberikan kemudahan dalam merespon perubahan ketika tim project melakukan requirement pada awal project. Skenario yang dipakai bisa diubah, daftar fungsi bisa bertambah, atau penulisan spesifikasi project bisa dilakukan perubahan. Biaya dari proses ini terlihat sangat minimal dan waktu yang diperlukan tidak akan berpengaruh terhadap pengeluaran dari sebuah project. Namun jika perubahan terus menerus terjadi dalam beberapa bulan kedepan sedangkan dalam waktu yang bersamaan tim masih dalam proses testing dan *stakeholder* menginginkan perubahan fungsi yang besar maka akan

terjadi perubahan yang sangat besar sehingga biaya yang dikeluarkan dengan cepat bertambah hanya untuk memastikan perubahan yang dibuat tidak berpengaruh terhadap software yang telah diberikan kepada pelanggan.



Gambar 2.1. Biaya Pengujian Perangkat Lunak

Gambar 2.1 pada kurva berwarna merah muda tebal menjelaskan proses *Agile* memiliki desain proses yang lebih datar serta fleksibel. Sehingga mendukung perubahan secara tiba-tiba dalam proses *software development* tanpa adanya biaya yang melonjak sangat cepat dibanding metode konvensional dan waktu yang terdampak akibat adanya perubahan sangat kecil Sedangkan kurva datar putus-putus tebal menggambarkan biaya ideal yang dikeluarkan pada proses *agile development*. Meskipun masih dalam perdebatan mengenai kurva datar pada gambar 2.1, namun hal ini menjadi acuan bagaimana biaya perubahan dapat ditekan

secara signifikan dapat tereralisasikan dengan didukung oleh *delivery incremental* disertai dengan kombinasi antara *continous unit testing* dan *pair programming*

2.3.2 Prinsip Agility

Agile memiliki beberapa prinsip yang menjadi pedoman dalam mencapai tujuan pengembangan software yang cepat namun tetap berkualitas. Prinsip tersebut antara lain;

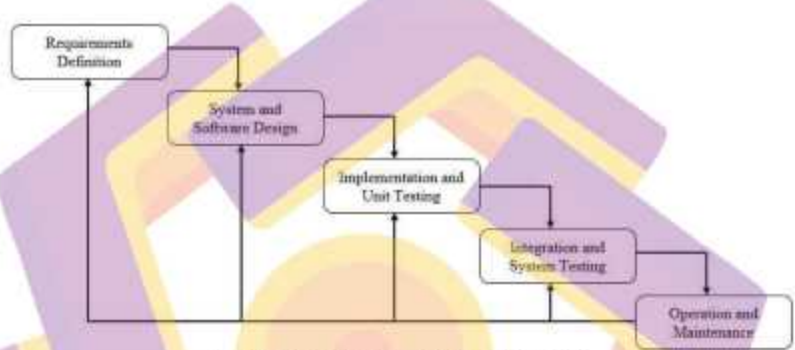
1. Prioritas utama adalah kepuasan *customer* lewat *deliver software* secara cepat dan kontinu serta tetap memperhatikan prinsip *valuable software*.
2. Terbuka terhadap perubahan *requirement* bahkan disaat terakhir proses *development*. Proses *Agile* memanfaatkan keuntungan yang kompetitif dari perubahan yang ditimbulkan dari perilaku *customer*.
3. Melakukan pembaharuan software secara berkala mulai dari 2 minggu sekali hingga 4 minggu sekali dengan mengacu pada standar waktu yang pendek.
4. Analis bisnis dan *developer* harus bekerja sama setiap hari selama project berjalan.
5. Membangun project dengan motivasi individu. Memberikan ruang dan kepercayaan kepada anggota tim dalam menyelesaikan tanggung jawab.
6. Metode yang paling efektif untuk mendapatkan informasi dari anggota tim project adalah dengan melakukan interaksi langsung.
7. Software yang berkerja dengan baik adalah pengukuran utama dari progress pengembangan software.

8. Proses *Agile* memperkenalkan bagaimana cara mengembangkan software secara berkelanjutan
9. Perhatian terhadap kesempurnaan teknis dan desain yang baik meningkatkan tingkat fleksibilitas
10. Arsitektur terbaik, *requirement*, dan dessain yang baik berasal dari pengelolaan tim secara mandiri
11. Memaksimalkan pekerjaan yang tidak bisa selesai untuk dikelola merupakan esensi yang utama
12. Anggota tim harus merefleksikan diri bagaimana menjadi lebih efektif dalam tim dan melakukan perbaikan kualitas secara kontinu.

2.3.3 Metode Waterfall

Model waterfall adalah model pengembangan perangkat lunak yang paling umum digunakan. Model pengembangan ini berfungsi secara linear dari tahap awal pengembangan sistem yakni tahap perencanaan hingga tahap akhir pengembangan sistem (Rosa dan Shalahudin, 2013). Jika tahapan sebelumnya belum selesai, tahapan berikutnya tidak dapat dilaksanakan. Dengan demikian proses tidak dapat kembali atau mengulangi tahapan sebelumnya. lebih lanjut (Rosa dan Shalahudin, 2013) menyebutkan bahwa model SDLC air terjun (waterfall) sering disebut juga model sekuensial linier (sequential linear) atau alur hidup klasik (classic life cycle). Model air terjun menyediakan pendekatan alur hidup perangkat lunak sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (support). Berdasarkan hasil analisis siklus pada metode waterfall menyebabkan anggota tim proyek saling menunggu hingga suatu proses yang

dikerjakan oleh tim lain selesai. Saat ini proyek perangkat lunak merupakan proyek yang bersifat dinamis dengan perubahan yang sangat tinggi sehingga membutuhkan penyesuaian dalam waktu yang sangat cepat. Namun metode klasik ini masih sesuai dipakai pada proyek perangkat lunak dengan spesifikasi yang jelas dan sistem yang tidak banyak mengalami perubahan yang besar.



Gambar 2.2. Tahapan Metode Waterfall

berikut ini merupakan tahap dari metode waterfall berdasarkan gambar 2.2 ;

a. Requirement Definition

Tahapan ini merupakan tahapan dimana hal yang dibutuhkan oleh sistem yang akan dibangun dipersiapkan dengan melakukan pencatatan seluruh kebutuhan sistem mulai dari dokumen flowchart, ERD (*Entity Relationship Diagram), Use Case Diagram, Sequence Diagram, Activity Diagram* dan dokumen lainnya baik berupa dokumen fisik atau non-fisik seperti penyediaan software atau penyimpanan *cloud*.

b. System and Software Design

Pada tahapan sistem dan software design seluruh requirement yang telah didefinisikan sebelumnya diterjemahkan ke dalam bentuk *user interface* sebagai tempat user melakukan interaksi dengan sistem dalam bentuk tampilan sistem. Selain itu dilakukan perancangan tabel di dalam database yang akan menjadi tempat transaksi data ketika sebuah fungsi dijalankan atas perintah dari *user interface* yang dikirimkan oleh user

c. Implementation and Unit Testing

Pada tahapan ini sistem dibangun dengan melakukan pembuatan kode program sehingga seluruh *input* dari pengguna melalui tampilan sistem bisa diolah dengan benar dan baik. Fungsi yang terlibat dalam sistem dibuatkan unit testing dengan tujuan ketika program dijalankan akan melalui proses pengecekan pengujian fungsi sehingga kode program bisa dipastikan tidak terjadi *error*.

d. Integration and System Testing

Kode program yang telah dibuat selanjutnya diuji kelayakan dari sisi fungsi dan tampilan yang akan digunakan oleh pengguna. Proses integrasi dan pengujian bertujuan memastikan antara fungsi dan tampilan sistem berjalan secara benar dan sesuai dengan *requirement* yang telah ditentukan pada proses *requirement definition* sehingga pengguna bisa menggunakan sistem dengan lancar tanpa ada kendalan atau kesalahan kode program yang terjadi.

e. Operation and Maintenance

Tahapan ini merupakan tahapan dimana sistem telah ter-*deploy* ke *environment* utama dimana pengguna menggunakan sistem. Sistem yang telah

dipakai oleh pengguna akan dipantau apakah ada kesalahan kode atau fungsi yang tidak seharusnya terjadi. Ketika terjadi error di dalam sistem yang telah dipakai oleh pengguna maka tim *development* akan menganalisis dan melakukan perbaikan sesuai dengan temuan yang terjadi pada *environment production*.

2.3.4 Pengertian Selenium

Selenium WebDriver merupakan *open-source framework automatic testing* yang digunakan untuk melakukan otomatisasi testing pada Web Apps. Selain itu selenium juga merupakan library yang terdiri dari *class* dan *functions* untuk mendefinisiakan sebuah test. Selenium API pada didesain untuk bekerja di dalam sebuah server sides. Pattrick Lightbody dan Paul Hammer (2004) mencari cara yang lebih mudah dalam melakukan kontrol terhadap test dan bagaimana sebuah library bisa digunakan disemua bahasa pemrograman. Kemudian terbentuklah selenium remote control dengan menggunakan bahasa pemrograman java sebagai client server yang akan membuat komunikasi melalui proxy server supaya terjadi komunikasi data antara selenium dan browser. Berikut ini merupakan jenis - jenis selenium;

1. Selenium IDE
2. Selenium Remote Control (RC)
3. WebDriver
4. Selenium Grid

Selenium RC dan WebDriver telah disatukan menjadi satu kerangka kerja yakni selenium 2 pada update terakhir. Hal ini dilakukan karena pada versi selenium 2 telah mengimplementasikan WebDriver code sehingga Selenium RC menjadi

lebih lengkap fungsinya. Sampai detik ini selenium masih dipakai diberbagai software automation sebagai library utama karena memiliki beberapa kelebihan antara lain;

1. *Opensource*

   Selenium merupakan tools pengujian bersifat open-source yang artinya bisa dikembangkan secara bebas tanpa menggunakan lisensi

2. Integrasi dengan banyak bahasa pemrograman

   Sifat selenium yang sangat fleksibel dan bekerja di sisi client server sehingga bisa digunakan dengan berbagai macam bahasa pemrograman.

3. Fleksibel

   Kelebihan selenium adalah kemampuan untuk mengelompokkan sebuah test case. Hal ini berguna untuk mempercepat proses jika terjadi perubahan kode, sehingga SQA (*Software Quality Assurance*) bisa melakukan pengujian denga lebih sederhana dan efisien

4. *Multiplatform*

   Selenium bisa digunakan pada banyak platform dan browser. Dengan kelebihan ini banyak developer atau tester bisa menuliskan kode dengan mudah tanpa harus memikirkan platform yang dipakai.

2.3.5 Infrastruktur Selenium

Selenium bekerja pada sisi client-server. Desain client server adalah model arsitektur perangkat lunak yang terdiri dari 2 bagian yakni client dan server yang melakukan komunikasi melalui jaringan antar komputer atau pada komputer yang sama. Berikut ini merupakan gambar arsitektur dari selenium

Gambar 2.3. Arsitektur Selenium WebDriver

Berikut ini merupakan penjelasan dari arsitektur selenium webdriver pada gambar 2.3 :

1. Selenium Client Library

Selenium client library memiliki peran sebagai client yang memfasilitasi berbagai macam bahasa pemrograman. Client library melakukan komunikasi antar bahasa pemrograman sehingga perintah bisa dikirimkan dengan tepat kepana jalur protokol website.

2. JSON Wire Protocol

JSON Wire Protocol berfungsi untuk melakukan komunikasi antara *client* dan server agar bisa saling mengerti. Komunikasi yang dilakukan yakni melakukan interaksi antara element dengan kode sehingga menghasilkan kontrol . Prinsip ini mirip seperti yang digunakan pada REST API untuk melakukan komunikasi.

3. Browser Driver

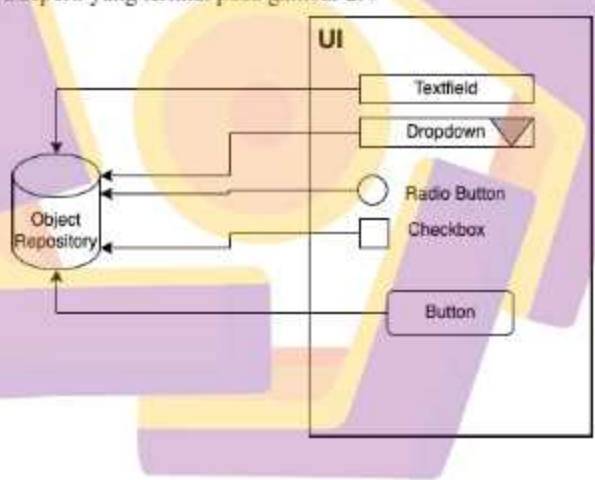
Browser driver berfungsi urnuk menerima request dari klien yang dikirimkan oleh JSON Wire Protocol

4. Real Browser

Browser yang digunakan sebagai sarana automatic testing. Setiap instruksi yang dikirimkan oleh selenium webdriver ke dalam script untuk dijalankan pada proses testing automation

2.3.6 Object Repository

Object repository adalah tempat semua objek pada sebuah UI disimpan. Masing masing UI terdiri atas beberapa tipe antara lain *text field, dropdown, radio button, button* dan lainnya. Object repository akan diakses pada setiap test case yang ada sehingga script mampu mengenali dan melakukan perintah yang telah diatur sebelumnya seperti yang terlihat pada gambar 2.4



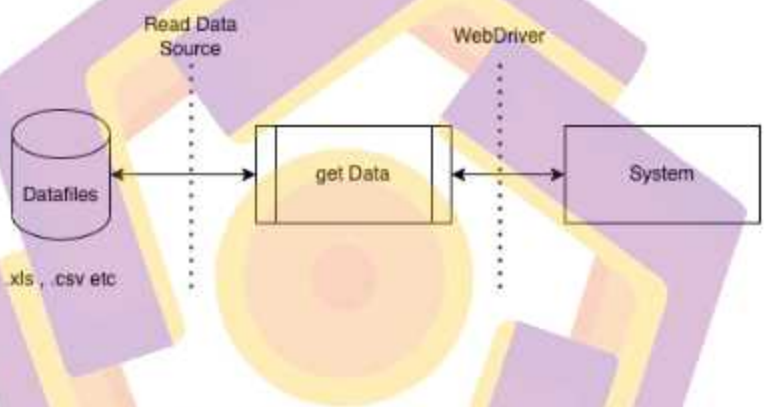
Gambar 2.4. Konsep Repositori Objek

2.3.7 Datafiles

Datafiles adalah tempat data yang akan digunakan pada *field* atau input pada sebuah sistem. Datafiles sendiri digunakan dengan tujuan agar mempermudah

melakukan modifikasi ketika automatic testing dijalankan. Tipe - tipe datafiles yang dapat digunakan antara lain;

1. File excell standalone
2. Tabel Database
3. CSV
4. Google Spreasheet



Gambar 2.5. Konsep datafiles

Pada gambar 2.5 terlihat bahwa data akan dibaca terlebih dahulu untuk kemudian akan dipakai oleh webdriver sebagai input data ke dalam sistem yang akan diuji. Data akan terus diakses oleh webdriver selama sistem berjalan sampai webdriver mengirimkan perintah untuk menghentikan proses akses data ke datafiles.

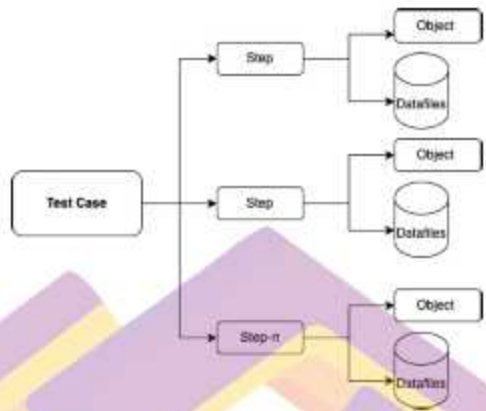
2.3.8 Locator

Locator merupakan properties yang berfungsi untuk mengidentifikasi di setiap element yang terdapat pada komponen dari aplikasi. Baik itu aplikasi mobile apps atau berbasis website. Dalam pembuatan automation script, automation driver bisa mengenali komponen dan menjalankan automation script karena locator yang telah didefinisikan sebelumnya. Jika dalam sebuah page terdapat beberapa input, maka masing masing input memiliki properties yang disebut dengan locator. Barikut ini merupakan tipe dari locator yang paling sering digunakan dalam web automation;

1. CSS id : *find_element_by_id*
2. CSS class name : *find_element_by_class_name*
3. name attribute : *find_element_by_name*
4. DOM structure atau Xpath : *find_element_by_xpath*
5. tagName : *find_element_by_tag_name()*
6. link text : *find_element_by_link_text*
7. partial link text : *find_element_by_partial_link_text*
8. HTML tag name : *find_element_by_tag_name*

2.3.9 Test Case

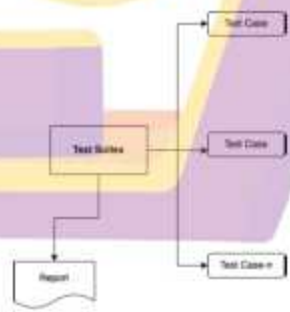
Test Case adalah susunan langkah yang terdiri atas gabungan antara test step, object repository dan datafiles sehingga membentuk sebuah satu kesatuan yang utuh seperti yang terlihat pada gambar 2.6. Test case disebut juga dengan skenario yang biasa dikerjakan manual oleh *Quality Assurance*. Namun dalam konteks ini semua step manual telah dikonversi kedalam bentuk script *automation*.



Gambar 2.6. Konsep Test Case

2.3.10 Test Suite

Test Suite Merupakan gabungan dari beberapa test case di dalam satu sistem. Test suite menggambarkan bagaimana proses aliran data terjadi dari awal hingga proses testing selesai. Setelah test suite dijalankan, maka akan ada report yang dihasilkan untuk selanjutnya dilakukan analisis berdasarkan test automation.



Gambar 2.7. Konsep Test Suite

Gambar 2.7 menjelaskan bagaimana test suite terbentuk dari beberapa test case yang kemudian akan menghasilkan report dari test. Test suite digunakan untuk

proses *end-to-end* dari sebuah sistem. Selain itu test suite juga bisa digunakan sebagai tempat mendefinisikan negative test yang biasa digunakan untuk memeriksa kondisi yang tidak sesuai dengan *requirement* yang ada.

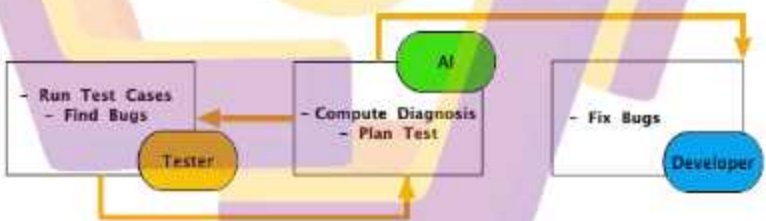
2.4 Pengujian Otomatis Berbasis Kecerdasan Buatan

Perkembangan dunia kecerdasan buatan atau biasa disebut dengan AI (Artificial Intelligence) banyak diimplementasikan pada kegiatan yang berhubungan dengan pemrosesan komputasi. Pengujian otomatis masuk ke dalam pengembangan pengujian berbasis kecerdasan buatan yang bertujuan agar proses testing lebih cepat dan adaptif. Salah satu contoh penerapan AI dalam pengujian otomatis perangkat lunak adalah membuat kode pengujian secara otomatis berdasarkan analisis fitur dan perilaku aplikasi. Penulisan kode otomatis pada umumnya merupakan serangkaian perintah pada bahasa pemrograman tertentu yang harus diatur secara manual mulai dari fungsi, konfigurasi perangkat lunak dan konfigurasi library yang terlibat. Namun pada AI hal tersebut tidak berlaku karena perilaku AI mempelajari langsung bagaimana perilaku aplikasi sehingga membuat proses pembuatan kode menjadi lebih cepat.

Selain pembuatan kode pengujian otomatis, AI juga dapat digunakan dalam identifikasi dan membuat skala prioritas dari kasus pengujian. Dengan memanfaatkan teknik pengelompokan dan klasifikasi data. AI dapat menganalisis riwayat pengujian dan perilaku pengguna untuk mengidentifikasi area yang rentan dan menyarankan kasus pengujian yang paling penting untuk dikerjakan. Selain itu, AI juga dapat digunakan dalam analisis dan pengujian data. Dalam pengujian perangkat lunak yang melibatkan manipulasi dan analisis data besar, AI dapat

digunakan untuk mengidentifikasi pola, anomali, dan tren yang tidak terdeteksi oleh pengujian secara manual. Contoh aplikasi yang memakai kecerdasan buatan dalam pembuatan kode pengujian otomatis antara lain Testim.io, Functionalize, Applitools dan Katalon Studio.

Pengujian otomatis memiliki peran yang sangat penting dalam proses pengembangan perangkat lunak khususnya meningkatkan efektifitas pengujian perangkat lunak. Ada banyak *tools* yang tersedia dalam mendukung pengujian otomatis, salah satunya adalah kecerdasan buatan (AI) dan *machine learning* (ML). Pemakaian teknologi kecerdasan banyak diimplementasikan pada pengembangan perangkat lunak menggunakan aplikasi berbasis AI dengan menggunakan teknik dan model dari *machine learning* (Minimol, 2021). Algoritma AI membantu proses pengujian perangkat lunak secara lebih efisien dalam membantu penguji dalam menemukan error dalam waktu yang lebih cepat.



Gambar 2.8. Implementasi AI dalam Proses Pengujian Perangkat Lunak

Gambar 2.8 menunjukkan bagaimana kecerdasan buatan terlibat dalam proses pengujian perangkat lunak. Proses pengujian dilakukan dengan menggunakan bantuan aplikasi pengujian berbasis kecerdasan buatan. Aplikasi AI akan mengirimkan pesan ke developer ketika terjadi galat sehingga proses

perbaikan lebih cepat. Beberapa pengujian dilakukan secara manual berdasarkan hasil diagnosis yang dilakukan aplikasi AI. Hal ini dikarenakan beberapa fungsi dari sebuah sistem aplikasi tidak diuji secara otomatis, namun tetap harus dilakukan secara manual.

AI membantu menghasilkan test case dan melakukan identifikasi test dalam melakukan proses verifikasi dan validasi sistem (Minimol, 2021). Proses pembuatan kode yang baik pada proses diagnosis test case harus disusun secara sistematis dan tepat oleh penguji. Oleh karena itu struktur yang tepat sangat dibutuhkan sehingga *machine learning* mampu memproses pengujian perangkat lunak secara efektif dan adaptif dalam menghadapi perubahan.

BAB III

METODE PENELITIAN

3.1. Jenis, Sifat, dan Pendekatan Penelitian

Penelitian ini menggunakan jenis penelitian eksperimental. Hal ini dikarenakan dilakukan serangkaian tindakan pengumpulan data berupa objek locator, masukkan sistem, pengujian menggunakan 4 desain kerangka kerja dan pengujian kerangka kerja berdasarkan aktifitas yang dilakukan pada saat perubahan sistem.. Sifat dari penelitian ini dilakukan secara mandiri menggunakan metode kuantitatif mulai dari pengumpulan data yang kemudian diuji. Hasil dari pengujian tersebut dicari nilai TS yang paling besar.. Pendekatan penelitian ini menggunakan metode pendekatan kuantitatif terhadap data hasil pengujian otomatis dan manual yang dikumpulkan kemudian dihitung menggunakan beberapa persamaan lainnya.

Hasil akhir dari penilitian ini kemudian akan ditampilkan dalam bentuk grafik. Hasil perbandingan nilai TS terbesar sebagai referensi dalam mengambil keputusan kerangka kerja yang tepat dalam merancang pengujian otomatis.

3.2. Metode Pengumpulan Data

Data yang dikumpulkan pada penelitian ini terdiri dari data locator web, input data sistem dan data website yang digunakan sebagai bahan eksperimen. Data locator web dikumpulkan dengan melakukan proses inspeksi halaman web dengan mencari nilai properti html. Nilai tersebut kemudian disimpan untuk digunakan pada kode otomatis. Sedangkan input data sistem disimpan pada penyimpanan

awan googlespreadsheet. Input data diakses dengan menggunakan API yang disediakan oleh google sesuai dengan kebutuhan dari sistem.

3.3. Metode Analisis Data

Metode yang digunakan dalam penelitian ini adalah studi pustaka, observasi dan eksperimen.

3.3.1 Studi Pustaka

Studi pustaka adalah suatu proses penyelidikan yang dilakukan terhadap literatur atau sumber informasi yang relevan dengan topik penelitian tertentu. Menurut Sugiyono (2016), studi pustaka merupakan proses mencari, membaca, memahami, dan menelaah berbagai literatur yang relevan dengan topik penelitian untuk mendapatkan informasi, data, teori, atau pandangan yang dapat digunakan sebagai landasan dalam merumuskan masalah penelitian. Studi pustaka pada penelitian ini menggunakan sumber dari jurnal, prosiding, *research article* dan buku.

3.3.2 Metode Observasi

Metode observasi adalah salah satu pendekatan yang digunakan dalam penelitian untuk mengumpulkan data, mengamati dan mencatat kejadian atau fenomena yang diamati secara langsung dari proses pengujian otomatis. Dalam metode ini, peneliti tidak melakukan intervensi aktif terhadap lingkungan atau subjek yang diamati, tetapi hanya mengamati apa yang terjadi secara alami. Peneliti mengumpulkan data melalu observasi terkait dengan testing software.

Data pengujian dikumpulkan berdasarkan tempat penyimpanan test case baik secara online maupun secara offline. Data online merupaka data yang disimpan

pada *cloud server* dan bersifat realtime untuk diedit. Data jenis ini bisa didapatkan dengan menggunakan test case management seperti google spreadsheet, test rail dan perangkat lunak test case management lainnya. Data data yang akan dikumpulkan peneliti antara lain data locator dan data masukkan dari sistem.

1. Data locator
2. Jumlah masukkan sistem
3. Total waktu yang dibutuhkan dalam proses testing baik secara manual dan automation untuk didapatkan nilai efektivitas

3.3.3 Metode Eksperimen

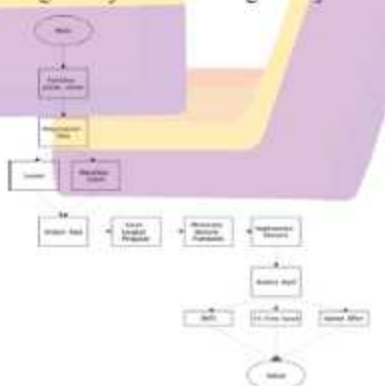
Metode eksperimen adalah salah satu pendekatan penelitian yang digunakan untuk menguji hipotesis kausal tentang hubungan antara dua atau lebih variabel. Dalam metode ini, peneliti secara sistematis memanipulasi variabel independen untuk mengamati efeknya terhadap variabel dependen, sambil mengontrol faktor-faktor lain yang dapat memengaruhi hasil. Variabel independen dalam penelitian adalah locator dan masukkan data. Sedangkan variabel dependen adalah waktu dari pengujian otomatis yang hasilnya akan diamati untuk menghasilkan hasil analisa dalam bentuk nilai parameter EMTE, T.S(*time saved*) dan *update effort.*

3.4. Alur Penelitian

Alur penelitian adalah tahapan atau langkah-langkah dalam penelitian, mengacu pada rangkaian proses yang harus dilalui oleh peneliti untuk merencanakan, melaksanakan, dan mengevaluasi sebuah penelitian. Pada penelitian ini proses pengambilan data hingga proses pengujian yang terlihat pada gambar 3.1 sehingga mempermudah pemahaman dalam penelitian yang dilakukan.

Gambar 3.1 menjelaskan bagaimana penelitian dilakukan dengan dilakukan terlebih dahulu pemilihan sistem serta pengumpulan data locator dan masukkan sistem. Kedua data tersebut penting diketahui untuk membangun langkah pengujian yang akan diimplementasikan pada kerangka kerja 1, kerangka kerja 2, kerangka kerja 3 dan kerangka kerja 4. Pada masing-masing framework akan memiliki langkah pengujian yang sama namun dalam hal teknik *scipting* akan dilakukan kombinasi sesuai dengan framework yang diimplementasi.

Pengujian akan menghasilkan 3 parameter sebagai data dalam menentukan framework mana yang paling optimal dan efektif. Parameter EMTE dan T.S (*Time saved*) digunakan untuk mendeskripsikan total waktu yang berhasil dioptimalkan pada saat proses pengujian berlangsung. Sedangkan parameter update effort merupakan total usaha yang dilakukan ketika sistem mengalami perubahan. Nilai dar parameter *update effort* didapatkan dengan melakukan pengujian perubahan sistem yang menghasilkan total langkah yang dilakukan oleh kerangka kerja 1, kerangka kerja 2, kerangka kerja 3 dan kerangka kerja 3.



Gambar 3.1. Alur Penelitian

Berdasarkan gambar 3.1 maka pembagian alur penelitian secara spesifik dapat terlihat pada tabel 3.1 berikut ini;

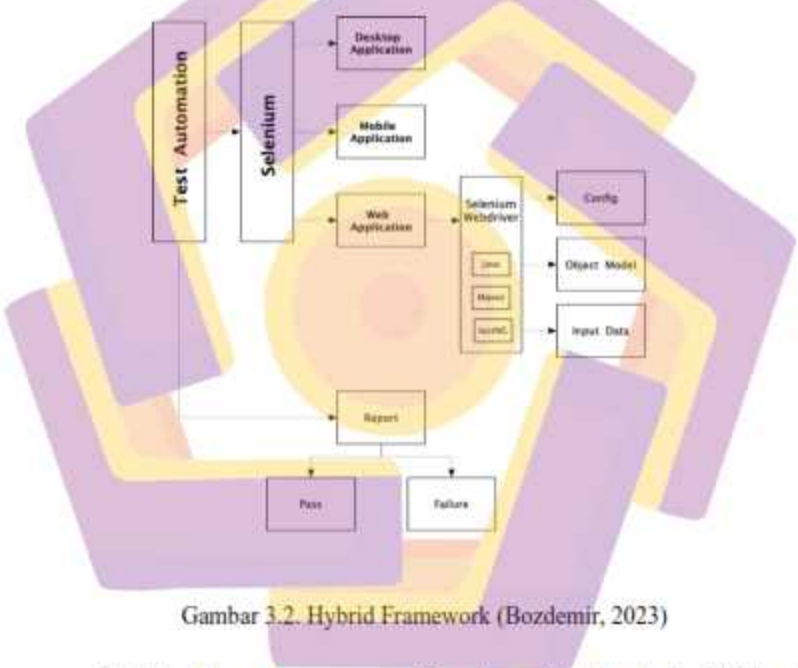
Tabel 3.1. Detail Alur Penelitian

Persiapan Pengujian
1. Menentukan proses dari 4 aplikasi berbasis website. Disarankan untuk memilih website yang berbeda untuk memastikan tingkat validitas pengujian lebih baik.
2. Mengambil dan menyimpan data dari locator yang dimiliki oleh *user interface* pada aplikasi.
3. Menentukan data masukkan sistem yang akan digunakan.
4. Membuat tempat penyimpanan data pada google spreadsheet sebagai tempat penyimpanan data realtime. 5. Menyusun langkah pengujian yang diimplementasikan dalam bentuk *language programming code* dalam bentuk 4 tipe sesuai dengan skenario. 6. Menggunakan locator dan data yang telah dikumpulkan sebelumnya pada kode otomatis sesuai dengan skenario yang digunakan.
Pengujian
1. Jalankan pengujian otomatis sesuai dengan 4 tipe skenario yang telah disusun sebanyak 30 kali untuk satu tipe skenario pada 1 proses. Serta 30 kali pengujian manual. Sehingga untuk 1 proses aplikasi membutuhkan 120 kali percobaan
2. Lakukan hal yang sama pada 3 aplikasi lainnya sesuai dengan langkah nomor 1
Analisis Hasil Pengujian
1. Cari nilai rata-rata keseluruhan pengujian otomatis pada masing masing skenario dan nilai rata- rata pengujian manual sehingga didapatkan nilai pengujian untuk satu kali pengujian.
2. Cari nilai EMTE berdasarkan persamaan 1 untuk menghasilkan total waktu pengujian otomatis
3. Cari nilai efisiensi dimana nilai EMTE menggambarkan total waktu yang berhasil meng-*cover* berapa persen dari total pengujian manual
4. Cari nilai fragility yang merupakan total dari pengujian manual ditambahkan dengan akitivitas ubah data. Nilai fragility merupakan nilai estimasi. 5. Cari nilai TS menggunakan persamaan 2

3.5 Kerangka Kerja Hybrid Sebelumnya (*Existing Hybrid Framework*)

Kerangka kerja dalam pengujian perangkat lunak didefinisikan sebagai kumpulan alat, konvensi, dan standar yang digunakan bersama untuk menguji komponen perangkat lunak atau sistem secara menyeluruh (Myers, G. J., 2004). Pada penelitian ini terdapat beberapa kerangka kerja yang sudah ada di dalam penelitian sebelumnya. Gambar 3.2 adalah kerangka kerja yang diimplementasikan

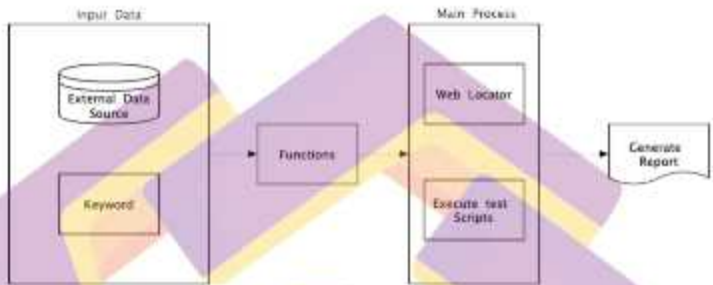
dengan menerapkan 3 konsep konfigurasi, pemodelan objek dan konsep input data. Konfigurasi merupakan bagian untuk membuat kondisi awal sebelum pengujian otomatis berjalan. Sedangkan pemodelan objek menitikberatkan pada manajemen web locator yang menjadi komunikasi antara webdriver dengan sistem aplikasi. Data data terkait test case kemudian disimpan di dalam sebuah file excel sehingga input sistem bisa dilakukan ketika proses berjalan.



Gambar 3.2. Hybrid Framework (Bozdemir, 2023)

Berbeda dengan konsep yang diterapkan oleh (Bozdemir, 2023), pada gambar 3.3 menunjukkan penelitian yang dilakukan oleh (Hanna, 2018) menunjukkan keterlibatan fungsi dalam proses pengujian otomatis. Fungsi tersebut mendapatkan akses input data dari penyimpanan eksternal data untuk selanjutnya dikirim ke kode utama pengujian otomatis. Setelah seluruh proses selesai

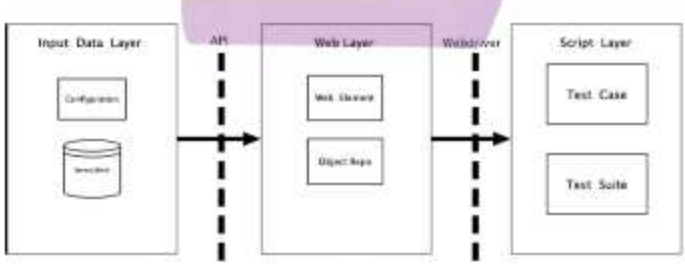
dilakukan, maka akan menghasilkan laporan dari pengujian otomatis. Namun pada kerangka kerja tersebut belum bisa menjelaskan bagaimana pembentukan kode otomatis atau struktur sehingga menghasilkan kerangka kerja yang optimal.



Gambar 3.3. Hybrid Framework (Hanna, 2018)

3.6 Pembentukan Kerangka Kerja *Hybrid* Usulan

Dalam proses penyusunan kerangka kerja, maka ada proses yang disebut dengan inisialisasi atau langkah pengoperasian awal sehingga menghasilkan kerangka kerja yang baik. Pada penelitian ini terdapat langkah pengoperasian awal seperti yang terlihat pada gambar 3.4.



Gambar 3.4. Langkah Pembentukan Kerangka Kerja

Gambar 3.4 menjelaskan pembentukan awal kerangka kerja yang terdiri atas 3 layer yakni input data, web, dan script. Layer input data berfungsi untuk mendefinisikan input sistem yang akan diberikan beserta pengaturan awal dari pengujian otomatis. Data input sistem disimpan pada penyimpanan awal yakni *google spreadsheet*. Data awan ini bertujuan meningkatkan tingkat kolaborasi secara real time dibandingkan dengan memakai penyimpanan pada *local storage*.

3.6 Spesifikasi Kode Otomatis

Pada proses pembuatan kode otomatis tentu akan memiliki spesifikasi yang merupakan hasil dari kesepakatan seluruh *stakeholder* ketika perangkat lunak dibuat. Detail yang jelas pada proses pembuatan spesifikasi akan menentukan apakah suatu menu atau alur kerja dapat dikonversi kedalam bentuk kode otomatis. Pembuatan kode otomatis membutuhkan beberapa kelengkapan agar mampu menggantikan proses pengujian manual. Berikut ini adalah spesifikasi yang dibutuhkan ketika proses pembuatan kode otomatis :

1. Jumlah halaman dari fitur
2. Input data sistem
3. Jenis input dari sistem
4. Locator yang melekat pada komponen input data
5. *System security* (apakah sistem diperbolehkan untuk di-*automatisasikan* dilihat dari sisi keamanan)

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

Berdasarkan hasil penelitan terdapat 4 skenario desain automatisasi. Setiap skenario memiliki alur yang dimodifikasi sehingga bisa terlihat perbedaan output serta memiliki kelebihan dan kekurangan dari tiap skenario. Desain standar merupakan desain yang digunakan ketika pembentukan *automation script* dilakukan pertama kali dalam proses konversi automation. Sedangkan desain rekomendasi adalah desain yang dirancang sebagai dasar dalam menyusun *framework* berdasarkan perbandingan dari 4 skenario yang diuji. Pada pengujian terdapat beberapa faktor yang berpengaruh terhadap waktu dari pengujian otomatis antara lain kecepatan internet, banyaknya user yang mengakses website dan kemampuan load sebuah website. Kecepatan load halaman sebuah website dipengaruhi oleh berbagai faktor antara lain;

1. Ukuran dan Kompleksitas Konten: Semakin besar dan kompleks konten sebuah halaman web, semakin lama waktu yang dibutuhkan untuk mengunduhnya.
2. Kinerja Server: Kinerja server tempat website di-host memainkan peran penting dalam kecepatan load halaman. Server yang lambat atau kelebihan beban dapat menyebabkan keterlambatan dalam proses pengujian otomatis.
3. Penggunaan Cache: Penggunaan cache memungkinkan browser untuk menyimpan salinan halaman web dan asetnya secara lokal. Jika halaman

web telah di-*cache* sebelumnya, waktu load dapat berkurang secara signifikan.

4. Koneksi Internet Pengguna: Kecepatan koneksi internet pengguna juga mempengaruhi waktu load halaman. Koneksi internet yang lambat akan memperlambat proses pengujian otomatis
5. Optimalisasi Gambar dan Aset: Ukuran dan format gambar serta aset lainnya dapat mempengaruhi kecepatan load halaman.
6. Optimalisasi Kode: Kualitas dan efisiensi kode HTML, CSS, dan JavaScript juga berpengaruh terhadap kecepatan load halaman. Kode yang terlalu panjang atau kompleks dapat memperlambat proses render halaman.
7. Desain Responsif: Desain responsif yang dioptimalkan untuk perangkat bergerak dapat membantu mempercepat waktu load halaman pada perangkat mobile.

4.1 Persiapan Pengujian

Pada tahapan ini dilakukan penentuan penggunaan aplikasi website. Website yang digunakan terdiri dari 4 jenis website yang berbeda dalam mengimplementasikan kerangka kerja yang akan dirancang. Website yang digunakan terdiri atas ;

1. Linkedin (sosial media professional)
2. KAI (PT Kereta Api Indonesia)
3. MyHorison
4. AutomationExercise

Masing masing website hanya diambil sample pada satu proses tertentu. Proses yang digunakan dalam penelitian ini antara lain proses login (linkedin), proses pencarian tiket (KAI), proses pendaftaran akun customer hotel (myHorison) dan proses checkout online shop (AutomationExercise).

Data yang digunakan pada penelitian ini merupakan data locator dan masukkan input yang dimiiliki oleh sistem. Tabel 4.1 Merupakan total locator dan masukkan sistem yang ada pada masing masing proses. Locator merupakan pengenal dari komponen website dalam menerima aksi dari pengujian otomatis ketika berinteraksi dengan sistem. Sedangkan masukkan sistem adalah data yang dimasukkan oleh pengujian otomatis ke dalam sistem. Masukkan yang dijalankan oleh pengujian otomatis diarahkan ke komponen yang tepat pada website sesuai dengan jenis masukkan sistem.

Tabel 4.1. Total Locator dan masukkan sistem

Proses	Jumlah locator	Jumlah masukkan sistem	Total
logIn	4	2	6
Pencarian tiket	15	5	20
create account	9	5	16
Checkout online shop	19	11	30

Komponen website terdiri dari beberapa input yang memiliki properties. Properties dari website berfungsi untuk identifikasi unik dari sebuah input sehingga webdriver bisa mengenali sebuah komponen, kemudian mengirim perintah untuk memasukkan data ke dalam komponen website yang telah didefinisikan dalam tipe CSS *selector id, name, className* atau *xpath*. berikut ini merupakan cara untuk mengambil nilai dari *selector* komponen website :

1. Buka software perambah website

2. Masukkan alamat website
3. Klik kanan pada komponen website
4. Pilih menu inspect
5. Ambil nilai dari *id, name, className, tag* dan lainnya.

Seluruh properties yang ada didalam sebuah komponen bisa bertindak sebagai locator. Berikut ini merupakan contoh dari komponen website;

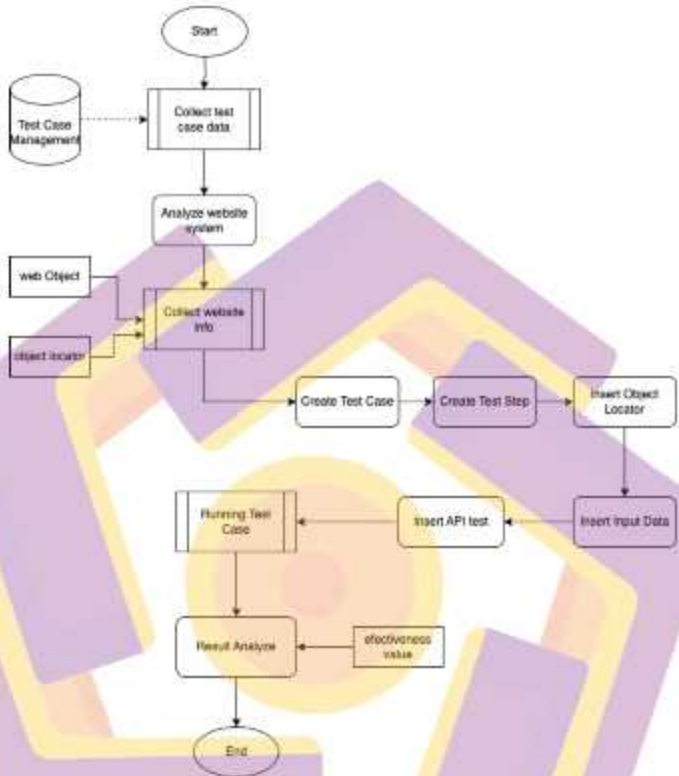
<input name="first_name" placeholder="First Name" class="form-control" type="text" data-bv-field="first_name">

Keseluruhan kode bisa digunakan sebagai pengenal sebuah komponen. Sebaiknya menggunakan locator yang memiliki keunikan dibanding komponen lainnya. Tipe CSS *selector* yang biasa digunakan adalah id atau name. Kedua tipe tersebut selain digunakan untuk pengenalan komponen website, juga digunakan sebagai wadah melempar nilai parameter dari satu fungsi ke fungsi yang lain..

4.2 Pengujian

Pengujian dilakukan dengan menggunakan skenario yang telah disusun. Skenario merupakan kumpulan langkah yang digunakan sebagai dasar pengujian dari *framework* rekomendasi yang akan muncul. Pada penelitian ini terdapat 4 skenario yang digunakan yakni;

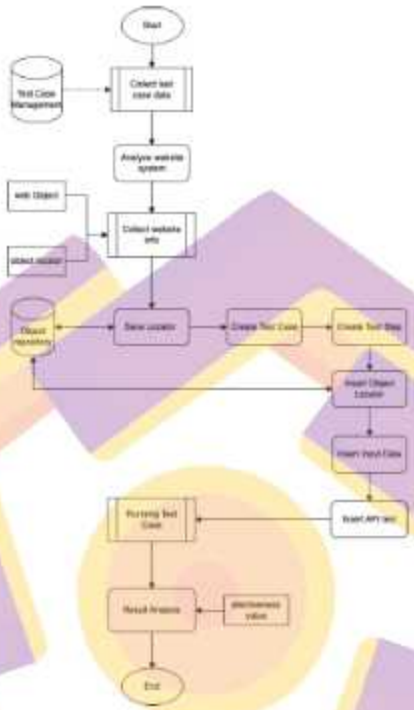
1) Skenario 1



Gambar 4.1. Desain Framework Skenario 1

Gambar 4.1. menunjukkan pembentukan desain framework skenario 1 dimana seluruh input data dan locator dimasukkan ke dalam sebuah test case. input data dan locator didefinisikan langsung pada kode otomatis atau secara *hard code*. Langkah pada skenario 1 dimulai dengan mengidentifikasi test case kemudian memasukkan seluruh data dan objek dari web kemudian test case dijalankan.

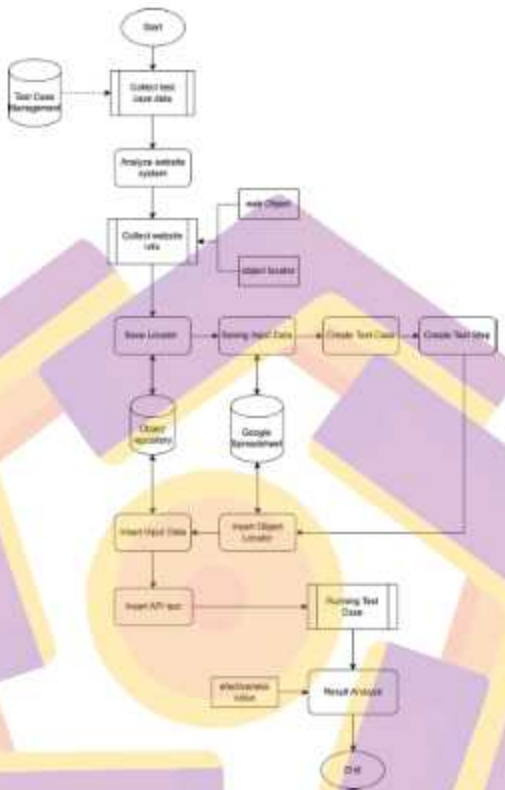
2) Skenario 2



Gambar 4.2. Desain Framework Skenario 2

Gambar 4.2 menunjukkan pembentukan desain framework skenario 2 dimana terdapat perbedaan pada cara menyimpan locator. Jika pada skenario 1 (gambar 4.1) locator langsung dipakai di dalam test case, maka pada skenario 2 locator tersentralisasi dan disimpan ke dalam sebuah tempat yang disebut dengan object repository. Ketika test case ingin menggunakan locator, maka cukup hanya memanggil nama parameter dari locator untuk menggunakan nilai locator. Hal ini berbeda dengan skenario 1 dimana nilai locator langsung didefinisikan pada langkah di dalam test case.

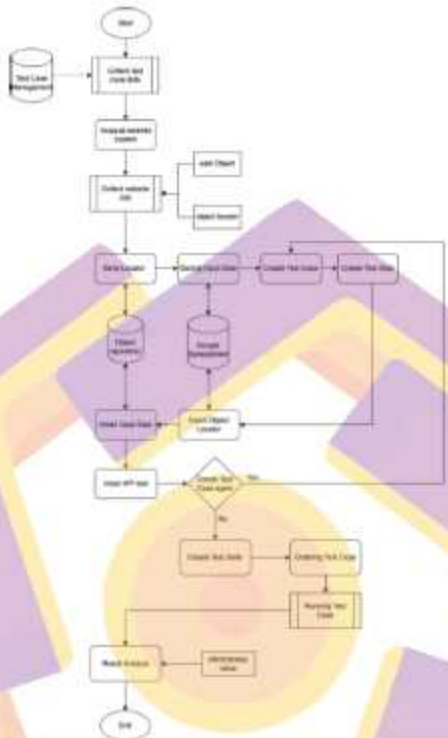
3) Skenario 3



Gambar 4.3. Desain Framework Skenario 3

Gambar 4.3 menunjukkan pembentukan desain framework skenario 3 dimana terdapat perbedaan pada cara menyimpan set data input. Jika pada skenario 1 dan skenario 2 data input langsung dipakai di dalam test case, maka pada skenario 3 data disimpan ke dalam penyimpanan berbagi file *atau cloud storage* dalam hal ini google spreadsheet. Hal ini sangat direkomendasikan sebab isi data tidak terlihat secara langsung pada langkah pengujian sehingga memperkuat tingkat privasi data dalam proses pengujian.

4) Skenario 4



Gambar 4.4. Desain Framework Skenario 4

Gambar 4.4 memperlihatkan perbedaan dari sisi test suite. pada skenario 1 hingga skenario 3 seluruh langkah pengujian disatukan ke dalam 1 test case. Sedangkan pada skenario 4 terdapat pemisahan terhadap test case yang ada. Hal ini berarti 1 test case dipecah menjadi beberapa bagian sehingga lebih mudah untuk di-*maintenance*.

4.3 Analisis Hasil Pengujian

Pengujian menggunakan 4 sistem yakni proses login, proses create account user hotel, proses pencarian tiket kerata api dan proses checkout belanja online.

Masing masing proses mengimplementasikan 4 skenario dimana masing masing seknario terdiri dari 30 kali percobaan. Selain itu juga dilakukan pengujian manual sebanyak 30 kali untuk masing masing proses. Jika ditotal maka 4 proses tersebut melakukan pengujian sebanyak 600 kali percobaan.

Berdasarkan hasil dari 600 kali pengujian maka didapatkan hasil waktu pengujian otomatis dan manual dari masing - masing proses seperti yang terlihat pada tabel 4.2 sampai 4.5. Satuan waktu yang digunakan adalah detik. Skenario 1 sampai 4 adalah pengujian otomatis sedangkan kolom manual menggambarkan pengujian manual atau yang dilakukan oleh manusia.

Tabel 4.2. Hasil pengujian skenario proses *login*

Percobaan ke-	Skenario 1 (detik)	Skenario 2 (detik)	Skenario 3 (detik)	Skenario 4 (detik)	Manual (detik)
1	10.44	10.65	11.28	10.54	15.43
2	9.51	10.08	10.76	11.24	13.16
3	9.38	9.33	10.67	9.17	15.02
4	10.15	11.68	8.71	10.25	15.04
5	9.94	10.24	10.39	10.22	17.02
6	11.04	9.99	9.17	10.89	13.11
7	9.74	12.61	10.17	9.24	12.11
8	9.72	9.98	10.24	8.64	15.19
9	8.94	10.04	9.68	8.72	14.86
10	7.56	10.65	8.31	9.57	12.34
11	7.84	10.11	9.79	8.77	11.11
12	9.34	9.68	10.78	10.12	15.22
13	8.72	9.17	8.9	8.72	13.22
14	8.29	10.6	8.99	10.8	15.08
15	7.88	10.06	9.58	9.2	15.42
16	9.28	11.92	8.97	9.91	16.89
17	9.66	14.29	9.45	10.86	16.74
18	8.09	9.3	9.54	14.16	16.77
19	10.61	9.52	7.86	11.96	17.62
20	8.69	9.02	10.35	8.38	18.66
21	9.59	9.03	9.38	8.76	19.22
22	10.95	9.12	9.3	8.78	20.11
23	10.73	9.47	9.52	9.07	17.35
24	9.46	8.73	9.14	8.54	19.56

Tabel 4.2. (Lanjutan)

Percobaan ke-	Skenario 1 (detik)	Skenario 2 (detik)	Skenario 3 (detik)	Skenario 4 (detik)	Manual (detik)
25	9.23	11.02	8.88	9.42	15.76
26	12.84	10.53	10.57	10.93	18.89
27	10.5	10.33	9.49	14.63	19.42
28	11.19	8.68	8.74	10.88	19.23
29	9.2	9.57	9.16	9.27	18.88
30	8.46	8.95	9.19	9.69	18.32
Rata- Rata	9.5657	10.145	**9.5653**	10.0443	16.2250
EMTE	286.97	304.35	**286.96**	301.33	

Pada tabel 4.2 terlihat nilai rata-rata pengujian otomatis yang paling rendah berada pada skenario 3 dengan nilai 9.5653. Nilai tersebut menjelaskan bahwa pengujian yang dilakukan menggunakan kode otomatis hanya membutuhkan waktu 9.5653 detik atau lebih cepat 6.6597 detik dibandingkan dengan pengujian manual. Hal ini juga berbanding lurus dengan nilai *automation effort* atau EMTE skenario 3 yang memiliki nilai paling kecil atau tercepat melakukan pengujian otomatis yakni 286.96 detik dibandingkan skenario yang lainnya. Nilai tersebut merupakan total waktu yang dipakai oleh pengujian otomatis selama 30 kali percobaan. Pada skenario ini locator sudah tersentralisasi pada objek repositori dan masukkan input sudah *realtime* pada googlespreadsheet.

Tabel 4.3. Hasil pengujian skenario proses pencarian tiket

Percobaan ke-	Skenario 1 (detik)	Skenario 2 (detik)	Skenario 3 (detik)	Skenario 4 (detik)	Manual (detik)
1	17.97	20.13	25.82	23.99	29.3
2	23.44	16.11	18.89	21.06	28.39
3	15.66	19.68	20.19	17.88	24.5
4	19.38	20.03	29.06	20.34	28.72
5	19.55	19.14	19.99	23.4	26.01
6	17.87	20.19	18.24	21.18	25.41
7	18.37	30.56	17.83	34.5	24.68
8	17.08	17.86	18.95	19.71	26.6

Tabel 4.3. (Lanjutan)

Percobaan ke-	Skenario 1 (detik)	Skenario 2 (detik)	Skenario 3 (detik)	Skenario 4 (detik)	Manual (detik)
9	16.83	19.24	28.25	29.31	24.68
10	18.05	19.78	17.33	19.71	27.64
11	17.73	22.09	19.2	20.3	26.68
12	16.57	22.46	19.81	30.72	26.06
13	18.27	22.25	19.46	25.99	23.89
14	18.02	20.49	18.33	19.97	25.46
15	18.66	20.24	25.33	24.93	28.63
16	20.77	22.13	19.11	18.59	28.59
17	19.13	32.13	18.76	20.38	24.34
18	18.21	20.75	19.59	20.02	28.64
19	17.65	20.92	26.12	21.26	26.72
20	16.9	29.48	19.25	23.48	26.1
21	18.11	17.71	18.46	22.44	24.06
22	16.6	19.13	18.89	19.72	24.58
23	18.2	18.38	19.17	29.42	26.95
24	35.84	21.48	26.82	20.65	28.75
25	17.68	20.41	18.93	19.59	25.4
26	17.97	18.09	19.24	20.11	29.16
27	16.43	33.76	21.11	19.89	28.96
28	19.4	22	20.21	24.48	28.03
29	22.02	18.98	18.7	19.86	23.97
30	19.95	20.2	36.92	21.12	24.81
Rata-Rata	**18.9437**	21.5267	21.2653	22.4667	26.5237
EMTE	**568.31**	645.8	637.96	674	

Pada tabel 4.3 terlihat nilai rata-rata pengujian otomatis yang paling rendah berada pada skenario 1 dengan nilai 18.9437. Nilai tersebut menjelaskan bahwa pengujian yang dilakukan menggunakan kode otomatis hanya membutuhkan waktu 18.9437 detik atau lebih cepat 7,58 detik dibandingkan dengan pengujian manual. Hal ini juga berbanding lurus dengan nilai *automation effort* atau EMTE skenario 1 yang memiliki nilai paling kecil atau tercepat melakukan pengujian otomatis yakni 568.31 detik. Pada skenario ini baik locator atau input data masih dilakukan manual pada kode otomatis atau *hard code*.

Tabel 4.4. Hasil pengujian skenario proses *create account*

Percobaan ke-	Skenario 1 (detik)	Skenario 2 (detik)	Skenario 3 (detik)	Skenario 4 (detik)	Manual (detik)
1	9.13	10.25	10.5	11.53	25.03
2	8.1	11.44	9.41	11.86	24.28
3	8.35	9.67	9.6	10.97	24.68
4	8.73	9.37	10.77	10.41	26.51
5	8.61	10.05	9.72	12.53	24.16
6	8.2	11.16	10.69	10.46	23.94
7	8.67	9.44	9.4	10.75	24.67
8	8.34	9.58	9.72	10.11	26.44
9	8.69	9.61	10.42	10.36	25.48
10	8.82	9.63	9.74	10.52	26.42
11	8.41	9.55	9.06	9.85	24.37
12	8.5	10.13	9.32	10.01	24.47
13	10.45	9.36	11.94	9.64	23.49
14	10.23	9.34	9.07	10	24.6
15	8.82	9.9	10.89	10.08	23.5
16	8.9	9.62	9.91	10.18	26.16
17	8.58	10.1	9.35	10.49	24.3
18	8.54	9.54	9.52	10.5	24.46
19	8.7	9.87	13.68	10.6	25.74
20	8.79	9.48	10.04	10.61	26.37
21	8.28	9.38	9.18	10.54	27.2
22	8.18	9.5	9.95	10.32	25.55
23	7.86	9.87	9.6	12.29	26.39
24	8.28	9.44	11.34	11.1	25.33
25	8.42	9.9	10.03	10.92	26.78
26	8.17	9.58	9.64	10.61	25.02
27	8.84	9.45	9.3	10.69	24.89
28	8.59	9.18	9.61	10.55	24.38
29	8.8	9.1	9.29	10.25	24.64
30	8.62	9.51	9.45	10.46	23.68
Rata-rata	**8.6533**	9.7333	10.0047	10.6397	25.0977
EMTE	**259.6**	292	300.14	319.19	

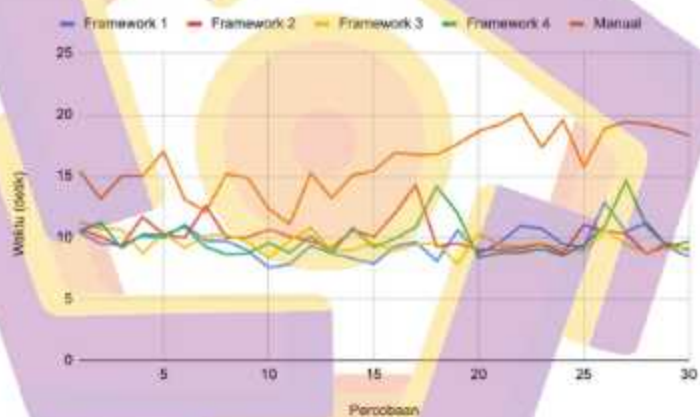
Pada tabel 4.4 terlihat nilai rata-rata pengujian otomatis yang paling rendah berada pada skenario 1 dengan nilai 8.6533. Nilai tersebut menjelaskan bahwa pengujian yang dilakukan menggunakan kode otomatis hanya membutuhkan waktu 8.6533 detik atau lebih cepat 16.4444 detik dibandingkan dengan pengujian

manual. Hal ini juga berbanding lurus dengan nilai *automation effort* atau EMTE skenario 1 yang memiliki nilai paling kecil atau tercepat yakni 259.6 detik. Nilai tersebut merupakan total waktu yang dipakai oleh pengujian otomatis selama 30 kali percobaan. Pada skenario ini baik locator atau input data masih dilakukan manual pada kode otomatis atau *hard code*.

Tabel 4.5. Hasil pengujian skenario proses *checkout shopping bag*

Percobaan ke-	Skenario 1 (detik)	Skenario 2 (detik)	Skenario 3 (detik)	Skenario 4 (detik)	Manual (detik)
1	31.11	28.69	24.7	40.75	53.42
2	27.48	21.12	24.34	29.51	51.92
3	32.18	24.51	25.11	29.47	53.07
4	28.71	26.26	27.71	29.99	50.28
5	27.1	24.43	22.89	31.26	50.73
6	23.91	26.51	24.91	25.44	48.91
7	29.33	21.26	101.57	27.66	52.03
8	29.08	28.17	22.59	26.32	50.08
9	47.43	24.52	23.08	23.89	49.54
10	34.31	23.84	24.44	28.1	52.99
11	24.13	29.02	26.92	26.85	48.54
12	8.5	10.13	9.32	10.01	24.47
13	10.45	9.36	11.94	9.64	23.49
14	10.23	9.34	9.07	10	24.6
15	22.02	27.16	28.64	29.34	53.26
16	22.07	21.8	22.78	31.04	53.38
17	26.19	26.04	25.43	30.64	48.41
18	25.98	28.4	23.73	28.67	53.73
19	29.73	24.86	22.45	28.26	52.45
20	23.21	29.68	21.59	29.69	51.29
21	29.34	24.38	22.55	37.04	52.81
22	28.07	24.13	27.77	30.89	51.88
23	22.62	40.67	21.87	27.36	50.35
24	30.34	26.52	28.44	27.29	51.86
25	23.13	23.16	22.73	31.09	49.93
26	27.49	22.53	28.35	37.89	49.35
27	26.04	25.09	25.33	27.67	52.72
28	28.36	28.24	25.58	25.9	48.9
29	28.78	21.56	27.31	30.82	52.76
30	22.72	26.52	28.5	27.67	49.53
Rata-rata	27.8967	**25.799**	27.3943	29.3857	51.1440
EMTE	836.9	**773.97**	821.83	881.57	

Pada tabel 4.5 terlihat nilai rata-rata pengujian otomatis yang paling rendah berada pada skenario 2 dengan nilai 25.799. Nilai tersebut menjelaskan bahwa pengujian yang dilakukan menggunakan kode otomatis hanya membutuhkan waktu 25.799 detik atau lebih cepat 25.345 detik dibandingkan dengan pengujian manual. Hal ini juga berbanding lurus dengan nilai *automation effort* atau EMTE skenario 2 yang memiliki nilai paling kecil yakni 773.97 detik. Nilai tersebut merupakan total waktu yang dipakai oleh pengujian otomatis selama 30 kali percobaan. Pada skenario ini locator sudah tersentralisasi pada objek repositori namun input data masih dilakukan manual pada kode otomatis atau *hard code*.



Gambar 4.5. Grafik hasil pengujian proses login

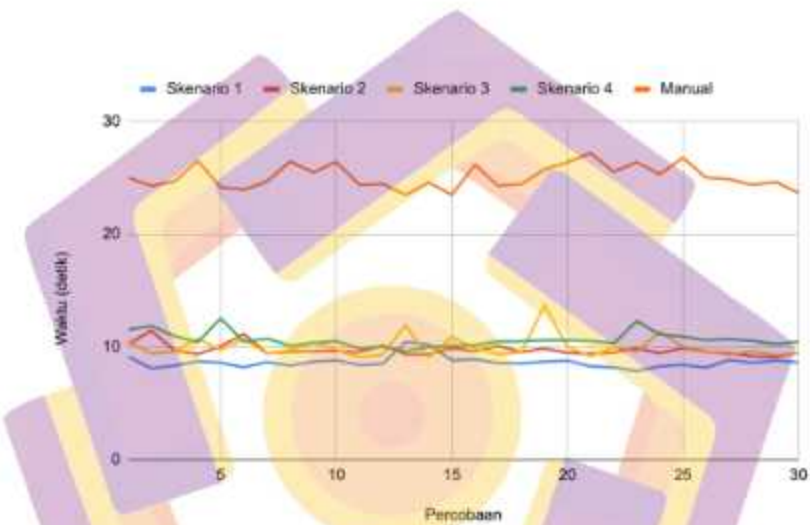
Gambar 4.5 merupakan representasi visual dari tabel 4.2 dimana masing-masing skenario diwakili oleh warna biru (skenario 1), merah (skenario 2), kuning (skenario 3), hijau (skenario 4) dan jingga (pengujian manual). Pada gambar 4.5 terlihat bahwa pengujian manual berada paling atas dari keseluruhan grafik yang ada. Hal ini menjelaskan bahwa pengujian manual memiliki waktu yang paling

lama dibandingkan dengan grafik lainnya yakni pengujian otomatis. Pada gambar 4.5 juga terlihat bentuk grafik yang naik turun. Hal ini merupakan pengaruh dari kecepatan load sebuah aplikasi yang berimpact kepada waktu proses pengujian manual atau otomatis. Pada gambar 4.5 pengujian yang paling cepat ditandai dengan warna kuning yakni skenario 3 dengan nilai 286.96 detik.

Gambar 4.6. Grafik hasil pengujian proses pencarian tiket

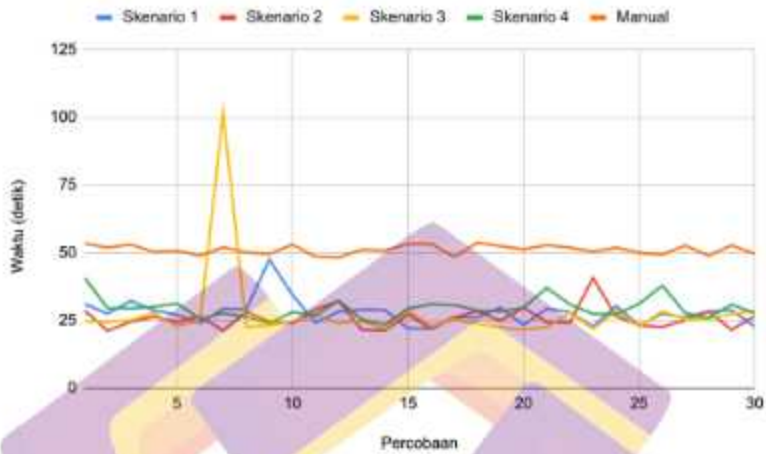
Gambar 4.6 merupakan representasi visual dari tabel 4.3 dimana masing-masing skenario diwakili oleh warna biru (skenario 1), merah (skenario 2), kuning (skenario 3), hijau (skenario 4) dan jingga (pengujian manual). Pada gambar 4.6 terlihat bahwa pengujian manual berada paling atas dari keseluruhan grafik yang ada. Hal ini menjelaskan bahwa pengujian manual memiliki waktu yang paling lama dibandingkan dengan grafik lainnya yakni pengujian otomatis. Pada gambar 4.6 juga terlihat bentuk grafik yang naik turun secara cukup tajam. Hal ini karena

pada proses penelitian bertepatan dengan banyaknya akses pada halaman pencarian tiket sehingga menyebabkan waktu loan website menjadi sangat tidak stabil. Pada gambar 4.6 pengujian yang paling cepat ditandai dengan warna biru yakni skenario 1 dengan nilai 568.31 detik.



Gambar 4.7. Grafik hasil pengujian proses *create account*

Gambar 4.7 merupakan representasi visual dari tabel 4.4 dimana masing-masing skenario diwakili oleh warna biru (skenario 1), merah (skenario 2), kuning (skenario 3), hijau (skenario 4) dan jingga (pengujian manual). Pada gambar 4.7 terlihat bahwa pengujian manual berada paling atas dari keseluruhan grafik yang ada. Hal ini menjelaskan bahwa pengujian manual memiliki waktu yang paling lama dibandingkan dengan grafik lainnya yakni pengujian otomatis.Pada gambar 4.7 pengujian yang paling cepat ditandai dengan warna biru yakni skenario 1 dengan nilai 259.6 detik.



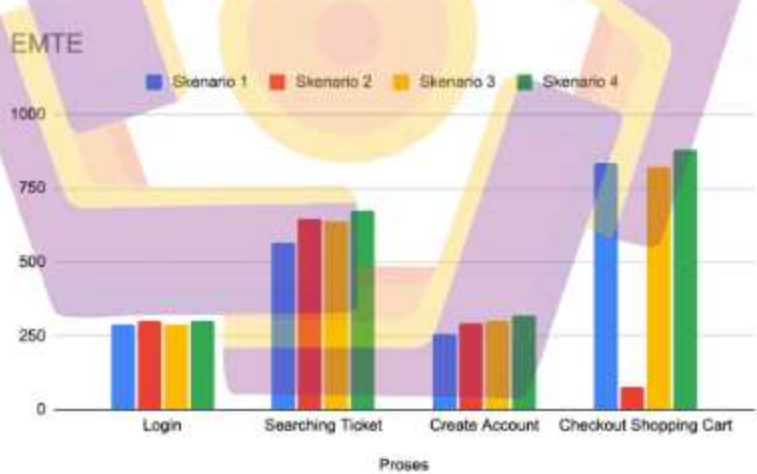
Gambar 4.8. Grafik hasil pengujian proses *checkout shopping cart*

Gambar 4.8 merupakan representasi visual dari tabel 4.5 dimana masing-masing skenario diwakili oleh warna biru (skenario 1), merah (skenario 2), kuning (skenario 3), hijau (skenario 4) dan jingga (pengujian manual). Pada gambar 4.8 terlihat bahwa pengujian manual berada paling atas dari keseluruhan grafik yang ada. Pada gambar 4.8 pengujian yang paling cepat ditandai dengan warna merah yakni skenario 2 dengan nilai 773.97 detik.

Pada grafik terlihat bahwa waktu yang dibutuhkan oleh pengujian selalu lebih lambat dibandingkan dengan pengujian yang dilakukan secara otomatis. Hal ini dikarenakan proses pengujian yang dilakukan secara manual selalu dipengaruhi oleh faktor kecepatan pengujian yang dilakukan oleh manusia. Manusia sebagai aktor dalam pengujian manual memiliki kecepatan refleks terhadap sebuah sistem yang berbeda beda dalam hal pengoperasian perangkat keras komputer seperti *mouse* dan *keyboard*. Perangkat keras input mempengaruhi dalam pengoperasian

sistem yang berimplikasi secara langsung terhadap total waktu pengujian. Semakin rendah kemampuan dalam melalukan input sistem maka pengujian akan semakin lama. Faktor lain yang terlibat adalah tingkat resistensi pengujian manual. Pada pengujian manual tingkat konsistensi akan berubah berubah dan cenderung tinggi karena keterbatasan kemampuan manusia dalam melakukan kegiatan yang berulang dalam jumlah yang banyak dalam satu waktu.

Pada proses testing perlu memperhatikan seberapa besar waktu yang berhasil dioptimalisasikan dan presentase optimalisasi pengujian otomatis terhadap manual. Oleh karena itu perlu dianalisis antara nilai EMTE dan nilai TS (*time saved*) dari sebuah pengujian. Berdasarkan hasil penelitian didapatkan nilai EMTE dari keseluruhan 4 proses seperti yang terlihat pada gambar 4.10.



Gambar 4.9. Grafik Equivalent Manual Test Effort

Pada Gambar 4.9 terlihat skenario 4 memiliki nilai EMTE yang paling tinggi yakni 2176.09 detik seperti yang terlihat pada tabel 4.6. Hal ini menunjukkan bahwa skenario 4 (hijau) memiliki waktu paling lama yakni 2176.09 detik atau 0.6044 jam ketika melakukan pengujian otomatis. Sedangkan skenario 2 memiliki waktu paling singkat yakni 1320.12 detik dalam melakukan proses pengujian otomatis. Tabel 4.6 merupakan data nilai EMTE dari masing proses sistem website pada 4 skenario yang diimplementasikan selama proses pengujian berlangsung sehingga bisa didapatkan nilai rata- rata.

Tabel 4.6. Hasil Pengujian Berdasarkan nilai EMTE

Proses sistem	Skenario 1 (detik)	Skenario 2 (detik)	Skenario 3 (detik)	Skenario 4 (detik)
Login	286.97	304.35	286.96	301.33
Searching Ticket	568.31	645.8	637.96	674
Create Account	259.6	292	300.14	319.19
Checkout Shopping Cart	836.9	77.97	821.83	881.57
Total	**1951.78**	1320.12	2046.89	2176.09

Pada proses testing juga diukur dari nilai TS (*time saved*) yang dihasilkan dari persamaan 2. Nilai TS merupakan nilai yang mendeskripsikan bagaimana pengujian otomatis mampu mengoptimalkan waktu pengujian perangkat lunak yang sertai dengan faktor berpengaruh yakni parameter pengujian manual dan *fragility*. Nilai kisaran *fragility* untuk 4 proses yakni *login* (60), pencarian tiket (120), *create account* (129) dan *checkout shopping chart* (180). Berdasarkan tabel 4.7 terlihat bahwa nilai rata - rata TS paling tinggi berada pada skenario 1 dengan nilai 1128.93 detik atau 0,31 jam. nilai TS juga bernilai positif yang artinya pengujian otomatis berhasil mengoptimalkan proses pengujian sistem jika

dibandingkan dengan pengujian manual. Jika dilihat dari tabel 4.7 terlihat bahwa nilai EMTE yang paling besar pada tiap proses sistem website menentukan skenario yang paling efektif dalam mengurangi waktu testing yang akan berbanding lurus dengan besarnya waktu yang berhasil dioptimalisasi yang dinyatakan pada nilai T.S.

Tabel 4.7. Nilai Time Saved

Proses	Skenario 1 (detik)	Skenario 2 (detik)	Skenario 3 (detik)	Skenario 4 (detik)
Login	139.78	122.4	**139.79**	125.42
Searching Ticket	**107.4**	29.91	37.75	1.71
Create Account	**364.33**	331.93	323.79	304.74
Checkout Shopping Cart	517.42	**580.35**	532.49	472.75
Total T.S	**1128.93**	1064.59	1033.82	904.62

4.4 Hasil Pengujian terhadap Spesifikasi Kode Otomatis

Hasil penelitian menunjukkan waktu yang dihasilkan dari proses pengujian dengan menggunakan kode otomatis. Implementasi kode otomatis secara langsung melakukan pengecekan terhadap spesifikasi yang telah didefinisikan sebelumnya.

Gambar 4.10. Screenshoot Mekanisme Kode Otomatis

Gambar 4.10 merupakan salah satu contoh hasil dari pengujian otomatis yang berhasil dijalankan dari percobaan pada penelitian ini. Seluruh percobaan menggunakan mekanisme yang sama. Sebelumnya pada proses spesifikasi kode otomatis akan ditentukan locator yang digunakan serta jumlah halaman website. Kode otomatis menggantikan pengecekan locator dan jumlah halaman website pada proses pengujian. terlihat pada gambar 4.10 proses validasi berlangsung selama kode otomatis berjalan ditandai dengan status keberhasilan validasi (*passed/failed*). Status keberhasilan menginterpretasikan proses validasi terhadap dokumen spesifikasi. Selain itu waktu yang dibutuhkan pengujian otomatis juga dihasilkan sampai level terkecil yakni locator. Detail yang dihasilkan dari pengujian otomatis secara teknis mampu menggantikan pengujian manual.

4.5 Strategi Pengujian

Strategi Pengujian yang dilakukan untuk menguji pola desain terdiri dari 4 pengujian yakni perubahan locator, perubahan input data, reposisi test case dan eliminasi test case.

1. Perubahan Locator : Merupakan proses memperbaharui locator atao objek dari komponen website. Proses ini biasanya dilakukan ketika ada perubahan proses sistem atau detail dari locator harus diubah sesuai dengan parameter yang akan dikirim.
2. Perubahan input data : Proses mengubah masukkan sistem untuk menguji sistem dengan kondisi data positif atau negatif.
3. Reposisi test case : Proses mengubah urutan langkah dari tahap pengujian

4. Eliminasi test case : Proses menghapus beberapa langkah pengujian yang disebabkan adanya perubahan alur proses sistem atau dihilangkan sementara sebuah langkah karena alasan strategi marketing.

Pengujian dilakukan untuk mengukur berapa banyak langkah yang dilakukan dalam proses perubahan pengujian. Tabel 4.8 merupakan detail dari langkah yang dilakukan pada tiap *framework* terhadap 4 pengujian. Tanda *checklist* menggambarkan langkah yang dilakukan oleh *framework*. Langkah tersebut harus dilakukan sesuai dengan karakteristik kerangka kerja yang telah disusun sedemikian rupa. Proses pengujian ini nilainya masih terkait dengan hasil waktu yang diperoleh dari pengujian otomatis. Nilai dari strategi pengujian bertujuan untuk memperkuat temuan berdasarkan hasil eksperimen uji waktu pengujian otomatis dari sisi efektifitas langkah dalam berinteraksi dengan proses perubahan dalam sistem.

Tabel 4.8. Implementasi Langkah Strategi Pengujian

Activity	Skenario 1				Skenario 2				Skenario 3				Skenario 4			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Find start test case	☐	☐	☑	☑	☐	☐	☑	☑	☐	☐	☑	☑	☐	☐	☐	☐
Find end test case	☐	☐	☑	☑	☐	☐	☑	☑	☐	☐	☑	☑	☐	☐	☐	☐
Searching test case	☑	☑	☑	☑	☐	☑	☑	☑	☐	☐	☑	☑	☐	☐	☑	☑
search locator	☑	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐
update locator	☑	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐
search input data	☐	☑	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐
update input data	☐	☑	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐
Update code	☑	☑	☑	☐	☐	☑	☑	☐	☐	☐	☑	☑	☐	☐	☐	☐
Change step posiiton	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐	☑	☑	☐	☐	☐	☐
Change test case position	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐
Delete test case	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑
Delete Step	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐	☐
Total Step	4	4	4	2	2	4	4	2	2	2	4	4	2	2	2	2

Nilai total langkah perubahan dari setiap framework bisa ditentukan seperti data pada tabel 4.8. Pada tabel 4.8 terlihat bahwa skenario 4 memiliki total langkah

yang paling efektif ketika diuji dengan 4 pengujian yang telah ditentukan. Penyebab dari efektifitas langkah pada skenario 4 antara lain manajemen objek repository, penyimpanan *realtime*, dan pemisahan *test step* ke dalam beberapa test case sehingga lebih mudah di-*maintenance*. skenario 1 memiliki total langkah yang paling panjang karena input data dan locator didefinisikan secara *hard code* pada *test step*. Selain itu seluruh *test step* disatukan ke dalam 1 *test case* saja.

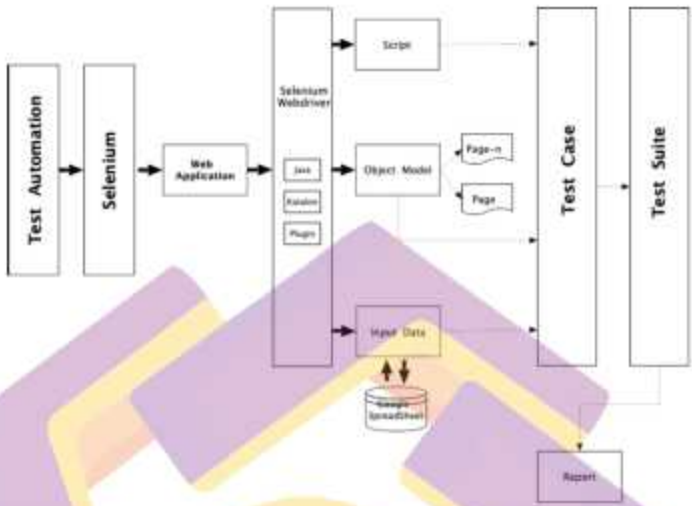
Tabel 4.9. Hasil Akhir Total Langkah Pengujian Tiap skenario

	Skenario 1	Skenario 2	Skenario 3	Skenario 4
Total Step	14	12	12	8

4.6 Perbandingan *Existing Framework* dan *Proposed Framework*

Dari hasil pembentukan awal kerangka kerja, terbentuklah sturktur kerangka kerja seperti yang terlihat pada gambar 4.11 . Berikut ini adalah penjelasan dari stuktur automation script seperti yang ditunjukkan pada gambar 4.11 yang terdiri atas ;

1. *Page* : merupakan tempat yang terdiri dari input field dan action
2. *Test Case* : cara sistem bekerja yang terdiri dari langkah-langkah tertentu
3. *Object Model* : Tempat object yang dimiliki dari sebuah input data atau action yang berfungsi sebagai pengenal bagi selenium driver dalam mengontrol sebuah sistem
4. *Test Suite* : Gabungan dari beberapa test case sehingga report bisa terbaca secara utuh sehingga menghasilkan report yang valid
5. *Input Data* : Tempat penyimpanan awa dimana data input akan dimasukkan ke dalam pengujian otomatis.



Gambar 4.11 Hybrid Framework Usulan

Jika dibandingkan dengan kerangka kerja penelitian sebelumnya maka terdapat perbedaan dari beberapa sisi antara lain ;

1. Pada penelitian sebelumnya tidak menjelaskan detail bagaimana proses pembentukan kode pengujian otomatis. Sedangkan pada penelitian ini dijelaskan bagaimana pembentukan sebuah kode mulai dari pengaturan input data, penyusunan test case hingga pembuatan test suite.
2. Dari sisi jenis penyimpanan data, kedua kerangka kerja sebelumnya memakai penyimpanan data *local storage*. Jenis penyimpanan data ini rentan terjadi perbedaan versi. Namun pada penelitian ini penyimpanan data diperkuat dengan memindahkan ke jenis

penyimpanan data awan (*cloud storage*) menggunakan *google spreadsheet*.

3. Pengujian sistem dilakukan pada *single application* sehingga tidak terlihat perbedaan waktu pengujian aplikasi ketika dilakukan pengujian secara otomatis. Pada penelitian ini digunakan 4 sistem yang berbeda dalam menilai tingkat efektifitas pengujian otomatis sehingga bisa didapatkan analisa yang mendalam pada sisi apa saja pengujian otomatis bisa dipercepat dan efektif.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan hasil penelitian, maka dapat disimpulkan bahwa penggunaan skenario 1 dalam melakukan pengujian secara otomatis menghasilkan nilai EMTE sebesar 1951.78 detik atau 0.5421 dibandingkan skenario yang lainnya pada 4 proses sistem yang diuji. Nilai tersebut menggambarkan waktu yang dibutuhkan dalam pengujian otomatis berjalan tanpa melibatkan parameter pengujian manual dan nilai kisaran *fragility*. Ketika parameter pengujian manual dan nilai kisaran *fragility* ikut serta dalam menentukan waktu yang berhasil dioptimalisasi, maka skenario yang paling efektif juga adalah skenario 1. Hal ini berbanding lurus dengan posisi skenario yang memiliki nilai EMTE terbesar pada masing masing proses sistem aplikasi website. Pendefinisian objek dan masukkan data sistem secara *hard code* mampu mencapai proses pengujian otomatis tercepat. Namun terdapat kekurangan ketika strategi pengujian perubahan sistem diterapkan. Skenario 1 memiliki langkah yang paling banyak yakni 12 sedangkan skenario 4 memiliki total langkah yang paling sedikit ketika terjadi perubahan yakni 8 langkah. Hal ini menunjukkan bahwa posisi locator dan masukkan data sistem secara *hard code* mempersulit proses pembaharuan ketika sistem mengalami perubahan atau modifikasi. Skenario 4 memiliki total langkah yang sedikit karena implementasi sentralisasi locator dan pemindahan masukkan sistem ke dalam penyimpanan awan (*cloud storage*) sehingga proses perubahan hanya dilakukan pada satu tempat

secara *realtime*. Proses tersebut mengurangi interaksi dengan kode dalam langkah pengujian otomatis yang berakibat efisiensi pada tahapan *updating* test case. selain itu skenario 4 menghasilkan beberapa kelebihan dibandingkan dengan skenario yang lainnya ketika proses 4 tipe pengujian dilakukan yakni ;

1. Efisiensi dalam merespon perubahan sistem baik secara perubahan alur sistem atau pengurangan sistem yang berpengaruh pada modul yang terlibat.
2. Efisiensi proses manajemen locator ketika jumlah locator semakin banyak dan digunakan pada test case yang berbeda.
3. Sistem data yang bersifat *realtime* sehingga mengurangi potensi adanya perbedaan versi ketika penyimpanan data disimpan secara *offline*.
4. Skenario 4 tepat untuk diterapkan pada lingkungan sistem yang mengalami perubahan yang cukup cepat sedangkan skenario 1 lebih tepat untuk digunakan pada proses sistem yang memiliki proses pembaharuan yang minim.

5.2. Saran

Penelitian ini memiliki keterbatasan dalam hal *platform* dan studi kasus yang kurang luas. Oleh karena itu penulis berharap pada penelitian selanjutnya bisa menjelaskan bagaimana implementasi desain kerangka kerja pada sistem aplikasi sejenis berbasis *mobile app* android atau iOS pada sistem yang lebih besar. Hal tersebut bertujuan untuk membandingkan kelebihan dan kekurangan pada kedua tipe sistem operasi perangkat lunak berbasis mobile app sehingga bisa didapatkan hasil analisis yang valid untuk perbaikan kerangka kerja pada proses *end-to-end* testing yang lebih baik pada *platform* yang berbeda. Kemudian bisa diusulkan untuk

mengukur bagaimana kecepatan pengujian ditinjau dari jenis *css selector* pada selenium *webdriver* mempengaruhi kecepatan pengujian otomatis. Faktor sumber daya manusia diharapkan diteliti dari sisi usaha dan waktu yang diperlukan ketika membuat kode pengujian otomatis pada saat pertama kali. Selain itu Faktor eksternal yang menyebabkan hasil pengujian dan pola pengujian tidak stabil juga perlu diteliti lebih lanjut dari sisi teknis maupun non teknis supaya menghasilkan penelitian yang lebih komprehensif.

DAFTAR PUSTAKA

Boezdemir, M., T. Bilgin dan N. Oylum, 2023, A New Hybrid Software Testing Automation Framework, The European Journal of Research and Development, ISSN: 2822-2296, Vol.3 Issue.1 March, 2023

Mahalakshmi A., et al, 2020, Web Application Automation Using Selenium, International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Vol. 9 Issue.7 May, 2020

Anil M, 2021, Automating and Optimizing Software Testing using Artificial Intelligence Techniques, International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 12 Issue. 5, 2021

Joe Lian Min, A. Istiqomah, dan A. Rahmani, 2020, Evaluasi Penggunaan Manual dan Automated Software Testing pada Pelaksanaan End-to-End Testing, Jurnal Teknologi Terapan (JTT), ISSN: 2549-1938, Vol. 6 Issue. 1 March, 2020

K. R. Halani, Kavita and R. Saxena, 2021, Critical Analysis of Manual Versus Automation Testing, International Conference on Computational Performance Evaluation (ComPE), Shillong, India, 2021, pp. 132-135

N. Pombo and C. Martins, 2021, Test Driven Development in Action: Case Study of a Cross-Platform Web Application, IEEE EUROCON 2021 - 19th International Conference on Smart Technologies, Lviv, Ukraine, 2021, pp. 352-356

Aniwange A., Nyishar dan Afolabi, 2021, A Hybrid Software Test Automation For Educational Portals, International Journal of Innovations in Engineering research and Technology, ISSN: 2394-3696,Vol. 8 Issue. 8 Augustus, 2021

P. P. Kore, M. J. Lohar, M. T. Surve and S. Jadhav, 2022, API Testing Using Postman Tool, International Journal for Research in Applied Science &

Engineering technology (IJRASET), ISSN: 2321-9653, Vol. 10 Issue. 12 December, 2022

Zulianto A, A. Purbasari, N. Suryani, A. I. Susanti, F. R. Rinawan, dan W. G. Purnama, 2021, Pemanfaatan Katalon Studio untuk Otomatisasi Pengujian Black-Box pada Aplikasi iPosyandu, Jurnal Edukasi dan Penelitian Informatika (JEPIN), ISSN: 2548-9364, Vol. 7 Issue. 3 Desember, 2021

Banjarnahor M.T, and L. S. Istiyowati, 2022, Smoke Automation and Regression Testing on a peer-to-peer lending Website with the Data-Driven Testing Method, Jurnal Rekayasa Sistem dan Teknologi Informasi, ISSN: 2580-0760, Vol. 6 Issue. 4, 2022

Riasat H., Z. Nazir, dan S. Zubair, 2021, Critical Analysisy of Software Testing techniques and Automation Testing Tools, International Journal of Scientific & Engineering Research, ISSN: 2229-5518, Vol. 12 Issue. 2 February, 2021

Anand R dan M. Arulprakash, 2018, Business Driven Automation Testing Framework, International Journal of Engineering and Technology (IJET), Vol. 7 Issue. 2.8 March, 2018, 345-349

Hanna M., A. Aboutabl dan M. Mostafa, 2018, Automated Software Testing Frameworks : A Review, International Journal of Computer Applications, ISSN: 0975-8887, Vol. 179 Issue. 46 June, 2018

Gojare S., R. Joshi, dan D. Gaigaware, 2015, Analysis and Design of Selenium WebDriver Automation Testing Framework, Elsevier Science Direct Procedia Computer Science 50, 2015, 341-346.

Mahajan P., H. Shedge, dan U. Patkar, 2016, Automation Testing in Software Oganization, International Journal of Computer Applications Technology and Research, ISSN: 2319-8656, Vol. 5 Issue. 4, 2016, 198-201

C. Rishab Jain, dan R. Kaluri, 2015, Design of Automation Scripts Execution Application for Selenium WebDriver and TestNG Framework, ARPN Journal of Engineering and Applied Science, Vol. 10 Issue. 6 January, 2015

Sharma M., dan R. Angmo, 2014, Web Based Automation Testing and Tools. International Journal of Computer Science and Information Technologies, Vol. 5 Issue. 1, 2014, 908-912

M. Adarsh, dan A. Mehta, 2022, Automation Testing - A Review, International Research Journal of Modernization in Engineering Technology and Science, ISSN: 2582-5208. Vol.4 Issue. 6 June, 2022

Ateşoğulları D., dan A. Misra, 2020, Automation Testing Tools: A Comparative View, International Journal of Computer Science and Information Technologies, Vol. 4 Issue. 12, 2020

R. Rudolf, dan C. Klammer, 2019, Enhancing Acceptance Test-Driver Development with Model-Based Test Generation, IEEE 19th International Conference on Software Quality, Realibility and Security Companion.

Ms. S. Karuturi, dan Mr. M. Gowda M, 2017, Research on Software Testing Techniques and Software Automation Testing Tools, International Conference on Energy, Communication, Data Analytics and Soft Computing

R. M. Sharma, 2014, Quantitative Analysis of Automation and Manual Testing, International Journal of Engineering and Innovative technology, ISSN: 2277-3754, Vol. 4 Issue. 1 June, 2014

Zhenyu L, Qiang C. dan Xu X, 2013, A Maintainability Spreadsheet-Driven RegressionTest Automation Framework, IEEE 16th International Conference on Computational Science and Engineering.

Maurya V.N., dan Rajender K, 2012, Analytical Study on Manual vs Automated Testing using with Simplistic Cost Model, International Journal of Electronics and Electrical Engineering, ISSN: 2277-7040, Vol. 2 Issue. 1 January, 2012

Ouriques R., Krzysztof W., dan Tony G, 2022, The Role of Knowledge-based resources in Agile Software Development Context, Elsevier Science Direct Transportation Research Proccedia 197, 2022, 0164-1212

Šimičková J., Katarina B., dan Erika M, 2021. Spesifics of the Agile Approach and Methods in Project Management and its Use in Transport, Elsevier Science Direct The Journal of Systems & Software 55, 2021, 1436-1443

Bhondokar B., Pooja R., dan Snehal J, 2015, Hybrid Test Automation Framework for Web Application, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 4 Issue. 4 April, 2015

Burns, D, 2012, Selenium Testing Tools Beginner's Guide, Second Edition, Packt Publishing Ltd, Birmingham

Roger S. Pressman. (2010). *Software Engineering : A Practitioner's Approach (7th ed)*. New York: McGraw-Hill.

J. Sutherland, 2014, SCRUM: The Art of Doing Twice the Work in Half the Time, Crown Business, New York

Shivani and R. K. Challa, 2020, CAPTCHA: A Systematic Review, 2020 IEEE International Conference on Advent Trends in Multidisciplinary Research and Innovation (ICATMRI), Buldhana, India, 2020, pp. 1-8,

Rosa, A. S. dan M Shalahuddin, 2013, Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek, Informatika, Bandung