BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Terdapat beberapa penelitian terkait tentang proses implementasi *automation test framework*

2.1.1. *A Comprehensive Review on Selenium Automation Testing* Tool[4], sebuah penelitian yang berfokus pada Selenium. Penelitian ini menjelaskan tentang beberapa faktor dari *testing* manual dan *testing* otomatis dari segi waktu, efisiensi, harga dan pelatihan. Penelitian ini membahas juga tentang siklus hidup pengujian otomatis dengan tahapan finalisasi *test*, membuat rencana *test*, pemilihan tools, pengembangan dan penulisan script, dan melakukan *test* dengan script yang telah dibuat dengan tujuan untuk memberikan penjelasan mengenai pentingnya dan definisi dari *test* otomatis.

2.1.2. *A Study on Functioning of Selenium Automation Testing Structure*, penelitian yang dilakukan oleh Jyoti Devi, Kirti Bhatia dan Rohini Sharma[5]. Penelitian ini menjelaskan tentang arsitektur pada Selenium, yang terdapat client, server, API, dan lain lain. Penelitian ini juga menjelaskan tentang kemudahan dari *software testing* secara otomatis. Fokus penelitian ini adalah pada bagian dari Selenium itu sendiri, salah satunya adalah mengenai Selenium Integrated Development Environment (IDE) dan memberikan pedoman mengenai proses dan langkah – langkah dari pengujian web dalam browser Firefox

2.1.3. *Analysis and Design of Selenium WebDriver Automation Testing Framework.* Penelitian ini dilakukan oleh Satish Gojare, Rahul Joshi, dan Dhanashree Gaigaware. Penelitian ini menjelaskan tentang tujuan dari *software testing* untuk mencari adanya kekurangan. Penelitian ini menyimpulkan tentang *software testing* yang mampu mengurangi waktu yang dibutuhkan untuk menulis *test* dan meningkatkan keberhasilan dalam *testing* dengan menggunakan framework *Test*NG.

2.1.4. *Intelligent Testing Tool :Selenium WebDriver*. Penelitian ini dilakukan oleh Renu Patil dan Rohini Temkar. Penelitian ini menjelaskan tentang definisi *test* otomatis meliputi kelebihannya seperti *regression testing*, juga menjelaskan tentang locators pada web driver dalam bentuk *id, name, className, tagName,cssSelector ,xpath* dan lainnya. Penelitian ini juga menjelaskan metode dan deskripsi yang dilakukan pada webdriver seperti clickAndHold(), doubleClick(), contextClick() dan lain lain.

2.1.5. *Selenium Test Automation Framework in On-line Based Application.* Penelitian yang dilakukan oleh Revathi.K dan Prof.V.Janani menjelaskan tentang pentingnya software testing pada program dengan mencari kesalahan dan meningkatkan kualitas. Penelitian ini juga menjelaskan tentang kelebihan dari penggunaan software testing seperti kode yang memiliki objek yang sama dan bisa dipakai pada aplikasi yang berbeda, kemudian juga pada script yang seragam. Pada testing otomatis, test dilakukan dengan langkah yang sama setiap kalinya dibanding dengan manual testing yang banyak terjadi kesalahan. Serta penyimpanan kasus test yang dapat kita cek jika ada kesalahan dan memperbaikinya.

2.1.6. *Test Automation Framework using Webdriverio based on Page Object Model.* Penelitian ini dilakukan oleh Anil Kulkarni dan Syeda Sana Nausheen. Penelitian ini menjelaskan tentang automation framework dan software testing WebdriverIO pada software quality assurance. Penelitian ini menggambarkan tentang kelebihan dari WebdriverIO yang mudah untuk digunakan, dapat diulang, kemudahan dalam menyusun test,dan kemudahan dalam eksekusi. Penelitian ini juga menjelaskan tentang *Page Object Model* yang merupakan desain planning yang biasanya dimanfaatkan pada Selenium.

2.1.7. *Systematic Study of A Web Testing Tool : Selenium.* Penelitian ini dilakukan oleh Chandraprabha, Ajeet Kumar, dan Sajal Saxena. Penelitian ini menjelaskan tentang software testing secara otomatis berarti menggunakan software khusus untuk mengontrol eksekusi pada test dan perbandingannya dengan output. Selenium merupakan testing tool open source untuk web yang fleksibel terhadap browser yang mendukung

beberapa macam bahasa pemrograman. Penelitian ini mengungkapkan beberapa kelebihan dari software test otomatis yakni, code yang dapat digunakan di berbagai aplikasi, meningkatkan akurasi, tidak perlu keahlian khusus, jika ada kesalahan atau *error*, maka kita mampu mengeceknya

2.1.8. *Automated Software Testing Framework :A Review*. Merupakan penelitian yang dilakukan oleh Milad Hanna, Amal Elsayed Abutabl dan Mostafa-Sami M. Mostafa. Penelitian ini menjelaskan tentang software manual yang sudah digunakan secara tradisional di industri software. Hal ini bergantung pada penguji dari manusia tanpa adanya bantuan dari tools untuk mendeteksi kesalahan pada aplikasi. Sehingga fokus penelitian ini ada pada kekurangan manual testing yang memakan waktu, tidak bisa digunakan ulang, dan mendeteksi kesalahan sangat sulit. Sehingga solusinya adalah dengan menggunakan testing secara otomatis yang menjalankan langkah yang diimplementasikan menggunakan bahasa pemrograman tingkat tinggi [6] dan membentuk test script yang mampu mendeskripsikan input, output serta hasil yang diinginkan.

2.1.9. *A Comparative Study of White Box, Black Box and Grey Box Testing Techniques.*[7] Penelitian yang dilakukan oleh Mohd. Ehmer Khan dan Farmeena Khan. Penelitian tersebut menjelaskan tentang proses dari software testing yang bertujuan untuk mencari kesalahan , atau *error* pada aplikasi yang harus diperbaiki dengan menjelaskan perbandingan teknik dari pengujian yang meliputi White Box, Black Box, dan Grey Box. Juga menjelaskan dua teknik dari pengujian Black Box yaitu *Equivalence Partitioning* dan *Boundary Value Analysis.*

2.1.10 Pengujian *Black Box* pada Sistem Aplikasi Informasi Data Kinerja Menggunakan Teknik *Equivalence Partitions*.[8] Penelitian ini dilakukan oleh Andri Ricat Sinulingga, Muhammad Zuhri, Rizky Budi Mukti, Ziasyifa dan Aries Sarifudin. Penelitian ini menjelaskan tentang pengujian sistem aplikasi yang dibuat dengan menggunakan teknik Equivalence Partitions untuk menguji aplikasi data kinerja dari perusahaan yang terdiri dari beberapa fitur seperti pembuatan *jobdesk* karyawan, penambahan target kerja, edit target kerja dan nilai.

Dari penelitian terkait yang dijelaskan diatas, penulis menyimpulkan bahwa penelitian terkait menguji adanya aplikasi, definisi pengujian otomatis dan manual, serta adanya metode pengujian pada teknik *Black Box* yang akan dipakai pada penelitian penulis. terdapat beberapa perbedaan dari penelitian terkait meliputi *tools* yang digunakan, metode dan teknik pengujian, serta cara menjalankan pengujian.

a. Penelitian ini menguji secara otomatis dengan teknik *Black Box* dari penelitian sebelumnya yang masih menggunakan pengujian manual
b. Penelitian terkait tidak berfokus pada aplikasi atau objek yang berjalan dengan prinsip CI/CD
c. Tools yang digunakan oleh penulis adalah WebdriverIO yang berbeda dengan *tools* dari penelitian terkait yaitu Selenium
d. *Command/Script* yang digunakan memiliki perbedaan dari penelitian terkait, serta bahasa pemrograman yang berbeda.
e. Untuk penelitian terkait yang menguji secara otomatis menggunakan *script* Selenium, command atau perintah yang digunakan memiliki tingkat kompleksitas yang lebih sulit dipahami
f. Fokus dari penelitian terkait bervariasi, dimulai dari uji regresi, framework yang berbeda, perbedaan browser yang digunakan, serta fitur yang diuji.
g. Penelitian ini akan menghasilkan output report melalui reporter dan juga melalui *logfile* yang dapat dijadikan dokumentasi pengujian aplikasi

2.2 CI/CD

CICD merupakan gabungan dari CI (*continuous integration)* dan CD(*continuous delivery)* pada CI, pengembang menggabungkan perubahan pada pusat *source code* pada repositori. Aplikasi akan di *re-build* dan pengujian otomatis akan dijalankan terhadap aplikasi yang dibangun. Jika *test case* gagal, maka pengembang dapat menganalisis kesalahan secepatnya. Dengan CI, aplikasi akan bekerja secara terus menerus dan berkelanjutan[3]. CD mengambil output dari integrasi yang telah memenuhi kriteria dan memperbarui aplikasi dengan output dari integrasi tersebut. CD bergantung pada kerja otomatis untuk mendeteksi keberhasilan atau kegagalan dari integrasi[9].

Tujuan dari CICD adalah menyelesaikan software pada prinsip yang kualitas tinggi, cepat, *reliable,* dan efisien

2.3 Software Testing

Merupakan prosedur dalam mengeksekusi program atau aplikasi dengan tujuan untuk mencari adanya kesalahan atau error. Proses ini memvalidasi dan memverifikasi seluruh komponen dari sistem dan memeriksa hasil sesuai dengan yang diharapkan [10]

Beberapa tingkatan pada pengujian sebuah aplikasi atau software [11]

1. Unit testing, merupakan pengujian pada level terendah. Pengujian ini menguji dasar unit dari software, atau disebut juga sebagai module testing. Pengujian ini memverifikasi fungsionalitas dari kode fungsi, *object-oriented,* atau level *class*.
2. Integration Testing, merupakan pengujian pada unit yang lebih besar pada segi strukuturalnya. Pengujian ini dilakukan antara komponen stuktur yang dibuat. Sehingga pengujian integrasi berfokus pada tipe dari aplikasi yang memverifikasi antarmuka antara komponen dengan rancangan aplikasi. Pengujian ini berguna untuk menemukan *error* pada antarmuka dan interaksi antara komponen didalamnya.
3. System Testing, merupakan pengujian *end-to-end* dari sistem. Pengujian sistem yang akan dilakukan berdasarkan fungsi dari spesifikasi sistem, dapat juga berupa Non-functional yang terdiri dari keamanan, *reliability, maintainablity*
4. Acceptance Testing, merupakan persiapan dari produk atau aplikasi. Pada bagian ini, pengujian fokus pada operasional atas persiapan sistem yang sudah selesai dari pengembang kepada pengguna. tujuan pengujian ini adalah untuk meyakinkan bahwa aplikasi sudah dapat berjalan
5. Alpha Testing, merupakan tahap dimana pengujian dilakukan oleh pengguna atau pelanggan
6. Beta Testing, merupakan pengujian yang dilakukan oleh pengguna khusus dimana mereka dapat menguji dan memeriksa bahwa produk dapat berjalan dan memiliki sedikit kesalahan
7. Regression Testing, merupakan fokus pengujian pada pencarian *error* ketika adanya perubahan kode. Biasanya pengujian ini berfokus pada *bug* lama yang

muncul kembali. Metode umum dari pengujian ini adalah menjalankan kembali pengujian yang sudah dilakukan sebelumnya dikarenakan adanya penambahan fitur tertentu, atau perubahan. Dijelaskan pada tabel 2.1 dibawah ini,

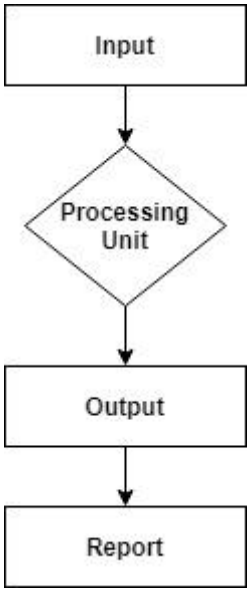
Tabel 2.1 Tingkatan Pengujian

Tipe Pengujian	Spesifikasi	Cakupan	Teknik	Penguji
Unit	Low level, code	Class	White Box	Programmer
Integration	Low level, high level	Multiple Class	White box, Black box	Programmer
Functional	High level	Produk	Black box	Tester
System	Requirement analysis	Produk	Black box	Tester
Acceptance	Requirement analysis	Produk	Black box	Customer
Beta	Ad hoc	Produk	Black box	Customer
Regression	High level	Class, produk, komponen	Black box, White box	Programmer, tester

2.4 Black Box Testing

Merupakan teknik dari pengujian tanpa adanya pengetahuan tentang pengerjaan internal dari sebuah aplikasi. Metode ini hanya menganalisis aspek fundamental dari sistem dan hanya memiliki relevansi yang sedikit atau bahkan tidak ada dengan logikal struktur dari sistem secara internal.

Prinsip pengujian ini dilakukan pada sudut pandang pengguna, bukan sudut pandang pengembang atau penguji. Kasus uji yang dibuat akan lebih mudah, serta penguji tidak memerlukan suatu keahlian khusus pada bahasa pemrograman seperti pengujian *white box.* Pengujian akan lebih cepat dikarenakan penguji hanya mementingkan pada bagian *graphical user interface*



Gambar 2.1 Diagram Black Box Testing

1.Input: merupakan persiapan dari spesifikasi fungsional dari sistem yang akan diuji. Penguji akan menggunakan input yang valid

2.Processing Unit: merupakan bagian dimana penguji tidak mementingkan bagian internal sistem. Proses ini penguji membuat kasus uji dengan pemilihan input dan menjalankannya. [12]

Terdapat beberapa teknik pengujian pada *black box* , diantaranya:

a. Equivalence Partitioning
b. Boundary Value Analysis
c. Orthogonal Array Testing
d. Fuzz Testing
e. Graph Based Testing
f. All-Pairs Testing
g. State Transition Testing

Pada penelitian ini, penulis menggunakan 2 metode pengujian *Black Box*, yakni Equivalence partitioning dan Boundary Value Analysis. Teknik Equivalence Partitioning merupakan pengujian *Black Box* yang membagi data input menjadi beberapa partisi yang mengurangi kasus uji yang banyak. Teknik ini dibagi menjadi

beberapa kelas yang merepresentasikan bagian keberhasilan dan kegagalan dari kondisi input, seperti angka, array, atau boolean. [12]

Boundary Value Analysis merupakan pengujian pada batas maksimum, minimum, dalam/luar batas dan nilai *error*.[12] Pengujian ini memiliki 6 batasan yang akan diuji

a. Min -1
b. Min
c. Min+1
d. Max -1
e. Max
f. Max+1

2.5 Functional Testing

Merupakan proses *quality assurance* dan tipe dari *black box testing* yang disebut juga sebagai *system testing* yang menentukan kasus uji berdasar pada spesifikasi aplikasi atau software. Fungsi akan diuji dengan menguji input dan memeriksa output dan isi dari internal tidak begitu diperhatikan. Pemilihan uji kasus pada pengujian fungsional berdasar pada rancangan spesifikasi software. [13]tujuan dari pengujian fungsional berfokus pada pengujian setiap fungsi dari aplikasi seperti *user interface*, API, *database*, *security*, *server*. Salah satu contoh pengujian fungsional adalah pengujian fungsi tombol *edit, submit, login, delete*. Pengujian fungsional dapat dilakukan secara manual atau otomatis. Pengujian fungsional dibagi menjadi 6 langkah:

1. Identifikasi fungsi dari perangkat lunak yang diharapkan
2. Pembuatan data input sesuai dengan spesifikasi fungsi
3. Penentuan output berdasarkan spesifikasi fungsi
4. Eksekusi kasus uji
5. Perbandingan dari hasil dan ekspektasi
6. Memeriksa proses aplikasi agar sesuai dengan keinginan pengguna

2.6 End-to-end Testing

Merupakan sebuah metodologi pengujian pada sebuah aplikasi yang dijalankan dari awal hingga selesai. pengujian ini sama seperti *system testing* yang menguji validasi atau keberhasilan dari fungsi fitur aplikasi. *End-to-end testing* tidak hanya menguji validasi tersebut, namun juga menguji keterhubungan alur fitur dari awal hingga akhir. tujuan dari teknik ini adalah menyesuaikan pengujian sesuai dengan kebutuhan dan sudut pandang *user*. End-to-end testing dibagi menjadi dua metode, yaitu *horizontal end-to-end testing* dan *verticl end-to-end testing.*

Horizontal end-to-end testing merupakan metode pengujian proses yang mencakup keseluruhan fitur pada aplikasi. Sehingga pengujian harus dilakukan dari awal sampai akhir. Pengujian ini dilakukan pada level UI.

Vertical end-to-end testing merupakan metode yang menguji dari API dan SQL sehingga proses pengujian ini pada umumnya lebih kompleks dibandingkan dengan *horizontal end-to-end testing.* Proses pengujian ini dilakukan dari level terbawah sampai teratas, metode ini membutuhkan seluruh *stakeholder* yang meliputi pengembang, penguji, manajer projek. Kedua teknik *end-to-end testing* ini dapat dilakukan secara manual maupun otomatis.[14]

2.7 Manual Testing

Merupakan proses testing secara maual pada software dengan tujuan mencari error pada software tanpa adanya testing tools. Testing ini tidak efektif jika dilakukan dalam projek skala besar karena membutuhkan waktu dan sumber daya yang besar [10]

2.8 Automated Testing

Automated testing merupakan proses dengan menggunakan tool dalam eksekusi menggunakan script yang sudah disiapkan dan bertujuan mencari error. Pengujian secaras otomatis meningkatkan akurasi, menyimpan banyak waktu dan sumber daya, cocok digunakan untuk projek skala besar dan dilakukan berulang ulang dimana banyak nya *regression testing* diperlukan serta meningkatkan efektivitas dan efisiensi dari software testing.[10] Pengujian manual mengacu pada konsep melakukan kasus uji menggunakan manusia Interaksi. Sebaliknya, dengan

memanfaatkan mesin atau perangkat lunak, pengembang dapat menjalankan sejumlah pengujian kasus dan bandingkan hasil dengan hasil yang diharapkan[15]. Dengan kata lain, otomatisasi pengujian mencakup kumpulan file skrip uji untuk dijalankan tanpa halangan manusia [16]. Ada 3 level pengujian otomatis yang dijelaskan dibawah ini,

a. Level 1 - Tes unit: Tes unit dianggap sebagai tes modul karena fakta bahwa itu menguji komponen tunggal dalam aplikasi [3]. xUnit (JUnit atau NUnit) mendukung penguji dengan memastikan tidak ada kesalahan di salah satu bagian kode sumber, yang artinya dengan masukan tertentu, fungsi tersebut harus mengembalikan hasil yang diharapkan. Di pengembangan perangkat lunak pengembang dapat membangun pengujian unit untuk fungsionalitas sebelum kode aktual ditulis [14].
b. Level 2 - Tes API: API (Application Programming Interface) - satu set dari fungsi yang mengembalikan kumpulan nilai setelah dipanggil atau berinteraksi dengan pengujian mengacu pada konsep pengujian API secara langsung untuk menentukan keandalan, kinerja, keamanan, dan lainnya penting dari semua - fungsionalitas [15]. Sekarang, dalam pengembangan perangkat lunak pengujian API membantu pengembang dan penguji untuk pengembangan yang cepat[16].
c. Level 3 - Tes GUI: pengujian GUI (Graphical User Interface) adalah pengujian otomatisasi tingkat terendah. Selama aplikasi memiliki file GUI, penguji dapat mengontrol mesin untuk meniru tugas berulang apa pun yang dapat dilakukan oleh pengguna akhir (pengujian manual). Tes GUI lebih menekankan pada sisi pengguna akhir, karena dapat mendeteksi tidak hanya gagal fungsionalitas tetapi juga tata letak palsu atau komponen hilang / tersembunyi. Masalah seperti itu tidak dapat dideteksi dengan uji Unit atau API. Namun, karena perubahan kecil di GUI dapat membuat kasus uji gagal, ini sangat penting untuk merekam dan memiliki cadangan saat mengembangkan kasus uji GUI[16]

2.9 WebdriverIO

WebdriverIO merupakan *testing tools open source utility* Node.js. sebuah testing tools yang sudah disederhanakan dibanding dengan Selenium. Test ini dilakukan menggunakan bahasa pemrograman Javascript. WebdriverIO merupakan *framework* automasi yang dibangun untuk mengotomasikan web dan aplikasi mobile. WebdriverIO menyederhanakan interasksi dengan aplikasi yang dibuat dengan menyediakan plugin yang membantu pengguna untuk membuat pengujian yang *salable, robust,* dan *flakiness.* WebdriverIO dirancang untuk:

a. *Extendable*. Fitur bantuan, kombinasi command yang simpel dan membantu
b. *Compatible*. Dapat dijalankan pada *cross-browser testing*
c. *Feature Rich.* Banyak variasi *built-in* dan *plugins* yang membantu pengguna untuk mengintegrasikan dan menyebarkan perancangan sesuai dengan kebutuhan[17]

WebdriverIO merupakan bagian luar Selenium Webdriver yang memberikan manfaat sangat baik dan pengaturan struktur dengan mudah untuk menjalankan instrumen dengan minim kesalahan yang mampu memberikan pengguna kemudahan dalam testing. WebdriverIO menggunakan pengujian lewat unit test seperti Cucumber, Mocha dan juga perangkat pelaporan seperti spec yang memberikan informasi terhadap test yang dilakukan. Minimal versi Node.js yang dibutuhkan untuk menjalankan WebdriverIO adalah versi 12.16.1

2.10 Node.js

Node.js merupakan salah satu platform pengembang yang dapat digunakan untuk membuat sebuah aplikasi[18][19]. Node.js dikembangkan dari engine javascript yang dibuat oleh Google untuk browser Chrome serta pustaka lainnya. Javascript digunakan sebagai bahasa pemrograman event-driven, non-blovking I/O model yang membuatnya efisien. Fitur built-in HTTP server yang memberikan kemudahan dalam sebuah webserver tanpa adanya bantuan software lainnya seperti Apache.

Node.js merupakan sebuah runtime environment dan script library. Sebuah software yang berfungsi untuk mengeksekusi dan mengimplementasi fungsi-fungsi serta cara kerja dari bahasa pemrograman.

2.11 NPM

NPM merupakan manajer paket untuk Node.js untuk membantu pengembang Javascript dalam kode paket modul. NPM kependekan dari node package manager yang menjadikannya sebuah online repository yang digunakan untuk mempublikasikan project Node.js. NPM juga merupakan command-line utility yang digunakan untuk berinteraksi dalam instalasi paket –paket berbasis node.js [20]

2.12 Yarn

Yarn merupakan sebuah package manager yang menggantikan proses kerja dari NPM client yang tetap kompatibel dengan registri NPM dengan fitur yang lebih cepat dalam beroperasi, lebih aman dan dapat diandalkan

2.13 Mocha

Mocha merupakan fitur JavaScript yang dijalankan pada Node.js dan di browser, membuat tes asinkronus lebih nyaman digunakan. Mocha dijalankan secara serial, menghasilkan laporan yang fleksibel dan akura ***Spec reporter***

Merupakan plugin dari WebdriverIO untuk memberikan laporan dari pengujian sebuah aplikasi dalam bentuk gaya spec

2.14 Chrome Driver

Webdriver merupakan tool open source untuk automated testing terhadap seluruh browser, digunakan untuk input, navigasi halaman, eksekusi JavaScript, dan lainnya. Chromedriver merupakan server standalone yang mengimplementasikan W3C Webdriver standard.

2.15 Docker

Merupakan software yang digunakan untuk memonitor database saat aplikasi atau web sedang berjalan

2.16 React

React adalah library Javascript terpopuler untuk membuat user interface (UI). Tool ini menawarkan respons cepat untuk user input dengan menggunakan metode baru dalam proses rendering website.

Komponen dari tool ini dikembangkan oleh Facebook. Dahulunya tool ini diluncurkan sebagai tool open-source JavaScript di tahun 2013. Saat ini, React tetap terdepan mendahului kompetitornya seperti Angular dan Bootstrap, dua library JavaScript terlaris.