BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Empiris

Kajian empiris adalah kajian yang didapatkan dari penelitian yang telah dilakukan sebelumnya oleh peneliti lain. Beberapa penelitian terkait automation testing yang dipaparkan di bawah ini.

Tabel 2. 1 Penelitian Sebelumnya

Judul Penelitian	**Peneliti & tahun Penelitian**
Perancangan *Automated Testing* Pada Studi Kasus *Website* Indicar	Bima Prakoso1*, Ari Sujarwo2 (2022)
Tujuan	**Ruang lingkup**
memastikan website IndiCar sudah terhubung dengan perangkat IoT	Research and Development (R&D) A. Analisis mengumpulkan data primer dari pihak Telkom dan melakukan wawancara kepada tim developer B. Perancangan merangkum fitur fitur menjadi dua hak akses yaitu Administrator dan pelanggan IndiCar C. Konstruksi / Pemrograman melakukan pemrograman Node JS yang dipasangkan tools Cypress untuk menjalankannya D. Pengujian melakukan pengujian dari kode pemrograman yang dibuat, apakah sudah sesuai apa belum, apakah ada *error* atau tidak

Hasil Penelitian
Proses pengujian manual di *website IndiCar* menghasilkan waktu kurang lebih 10 menit. Pada proses pengujian dengan menggunakan *Cypress* pada *website admin IndiCar*, didapatkan waktu 82 detik dengan 42 pengujian telah berhasil melewati test uji. Pada proses pengujian dengan menggunakan *Cypress* pada *website* pelanggan *IndiCar*, didapatkan waktu 33 detik dengan 13 pengujian telah berhasil melewati test uji.
Perbedaan dengan penelitian yang akan dilakukan
Penelitian saya akan melakukan automation testing yang dilakukan menggunakan Katalon Studio sebagai *tools*.

Judul Penelitian	**Peneliti & tahun Penelitian**
Pemanfaatan Katalon Studio untuk Otomatisasi Pengujian *Black-Box* pada Aplikasi iPosyandu	Arief Zulianto#1, Ayi Purbasari*2, Neni Suryani*3, Ari Indra Susanti+4, Fedri R. Rinawan+5, Wanda G. Purnama*6 (2021)
Tujuan	**Ruang lingkup**
meminimalisir apa yang tidak bisa di *back up* oleh pengujian secara manual dan menghindari *human error*.	*A. Mapping data* Membuat skenario *test case* terlebih dahulu secara manual B. Pembuatan *Test Case* Berikut daftar fungsionalitas dan *test case* yang untuk pengujian C. Pembuatan *Test Suite* *Test suite* berisi *test case* dan *data files* yang telah dibuat sebelumnya D. Hasil Pengujian Secara hasil eksekusi, pengujian manual dan otomatis

	menghasilkan status yang sama: 10 pass dan 3 fail.
Hasil Penelitian	
Pada saat menjalankan pengujian dapat diketahui *response time* setiap *test case* yang. *Response time* pada satu test case berbeda-beda apabila dijalankan beberapa kali. Total waktu eksekusi pengujian otomatis total adalah 283,08 detik, dan 719,27 detik untuk pengujian secara manual. Dengan demikian terjadi peningkatan kecepatan 2,54 kali atau 254%.	
Perbedaan dengan penelitian yang akan dilakukan	
Penelitian saya akan melakukan analisis perbandingan *automation testing* dan *manual testing* pada *website* STIKI TECHNOFEST	

Judul Penelitian	**Peneliti & tahun Penelitian**
Automation Testing Tool Dalam Pengujian Aplikasi *The Point Of Sale* (Studi Kasus TPOS PT. JAVASIGNA INTERMEDIA)	Yoga Kosasih, Andhik Budi Cahyono (2020)
Tujuan	**Ruang lingkup**
Menganalisis efektivitas aplikasi yang sudah dibangun, yang kemudian apakah *Automation testing tool* Katalon lebih efektif jika dibandingkan dengan pengujian manual.	Pada proses *Black Box Testing* dengan cara mencoba program yang telah dibuat dengan mencoba memasukan data pada setiap formnya. Pengujian ini diperlukan untuk mengetahui program tersebut berjalan sesuai dengan yang dibutuhkan oleh perusahaan. Hasil test case pengujian terbagi menjadi 2 yaitu manual menggunakan Google Spreadsheet dan otomatis menggunakan *test case* yang dibuat dengan metode *record & playback* yang disediakan oleh katalon.

Hasil Penelitian
Total keseluruhan fungsi yang diuji ada 5 fungsi dan 42 item/*object* ditemukan error 0 atau tidak ditemukan adanya *error* pada saat pengujian, Berdasarkan eksperimen yang dilakukan dapat diambil kesimpulan bahwa kesimpulan tergantung dari kebutuhan *testing* yang dilakukan jika membutuhkan pengujian yang repetitif atau harus menguji banyak *platform* data yang besar sebaiknya itu di buat *automation script* dan jika sifatnya cuma 1 kali testing atau dibutuhkan perasaan atau *experience* langsung biasanya sifatnya *emergency* sebaiknya manual.
Perbedaan dengan penelitian yang akan dilakukan
Penelitian saya akan melakukan *automation testing* menggunakan Katalon Studio dengan memanfaatkan *spy object*.

Judul Penelitian	**Peneliti & tahun Penelitian**
Implementasi *Black Box Testing* pada Sistem Informasi Manajemen Dosen	Ni Made Dwi Febriyantia1, A.A. Kompiang Oka Sudana a2, I Nyoman Piarsa b3 (2021)
Tujuan	**Ruang lingkup**
Untuk mendapatkan hasil yang optimal maka perlu dilakukan pengujian dengan menggunakan *black box testing* dengan bantuan *automation tools* TestProject memiliki tujuan untuk meminimalisir adanya kesalahan dan memastikan apakah fungsi-fungsi pada sistem berjalan sesuai yang diharapkan	Metodologi penelitian yang digunakan pada pengujian perangkat lunak SIMDOS menggunakan metode *black box testing*. Metode *Blackbox Testing* adalah sebuah metode yang dipakai untuk menguji sebuah *software* tanpa harus memperhatikan detail *software*. Proses *Black Box Testing* dengan cara mencoba program yang telah dibuat dengan mencoba memasukkan data pada setiap formnya.
Hasil Penelitian	

Pengujian dengan menggunakan metode *black box testing* dengan bantuan *automation tools* pada Sistem Informasi Manajemen Dosen (SIMDOS). *Black box testing* digunakan untuk mengetahui kesesuaian alur fungsi dengan bisnis proses yang diinginkan oleh pengguna. Terdapat 53 butir uji yang diujikan pada SIMDOS didapatkan 51 hasil pengujian yang sesuai dan 2 hasil pengujian yang tidak sesuai dengan harapan pengguna.
Perbedaan dengan penelitian yang akan dilakukan
Penelitian saya akan melakukan *Automation testing* dengan menggunakan *tool* Katalon Studio

Judul Penelitian	**Peneliti & tahun Penelitian**
AUTOMATION TESTING TOOL DALAM PENGUJIAN APLIKASI BELAJAR TAJWID PADA *PLATFORM* ANDROID	Herlinda1, Dona Katarina2, Erlin Windia Ambarsari3 (2019)
Tujuan	**Ruang lingkup**
Menganalisis efektivitas aplikasi yang sudah dibangun, yang kemudian apakah *Automation Testing Tool* Katalon berdaya guna dibandingkan pengujian manual.	Pada penelitian ini, pengujian yang dilakukan adalah pengujian aplikasi berbasis Android *(Mobile)*
Hasil Penelitian	
Aplikasi tajwid ketika diuji membutuhkan 326,128 detik atau rata-rata sekitar 26,128 detik sampai dengan 5 menit dengan 121 langkah pengujian. Dari 121 langkah tersebut, terdapat 3 kegagalan, antara lain tampilan layar tidak dapat menyesuaikan ketika diubah ke dalam mode *Landscape*.	
Perbedaan dengan penelitian yang akan dilakukan	
Penelitian saya akan melakukan *automation testing* pada *website* STIKI TECHNOFEST	

2.2 Tinjauan Teoritis

2.2.1 Software Quality Assurance

Software Quality Assurance merupakan rancangan sistematis terkait dengan penilaian kualitas, standar produk, prosedur dan proses dari *software* yang bertujuan agar *software* yang dibangun bermutu tinggi (Tohirin et al., 2019). *SQA* adalah proses formal untuk mengevaluasi dan mendokumentasikan kualitas pekerjaan produk yang dihasilkan selama setiap tahap *Software Development Lifecycle (SDLC)* (Dhawan, 2015). Tujuan utama dari proses *SQA* adalah untuk memastikan produksi produk kerja berkualitas tinggi sesuai dengan persyaratan yang dinyatakan dan standar yang ditetapkan (Dhawan, 2015). Pengujian dilakukan untuk memberikan *stakeholder* informasi dari kualitas software yang diuji (Desi Dyah Sulistiyarini, 2016).

2.2.2 STLC (Software Testing Life Cycle)

Proses pengujian dibagi menjadi urutan langkah yang didefinisikan dengan baik disebut sebagai *STLC* yang terdiri dari *test plan, test design, test execution, post execution/ test review* (Honest, 2019).

a. Test Plan

Tujuan utama dari fase ini adalah untuk mempertimbangkan isu-isu penting dari pengujian strategi seperti sumber daya, jadwal, tanggung jawab, risiko dan prioritas sebagai roadmap. Keluaran dari tahap ini adalah dokumen *test plan. Test plan* dikembangkan setiap fase *SDLC* (Honest, 2019).

b. Test Design

Pada fase ini *Test Cases* dirancang. Dimulai dari menentukan tujuan pengujian, penyusunan daftar item yang akan diuji, identifikasi kasus uji, pemilihan teknik desain kasus uji, membuat kasus uji dan data uji, menyiapkan lingkungan pengujian dan alat pendukung, membuat spesifikasi prosedur pengujian (Honest, 2019).

c. Test Execution

Pada fase ini, semua kasus uji dijalankan termasuk verifikasi dan validasi. Hasil tes didokumentasikan dalam laporan insiden pengujian, log pengujian, status pengujian, dan laporan ringkasan pengujian (Honest, 2019).

d. Post Execution/Test Review

Setelah eksekusi pengujian berhasil, bug akan dilaporkan ke pengembang terkait. Fase ini adalah untuk menganalisis masalah terkait bug dan mendapatkan umpan balik bahwa jumlah maksimum *bug* dapat dihapus. Setelah memperbaiki *bug*, pengembang laporan ke tim pengujian dan bagian perangkat lunak yang dimodifikasi diuji sekali lagi laporan *bug* akhir ditinjau dan dianalisis untuk proses pengujian keseluruhan (Honest, 2019).

2.2.3 Jenis – Jenis Pengujian

2.2.3.1 Manual Testing

Manual testing adalah teknik pengujian di mana *tester* menyiapkan kasus uji secara manual dan menjalankannya untuk mengidentifikasi cacat pada *software*. Ini merupakan metode paling tua dalam pengujian *software* (Maurya & Kumar,

2012). *Manual testing* itu melelahkan aktivitas yang mengharuskan penguji untuk memiliki kesabaran, pengamat, spekulatif, kreatif, inovatif, berpikiran terbuka, dan terampil (Sharma, 2014). Manual testing berulang bisa sulit dilakukan pada aplikasi besar atau aplikasi yang memiliki kumpulan data yang sangat besar (Sharma, 2014).

Permasalahan dalam manual testing antara lain (Sharma, 2014):

a. Memakan waktu dan membosankan: Karena kasus uji dieksekusi oleh sumber daya manusia sehingga sangat lambat dan membosankan.
b. Investasi besar dalam sumber daya manusia: Sesuai kebutuhan kasus uji untuk dieksekusi secara manual sehingga lebih banyak penguji diperlukan dalam manual tes.
c. Kurang dapat diandalkan: Pengujian manual kurang dapat diandalkan karena pengujian mungkin tidak dilakukan dengan presisi setiap kali karena kesalahan manusia.
d. Tidak dapat diprogram: Tidak ada pemrograman yang dapat dilakukan untuk menulis tes canggih yang mengambil informasi tersembunyi. Pengujian Manual bisa menjadi membosankan dan karenanya rawan kesalahan.

2.2.3.2 Automation Testing

Automation testing adalah teknik pengujian perangkat lunak yang dilakukan dengan menggunakan software pengujian otomatis khusus untuk menjalankan rangkaian kasus uji. Sebaliknya, pengujian manual dilakukan oleh manusia yang menjalankan langkah-langkah pengujian (Thooriqoh et al., 2021). *Software* pengujian otomatis juga dapat memasukkan data pengujian ke dalam sistem yang

diuji, membandingkan hasil yang diharapkan dan menghasilkan laporan pengujian yang terperinci (Thooriqoh et al., 2021). Dengan menggunakan software pengujian otomatis, dimungkinkan untuk merekam rangkaian pengujian ini dan menjalankan ulang sesuai kebutuhan (Thooriqoh et al., 2021). Dalam menjalankan pengujian otomatis tidak diperlukan campur tangan manusia sehingga meningkatkan *ROI* dari pengujian otomatis (Thooriqoh et al., 2021). Tujuan automation testing adalah untuk mengurangi jumlah kasus uji yang dijalankan secara manual dan tidak menghilangkan pengujian manual (Thooriqoh et al., 2021).

Automation Testing Software melibatkan pengembangan pengujian skrip menggunakan bahasa skrip seperti *Python, JavaScript atau Tcl (Tool Command Language)*, sehingga kasus uji dapat dijalankan oleh komputer dengan intervensi manusia yang minimal (Sharma, 2014). Test design dan development bersama-sama dapat diotomatisasi untuk mengurangi upaya manusia dan menghemat biaya (Yadav, S., & Yadav, M, 2016). *Automation software* dapat menginputkan data ke dalam sistem yang sedang diuji, lalu melakukan perbandingan hasil yang diharapkan dan aktual dari *test report* (Maurya & Kumar, 2012). *Test Automation* menuntut investasi uang dan sumber daya yang cukup besar (Ramler & Wolfmaier, 2006). Dalam siklus pengembangan akan membutuhkan eksekusi rangkaian pengujian yang sama berkali-kali (Ramler & Wolfmaier, 2006). Menggunakan alat automation testing, anda dapat membuat *test suite* dan dapat dijalankan ulang sesuai kebutuhan. Setelah *test suite* diotomatisasi, tidak diperlukan campur tangan manusia (Ramler & Wolfmaier, 2006). Tujuan dari otomatisasi adalah untuk

mengurangi jumlah kasus uji yang akan dijalankan secara manual dan tidak menghilangkan pengujian manual secara bersamaan (Sharma, 2014).

Berikut manfaat dari automation testing (Sharma, 2014):

a. Cepat: Ini lebih cepat daripada pengujian manual.
b. Hemat Biaya: *Test case* dijalankan dengan menggunakan alat otomatisasi sehingga lebih sedikit penguji yang diperlukan dalam *automation testing.*
c. Dapat diulang: *Test case* yang sama (merekam dan memutar ulang) dapat dieksekusi kembali menggunakan alat pengujian.
d. Dapat digunakan kembali: Setelan uji dapat digunakan kembali pada berbagai versi perangkat lunak.
e. Dapat diprogram: Penguji dapat memprogram tes yang membawa informasi tersembunyi.
f. Komprehensif: Penguji dapat membuat rangkaian pengujian yang mencakup setiap fitur dalam aplikasi perangkat lunak.
g. Lebih andal: Automation testing bekerja sama persis operasi setiap kali dijalankan.
h. Cakupan Uji: Cakupan uji yang lebih luas dari fitur aplikasi

2.2.3.3 Functional Testing

a. Unit Testing

Jenis pengujian perangkat lunak yang dilakukan pada unit atau komponen individu . Biasanya, pengujian unit dilakukan oleh pengembang

pada fase pengembangan aplikasi. Setiap unit dalam pengujian unit dapat dilihat sebagai metode, fungsi, prosedur, atau objek. Pengembang sering menggunakan alat otomatisasi pengujian seperti NUnit, Xunit, JUnit untuk eksekusi pengujian (Vijay, 2023).

i. White Box Testing

Teknik pengujian di mana struktur internal atau kode aplikasi terlihat dan dapat diakses oleh penguji. Dalam teknik ini, mudah untuk menemukan celah dalam desain aplikasi atau kesalahan dalam logika bisnis. Cakupan pernyataan dan cakupan keputusan/cabang adalah contoh teknik pengujian kotak putih (Vijay, 2023).

ii. Gorilla Testing

Teknik pengujian di mana penguji dan/atau pengembang menguji modul aplikasi secara menyeluruh dalam semua aspek. Pengujian gorila dilakukan untuk memeriksa seberapa tangguh aplikasi Anda (Vijay, 2023).

b. Integration Testing

Jenis pengujian perangkat lunak di mana dua atau lebih modul aplikasi dikelompokkan secara logis dan diuji sebagai satu kesatuan. Fokus dari jenis pengujian ini adalah untuk menemukan cacat pada antarmuka, komunikasi, dan aliran data di antara modul-modul. Pendekatan atas-bawah

atau bawah-atas digunakan saat mengintegrasikan modul ke dalam sistem keseluruhan (Vijay, 2023).

i. Gray box testing

Kombinasi dari pengujian kotak putih dan pengujian kotak hitam. Penguji memiliki pengetahuan parsial tentang struktur internal atau kode aplikasi (Vijay, 2023).

c. System Testing

Jenis pengujian di mana penguji mengevaluasi seluruh sistem berdasarkan persyaratan yang ditentukan (Vijay, 2023).

i. End to End Testing

Melibatkan pengujian lingkungan aplikasi lengkap dalam situasi yang meniru penggunaan dunia nyata, seperti berinteraksi dengan basis data, menggunakan komunikasi jaringan, atau berinteraksi dengan perangkat keras, aplikasi, atau sistem lain jika sesuai (Vijay, 2023).

ii. Black Box Testing

Teknik pengujian perangkat lunak di mana pengujian dilakukan tanpa mengetahui struktur internal, desain, atau kode dari sistem yang diuji. Penguji harus fokus hanya pada masukan dan keluaran objek pengujian (Vijay, 2023).

iii. Smoke Testing

Dilakukan untuk memverifikasi bahwa fungsionalitas dasar dan kritis dari sistem yang diuji berfungsi dengan baik pada tingkat yang sangat tinggi (Vijay, 2023).

iv. Sanity Testing

Dilakukan pada suatu sistem untuk memastikan bahwa fungsionalitas baru yang ditambahkan atau perbaikan bug berjalan dengan baik. Pengujian sanity dilakukan pada build yang stabil. Ini merupakan bagian dari pengujian regresi (Vijay, 2023).

v. Happy path Testing

Menguji aplikasi dengan sukses pada alur positif. Ini tidak mencari kondisi negatif atau kesalahan. Fokus hanya pada masukan valid dan positif melalui mana aplikasi menghasilkan keluaran yang diharapkan (Vijay, 2023).

vi. Monkey Testing

Dilakukan oleh seorang penguji, dengan asumsi bahwa jika seekor monyet menggunakan aplikasi, maka bagaimana masukan dan nilai acak akan dimasukkan oleh Monyet tanpa pengetahuan atau pemahaman tentang aplikasi (Vijay, 2023).

d. Acceptance Testing

Jenis pengujian di mana klien/usaha/pelanggan menguji perangkat lunak dengan skenario bisnis waktu nyata. Klien menerima perangkat lunak hanya ketika semua fitur dan fungsionalitas berfungsi seperti yang diharapkan. Ini adalah fase terakhir dari pengujian, setelah itu perangkat lunak masuk ke produksi. Ini juga disebut Pengujian Penerimaan Pengguna (UAT) (Vijay, 2023).

i. Alpha Testing

Jenis pengujian penerimaan yang dilakukan oleh tim di organisasi untuk menemukan sebanyak mungkin cacat sebelum merilis perangkat lunak kepada pelanggan (Vijay, 2023).

ii. Beta Testing

Jenis pengujian perangkat lunak yang dilakukan oleh klien/pelanggan. Ini dilakukan di Lingkungan Nyata sebelum produk dilepas ke pasar untuk pengguna akhir yang sebenarnya.

iii. Operational acceptance testing (OAT)

Pengujian penerimaan operasional dari sistem dilakukan oleh staf operasi atau administrasi sistem di lingkungan produksi. Tujuan dari pengujian penerimaan operasional adalah memastikan bahwa administrator sistem dapat menjaga sistem berfungsi dengan baik bagi pengguna dalam lingkungan waktu nyata (Vijay, 2023).

2.2.3.4 Non Functional Testing

a. Security Testing

Dilakukan untuk memeriksa seberapa aman perangkat lunak, aplikasi, atau situs web dari ancaman internal dan/atau eksternal. Pengujian ini mencakup seberapa aman perangkat lunak dari program jahat, virus, dan seberapa aman & kuat proses otorisasi dan otentikasi (Vijay, 2023).

i. Penetration Testing

Jenis pengujian keamanan yang dilakukan sebagai serangan siber yang sah pada sistem untuk menemukan titik-titik lemah sistem dalam hal keamanan (Vijay, 2023).

b. Performance Testing

Pengujian stabilitas dan waktu respons aplikasi dengan memberlakukan beban (Vijay, 2023).

i. Load testing

Pengujian stabilitas dan waktu respons aplikasi dengan memberlakukan beban, yang sama atau kurang dari jumlah pengguna yang dirancang untuk aplikasi (Vijay, 2023).

ii. Stress Testing

Pengujian stabilitas dan waktu respons aplikasi dengan memberlakukan beban, yang lebih dari jumlah pengguna yang dirancang untuk aplikasi (Vijay, 2023).

iii. Scalability Testing

Pengujian stabilitas dan waktu respons aplikasi dengan memberlakukan beban, yang lebih dari jumlah pengguna yang dirancang untuk aplikasi (Vijay, 2023).

iv. Volume testing (flood testing)

Pengujian stabilitas dan waktu respons aplikasi dengan mentransfer volume data besar ke basis data. Pada dasarnya, ini menguji kapasitas basis data untuk menangani data (Vijay, 2023).

v. Endurance Testing (Soak Testing)

Pengujian stabilitas dan waktu respons aplikasi dengan memberlakukan beban secara terus menerus untuk jangka waktu yang lebih lama untuk memverifikasi bahwa aplikasi berfungsi dengan baik (Vijay, 2023).

c. Usability Testing

Pengujian aplikasi dari perspektif pengguna untuk memeriksa tampilan, nuansa, dan kegunaan (Vijay, 2023).

i. Exploratory testing

Pengujian informal yang dilakukan oleh tim pengujian. Tujuan dari pengujian ini adalah untuk menjelajahi aplikasi dan mencari cacat yang ada dalam aplikasi. Penguji menggunakan pengetahuan tentang domain bisnis untuk menguji aplikasi (Vijay, 2023).

ii. Cross browser testing

Pengujian aplikasi di berbagai browser, sistem operasi, perangkat seluler untuk melihat tampilan, nuansa, dan kinerjanya (Vijay, 2023).

iii. Accessibility Testing

Menentukan apakah perangkat lunak atau aplikasi dapat diakses oleh orang disabilitas atau tidak (Vijay, 2023).

d. Compatibility testing

Jenis pengujian di mana validasi dilakukan terhadap bagaimana perangkat lunak berperilaku dan berjalan di lingkungan yang berbeda, server web, perangkat keras, dan lingkungan jaringan (Vijay, 2023).

2.2.4 Metode Black Box Testing

Metode pengujian *black box* merupakan metode yang mudah digunakan karena hanya membutuhkan batas bawah dan batas atas dari data yang diharapkan. Estimasi jumlah data uji dapat dihitung dengan jumlah bidang entri data yang perlu diuji, aturan entri yang harus dipenuhi, dan apakah batas atas dan batas bawah terpenuhi, dan dengan metode ini dapat diketahui bahwa jika fungsi tersebut masih dapat menerima input data yang tidak diharapkan, maka akan menyebabkan data yang disimpan menjadi tidak valid (Cholifah et al., 2018). Metode *black box* merupakan salah satu metode pengujian yang berfokus pada bagaimana aplikasi terlihat, apa fungsinya, dan bagaimana fungsionalitasnya sesuai dengan proses bisnis (Ardi & Putro, 2021).

Pendekatan black box testing adalah metode pengujian yang memperoleh data pengujian dari persyaratan fungsional yang ditentukan, terlepas dari struktur program (Komarudin, 2016). Pengujian black-box sendiri merupakan pengujian yang hanya memverifikasi fungsionalitas perangkat lunak dengan mengamati hasil eksekusi menggunakan data pengujian (Maulana Syaban & Bunyamin, 2016). Tujuan dari pengujian *black box* pada sistem adalah untuk memastikan bahwa bagian dari sistem aplikasi menampilkan pesan kesalahan dengan benar ketika terjadi kesalahan entri data (Hanifah et al., 2016).

2.2.5 Testing Tools

Alat otomatisasi adalah perangkat lunak itu sendiri dan dapat digunakan untuk fokus pada pengujian perangkat lunak yang sebenarnya. Dengan kata lain, alat otomatisasi bertindak sebagai sarana untuk melakukan pengujian perangkat lunak

(M. A. Umar & C. Zhanfang, 2019). Ada berbagai alat otomatisasi yang tersedia. Identifikasi alat otomatisasi yang tepat sangat penting untuk memastikan keberhasilan proyek pengujian (M. A. Umar & C. Zhanfang, 2019). Tabel dibawah ini menyajikan perbandingan beberapa alat pengujian otomatisasi yang popular (M. A. Umar & C. Zhanfang, 2019):

Tabel 2. 2 Perbandingan Alat Pengujian Otomatisasi

Features	**Katalon Studio**	**Selenium**	**UFT**	**TestComplete**
Test development platform	*Cross platform*	*Cross platform*	*Windows*	*Windows*
Application under test	*Windows desktop, Web, Mobile apps, API*	*Web*	*Windows desktop, Web, Mobile apps, API*	*Windows desktop, Web, Mobile apps, API*
Scripting languages	*Java/Groovy*	*Java, C#, Perl, Python, JavaScript, Ruby, PHP*	*VBScript*	*JavaScript, Python, VBScript, JScript, Delphi, C++ and C#*
Programming skills	*Not required. Recommended for advanced test scripts*	*Advanced skills needed to integrate various tools*	*Not required. Recommended for advanced test scripts*	*Not required. Recommended for advanced test scripts*
Ease of installation and use	*Easy to set up and run*	*Require installing and integrating various tools*	*Easy to set up and run*	*Easy to set up and run*

Product support	*Community, Business support service, Dedicated staff*	*Open source community*	*Dedicated staff, Community*	*Dedicated staff, Community*
License type	*Freeware*	*Open source (Apache 2.0)*	*Proprietary*	*Proprietary*
Cost	*Free*	*Free*	*License and maintenance fees*	*License and maintenance fees*

Pemilihan alat tes otomatis seperti Katalon bisa dipengaruhi oleh sejumlah faktor. Berikut adalah beberapa alasan mengapa memilih Katalon sebagai alat tes otomatis (M. A. Umar & C. Zhanfang, 2019):

a. Kemudahan Penggunaan: Katalon memiliki antarmuka yang ramah pengguna dan mudah digunakan, bahkan oleh orang yang tidak memiliki pengetahuan pemrograman yang mendalam. Ini memungkinkan pengujian yang lebih cepat dan lebih mudah tanpa memerlukan keterampilan teknis yang tinggi.
b. Dukungan Multi-Platform: Katalon mendukung pengujian di berbagai platform termasuk aplikasi web, API, desktop, dan mobile. Ini sangat bermanfaat jika proyek memerlukan pengujian lintas platform.
c. Skrip Otomatisasi: Meskipun Katalon dirancang untuk non-programmer, ia juga mendukung penggunaan skrip dengan bahasa pemrograman seperti

Groovy. Ini memungkinkan fleksibilitas bagi tim pengujian yang memiliki latar belakang teknis yang lebih kuat

d. Komunitas dan Dukungan: Katalon memiliki komunitas aktif dan dukungan yang solid. Ini berarti ada banyak sumber daya yang tersedia untuk membantu pengguna, termasuk dokumentasi, tutorial, dan forum komunitas.
e. Integrasi dengan Alat Lain: Katalon dapat diintegrasikan dengan alat pengembangan dan kolaborasi seperti JIRA dan Git. Ini membantu dalam manajemen uji yang lebih baik dan integrasi dengan siklus pengembangan yang ada.
f. Fleksibilitas dalam Pengujian UI dan API: Katalon dapat digunakan untuk pengujian UI (antarmuka pengguna) dan juga pengujian API. Ini memungkinkan tim pengujian untuk melakukan pengujian lintas lapisan aplikasi.
g. Pengujian Ke Dua (Regression Testing): Katalon mendukung pengujian regresi, yang memungkinkan pengujian otomatis untuk memastikan bahwa perubahan baru tidak merusak fungsionalitas yang sudah ada.
h. Biaya: Katalon menyediakan versi gratis yang memiliki banyak fitur, serta versi berbayar yang menawarkan fitur tambahan dan dukungan teknis.

Jaringan juga dapat mempengaruhi penggunaan alat tes otomatis. Jika jaringan tidak stabil atau memiliki keterbatasan akses ke sumber daya luar, ini dapat mempengaruhi kemampuan alat tes untuk berkomunikasi dengan komponen yang diuji (Silpa Sasidharan, 2023).

2.2.6 Katalon Studio

Katalon Studio diluncurkan pada tahun 2016 dengan produk otomatisasi pengujian yang merupakan alternatif yang lebih mudah daripada alat lama yang tersedia. Sejak itu, Katalon LLC terus berinvestasi dalam komunitas kualitas perangkat lunak dengan produk tambahan, konten pembelajaran, dan jaringan mitra di seluruh dunia. Seperti basis user yang luas, tim menjangkau seluruh dunia. Katalon LLC berkantor di Atlanta, Kota Ho Chi Minh, dan New Delhi (Katalon LLC, 2022.).

Katalon Studio adalah integrated development environment (IDE) untuk pembuatan pengujian otomatis, dapat membuat, menjalankan, dan melihat laporan untuk pengujian otomatis. Dengan tampilan manual dan tampilan skrip, Katalon Studio cocok bahkan ketika Anda memiliki sedikit pengalaman pemrograman, atau Anda sudah ahli di bidang pengujian otomatis (*Katalon Studio Overview | Katalon Docs*, 2022).

Katalon Studio adalah alat pengujian otomatisasi kode rendah dan dapat digunakan untuk web, API, desktop (Windows), dan aplikasi seluler. Setelah menghapus persyaratan pengkodean dan membangun kerangka kerja otomatisasi pengujian dari awal, pengguna cukup mengunduh alat dan hanya fokus pada pengujian. Selain itu, Studio sering menawarkan rilis agar tetap kompatibel dengan platform/browser/OS terbaru (Katalon LLC, 2022).

Fitur Fitur Katalon Studio (Katalon LLC, 2022):

a. Metode fleksibel untuk desain pengujian: *Record & Playback, Manual and Scripting Mode*
b. Metodologi pengujian yang didukung : *BDD, DDT, Keyword-Driven Testing, Cross-Browser Testing (Headless, Chrome, Edge, Firefox and Safari) and Cross-Platform Mobile Testing (iOS, Android)*
c. *Automatic Retry Failed Tests, Smart Wait, and Self-Healing mechanisms*
d. *Test object, keywords,* dan *test case* yang dapat digunakan kembali dengan berbagai pengujian
e. Integrasi dengan *tool CI/CD* dan *ALM* populer *(Jira, GitLab, Jenkins, BitBucket, Azure DevOps, dll.)*
f. *UI Debugging* Cerdas dan pelaporan pengujian untuk memecahkan masalah kegagalan dengan cepat
g. Dokumentasi alat terperinci dan tutorial video di Katalon Academy

2.2.7 Activity Diagram

Activity diagram dapat digunakan untuk memodelkan proses yang terjadi pada sistem (Rendi Juliarto, 2021). Urutan proses sistem direpresentasikan secara vertikal (Rendi Juliarto, 2021). Activity diagram adalah pengembangan dari use case yang memiliki alur aktivitas (Rendi Juliarto, 2021). Activity diagram harus digunakan secara paralel (horizontal) dengan teknik pemodelan lainnya seperti use case diagram dan state diagram (Rendi Juliarto, 2021). Activity diagram berfungsi dalam menganalisis diagram use case dengan menggambarkan aktor, tindakan yang diperlukan, dan kapan itu terjadi (Rendi Juliarto, 2021).

Tabel 2. 3 Komponen Activity Diagram

Simbol	Nama	Keterangan
	Status awal	Sebuah diagram aktivitas memilki sebuah status awal.
	Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
	Percabangan / decision	Percabangan dimana ada pilihan aktivitas yang lebih dari satu
	Penggabungan / join	Penggabungan yang mana lebih dari satu aktivitas lalu digabungkan jadi satu.
	Status akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebua status akhir
	Swimlane	Swimlane memisakahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi