ANALISIS PERBANDINGAN *UI TEST AUTOMATION FRAMEWORK* DENGAN METODE *THE DISTANCE TO THE IDEAL ALTERNATIVE*

Disusun Oleh:

N a m a : Gilang Aries Prasetyo
NIM : 20523112

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2024

HALAMAN PENGESAHAN DOSEN PEMBIMBING

ANALISIS PERBANDINGAN *UI TEST AUTOMATION FRAMEWORK* DENGAN METODE *THE DISTANCE TO THE IDEAL ALTERNATIVE*

TUGAS AKHIR

Disusun Oleh:

N a m a : Gilang Aries Prasetyo

NIM : 20523112

Yogyakarta, 10 Juli 2024

Pembimbing,

(Dr. Novi Setiani, S.T, M.T)

HALAMAN PENGESAHAN DOSEN PENGUJI

ANALISIS PERBANDINGAN *UI TEST AUTOMATION FRAMEWORK* DENGAN METODE *THE DISTANCE TO THE IDEAL ALTERNATIVE*

TUGAS AKHIR

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

Yogyakarta, 29 Juli 2024

Tim Penguji

Dr. Novi Setiani, S.T, M.T. _____

Anggota 1

Kholid Haryono, S.T., M.Kom. _____

Anggota 2

Moh. Idris, S.Kom., M.Kom. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Gilang Aries Prasetyo

NIM : 20523112

Tugas akhir dengan judul:

ANALISIS PERBANDINGAN *UI TEST AUTOMATION FRAMEWORK* DENGAN METODE *THE DISTANCE TO THE IDEAL ALTERNATIVE*

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

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

Yogyakarta, 10 Juli 2024

(Gilang Aries Prasetyo)

HALAMAN PERSEMBAHAN

Alhamdulillahi Robbil'alamin dengan penuh rasa syukur, karya tugas akhir ini kupersembahkan kepada kedua Orang tua tercinta sebagai ungkapan terima kasih atas segala dukungan, doa, nasihat, dan kasih sayang yang telah diberikan selama ini. Semoga, ini menjadi langkah awal untuk membuat Ayah dan Ibu bahagia.

HALAMAN MOTO

"Bahwa manusia hanya memperoleh apa yang telah diusahakannya, bahwa sesungguhnya usahanya itu kelak akan diperlihatkan (kepadanya), kemudian dia akan diberi balasan atas (amalannya) itu dengan balasan yang paling sempurna, bahwa sesungguhnya kepada Tuhanmulah kesudahan (segala sesuatu)"

(QS. An-Najm:39-42)

"Apa yang sudah digariskan menjadi milik kita, akan tetap milik kita"

(Wira Nagara)

"Sesungguhnya beserta kesulitan ada kemudahan. Apabila engkau telah selesai (dengan suatu kebajikan), teruslah bekerja keras (untuk kebajikan yang lain)."

(Q.S. Al-Insyirah: 6-7)

KATA PENGANTAR

Assalamu'alakum Warrahmatullahi Wabarakatuh

Puja dan puji dan syukur penulis panjatkan kepada Allah SWT atas segala rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir penulisan ini. Perasaan senang dan juga kesedihan saya curahkan dalam penulisan laporan ini atas segala waktu dan kenangan yang telah saya alami. Dalam penulisan laporan ini, penulis mendapatkan segala bentuk dukungan dari berbagai pihak yang membuat penulis dapat menyelesaikan rangkaian pengerjaan tugas akhir ini. Maka dari itu, penulis ingin mengucapkan terima kasih kepada:

1. Allah SWT yang telah memberikan segala Rahmat dan hidayahnya sehingga penulis dapat menyelesaikan pengerjaan tugas akhir ini.
2. Kedua orang tua serta kakak dan adik yang senantiasa berdoa dan memberikan dukungan dalam bentuk apa pun kepada penulis.
3. Bapak Prof. Fathul Wahid, S.T., M.Sc, Ph.D selaku Rektor Universitas Islam Indonesia.
4. Bapak Prof., Dr., Ir., Hari Purnomo, M.T., IPU, ASEAN.Eng. Pimpinan Fakultas Teknologi Industri Universitas Islam Indonesia.
5. Bapak Dr. Raden Teduh Dirghayu, S.T., M.Sc. Ketua Jurusan Informatika Universitas Islam Indonesia.
6. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. Ketua Program Studi Informatika Universitas Islam Indonesia dan selaku dosen pembimbing akademik yang sempat memberikan motivasi kepada penulis.
7. Ibu Dr. Novi Setiani, S.T., M.T. selaku dosen pembimbing tugas akhir yang senantiasa memberikan ilmu, arahan serta waktu untuk membimbing penulis mengerjakan tugas akhir ini.
8. Seluruh dosen serta staf pengajar Program Studi Informatika Universitas Islam Indonesia.
9. Rekan-rekan perkuliahan saya yang senantiasa memberikan dukungan serta menemani saya dalam pengerjaan tugas akhir ini.
10. Sahabat Badan Sistem Informasi, khusus nya Tim Akurasi yang telah memberikan kenangan serta dukungan yang baik kepada penulis selama pengerjaan tugas akhir ini disela sela waktu magang.
11. Sahabat-sahabat kontrakan saya, khusus nya Fauzan, Fahrizal, Hawada, Arya, Nanda, Pradipta, Raffry, Leo, Derry, Zidan dan yang tidak bisa saya sebutkan satu

persatu, yang selalu peduli atas segala permasalahan dan memberikan dukungan dalam bentuk apa pun kepada penulis, semoga kalian sehat selalu.

Penulis menyadari bahwa dalam penulisan laporan tugas akhir ini terdapat banyak kekurangan. Oleh sebab itu dengan penulis menerima segala bentuk kritik dan saran yang membangun. Semoga Allah SWT memberikan taufik dan hidayah-Nya kepada kita semua Aamiin.

Wassalamu'alaikum warrahmatullahi wabarakatuh

Yogyakarta, 10 Juli 2024

(Gilang Aries Prasetyo)

SARI

Seiring dengan maraknya berbagai alat pengujian otomatis, pemilihan alat yang sesuai menjadi tugas yang semakin rumit. Penelitian ini bertujuan untuk membandingkan kinerja tiga *framework* otomatisasi pengujian, yaitu Selenium, Cypress, dan Playwright, dalam pengujian UI pada *platform website*. Metode yang digunakan adalah The Distance to the Ideal Alternative (DIA) untuk menilai performa *framework* berdasarkan parameter *Technical and Economic View, Testing Process View,* dan *Quality View*. Studi ini diawali dengan analisis literatur untuk menentukan parameter pengujian, diikuti oleh pembobotan parameter melalui survei pada *software tester* dan *programmer* profesional. *Framework* diuji pada *website* iconhub.io, dan hasil pengujian menunjukkan bahwa Cypress adalah *framework* terbaik dengan skor tertimbang terendah, menunjukkan keunggulan dalam kecepatan dan efisiensi. Hasil penelitian ini memberikan panduan bagi pengembang dalam memilih *framework* pengujian UI yang tepat dan efisien, berkontribusi pada peningkatan kualitas dan efektivitas pengujian perangkat lunak.

Kata kunci: *Test automation framework*, *UI Testing*, Selenium, Cypress, Playwright

GLOSARIUM

Pengujian UI	Proses untuk memastikan bahwa antarmuka pengguna dari aplikasi perangkat lunak berfungsi sebagaimana mestinya dan memberikan pengalaman pengguna yang diharapkan.
Selenium	*Framework* pengujian otomatisasi *open-source* yang digunakan untuk mengotomatisasi tugas pengujian pada berbagai *browser web*.
Distance to the Ideal Alternative (DIA)	Metode analisis Multi-kriteria yang digunakan untuk membandingkan beberapa alternatif berdasarkan kedekatan mereka dengan alternatif ideal.
Open Source	Perangkat lunak yang *source code*-nya tersedia untuk umum dan dapat digunakan, dimodifikasi, serta didistribusikan kembali oleh siapa saja.
Cypress	Alat pengujian modern untuk aplikasi *web* yang dibangun untuk memudahkan pengujian *end-to-end* secara otomatis dengan kecepatan tinggi.
Playwright	*Framework* pengujian otomatisasi yang dikembangkan oleh Microsoft, memungkinkan pengujian pada berbagai *browser* modern dengan API yang konsisten.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR FORMULA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada pengembangan perangkat lunak, pengujian perangkat lunak memainkan peran yang sangat penting dalam menghasilkan *output* sistem berkualitas tinggi dan mendeteksi kesalahan sistem. Pengujian perangkat lunak merupakan bagian yang tersusun dalam pembuatan sistem *development* (Saputra & Stefanie, 2023). Pengujian perangkat lunak merupakan komponen penting dalam proses pengembangan perangkat lunak. Penggunaan pengujian otomatis memperbaiki keakuratan dan efisiensi dalam proses tersebut. Seiring dengan maraknya berbagai alat pengujian otomatis, pemilihan alat yang sesuai menjadi tugas yang semakin rumit (Abdulwareth & Al-Shargabi, 2021).

Pengujian perangkat lunak dapat dilakukan melalui dua pendekatan, yaitu pengujian manual dan pengujian otomatis. Menurut (Nagabushanam dkk., 2022) pengujian perangkat lunak memungkinkan kita untuk mengevaluasi sejauh mana desain dan arsitektur perangkat lunak sesuai dengan fungsionalitasnya, sehingga kita dapat memahami sejauh mana ketepatan yang tercapai dalam proses pengembangan. Dalam metode pengujian manual, penguji berperan sebagai pengguna akhir yang mencoba berbagai fitur perangkat lunak, namun demikian, metode ini bisa memakan waktu dan menguras tenaga, dan tidak selalu efektif dalam mengidentifikasi seluruh cacat yang mungkin ada (Gamido & Gamido, 2019). Sementara itu, menggunakan pengujian otomatis akan mengurangi waktu yang dibutuhkan dalam pengujian dibandingkan dengan pengujian manual (Saputra & Stefanie, 2023). Penerapan metode pengujian otomatis membuat pengujian menjadi lebih sederhana, efisien, dan efektif. Selain mempercepat proses pengujian, pendekatan ini juga memungkinkan pengujian dengan cakupan yang lebih luas dalam jangka waktu terbatas, yang pada gilirannya meningkatkan kualitas perangkat lunak yang dihasilkan (Nagabushanam dkk., 2022). Selain itu, skenario pengujian yang dibuat oleh penguji perangkat lunak dalam pengujian otomatis dapat digunakan kembali atau dieksekusi ulang. Inilah alasan utama yang mendorong sebagian besar penguji untuk beralih dari pengujian manual ke pengujian perangkat lunak otomatis.

Terdapat beberapa *test automation framework* yang baik, terutama dengan keunggulan *open-source* yang menghemat biaya lisensi pada pengujian web. Terdapat 8 *test automation framework* terbaik yang digunakan oleh *developer* di seluruh dunia yaitu Selenium, Cypress,

Playwright, WebDriverIO, TestCafe, NightwatchJS, Appium, dan Cucumber (Badkar, 2023). Penulis akan berfokus pada 3 *test automation framework* saja yaitu Selenium, Cypress dan Playwright, yang masing-masing akan penulis uji. Penulis memilih untuk membandingkan ketiga *test automation framework* ini karena popularitas diakui pada industri perangkat lunak dibuktikan pada *used by* di github pada *framework* cypress sebanyak 660 ribu pengguna, 175 ribu pengguna pada selenium, dan 46 ribu pengguna pada playwright. Serta populer karna banyaknya penelitian yang membahas dari ketiga *framework* tersebut dari hasil riset mereka, serta *test automation framework* tersebut merupakan yang terbaik di tahun 2023 (Badkar, 2023).

Penulis akan menguji dan menganalisis pengujian pada *website* iconhub.io yang mana *website* ini merupakan sebuah *website* dan juga *plugin* yang sangat berguna bagi desainer yang menggunakan aplikasi Figma dalam pekerjaan mereka. Dikembangkan oleh Javan Cipta Solusi dan *One Week Wonders*, IconHub memungkinkan pengguna untuk dengan mudah mencari, memilih, dan mengorganisir ikon yang diperlukan dalam desain mereka. *Plugin* ini memiliki koleksi ikon yang luas dan bervariasi, yang dapat digunakan untuk berbagai jenis proyek desain.

Untuk menguji UI *website* iconhub.io penulis akan menggunakan 3 *test automation framework*, yaitu Selenium, Cypress, dan Playwright. Parameter yang dipilih untuk dilakukan pengujian UI *framework* yaitu *Technical and Economic View*, *Testing Process View* dan *Quality view* (Abdulwareth & Al-Shargabi, 2021). Penulis akan menerapkan metode *The Distance to The Ideal Alternative* (DIA) sebagai metode pengambilan keputusan dalam menentukan *test automation framework* terbaik. Ini didasarkan pada penelitian yang dilakukan oleh (Raihansyah dkk., 2022) Metode DIA adalah sebuah pendekatan dalam MADM yang berdasarkan pada prinsip-prinsip yang mirip dengan *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS) Metode ini dikembangkan untuk menyempurnakan metode TOPSIS, dimana metode DIA juga menentukan nilai ideal positif dan negatif dari setiap atribut. Perbedaannya terletak pada penggunaan jarak dengan metode *Manhattan distance*, penentuan Positif Ideal Alternatif (PIA) yang berdasarkan urutan minimal dan maksimal dari nilai Ri, serta rumus yang digunakan dalam proses perangkingan alternatif. Oleh karena itu hal ini mendasari penulis untuk membuat penelitian dengan judul "Analisis Perbandingan *UI Test Automation Framework* dengan Metode *The Distance to The Ideal Alternative*".

1.2 Rumusan Masalah

Perumusan masalah pada penelitian ini adalah sebagai berikut:

a. Bagaimana menganalisis perbandingan *test automation framework* berdasarkan parameter *Technical and Economic View*, *Testing Process View*, dan *Quality view*?
b. Bagaimana hasil analisis perbandingan test automation *framework* berdasarkan parameter *Technical and Economic View, Testing Process View*, dan *Quality view* dengan metode *the distance to the ideal alternatives* (DIA)

1.3 Batasan Masalah

Batasan-batasan masalah dalam penelitian ini adalah sebagai berikut:

a. Pengujian yang dilakukan merupakan pengujian UI *(User Interface test* atau *functional test)*.
b. Pengujian dilakukan pada *platform website*
c. Pengujian dilakukan dengan 3 *test automation framework*, yaitu Selenium, Cypress, dan Playwright karena *test automation framework* tersebut dapat berkomunikasi dengan *web-browser* yang paling sering digunakan dan *test automation framework* terbaik berdasarkan studi literatur yang dipelajari penulis.
d. Parameter perbandingan pengujian dari segi kinerja yaitu *Technical and Economic View*, *Testing Process View* dan *Quality view*
e. Terdapat satu fitur yang tidak dilakukan pengujian karena keterbatasan *tools* saat mengidentifikasi *element* fitur yang diuji, fitur yang tidak diuji yakni memilih warna *icon*.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah menganalisis perbandingan UI *test automation framework* dengan *framework*, yaitu Selenium, Cypress, dan Playwright berdasarkan parameter *Technical and Economic View, Testing Process View, dan Quality View* dengan menggunakan *The Distance to The Ideal Alternative* (DIA) sebagai metode pengambilan keputusan.

1.5 Manfaat Penelitian

a. Mengetahui hasil dari pengujian *user interface* (UI) atau *functional test* sesuai parameter perbandingan yang telah ditentukan.
b. Pembaca dapat mengetahui *test automation framework* mana yang terbaik dari segi kinerja.

1.6 Metode Penelitian

Dalam penelitian metode yang akan penulis gunakan adalah sebagai berikut:

a. Proses Studi Literatur : Penelitian ini menggunakan studi literatur melalui jurnal, *internet*, dan sumber lain. Studi literatur dijadikan sebagai bahan referensi untuk penelitian ini.
b. Proses Pengujian: Pengujian dilakukan dengan *User Interface* (UI) *testing* dengan alat sebagai perbandingan.
c. Proses analisis hasil : Setelah dilakukannya pengujian unit akan dilakukan analisis kedua alat sebagai perbandingan untuk menentukan ideal atau tidaknya alat yang digunakan dengan menggunakan metode *The Distance to The Ideal Alternatif* (DIA).

1.7 Sistematika Penulisan

Struktur penulisan laporan dari penelitian dijelaskan pada pembahasan dibawah ini:

a. BAB I PENDAHULUAN

   Bab ini berisi tentang gambaran umum tentang topik yang dibahas. Mencakup tentang pembahasan latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metode penelitian, dan struktur laporan.

b. BAB II LANDASAN TEORI DAN KAJIAN PUSTAKA

   Bab ini memuat teori dan definisi yang terkait dengan penelitian baik dari buku, jurnal, maupun *website*. Selain itu bab ini juga berisi referensi artikel penulisan yang mengimplementasikan setiap sub bab.

c. BAB III METODOLOGI PENELITIAN

   Bab ini berisi tentang penjelasan tentang metode penelitian yang digunakan serta langkah-langkah yang digunakan dalam penelitian ini.

d. BAB IV HASIL DAN PEMBAHASAN

   Bab ini berisi tentang hasil pengujian UI (*User Interface*) / *functional testing* yang telah dilakukan dan menguraikan analisis dari pembahasan setiap *test automation framework* yang penulis uji dengan metode yang telah penulis pilih.

e. BAB V KESIMPULAN DAN SARAN

   Bab ini berisi tentang rangkuman dan kesimpulan dari hasil penelitian serta saran-saran yang dapat membantu untuk pengembangan analisis di masa yang akan datang.

BAB II
LANDASAN TEORI DAN KAJIAN PUSTAKA

2.1 *Test Automation Framework*

Menurut (Dias dkk., 2023) Pengujian perangkat lunak adalah proses verifikasi dan validasi perangkat lunak, yang tujuannya adalah menjawab dua pertanyaan masing-masing: apakah sistem yang benar sedang dibangun, dan apakah sistem tersebut dibangun dengan benar. Pertanyaan validasi dijawab oleh pemilik produk dan pengguna akhir yang memvalidasi persyaratan pengguna dan kepuasan solusi yang dikembangkan, masing-masing. Pertanyaan verifikasi dapat dijawab dengan memastikan, sejauh mungkin, bahwa perangkat lunak telah dibangun dan diuji dengan benar.

Test Automation Framework merupakan kumpulan peraturan dan alat yang sesuai yang digunakan untuk membuat kasus uji (*test case*). Tujuannya adalah untuk meningkatkan efisiensi fungsi rekayasa. Aturan umum untuk kerangka otomatisasi melibatkan standar pengkodean yang membantu menghindari pemasukan data manual, teknik penanganan dan keuntungan data uji, penyimpanan yang dapat diakses untuk hasil data uji, repositori objek, dan informasi tambahan yang mungkin digunakan untuk menjalankan pengujian secara optimal (Badkar, 2023).

Berikut adalah tabel keunggulan dari penggunaan *Test Automation Framework*

Tabel 2.1 Keunggulan penggunaan *Test Automtion Framework*

Sumber: (Badkar, 2023)

Keunggulan	Deskripsi
Penggunaan Kode Ulang	*Test automation framework* ketika sudah memiliki data kode yang dibutuhkan untuk keberhasilan pengujian otomatisasi, kode (skrip) berharga ini disimpan untuk digunakan lagi di masa depan dan bisa digunakan kembali kapan saja. Kita tidak perlu memasukkan atau menyusun ulang kode secara manual. Kode ini juga bisa dipakai untuk menjalankan pengujian otomatisasi lainnya. Skrip yang sudah dibuat juga bisa disimpan.

Keunggulan	Deskripsi
Biaya Rendah	Kasus uji (*test case*) dapat dikembangkan dengan biaya yang relatif murah karena pada *Test Automation Framework* sudah memiliki aturan yang mapan. Selain itu, karena kode ini dapat digunakan berulang kali, biaya dan waktu untuk membuat kasus uji untuk fitur baru menjadi rendah.
Efisiensi Meningkat	*Test Automation Framework* meningkatkan produktivitas karena adanya standarisasi. Ini menjamin cakupan uji maksimum karena rangkaian kode dalam suatu kerangka dijalankan dengan cara standar sejak awal.

2.1.1 Selenium

Selenium adalah sebuah perangkat lunak yang digunakan untuk mengotomatisasi pengujian perangkat lunak berbasis *web*. Selenium memungkinkan pengembang perangkat lunak untuk menulis skrip pengujian otomatis dalam berbagai bahasa pemrograman yang mendukung *WebDriver*. Dengan Selenium, pengguna dapat mengotomatisasi tindakan-tindakan seperti mengisi formulir, mengeklik tautan, dan memverifikasi teks pada halaman *web*, sehingga memungkinkan pengujian otomatis terhadap aplikasi *web*. Selenium telah menjadi salah satu alat otomatisasi pengujian yang paling populer dan andal dalam industri perangkat lunak (Mobaraya & Ali, 2019).

Selenium *web driver* juga dikenal sebagai selenium 2.0. Selenium *web driver* berkomunikasi langsung dengan *browser*, sehingga selenium *web driver* lebih cepat daripada Selenium RC. Selenium *web driver* mendukung beberapa *browser web* dan mendukung aplikasi Ajax. Tujuan utama dari selenium *web driver* adalah untuk meningkatkan dukungan terhadap masalah pengujian aplikasi *web modern*. Selenium *web driver* mendukung beberapa bahasa untuk menulis skrip uji. Namun, meskipun semua kelebihan dari selenium *web driver*, ada beberapa keterbatasan ketika menguji aplikasi *web*. Selenium *web driver* tidak memiliki fungsionalitas bawaan untuk menghasilkan tangkapan layar untuk kasus uji yang gagal. Selenium *web driver* tidak memiliki kemampuan bawaan untuk menghasilkan hasil uji. Ini bergantung pada alat pihak ketiga untuk menghasilkan laporan uji (Thooriqoh dkk., 2021).

2.1.2 Cypress

Cypress merupakan *automation testing tools end-to-end* yang dikembangkan dan dirancang khusus untuk aplikasi web *modern*. Fokus utama Cypress adalah mengatasi ketidaksesuaian dalam pengujian dengan memastikan bahwa penguji dapat menulis, melakukan *debugging*, dan menjalankan pengujian di *browser* tanpa memerlukan konfigurasi atau paket tambahan. Cypress beroperasi sebagai aplikasi independen dan dapat di-*instal* pada sistem operasi macOS, Unix/Linux, dan Windows, baik menggunakan *hyphenate application* maupun melalui *command-line tools*. *Framework* Cypress dirancang terutama untuk para pengembang yang menggunakan JavaScript dalam penulisan aplikasi mereka, sehingga dapat digunakan untuk menguji semua aplikasi yang berjalan di dalam *browser* (Mwaura, 2021).

2.1.3 Playwright

Playwright adalah kerangka pengujian *end-to-end* yang relatif baru yang dirancang untuk aplikasi web *modern*. Diciptakan oleh Microsoft pada tahun 2019 sebagai alternatif untuk kerangka pengujian yang sudah ada seperti Puppeteer dan WebDriver. Salah satu keunggulan Playwright adalah kemampuannya untuk mengotomatisasi aplikasi web dalam beberapa *browser*, termasuk Chromium, Firefox, dan WebKit. Selain itu, Playwright juga kompatibel di berbagai *platform*, memungkinkan pengembang menulis dan menjalankan pengujian di Windows, macOS, dan Linux (Brahmbhatt, 2023).

2.2 *User Interface* (*UI*) *Testing*

Pengujian *User Interface* (UI) adalah jenis pengujian perangkat lunak yang bertujuan untuk memvalidasi fungsi dan tampilan antarmuka pengguna sebuah aplikasi. Dalam *UI testing*, pengujian dilakukan untuk memastikan bahwa antarmuka pengguna (seperti layar, tombol, menu, dan elemen *visual* lainnya) berfungsi dengan benar dan memberikan pengalaman pengguna yang baik. Dalam esensinya, pengujian UI mengikuti paradigma *Black Box Testing*, di mana akses terhadap implementasi *Application Under* Test (AUT) tidak tersedia. Pentingnya pengujian UI adalah untuk memvalidasi kebenaran suatu aplikasi melalui petunjuk *visual* dan peristiwa interaktif yang muncul dalam penggunaan dunia nyata. Ini memastikan bahwa antarmuka pengguna berfungsi sesuai yang diharapkan dan memberikan pengalaman yang lancar bagi pengguna akhir (Nguyen dkk., 2019).

Tujuan utama dari pengujian UI otomatis adalah untuk mengurangi tugas yang berulang-ulang bagi seorang *tester* dan mengurangi kesalahan yang terkait dengan manusia dengan membuat skrip pengujian yang dapat menyimulasikan tindakan-tindakan seperti mengeklik

tombol, mengisi kotak teks, atau menavigasi melalui berbagai layar aplikasi (Nguyen dkk., 2019).

UI *testing* dapat dianggap sebagai bagian dari *functional testing* karena juga memeriksa fungsionalitas aplikasi dari perspektif pengguna. Kedua jenis pengujian ini bersama-sama memberikan gambaran menyeluruh tentang kinerja dan kualitas aplikasi, membantu mengidentifikasi masalah yang mungkin tidak terlihat jika hanya satu metode pengujian yang digunakan (Banerjee dkk., 2013; Khaliq dkk., 2022).

2.3 Metode The Distance to The Ideal Alternative (DIA)

Metode DIA merupakan metode yang dimiliki oleh MADM yang baru dikembangkan oleh beberapa peneliti. Metode DIA didasarkan pada prinsip-prinsip sebagaimana pada metode TOPSIS, DIA juga menentukan nilai ideal positif dan negatif dari setiap atribut. Perbedaan terletak pada penentuan jarak yang menggunakan *manhattan distance*, penentuan Positif Ideal Alternatif (PIA) yang memiliki minimal + Dj , dan maksimal − Dj serta formula dalam urutan nilai pada Ri sebagai penentu perangkingan alternatif (Rijoly dkk., 2023). Adapun Langkah-langkah yang dilakukan dalam penyelesaian masalah menggunakan metode DIA adalah sebagai berikut (Rijoly dkk., 2023).

a. Menentukan matriks keputusan

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \quad (2.\ 1)$$

dengan:

X adalah matriks keputusan

x_{ij} adalah setiap elemen pada matriks X, $i = 1,2,3,\ldots, m; j = 1,2,3,\ldots, n$.

b. Normalisasi matriks keputusan

Matriks normalisasi diperoleh dari setiap elemen pada matriks keputusan dibagi dengan akar dari jumlah setiap elemen kolom matriks keputusan yang dikuadratkan. Persamaan ini dapat dilihat sebagai berikut :

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x^2{}_{ij}}} \quad (2.\ 2)$$

Dimana : r_{ij} adalah setiap elemen pada matriks normalisasi. $x^2{}_{ij}$ adalah nilai kuadrat setiap elemen pada matriks keputusan sehingga diperoleh matriks normalisasi (R) :

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix} \quad (2.\ 3)$$

c. Pembobotan pada matriks yang telah di normalisasi

Setelah proses normalisasi matriks keputusan, kemudian menentukan matriks pembobotan. Dimana setiap elemen matriks pembobotan diperoleh dari hasil setiap elemen bobot dikali dengan setiap elemen pada matriks normalisasi. Persamaan ini dapat dilihat sebagai berikut:

$$v_{ij} = r_{ij} w_i \quad (2.\ 4)$$

Dimana: v_{ij} adalah setiap elemen pada matriks pembobotan. w_i adalah setiap elemen pada matriks bobot (W).

Sehingga diperoleh matriks pembobotan (V) :

$$R = \begin{bmatrix} r_{11}w_1 & r_{12}w_2 & \cdots & r_{1n}w_n \\ r_{21}w_1 & r_{22}w_2 & \cdots & r_{2n}w_n \\ \vdots & \vdots & & \vdots \\ r_{m1}w_1 & r_{m2}w_2 & \cdots & r_{mn}w_n \end{bmatrix} \quad (2.\ 5)$$

d. Menentukan solusi ideal positif dan ideal negatif

Solusi ideal positif dinotasikan dengan A^+ dan solusi ideal negatif dinotasikan dengan A^-. Persamaan ini dapat dilihat sebagai berikut :

$$A^+ = \max v_{ij} = [a_1^+, a_2^+, \ldots, a_n^+] \quad (2.\ 6)$$

Dimana, max v_{ij} adalah nilai terbesar dari setiap kolom matriks V.

$$A^- = \min v_{ij} = [a_1^-, a_2^-, \ldots, a_n^-] \quad (2.\ 7)$$

Dimana, min v_{ij} adalah nilai terkecil dari setiap kolom matriks V.

e. Hitung jarak *manhattan* untuk atribut positif dan negative

$$D_i^+ = \sum_{i=1}^{m} \left[v_{ij} - a_i^+ \right] \quad (2.\ 8)$$

Dimana, a_i^+ adalah nilai dari setiap solusi ideal positif.

$$D_i^- = \sum_{i=1}^{m} \left[v_{ij} - a_i^- \right] \quad (2.\ 9)$$

Dimana, a_i^- adalah nilai dari setiap solusi ideal negatif.

f. Menentukan Positif Ideal Alternatif (PIA)

PIA minimal dinotasikan dengan min (D_i^+), nilai ini diperoleh dari pencarian nilai terkecil pada setiap baris D_i^+. Untuk PIA maksimal dinotasikan dengan max (D_i^-), dan nilai ini diperoleh dari pencarian nilai terbesar pada setiap baris D_i^-. Persamaan ini dapat dilihat sebagai berikut :

$$PIA = (\min(D_i^+), \max(D_i^-)) \quad (2.\ 10)$$

g. Melakukan identifikasi peringkat

Peringkat dapat ditentukan dengan membandingkan nilai identifikasi peringkat (R_i) yang diperoleh dari persamaan sebagai berikut :

$$R_i = \sqrt{\left(D_i^+ - \min\left(D_i^+\right)\right)^2 + \left(D_i^- - \max\left(D_i^-\right)\right)^2} \quad (2.\ 11)$$

Nilai R_i yang minimum mengindikasikan bahwa alternatif tersebut lebih terpilih sebagai peringkat yang lebih utama.

2.4 Parameter Perbandingan Pengujian

Terdapat beberapa parameter untuk bisa membandingkan kinerja sebuah *automation testing framework*. Menurut (Abdulwareth & Al-Shargabi, 2021) untuk membandingkan kinerja *automation testing framework* terdapat 3 parameter, yaitu:

a. *Technical and Economic View,* merupakan aspek untuk menilai *automation testing framework* berdasarkan sisi teknis, yaitu kemampuan yang dimiliki dan sisi ekonomi. Dengan mempertimbangkan aspek ini, pengembang dapat membandingkan antara keunggulan setiap *framework* serta biaya yang harus dikeluarkan.
b. *Testing Process View,* yaitu parameter saat pengembang mengklasifikasikan *automation testing framework* berdasarkan proses yang dialami dan seperti apa proses yang dirasakan

pengembang saat mencoba alat tersebut yang mencakup hasil eksekusi, seberapa sulit untuk melakukan *setup* serta proses pembuatan *code* nya.

c. *Quality View,* yaitu salah satu aspek penting dalam pengujian pada *automation testing framework*. Berdasarkan model kualitas perangkat lunak, terdapat berbagai karakteristik yang umumnya diidentifikasi, antara lain kegunaan, dan portabilitas. Oleh karena itu, karakteristik kualitas dianggap sebagai pandangan penting dalam pengujian.

Adapun parameter yang dimaksud adalah:

a. *Technical and Economic View* yang terdiri dari beberapa sub parameter yakni
   1. *Scripting Language*
   2. *Cost*
   3. *Product Support*

b. *Testing Process View* yang terdiri dari beberapa sub parameter yakni
   1. *Easy of Setup*
   2. *Script Creation Time*
   3. Dokumentasi hasil report
   4. Waktu hasil eksekusi
   5. *Inspection Element*

c. *Quality View* yang terdiri dari beberapa sub parameter yakni
   1. *Usability*
   2. *Portability*

2.5 Kajian Pustaka

Dalam penelitian ini, kajian Pustaka digunakan untuk melakukan analisis terhadap objek yang digunakan dalam penelitian.

2.5.1 Penelitian Terdahulu

Bagian ini akan membahas penelitian-penelitian sebelumnya yang telah dilakukan melalui berbagai referensi terkait dengan topik penelitian ini. Tujuannya adalah untuk mengidentifikasi perbedaan pada setiap topik yang dipilih oleh penulis. Tabel Penelitian terdapat pada 2.2

Tabel 2.2 Penelitian Terdahulu

No	Judul	Peneliti dan Tahun	Batasan	Metode	Hasil
1	*Toward a Multi-Criteria Framework for Selecting Software Testing Tools*	Asma J. Abdulwareth, Asma A. Al-shargabi 2021	Taksonomi yang rumit, perlunya validasi lebih luas, dan keterbatasan implementasi praktis.	*Analytic Hierarchy Process* (AHP) dan TOPSIS	Pengembangan *framework* yang mempermudah pemilihan alat uji perangkat lunak berdasarkan taksonomi komprehensif dan peringkat yang akurat.(Abdulwareth & Al-Shargabi, 2021).
2.	*A Comparative Analysis of Test Automation Frameworks Performance for Functional Testing in Android-Based Applications using the Distance to*	Calysta Merina, Nenny Anggraini, Nashrul Hakiem. 2018	Fokus pada kinerja Espresso, Calabash, dan Appium untuk aplikasi Android.	*The Distance to The Ideal Alternative* (DIA)	Calabash teridentifikasi sebagai *framework* terbaik dalam hal kinerja. Kelebihan dan kekurangan masing-masing *framework* dijelaskan.(Merina dkk., 2018).

No	Judul	Peneliti dan Tahun	Batasan	Metode	Hasil
	the Ideal Alternative Method				
3.	Studi Perbandingan Alat Pengujian Otomatis untuk Aplikasi Android	Arlinta Christy, Leo Siburian. 2019	Perbandingan Selendroid, Calabash, dan UI *Automator* untuk pengujian fungsional aplikasi Android.	*Software Testing Life Cycle* (STLC)	Selendroid dan Calabash lebih cocok untuk aplikasi *hybrid* dan *mobile web*, UI *Automator* untuk aplikasi native. Keunggulan berdasarkan fleksibilitas dan stabilitas disoroti. (Barus & Siburian, 2019).
4.	*A comparative study of automation testing tools for web applications*	Elis Pelivani, Betim Cico 2021	Evaluasi dan perbandingan Katalon Studio dan Selenium untuk pengujian aplikasi *web*.	*Software Testing Life Cycle* (STLC)	Perbandingan kinerja dan fitur antara Katalon Studio dan Selenium. Menyoroti kelebihan dan kekurangan serta implikasi AI dalam pengujian otomatis. (Pelivani dkk., 2021).
5.	*Technical Analysis of Selenium and Cypress as Functional Automation Framework for*	Fatini Mobaraya, Shahid Ali	Analisis teknis antara Selenium dan Cypress untuk aplikasi *web modern*.	*Agile Scrum*	Cypress lebih unggul dalam waktu eksekusi dan efisiensi. Saran untuk pengujian lebih lanjut pada berbagai *browser* dan penggunaan kerangka kerja pelaporan khusus. (Mobaraya & Ali, 2019).

No	Judul	Peneliti dan Tahun	Batasan	Metode	Hasil
	Modern Web Aplication				

2.5.2 Telaah Kajian Pustaka

Sub-bab ini memberikan ringkasan umum dari jurnal yang telah disampaikan sebelumnya, dengan fokus khusus pada Penelitian nomor 2 pada Tabel 2.2, "*A Comparative Analysis of Test Automation Frameworks Performance for Functional Testing in* Android *Based Applications*" oleh Calysta Merina, Nenny Anggraini, dan Nashrul Hakiem, tahun 2018. Jurnal ini membandingkan kinerja dari tiga *framework* pengujian otomatis Espresso, Calabash, dan Appium dalam konteks aplikasi berbasis Android. Metode yang digunakan adalah *Distance to The Ideal Alternative* (DIA) untuk mengukur dan membandingkan kinerja masing-masing *framework* berdasarkan parameter yang telah ditetapkan.

Penelitian yang dilakukan memiliki beberapa pembeda signifikan dari jurnal tersebut. Pertama, lingkup penelitian ini berfokus pada pengujian aplikasi *web*, bukan aplikasi *mobile*. Hal ini menimbulkan tantangan dan kebutuhan yang berbeda dalam hal alat dan pendekatan yang digunakan. Kedua, penelitian ini menggunakan rangkaian *framework* pengujian otomatis yang berbeda, yang lebih sesuai untuk aplikasi *web*. Ketiga, dalam penelitian ini, dilakukan pengembangan dan menggunakan lebih banyak parameter dan sub-parameter pembanding yang kompleks untuk menilai efektivitas berbagai *framework* pengujian. Parameter-parameter ini dirancang untuk mencakup aspek-aspek kritis yang lebih luas dan spesifik dari pengujian aplikasi web, seperti kecepatan, skalabilitas, dan kemudahan integrasi dengan alat pengembangan lainnya.

Selain itu, penentuan bobot untuk setiap parameter dalam penelitian ini dilakukan secara empiris, melibatkan *programmer* dan *tester* profesional. Ini bertujuan untuk mendapatkan perspektif praktis yang lebih dalam dan memastikan bahwa bobot yang ditetapkan mencerminkan pentingnya setiap parameter dalam konteks pengujian aplikasi *web* di dunia nyata. Pendekatan empiris ini memberikan validasi tambahan pada metodologi yang digunakan, menjamin bahwa penelitian tidak hanya teoritis tetapi juga teraplikasi dengan baik dalam praktik pengembangan *software*.

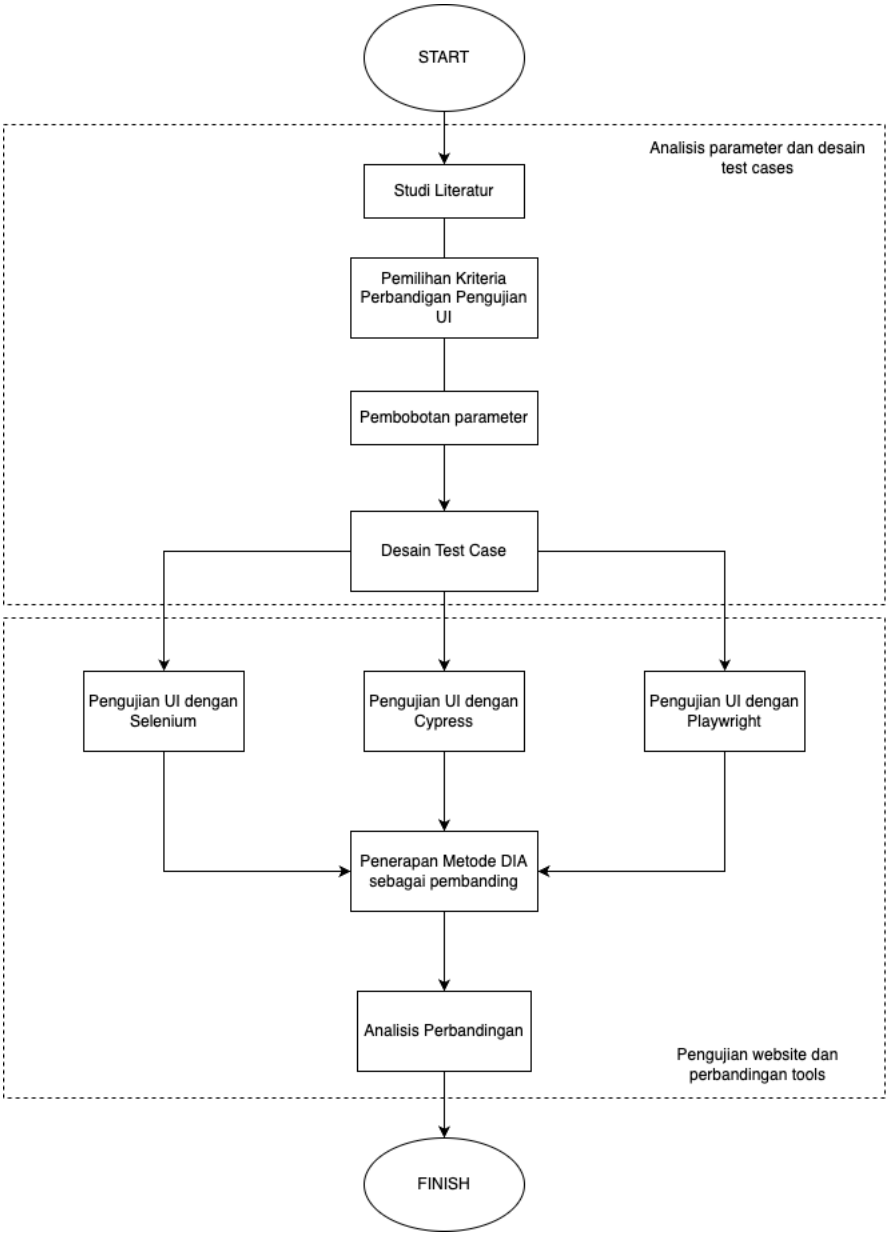
Keseluruhan telaah kajian pustaka ini tidak hanya memperluas pemahaman tentang penggunaan dan efektivitas alat pengujian otomatis dalam konteks yang berbeda, tetapi juga menegaskan pentingnya adaptasi metodologi pengujian untuk mendukung kebutuhan spesifik dari platform dan teknologi yang ditargetkan.

BAB III
METODOLOGI PENELITIAN

Pada Bab 3 ini, akan dijelaskan secara rinci mengenai metodologi penelitian yang digunakan dalam studi ini. Penelitian ini mengikuti alur proses yang digambarkan dalam diagram alur pada gambar 3.1, yang terdiri dari dua tahapan besar. Tahap pertama yakni Analisis parameter dan desain *test cases*, yang mana dilakukan studi literatur untuk mengumpulkan informasi terkait parameter dan desain *test case* yang relevan dengan pengujian UI. Selanjutnya, dilakukan pemilihan parameter untuk perbandingan pengujian UI serta pembobotan parameter-parameter yang telah dipilih. Setelah itu, *test case* dirancang berdasarkan parameter dan parameter yang telah ditentukan. Bab ini akan membahas setiap tahap tersebut secara detail, namun pembahasan hanya akan sampai pada tahap desain *test case* sesuai dengan alur proses yang digambarkan. Tahap selanjutnya yakni pengujian *website* dan perbandingan *tools* akan dibahas pada bab berikutnya.

3.1 Alur Proses

Penelitian ini dilakukan sesuai dengan metode penelitian yang telah di tentukan. Tahap pertama adalah analisis parameter dan desain *test cases*. Tahap kedua adalah melakukan pengujian *website* menggunakan ketiga alat dan melakukan perbandingan dengan metode yang sudah ditentukan seperti pada gambar 3.1.



Gambar 3.1 Alur Proses

3.2 Studi Literatur

Dalam melakukan penelitian ini, penulis membutuhkan bahan, referensi, informasi, dan data untuk mendukung pembahasan penelitian yang sedang dilakukan. Dengan mengumpulkan data dan informasi yang relevan dengan topik penelitian, diharapkan dapat memperkuat kevalidan dan kebenaran hasil penelitian. Melalui studi literatur, yaitu dengan membaca dan mempelajari penelitian serupa yang telah dilakukan sebelumnya, diharapkan dapat menjadi sumber referensi, informasi, dan dasar teoritis bagi penulis. Literatur yang diacu meliputi jurnal-jurnal baik nasional maupun internasional.

3.3 Pemilihan Parameter Perbandingan Pengujian

Pemilihan parameter menjadi salah satu hal yang paling diprioritaskan penulis dalam melakukan penelitian ini. Menentukan parameter yang didapatkan dari hasil studi literatur ini akan membantu mempertajam dan juga memperkuat keabsahan dari perbandingan kinerja pada pembahasan penelitian yang sedang dilakukan. Maka dari itu penulis melihat dari penelitian sebelumnya dan menekankan untuk penelitian ini pada tahap pemilihan parameter perbandingan pengujian, dengan menyajikan tiga parameter perbandingan yang mana dari masing-masing tiga parameter tersebut terdapat beberapa sub parameter.

Penulis menentukan parameter dan sub parameter berdasarkan studi literatur yang telah dipelajari yang mana menyebutkan bahwa parameter atau parameter dalam memilih *test automation framework* ada tiga. Parameter pertama adalah *Technical Economic View* yaitu kemudahan untuk penulis mengklasifikasikan *automation testing framework* berdasarkan aspek teknis dan ekonomi (Abdulwareth & Al-Shargabi, 2021). Terdapat juga Sub Parameter pada *Technical economic view* yaitu:

a. *Scripting Language*: Semakin banyak pilihan Bahasa pemrograman yang bisa digunakan dalam satu *framework*, semakin baik fleksibilitas *framework* tersebut (Pelivani dkk., 2021).
b. *Cost*: Biaya yang dikeluarkan untuk melakukan pengujian atau menggunakan *framework* tersebut, apakah gratis ataukah berbayar (Pelivani dkk., 2021).
c. *Product Support*: Mempunyai komunitas yang besar dan aktif dengan dokumentasi dan sumber daya pendukung yang baik (Pelivani dkk., 2021).

Parameter kedua adalah *Testing Process View* yaitu parameter saat penulis mengklasifikasikan *automation testing framework* berdasarkan proses yang dialami dan seperti apa proses yang dirasakan penguji dan penulis saat mencoba alat tersebut yang mencakup hasil eksekusi, seberapa sulit untuk melakukan *setup* serta proses pembuatan *code* nya (Abdulwareth & Al-Shargabi, 2021). Terdapat juga Sub Parameter pada *Testing Process View* yaitu:

a. *Easy of setup*: Kemudahan saat proses instalasi atau konfigurasi awal, dan juga *setup* pada *framework* tersebut, tidak memerlukan banyak *tools* untuk terintegrasi (Pelivani dkk., 2021).
b. *Script creation time*: Seberapa banyak memakan waktu saat membuat *script* atau *code* pengujian pada *framework* tersebut (Pelivani dkk., 2021).

c. Dokumentasi hasil *report*: Apakah terdapat dokumentasi dari hasil *report* pengujian pada *framework* tersebut ataukah sudah *built in* atau perlu tambahan *library tools* lagi (Melia & Putra, 2023).
d. Waktu hasil eksekusi: Seberapa banyak memakan waktu saat menjalankan pengujian dari *script* sesuai *test case* yang sudah dibuat (Melia & Putra, 2023).
e. *Inspection Element*: Apakah ada kemudahan untuk mencari atau menemukan *element (locator)* pada objek yang akan diuji pada *framework* tersebut (Merina dkk., 2018).

Parameter ketiga adalah *Quality View* yaitu salah satu aspek penting dalam pengujian pada *automation testing framework* (Abdulwareth & Al-Shargabi, 2021). Berdasarkan model kualitas perangkat lunak, terdapat berbagai karakteristik yang umumnya diidentifikasi pada *Quality View* yaitu:

a. *Usability*: Mencakup *Understandability, Learnability, Operability, Attractiveness, Usability compliance* pada *framework* tersebut atau secara mudahnya adalah *System Usability Scale* (SUS) (Abdulwareth & Al-Shargabi, 2021). Berikut merupakan pertanyaan dari *System Usability Scale* (SUS)
    1. Saya berpikir akan menggunakan sistem ini lagi
    2. Saya merasa sistem ini rumit untuk digunakan
    3. Saya merasa sistem ini mudah digunakan
    4. Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini
    5. Saya merasa fitur-fitur sistem ini berjalan dengan semestinya
    6. Saya merasa ada banyak hal yang tidak konsisten (tidak serasi pada sistem ini)
    7. Saya merasa orang lain akan memahami cara menggunakan sistem ini dengan cepat
    8. Saya merasa sistem ini membingungkan
    9. Saya merasa tidak ada hambatan dalam menggunakan sistem ini
    10. Saya perlu membiasakan diri terlebih dahulu sebelum menggunakan sistem ini
b. *Portability*: Apakah *framework* bisa *cross platform* atau tidak (Abdulwareth & Al-Shargabi, 2021).

3.4 Pembobotan Parameter dan Nilai

Pada tahap ini penulis akan melakukan pembobotan parameter. Penulis menentukan bobot untuk parameter dan sub parameter, yang akan digunakan untuk membandingkan efektivitas berbagai *framework* menggunakan metode *The Distance to The Ideal Alternative*

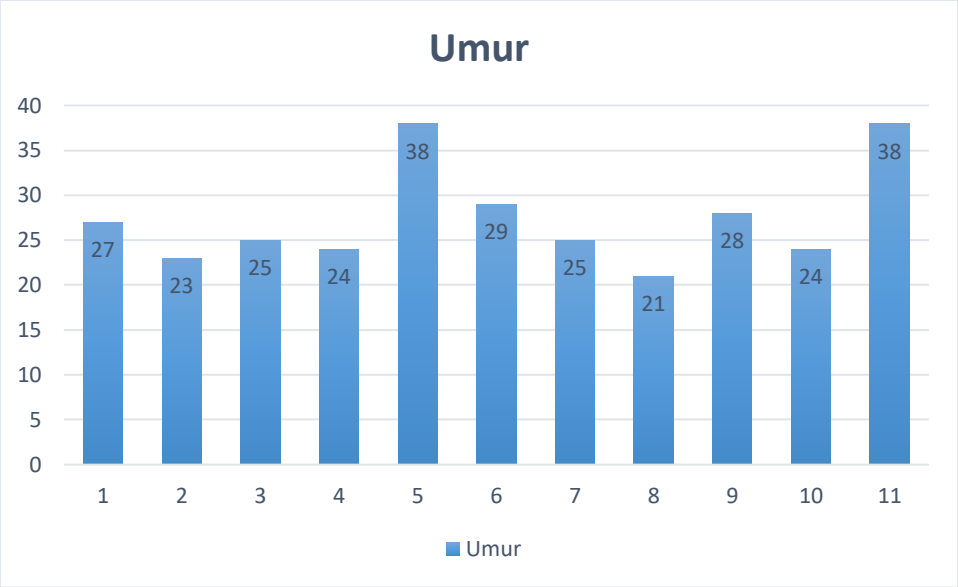
(DIA) Bobot tersebut ditetapkan berdasarkan pengaruhnya dalam mengukur kinerja dari *test automation framework*. Dalam proses penetapan bobot ini, penulis menggunakan prinsip-prinsip umum *Multiple Attribute Decision Making* (MADM). Perbedaan nilai bobot tidak mempengaruhi hasil analisis secara keseluruhan bobot hanya digunakan untuk menilai kepentingan relatif dari setiap parameter, dengan bobot yang lebih tinggi menunjukkan pengaruh yang lebih besar terhadap hasil akhir. Sesuai dengan ketentuan metode DIA, total semua bobot harus sama dengan 1.

Pembobotan parameter dalam penelitian ini dilakukan secara empiris melalui distribusi kuesioner yang ditujukan kepada para *software tester* dan *programmer* profesional. Kuesioner ini dirancang untuk mendapatkan pemahaman yang lebih dalam mengenai prioritas penggunaan dan efektivitas berbagai parameter dalam *test automation framework*.

3.4.1 Demografi Responden

Data demografi dari 11 responden yang terlibat dalam survei ini memberikan konteks yang berharga mengenai keberagaman latar belakang dan pengalaman mereka:

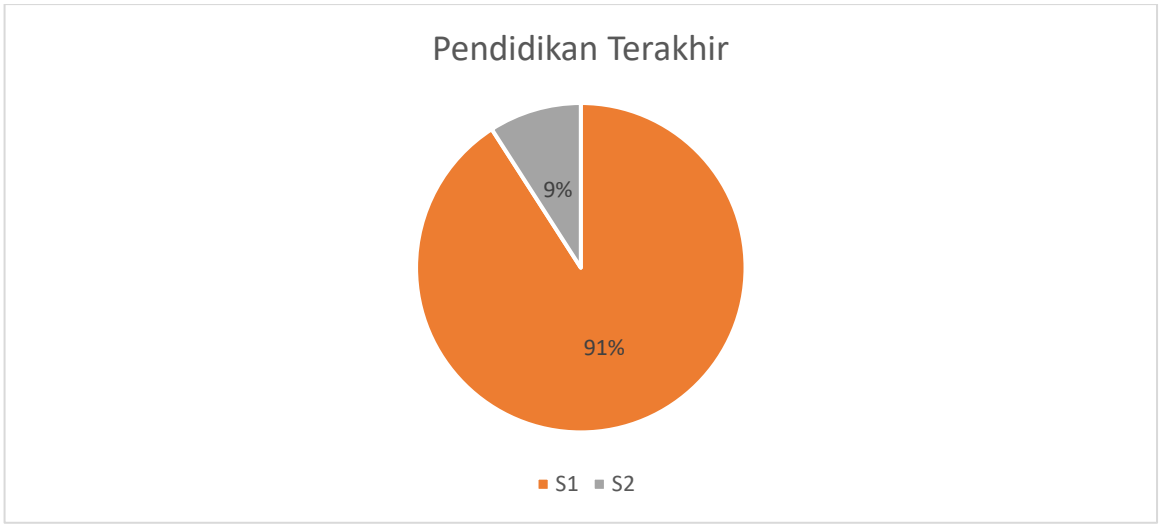
a. Umur



Gambar 3.2 Grafik umur tiap responden

Sumbu x atau nilai 1 – 11 merupakan jumlah responden pada grafik umur. Distribusi umur responden bervariasi, dengan sebagian besar berada di kisaran 21 hingga 38 tahun, mencerminkan keaktifan generasi muda dalam industri pengembangan perangkat lunak.

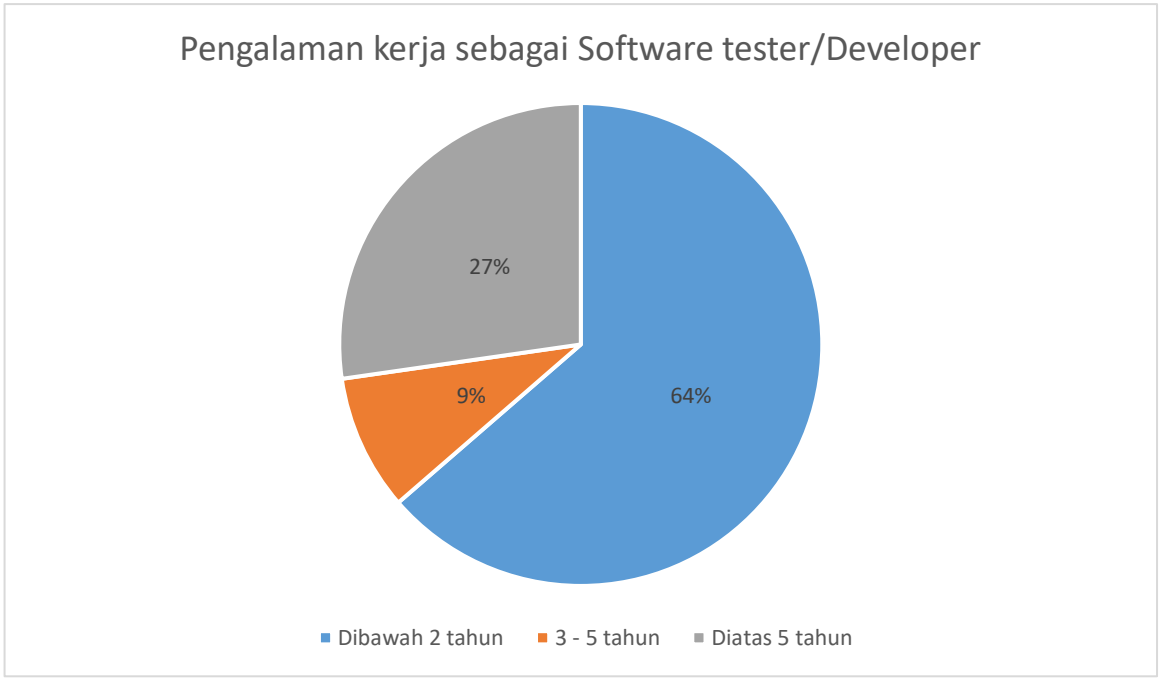
b. Pendidikan Terakhir



Gambar 3.3 Grafik pendidikan terakhir responden

Sebagian besar responden (91%) memiliki gelar sarjana (S1), dan adapula responden yang memiliki gelar magister (S2) sebanyak (1%).

c. Pengalaman Kerja



Gambar 3.4 Grafik pengalaman kerja responden

Responden menunjukkan variasi pengalaman yang luas, mulai dari mereka yang memiliki pengalaman di bawah dua tahun hingga yang memiliki pengalaman lebih dari lima tahun. Variasi ini memungkinkan penelitian untuk memperoleh perspektif yang kaya dan

beragam tentang penggunaan dan efektivitas *framework* pengujian, mencerminkan berbagai tingkat keahlian dan pendekatan dalam pengujian perangkat lunak.

d. *Framework testing* dan Bahasa pemrograman yang pernah digunakan

Hasil kuesioner mengungkap variasi kemahiran dalam bahasa pemrograman dan penggunaan berbagai *framework testing* di antara sekelompok *software tester* dan *developer*. JavaScript dan Java dominan dikuasai oleh responden, dengan penguasaan 63,6% untuk JavaScript dan 54,5% untuk Java, sementara Python juga menunjukkan penguasaan yang signifikan (36,4%). Bahasa lain seperti TypeScript, Ruby, Groovy, PHP, Php dengan golang, dan C++ masing-masing hanya dikuasai oleh satu responden. Dalam konteks penggunaan *framework testing*, Selenium menjadi yang paling sering digunakan, dengan beberapa responden juga menggunakan kombinasi alat seperti Selenium dengan Cypress, WebDriver.io, Mocha, Chai, dan Katalon, serta Go Unit Test, dan kombinasi Selenium dengan Postman, Gradle, Katalon Studio, dan Serenity. Kuesioner ini menonjolkan keberagaman keterampilan pemrograman dan preferensi kuat terhadap Selenium sebagai alat pengujian yang terpercaya dan banyak diadopsi dalam industri pengujian software.

3.4.2 Parameter *Technical Economic View*

Dari hasil kuesioner yang telah disebar selanjutnya, dihitung nilai bobot untuk sub parameter. Untuk menentukan bobot yakni dengan menghitung jumlah responden yang memilih sub parameter sebagai aspek paling penting, lalu dibagi dengan jumlah responden keseluruhan, sesuai rumus berikut:

$$Pembobotan = \frac{Responden\ yang\ memilih\ sub\ parameter}{Jumlah\ responden}$$

Tabel 3.1 Bobot pada parameter *Technical Economic View*

Sub Parameter	Jumlah responden	Tipe	Jenis	Bobot
Scripting Language	5 orang	*Benefit*	Numerik	0.46
Cost	4 orang	*Cost*	Kategori	0.36
Product Support	2 orang	*Benefit*	Kategori	0.18

Pembobotan nilai sub parameter *Scripting Language*

Berikut pada Tabel 3.2 adalah bobot untuk nilai dari sub parameter *Scripting language* yang nantinya akan memudahkan penulis dalam pembuatan matriks keputusan yang mana terdapat dalam ketentuan metode DIA.

Tabel 3.2 Pembobotan nilai sub parameter *Scripting Language*

Jumlah bahasa	Rentang nilai
$5 \leq x$	1
$4 \leq x < 5$	0.75
$3 \leq x < 4$	0.5
$2 \leq x < 3$	0.25
$1 \leq x < 2$	0

Pembobotan nilai sub parameter *Cost*

Tabel 3.3 Pembobotan nilai sub parameter *Cost*

Sub parameter	Rentang nilai	
Cost	*Framework* yang gratis (tidak menggunakan biaya) = 1	*Framework* yang berbayar = 0

Bobot nilai untuk sub parameter *cost* adalah 1 untuk *framework* yang gratis (tidak menggunakan biaya) dan 0 untuk *framework* yang berbayar.

Pembobotan nilai sub parameter *Product Support*

Tabel 3.4 Pembobotan nilai sub parameter *Product Support*

Sub parameter	Rentang nilai	
Product Support	*framework* yang memiliki komunitas yang besar dan aktif, serta sumber daya pendukung yang baik = 1	*framework* yang kurang dalam support terhadap *user* sulit nya mendapat sumber daya pendukung yang baik = 0

Bobot nilai untuk sub parameter *cost* adalah 1 untuk *framework* yang memiliki komunitas yang besar dan aktif, serta sumber daya pendukung yang baik dan 0 untuk *framework* yang kurang dalam *support* terhadap *user* sulit nya mendapat sumber daya pendukung yang baik.

3.4.3 Parameter *Testing Process View*

Tabel 3.5 Bobot pada parameter Testing Process View

Sub Parameter	Jumlah responden	Tipe	Jenis	Bobot
Easy of Setup	3 orang	*Benefit*	Kategori	0.27
Script creation time	1 orang	*Cost*	Numerik	0.09
Dokumentasi hasil *report*	5 orang	*Benefit*	Kategori	0.46
Waktu hasil eksekusi	1 orang	*Cost*	Numerik	0.09
Inspection element	1 orang	*Benefit*	Kategori	0.09

Pembobotan nilai sub parameter *Easy of Setup*

Tabel 3.6 Pembobotan nilai sub parameter *Easy of Setup*

Sub parameter	Rentang nilai	
Easy of Setup	*Framework* yang mudah dilakukan *setup* = 1	*Framework* yang sulit dan banyak konfigurasi = 0

Bobot nilai untuk sub parameter *Easy of Setup* adalah 1 untuk *framework* yang mudah untuk dilakukan *setup* dan 0 untuk *framework* yang sulit dan banyak konfigurasi.

Pembobotan nilai sub parameter *Script Creation Time*

Berikut pada tabel 3.7 adalah bobot untuk nilai dari sub parameter *Script Creation Time*.

Tabel 3.7 Pembobotan nilai sub parameter *Script Creation Time*

Rentang waktu (jam)	Rentang nilai
$x \leq 2$	1
$3 \leq x < 4$	0.75
$4 \leq x < 5$	0.5
$5 \leq x < 6$	0.25
$x \geq 7$	0

Pembobotan nilai sub parameter Dokumentasi hasil report

Tabel 3.8 Pembobotan nilai sub parameter dokumentasi hasil report

Sub parameter	Rentang nilai	
Dokumentasi hasil *report*	Terdapat dokumentasi hasil *report* atau sudah *built in* = 1	Tidak terdapat dokumentasi atau masih perlu menambahkan library *tools* lagi = 0

Bobot nilai untuk sub parameter Dokumentasi hasil report adalah 1 untuk *framework* yang terdapat dokumentasi dari hasil *report* pengujian pada *framework* tersebut ataukah sudah *built in* atau perlu tambahan *library tools* lagi dan 0 untuk *framework* yang tidak terdapat dokumentasi atau masih harus perlu menambahkan *library tools* lagi.

Pembobotan nilai sub parameter Waktu hasil eksekusi

Berikut pada tabel 3.9 adalah bobot untuk nilai dari sub parameter Waktu hasil eksekusi.

Tabel 3.9 Pembobotan nilai sub parameter waktu hasil eksekusi

Rentang waktu (detik)	Rentang nilai
$x \leq 23$	1
$23 \leq x \leq 24$	0.75
$24 \leq x \leq 25$	0.5
$25 \leq x \leq 26$	0.25
$x \geq 28$	0

Pembobotan nilai sub parameter *Inspection Element*

Tabel 3.10 Pembobotan nilai sub parameter *Inspection Element*

Sub parameter	Rentang nilai	
Inspection Element	Mudah untuk melakukan *inspection element* = 1	Masih manual dalam *inspection element* = 0

Bobot nilai untuk sub parameter *Inspection Element* adalah 1 untuk *framework* yang memiliki kemudahan untuk mencari atau menemukan *element (locator)* pada objek yang akan diuji dan 0 untuk *framework* yang masih manual dalam mencari atau menemukan *element* (*locator*) pada objek yang akan diuji.

3.4.4 Parameter *Quality View*

Tabel 3.11 Bobot pada Parameter *Quality View*

Sub Parameter	Jumlah responden	Tipe	Jenis	Bobot
Usability	10 orang	*Benefit*	Numerik	0.91
Portability	1 orang	*Benefit*	Kategori	0.09

Pembobotan nilai sub parameter *Usability*

Berikut adalah bobot untuk nilai dari sub parameter *Usability*, yang mana menurut (Brooke, 2013) standar rata-rata SUS *Score* adalah 68, berikut pada tabel 3.12 merupakan penjabaran SUS *Score* yang sudah dimodifikasi ke bentuk rentang nilai pembobotan.

Tabel 3.12 Pembobotan nilai sub parameter *Usability*

Rentang SUS *Score*	Rentang nilai
$x \geq 80.3$	1
$68 \leq x \leq 80.3$	0.75
68	0.5
$51 \leq x \leq 68$	0.25
$x \leq 51$	0

Pembobotan nilai sub parameter *Portability*

Tabel 3.13 Pembobotan nilai sub parameter *Portability*

Sub parameter	Rentang nilai	
Portability	Dapat dijalankan secara *cross platform* = 1	Tidak bisa *cross platform* = 0

Bobot nilai untuk sub parameter *Portability* adalah 1 untuk *framework* yang dapat dijalankan secara *cross platform* dan 0 untuk *framework* tidak bisa.

3.4.5 Pembobotan Seluruh Parameter

Tabel 3.14 Pembobotan tiap parameter

Parameter	Peringkat 1	Peringkat 2	Peringkat 3	Bobot
Technical economic view	3 orang	5 orang	3 orang	0.27
Testing Process view	7 orang	3 orang	1 orang	0.64
Quality view	1 orang	3 orang	7 orang	0.09

Telah didapat hasil bahwa bobot untuk parameter *Technical economic view* adalah 0.27 lalu untuk parameter *Testing process view* adalah 0.64 dan yang terakhir parameter *Quality view* adalah 0.09.

3.5 Desain *Test Case*

Pada tahap ini, penulis mendesain *test case* yang akan diujikan pada masing-masing fitur, *test case* yang penulis desain berupa *test case* positif maupun *test case negative*. Berikut merupakan desain *test case* yang penulis buat.

a. Menampilkan halaman utama

Tabel 3.15 Test Case Halaman Utama

Name	TC-01 : Verifikasi bahwa sistem berhasil menampilkan
Requirement	Koneksi *internet* yang stabil
Preconditions	Sudah membuka situs https://iconhub.io
Steps	1. Membuka situs iconhub.io
Expected Results	Berhasil menampilkan halaman utama

b. Membuat akun baru

Tabel 3.16 Test Case Membuat Akun Baru

Name	TC-02 : Verifikasi bahwa *user* dapat membuat akun baru
Requirement	Koneksi *internet* yang stabil
Preconditions	Sudah membuka situs https://iconhub.io
Steps	1. Klik tombol login 2. Klik tombol register 3. Isi bagian email “keepsfly@gmail.com” 4. Isi bagian password “cobacoba123” 5. Isi bagian username “akuncoba1” 6. Klik tombol register

c. Masuk ke akun

Tabel 3.17 Test Case Masuk ke Akun

Name	TC-03-1 : Verifikasi bahwa *user* dapat masuk ke akun
Requirement	Koneksi *internet* yang stabil
Preconditions	Sudah membuka situs https://iconhub.io
Steps	1. Klik tombol login 2. Isi bagian email “keepsfly@gmail.com” 3. Isi bagian password “cobacoba123” 4. Klik tombol login
Expected Results	Berhasil masuk ke akun dan tertampil halaman utama

Tabel 3.18 Test Case Masuk ke Akun gagal

Name	TC-03-2 : Verifikasi bahwa *user* tidak dapat masuk ke akun
Requirement	Koneksi *internet* yang stabil
Preconditions	Sudah membuka situs https://iconhub.io
Steps	1. Klik tombol login 2. Isi bagian email “keepsflyyyy@gmail.com” 3. Isi bagian password “cobacoba12345” 4. Klik tombol login
Expected Results	Tidak berhasil masuk ke akun dan tertampil halaman utama

d. Memilih *icon style*

Tabel 3.19 Test Case Memilih *Icon Style*

Name	TC-04 : Verifikasi bahwa icon berubah sesuai *icon style* yang dipilih
Requirement	Koneksi *internet* yang stabil
Preconditions	Sudah membuka situs https://iconhub.io
Steps	1. Pilih *icon style line*

	2. Pastikan icon berubah 3. Pilih *icon style glyph* 4. Pastikan *icon* berubah 5. Pilih *icon style line color* 6. Pastikan *icon* berubah 7. Pilih *icon style flat color* 8. Pastikan *icon* berubah 9. Pilih *icon style flat line* 10. Pastikan *icon* berubah 11. Pilih *icon style multi color* 12. Pastikan *icon* berubah
Expected Results	Berhasil memilih *icon style* dan *icon* sepenuhnya berubah sesuai pilihan

e. Memilih *icon size*

Tabel 3.20 Test Case Memilih *Icon Size*

Name	TC-05 : Verifikasi bahwa icon berubah sesuai *icon size* yang dipilih
Requirement	Koneksi *internet* yang stabil
Preconditions	Sudah membuka situs https://iconhub.io
Steps	1. Pilih *icon size* 16 2. Pastikan *icon* berubah 3. Pilih *icon size* 24 4. Pastikan *icon* berubah 5. Pilih *icon size* 32 6. Pastikan *icon* berubah 7. Pilih *icon size* 48 8. Pastikan *icon* berubah 9. Pilih *icon size* 64 10. Pastikan *icon* berubah 11. Pilih *icon size* 96 13. Pastikan *icon* berubah

Expected Results	Berhasil merubah *icon size* dan *icon* sepenuhnya berubah sesuai pilihan

f. Memilih *stroke size*

Tabel 3.21 Test case memilih *stroke size*

Name	TC-06 : Verifikasi bahwa icon berubah sesuai *icon stroke size* yang dipilih
Requirement	Koneksi *internet* yang stabil
Preconditions	Sudah membuka situs https://iconhub.io
Steps	1. Pilih *icon stroke size* 0.5 2. Pastikan *icon* berubah 3. Pilih *icon stroke size* 1 4. Pastikan *icon* berubah 5. Pilih *icon stroke size* 1.5 6. Pastikan *icon* berubah 7. Pilih *icon stroke size* 2 8. Pastikan *icon* berubah 9. Pilih *icon stroke size* 2.5 10. Pastikan *icon* berubah 11. Pilih *icon stroke size* 3 12. Pastikan *icon* berubah
Expected Results	Berhasil merubah *icon stroke size* dan *icon* sepenuhnya berubah sesuai pilihan

g. Memilih *icon*

Tabel 3.22 Test Case Memilih *Icon*

Name	TC-07 : Verifikasi bahwa *user* dapat memilih *multiple icon*
Requirement	Koneksi *internet* yang stabil
Preconditions	Sudah membuka situs https://iconhub.io
Steps	1. Pilih icon “1st-place”

	2. Pilih icon “2k” 3. Pilih icon “2nd-place” 4. Pilih icon “30fps” 5. Pilih icon “3rd-place”
Expected Results	Berhasil memilih icon dan tertampil jumlah icon yang dipilih

h. Mengunduh *icon*

Tabel 3.23 Test Case Mengunduh Icon

Name	TC-08 : Verifikasi bahwa user dapat mengunduh *icon*
Requirement	Koneksi *internet* yang stabil
Preconditions	Sudah membuka situs https://iconhub.io
Steps	1. Klik tombol login 2. Isi bagian email “keepsfly@gmail.com” 3. Isi bagian password “cobacoba123” 4. Klik tombol login 5. Pilih icon “1st-place” 6. Pilih icon “2k” 7. Pilih icon “2nd-place” 8. Pilih icon “30fps” 9. Pilih icon “3rd-place” 10. Klik *button download icon* 6. Klik *button* “svg”
Expected Results	Berhasil mengunduh *icon*

i. Memilih *icon* berdasarkan kategori

Tabel 3.24 Test Case pilih *icon* berdasarkan kategori

Name	TC-09 : Verifikasi bahwa user dapat memilih *icon* berdasarkan kategori
Requirement	Koneksi *internet* yang stabil

Preconditions	Sudah membuka situs https://iconhub.io
Steps	1. Klik dropdown kategori "*Pick from our icon collection*" 2. Pilih kategori "*Brand*" 3. Pilih icon "disney-plus"
Expected Results	Berhasil memilih icon berdasarkan kategori yang diminta

BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini melanjutkan tahap berikutnya dari alur proses yaitu Pengujian *website* dan perbandingan *tools*, yang mana terdiri dari pengujian dari masing-masing *test automation framework* dan Penerapan metode DIA sebagai pembanding. Selain itu pada bab ini penulis juga akan membahas mengenai hasil analisis pengujian yang sudah melewati proses perbandingan dengan metode DIA.

4.1 Pengujian UI dengan Selenium

Untuk melakukan pengujian menggunakan Selenium, penulis harus melakukan *setup* terlebih dahulu pada *framework* Selenium. Selenium bisa dijalankan dengan beberapa bahasa seperti Java, C#, Ruby, Python dan JavaScript, penulis memilih menggunakan bahasa Java karena sebelumnya penulis telah mencoba semua Bahasa namun mendapati *error* dalam pengujian, selain itu penulis mempunyai pengalaman menggunakan Selenium dengan Bahasa Java yang mana dapat memudahkan penulis saat melakukan pengujian.

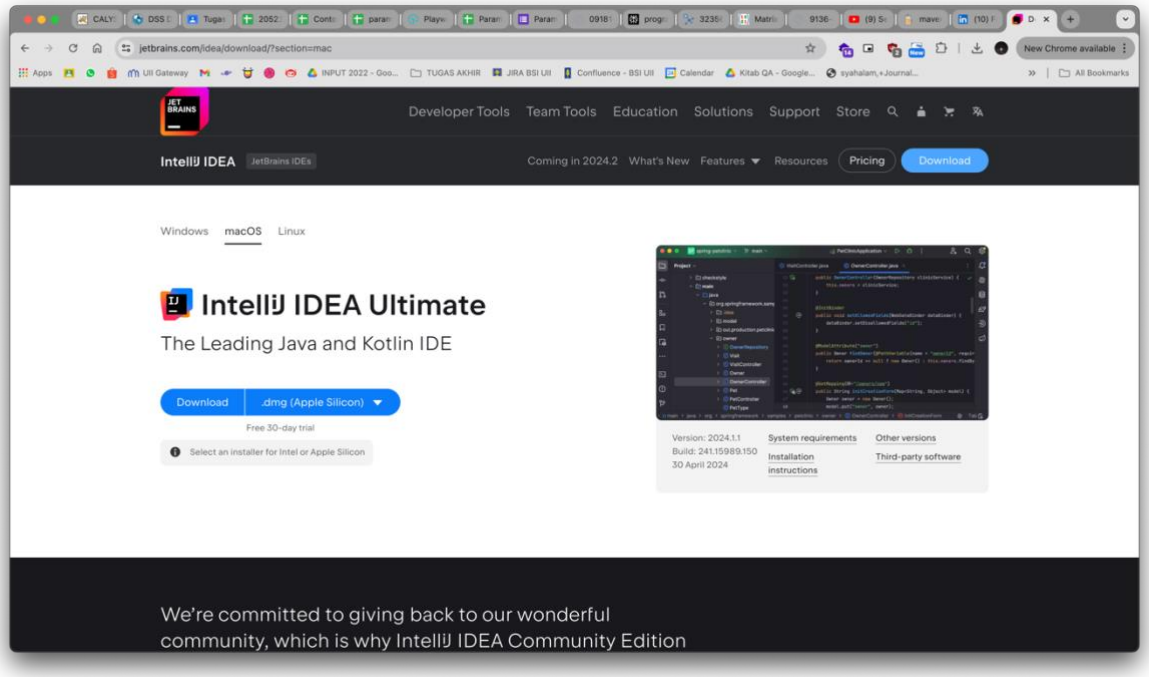
4.1.1 Setup *Framework* Selenium

Hal yang dilakukan pertama kali penulis yaitu meng-*install* Maven dengan menggunakan bantuan Brew untuk melakukan *setup* awal.

```
Brew install maven
```

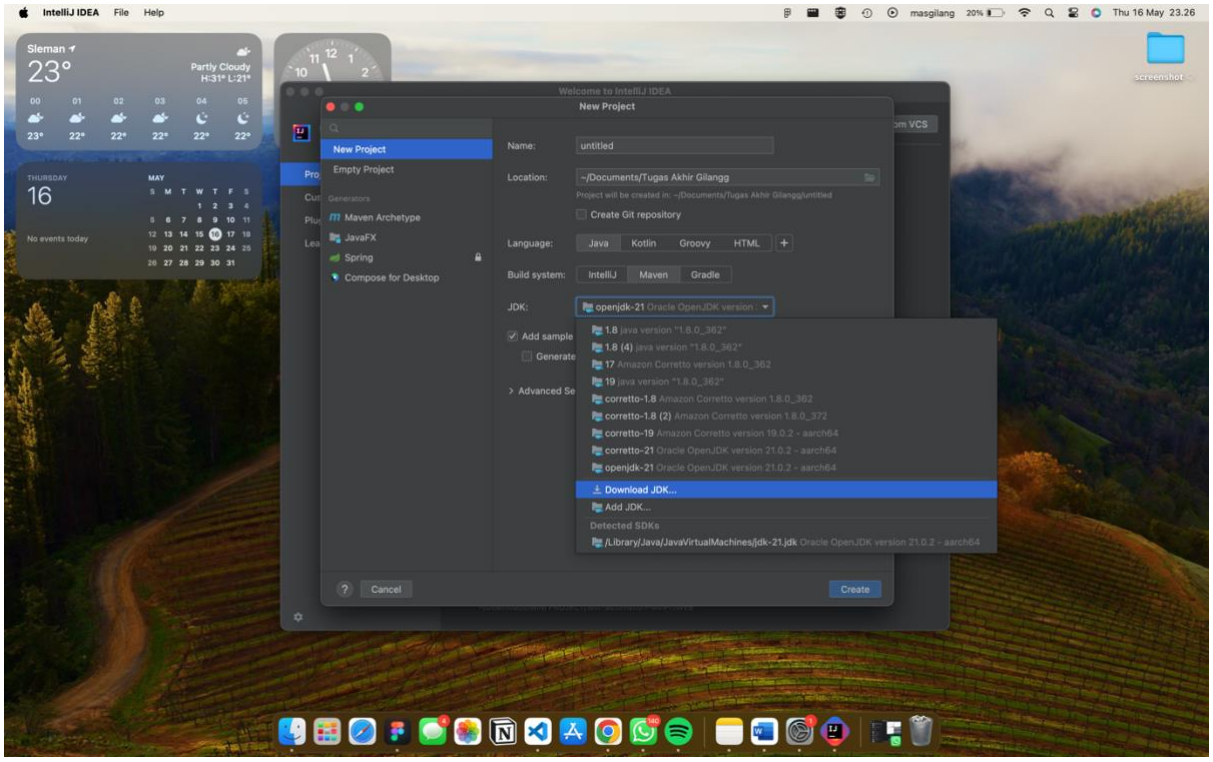
Gambar 4.1 Menginstall maven

Lalu selanjutnya meng-*install* IDE yang akan penulis gunakan yaitu IntelliJ IDEA, penulis mengunduh nya dari laman resmi dari IntelliJ IDEA.



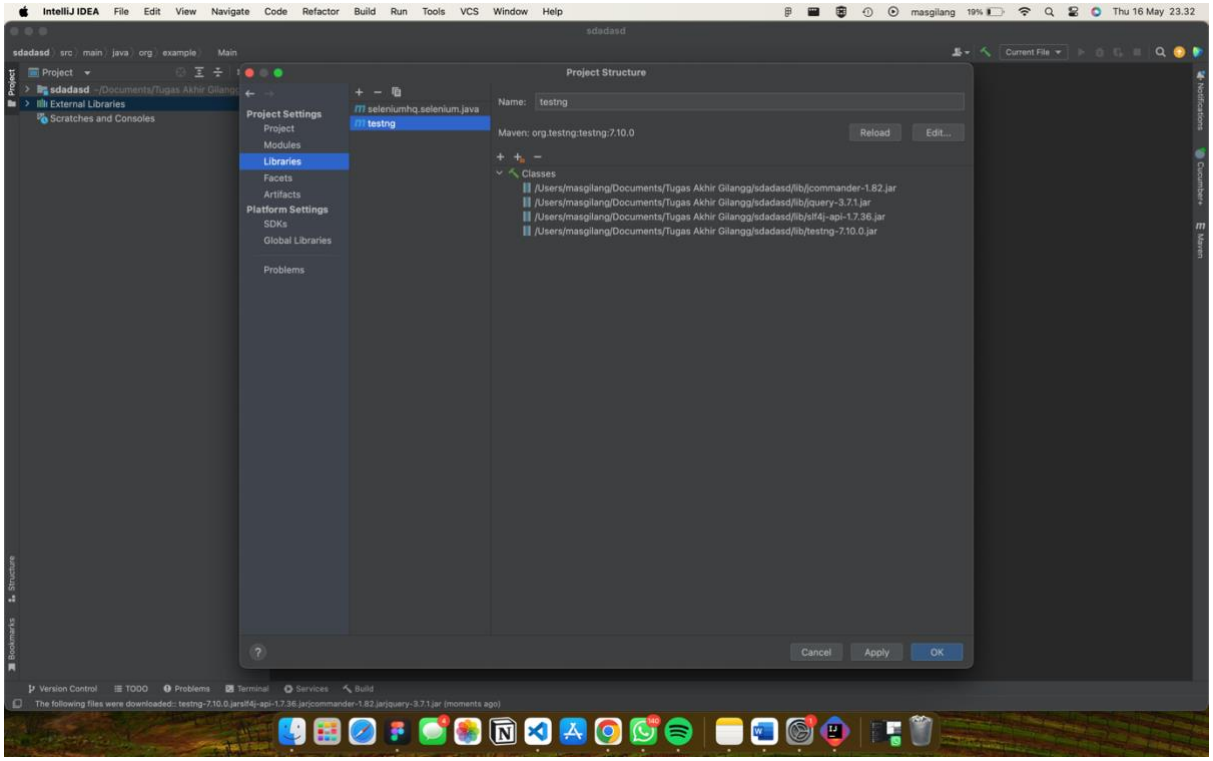
Gambar 4.2 Meng-*install* IDE yang akan dipakai

Setelah itu penulis melakukan *setup* pada IntelliJ IDEA dengan membuat *project* baru dengan memilih *build system by* Maven dan memilih JDK jika sudah meng-*install* atau bisa men*download* langsung pada *setting project* seperti gambar 4.3 berikut.



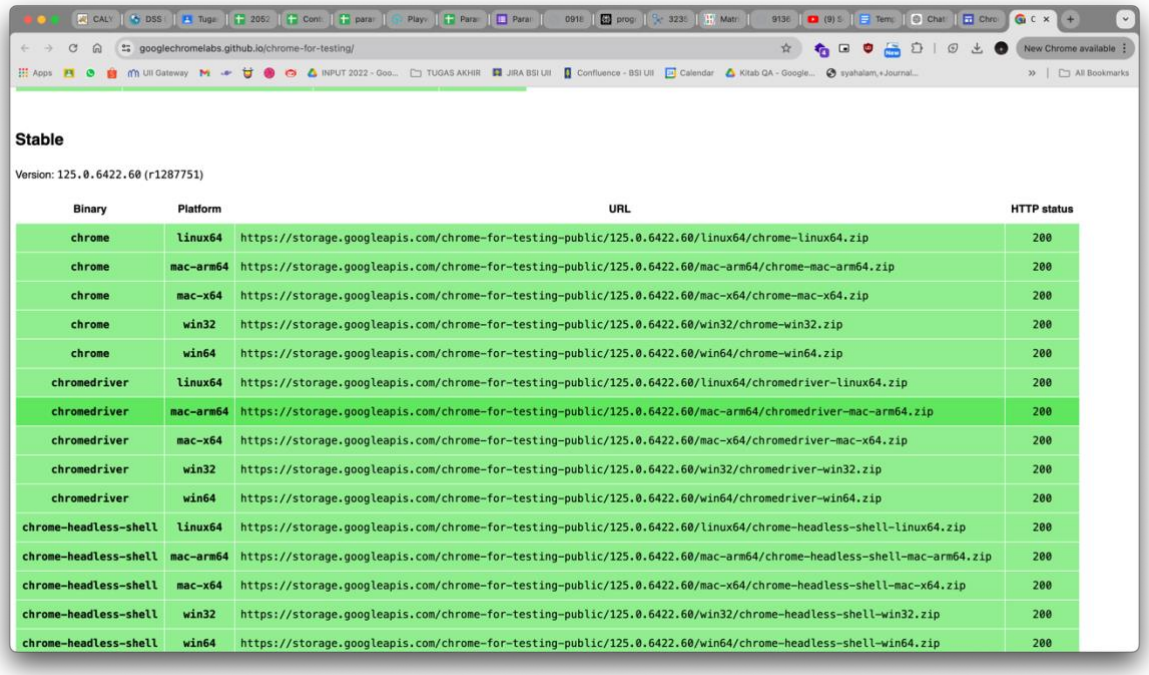
Gambar 4.3 Melakukan setting project baru

Setelah berhasil melakukan *setting project*, maka akan tertampil halaman awal pada IntelliJ lalu selanjutnya penulis melakukan *setup* pada *libraries* pada menu *setting project structure*. Disini penulis menggunakan *libraries* testng dan selenium.java dengan menggunakan versi terbaru seperti gambar 4.4 berikut.



Gambar 4.4 Melakukan *setup project structure*.

Langkah selanjutnya yang penulis lakukan adalah mengunduh Chrome Driver. Pada pengujian ini penulis menggunkan Google Chrome sebagai *browser* untuk melakukan pengujian dan ChromeDriver adalah alat yang digunakan untuk mengotomatisasi pengujian aplikasi *web* di *browser* Google Chrome. Ini merupakan bagian dari proyek Selenium, yang menyediakan *framework* untuk mengendalikan *browser* secara otomatis. ChromeDriver yang diunduh pun harus sesuai dengan versi Google Chrome pada saat proses pengujian.



googlechromelabs.github.io/chrome-for-testing/

Stable

Version: 125.0.6422.60 (r1287751)

Binary	Platform	URL	HTTP status
chrome	linux64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/linux64/chrome-linux64.zip	200
chrome	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/mac-arm64/chrome-mac-arm64.zip	200
chrome	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/mac-x64/chrome-mac-x64.zip	200
chrome	win32	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/win32/chrome-win32.zip	200
chrome	win64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/win64/chrome-win64.zip	200
chromedriver	linux64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/linux64/chromedriver-linux64.zip	200
chromedriver	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/mac-arm64/chromedriver-mac-arm64.zip	200
chromedriver	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/mac-x64/chromedriver-mac-x64.zip	200
chromedriver	win32	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/win32/chromedriver-win32.zip	200
chromedriver	win64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/win64/chromedriver-win64.zip	200
chrome-headless-shell	linux64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/linux64/chrome-headless-shell-linux64.zip	200
chrome-headless-shell	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/mac-arm64/chrome-headless-shell-mac-arm64.zip	200
chrome-headless-shell	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/mac-x64/chrome-headless-shell-mac-x64.zip	200
chrome-headless-shell	win32	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/win32/chrome-headless-shell-win32.zip	200
chrome-headless-shell	win64	https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.60/win64/chrome-headless-shell-win64.zip	200

Gambar 4.5 Mengunduh ChromeDriver

4.1.2 Menulis *Script Test*

Penulis menulis kode atau *script* untuk menjalankan pengujian fungsional otomatis berdasarkan *test case* yang telah dibuat.

a. Fungsi BeforeTest

```
    WebDriver driver;

    @BeforeTest public void init(){
        //initiate browser
        System.setProperty("webdriver.chrome.driver",
"/Users/masgilang/Downloads/chromedriver-mac-arm64/chromedriver");
        driver = new ChromeDriver();

        //go to homepage
        driver.navigate().to("https://iconhub.io");
        driver.manage().window().maximize();
}
```

Gambar 4.6 Fungsi *before test*

b. Menampilkan halaman utama

```
@Test
public void checkElement(){
```

```
Assert.assertEquals(driver.findElements(By.xpath("//a[contains(text(),
'Login')]")).size(), 1, "Menampilkan halaman utama");
}
```

Gambar 4.7 Menampilkan halaman utama

c. Membuat akun baru

```
@Test
public void signupWithValidAccount() {
        // Simulate generating random email using faker
        String randomEmail = "test@example.com"; // replace with actual
random email generation
        driver.findElement(By.cssSelector(".pr-2")).click();
        driver.findElement(By.cssSelector("body > div > div > div > form
> div:nth-child(10) > a")).click();
        driver.findElement(By.id("email")).sendKeys(randomEmail);
        driver.findElement(By.id("password")).sendKeys("cobacoba123");
        driver.findElement(By.id("fullname")).sendKeys("cobacobain");
        driver.findElement(By.cssSelector("body > div > div > div > form
> div.w-full.my-2 > button")).click();
        Assert.assertTrue(driver.findElement(By.cssSelector(".btn      >
.text-black")).getText().contains("cobacobain"));
}
```

Gambar 4.8 Membuat akun baru

d. Masuk ke akun berhasil

```
@Test
public void loginWithValidAccount() {
        driver.findElement(By.cssSelector(".pr-2")).click();

driver.findElement(By.id("email")).sendKeys("cobacoba4@gmail.com");
        driver.findElement(By.id("password")).sendKeys("cobacoba123");
        driver.findElement(By.cssSelector(":nth-child(8)               >
.btn")).click();
        Assert.assertTrue(driver.findElement(By.cssSelector(".btn      >
.text-black")).getText().contains("cobatestTA"));
}
```

Gambar 4.9 Masuk ke akun berhasil

e. Gagal masuk ke akun

```
@Test
public void loginWithInvalidAccount() {
driver.findElement(By.cssSelector(".pr-2")).click();

driver.findElement(By.id("email")).sendKeys("cobacoba56@gmail.com");
        driver.findElement(By.id("password")).sendKeys("cobacoba12345");
        driver.findElement(By.cssSelector(":nth-child(8)               >
.btn")).click();
        Assert.assertTrue(driver.findElement(By.cssSelector(".btn      >
.text-black")).getText().contains("cobatestTA"));
}
```

Gambar 4.10 Gagal masuk ke akun

f. Memilih *icon style*

```
@Test
private void selectingIconStyleAppliesCorrectStyle() {
// Define an array of style names
String[] styleNames = {"line", "glyph", "line-color", "flat-color", "flat-
line", "multi-color"};

// Iterate over each icon style
for (String styleName : styleNames) {
// Click on the icon style button
WebElement styleButton = driver.findElement(By.cssSelector("button[data-
name='" + styleName + "']"));
            styleButton.click();

// Check if the selected icon style is displayed after clicking
try {
WebElement  iconStyle  =  driver.findElement(By.cssSelector(".icon[data-
style='" + styleName + "']"));
    if (iconStyle.isDisplayed()) {
System.out.println("Icon style '" + styleName + "' is applied correctly.");
}   else {
System.out.println("Icon style '" + styleName + "' is applied correctly
but not showed");
}
            }
catch (org.openqa.selenium.NoSuchElementException e) {
```

```
System.out.println("Icon style '" + styleName + "' is NOT applied
correctly.");
            }
          }
    }
```

Gambar 4.11 Memilih *icon style*

g. Memilih *icon size*

```
@Test
public void selectingIconSizeAppliesCorrectStyle() {
        // Temukan elemen yang sesuai dengan selektor CSS dan klik
        driver.findElement(By.cssSelector("#sidebar > div > div:nth-
child(3) > div > div:nth-child(1) > div.flex.flex-wrap > div:nth-child(1)
> div")).click();
        // Verifikasi ukuran ikon yang diperbarui
        WebElement svgElement =
driver.findElement(By.cssSelector("svg[data-icon-name='1st-
place'][width='16'][height='16']"));
        Assert.assertEquals("16", svgElement.getAttribute("width"));
        Assert.assertEquals("16", svgElement.getAttribute("height"));

        // Simulate clicking to change the size to 24
        driver.findElement(By.cssSelector("#sidebar > div > div:nth-
child(3) > div > div:nth-child(1) > div.flex.flex-wrap > div:nth-child(2)
> div")).click();
        // Verify the updated icon size
        WebElement svgElement24 =
driver.findElement(By.cssSelector("svg[data-icon-name='1st-
place'][width='24'][height='24']"));
        Assert.assertEquals("24", svgElement24.getAttribute("width"));
        Assert.assertEquals("24", svgElement24.getAttribute("height"));

        // Simulate clicking to change the size to 32
        driver.findElement(By.cssSelector("#sidebar > div > div:nth-
child(3) > div > div:nth-child(1) > div.flex.flex-wrap > div:nth-child(3)
> div")).click();
        // Verify the updated icon size
        WebElement svgElement32 =
driver.findElement(By.cssSelector("svg[data-icon-name='1st-
place'][width='32'][height='32']"));
```

```
        Assert.assertEquals("32", svgElement32.getAttribute("width"));
        Assert.assertEquals("32", svgElement32.getAttribute("height"));

        // Simulate clicking to change the size to 48
        driver.findElement(By.cssSelector("#sidebar > div > div:nth-
child(3) > div > div:nth-child(1) > div.flex.flex-wrap > div:nth-child(4)
> div")).click();
        // Verify the updated icon size
        WebElement                    svgElement48                    =
driver.findElement(By.cssSelector("svg[data-icon-name='1st-
place'][width='48'][height='48']"));
        Assert.assertEquals("48", svgElement48.getAttribute("width"));
        Assert.assertEquals("48", svgElement48.getAttribute("height"));

        // Simulate clicking to change the size to 64
        driver.findElement(By.cssSelector("#sidebar > div > div:nth-
child(3) > div > div:nth-child(1) > div.flex.flex-wrap > div:nth-child(5)
> div")).click();
        // Verify the updated icon size
        WebElement                    svgElement64                    =
driver.findElement(By.cssSelector("svg[data-icon-name='1st-
place'][width='64'][height='64']"));
        Assert.assertEquals("64", svgElement64.getAttribute("width"));
        Assert.assertEquals("64", svgElement64.getAttribute("height"));

        // Simulate clicking to change the size to 96
        driver.findElement(By.cssSelector("#sidebar > div > div:nth-
child(3) > div > div:nth-child(1) > div.flex.flex-wrap > div:nth-child(6)
> div")).click();
        // Verify the updated icon size
        WebElement                    svgElement96                    =
driver.findElement(By.cssSelector("svg[data-icon-name='1st-
place'][width='96'][height='96']"));
        Assert.assertEquals("96", svgElement96.getAttribute("width"));
        Assert.assertEquals("96", svgElement96.getAttribute("height"));
}
```

Gambar 4.12 Memilih *icon size*

h. Memilih *stroke size*

```
@Test
```

```
public void selectingIconStrokeSize(){

        // Simulate clicking to change the stroke size to 0.5
        driver.findElement(By.cssSelector("#sidebar  >  div  >  div:nth-
child(3) > div > div:nth-child(2) > div.flex.flex-wrap > div:nth-child(1)
> div")).click();
        // Verifikasi ukuran stroke ikon yang diperbarui
        WebElement                      pathElement                     =
driver.findElement(By.cssSelector("path[data-name='primary']"));

        // Simulate clicking to change the stroke size to 1
        driver.findElement(By.cssSelector("#sidebar  >  div  >  div:nth-
child(3) > div > div:nth-child(2) > div.flex.flex-wrap > div:nth-child(2)
> div")).click();
        // Verify the stroke size has been updated to 1
        WebElement                      svgElement2                     =
driver.findElement(By.cssSelector("svg[data-icon-name='1st-place']"));

        // Simulate clicking to change the stroke size to 1.5
        driver.findElement(By.cssSelector("#sidebar  >  div  >  div:nth-
child(3) > div > div:nth-child(2) > div.flex.flex-wrap > div:nth-child(3)
> div")).click();
        // Verify the stroke size has been updated to 1.5
        WebElement                      svgElement3                     =
driver.findElement(By.cssSelector("svg[data-icon-name='1st-place']"));

        // Simulate clicking to change the stroke size to 2
        driver.findElement(By.cssSelector("#sidebar  >  div  >  div:nth-
child(3) > div > div:nth-child(2) > div.flex.flex-wrap > div:nth-child(4)
> div")).click();
        // Verify the stroke size has been updated to 2
        WebElement                      svgElement4                     =
driver.findElement(By.cssSelector("svg[data-icon-name='1st-place']"));

        // Simulate clicking to change the stroke size to 2.5
        driver.findElement(By.cssSelector("#sidebar  >  div  >  div:nth-
child(3) > div > div:nth-child(2) > div.flex.flex-wrap > div:nth-child(5)
> div")).click();
        // Verify the stroke size has been updated to 2.5
```

```
        WebElement                      svgElement5                     =
driver.findElement(By.cssSelector("svg[data-icon-name='1st-place']"));

        // Simulate clicking to change the stroke size to 3
        driver.findElement(By.cssSelector("#sidebar  >  div  >  div:nth-
child(3) > div > div:nth-child(2) > div.flex.flex-wrap > div:nth-child(6)
> div")).click();
        // Verify the stroke size has been updated to 3
        WebElement                      svgElement6                     =
driver.findElement(By.cssSelector("svg[data-icon-name='1st-place']"));
}
```

Gambar 4.13 Memilih *stroke size*

i. Memilih *icon*

```
@Test
public void selectingMultipleIcon(){
        // Daftar selector yang akan diklik
        String[] selectors = {
                "div:nth-child(1) > .relative > .absolute > .btn-select",
                "div:nth-child(2) > .relative > .absolute > .btn-select",
                "div:nth-child(3) > .relative > .absolute > .btn-select",
                "div:nth-child(4) > .relative > .absolute > .btn-select",
                "div:nth-child(5) > .relative > .absolute > .btn-select"
        };

        for (String selector : selectors) {
            // Klik elemen
            driver.findElement(By.cssSelector(selector)).click();

            // Dapatkan elemen target
            WebElement                      element                     =
driver.findElement(By.cssSelector("span[x-text=\"selectedLine.length  ===
0 ?'' : selectedLine.length\"]"));

            // Dapatkan konten teks dari elemen target
            String textContent = element.getText().trim();

            // Verifikasi bahwa konten teks cocok dengan nilai yang
diharapkan (yang merupakan indeks loop ditambah 1)
```

```
            Assert.assertEquals(textContent,
Integer.toString(Arrays.asList(selectors).indexOf(selector) + 1));
        }
    }
```

Gambar 4.14 Memilih *icon*

j. Mengunduh *icon*

```
@Test
public void selectingAndDownloadingIcon() {
        //Login first
        driver.findElement(By.cssSelector(".pr-2")).click();

driver.findElement(By.id("email")).sendKeys("cobacoba4@gmail.com");
        driver.findElement(By.id("password")).sendKeys("cobacoba123");
        driver.findElement(By.cssSelector(":nth-child(8)               >
.btn")).click();
        Assert.assertTrue(driver.findElement(By.cssSelector(".btn      >
.text-black")).getText().contains("cobatestTA"));
        // wait until "cobatestTA" appear after login

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(3));
        // list selector that clicked
        String[] selectors = {
                "div:nth-child(1) > .relative > .absolute > .btn-select",
                "div:nth-child(2) > .relative > .absolute > .btn-select",
                "div:nth-child(3) > .relative > .absolute > .btn-select",
                "div:nth-child(4) > .relative > .absolute > .btn-select",
                "div:nth-child(5) > .relative > .absolute > .btn-select"
        };

        for (String selector : selectors) {
            // click element
            driver.findElement(By.cssSelector(selector)).click();

            // get the element target
            WebElement                    element                    =
driver.findElement(By.cssSelector("span[x-text=\"selectedLine.length  ===
0 ?'' : selectedLine.length\"]"));

            // Get the content from target element
```

```
            String textContent = element.getText().trim();

            // Verify that the text content matches the expected value
(which is the loop index plus 1).
            Assert.assertEquals(textContent,
Integer.toString(Arrays.asList(selectors).indexOf(selector) + 1));
        }
        //delay

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(2));

        // Click download

driver.findElement(By.xpath("//*[@id=\"sidebar\"]/div/div[3]/div/div[4]/
button[2]")).click();

        // Click svg

driver.findElement(By.xpath("/html/body/div[1]/div/div[3]/div[2]/div/div
[2]/div[3]/button[2]")).click();
    }
```

Gambar 4.15 Mengunduh *icon*

k. Memilih *icon* berdasarkan kategori

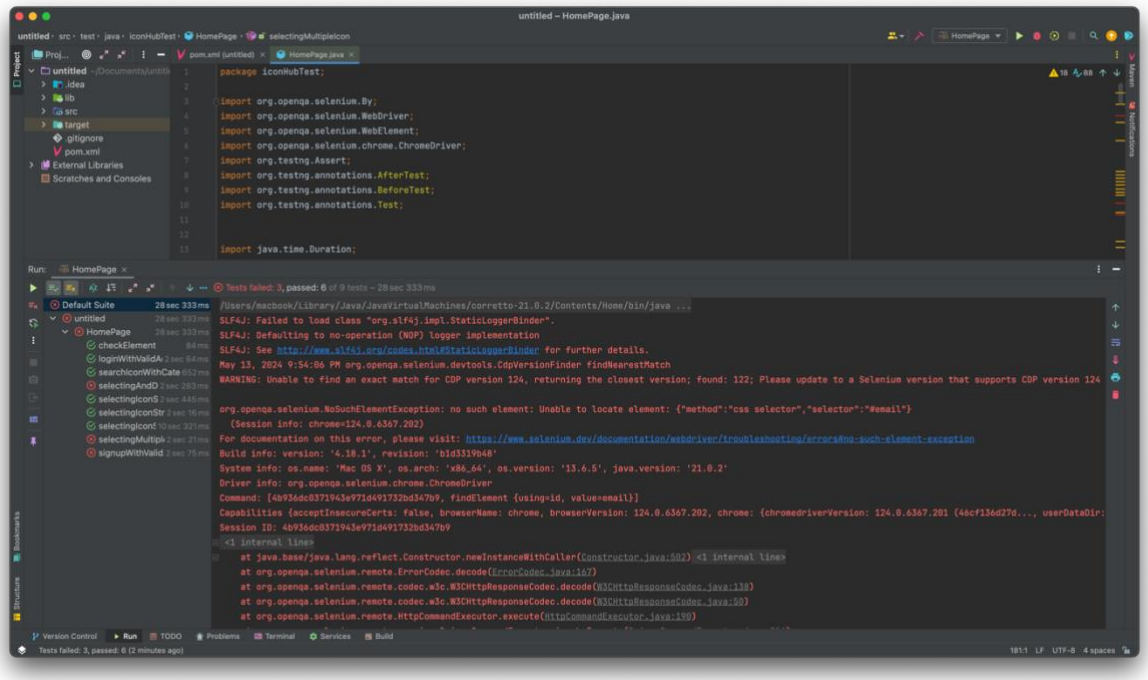
```
@Test
public void searchIconWithCategory(){

        // Select the "Brand" option from the dropdown
        driver.findElement(By.id("dropdownCategory")).click();

driver.findElement(By.xpath("//option[text()='Brand']")).click();
        //delay

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(2));
        // Verifikasi apakah teks mengandung kata "disney-plus"
        boolean                    isElementVisible                    =
driver.findElement(By.cssSelector(".text-xs.font-thin.text-gray-700.ml-
2.break-words")).isDisplayed();
        Assert.assertEquals(isElementVisible, true);
    }
```

```
@AfterTest
Public void teardown(){
    // Close browser after finish
    If (driver != null) {
        Driver.quit();
     }
   }
}
```

Gambar 4.16 Memilih *icon* berdasarkan kategori

4.1.3 Menjalankan *test* dan hasil *test*

Untuk menjalankan *test* dan melihat hasil *test* penulis hanya perlu mengklik tombol *run* di setiap *case* ataupun *scenario* yang telah dibuat.



Gambar 4.17 Menjalankan *test*

Hasil dari pengujian pada selenium terlihat bahwa berhasil menjalankan *test case* selama 28 detik yang mana terdapat 3 gagal dan 6 berhasil. Dari hasil tersebut terdapat beberapa *error* yang terjadi akibat *bug* yang terjadi pada selenium yang mana umum terjadi ketika mendapat pembaruan dari google chrome dan chromedriver.

4.2 Pengujian UI dengan Cypress

Untuk melakukan pengujian menggunakan Cypress, penulis harus melakukan *setup* terlebih dahulu pada *framework* Cypress. Cypress hanya bisa menggunakan Bahasa JavaScript, maka dari itu penulis menggunakan Bahasa JavaScript untuk pengujian kali ini.

4.2.1 Setup *Framework* Cypress

Hal yang dilakukan pertama kali penulis yaitu meng-*install* nya. Meng-*install* Cypress cukup mudah, bisa dilakukan dengan *install* via NPM melalui Terminal.

```
CD /your/project/path
```

Gambar 4.18 Contoh menavigasi ke direktori *project*.

```
npm install cypress --save-dev
```

Gambar 4.19 Contoh meng-*install* Cypress ke direktori *project*.

Setelah melakukan seperti hal diatas, maka akan ter-*install* secara otomatis dan tidak ada lagi *setup* yang perlu penulis lakukan, serta bisa langsung menulis *test script* nya.

4.2.2 Menulis *Script Test*

Penulis menulis kode atau *script* untuk menjalankan pengujian fungsional otomatis berdasarkan *test case* yang telah dibuat.

a. Fungsi beforeEach

```
describe('Sebelum melakukan pengujian', () => {
          beforeEach(() => {

            cy.visit(' https://iconhub.io')
          })
```

Gambar 4.20 Fungsi *beforeEach*

b. Menampilkan Halaman Utama

```
  describe('First IconHub Tests', () => {
    it('displays the main page', () => {
      cy.get('.text-lg > img').should('be.visible');
    });
  });
```

Gambar 4.21 Menampilkan halaman utama

c. Membuat Akun baru

```
    it('signup with valid account', () => {
      // Generate random email using faker
      const randomEmail = faker.internet.email();

      cy.get('.pr-2').click();
      cy.get('body > div > div > div > form > div:nth-child(10) > a').click();
      cy.get('#email').type(randomEmail);
      cy.get('#password').type('cobacoba123');
      cy.get('#fullname').type('cobacobain');
      cy.get('body > div > div > div > form > div.w-full.my-2 > button').click();
      cy.get('.btn > .text-black').should('contain.text', 'cobacobain');
    });
  });
```

Gambar 4.22 Membuat akun baru

d. Masuk ke akun berhasil

```
describe('Login account for iconhub', () => {
  it('login with valid account', () => {
      cy.get('.pr-2').click();
      cy.get('#email').type('cobacoba4@gmail.com');
      cy.get('#password').type('cobacoba123');
      cy.get(':nth-child(8) > .btn').click();
      cy.get('.btn > .text-black').should('contain.text', 'cobatestTA');
    });
  });
```

Gambar 4.23 Masuk ke akun berhasil

e. Gagal masuk ke akun

```
describe('Login account for iconhub', () => {
  it('login with invalid account', () => {
      cy.get('.pr-2').click();
      cy.get('#email').type('cobacoba444@gmail.com');
      cy.get('#password').type('cobacoba12345');
      cy.get(':nth-child(8) > .btn').click();
      cy.get('.btn > .text-black').should('contain.text', 'cobatestTA');
    });
  });
```

Gambar 4.24 Gagal masuk ke akun

f. Memilih *icon style*

```
describe('Icon Style Selection Test', () => {
    it('Should apply the correct style when selecting each icon style', () => {
      // Visit the page with the icon selection buttons
      cy.visit('/');

      // Define an array of style names
      const styleNames = ['line', 'glyph', 'line-color', 'flat-color', 'flat-line',
'multi-color'];

      // Iterate over each icon style
      styleNames.forEach(styleName => {
        // Click on the icon style button
        cy.get(`button[data-name="${styleName}"]`).click();



        // Assert that the selected icon style has the correct style applied
        cy.get(`.icon[data-style="${styleName}"]:visible`).should('exist');
      });
    });
  });
```

Gambar 4.25 Memilih *icon style*

g. Memilih *icon size*

```
describe('Icon Size Selection Test', () => {
    it('Should apply the correct size when selecting each icon size', () => {
      // Visit the page with the icon size options
      cy.visit('/');

      // Define an array of icon sizes
      const iconSizes = [16, 24, 32, 48, 64, 96];

      // Iterate over each icon size
      iconSizes.forEach(size => {
        // Click on the icon size button
        cy.get(`.btn-icon-size:contains(${size})`).click();

        // Assert that the selected icon size has the correct size applied
        cy.get(`.btn-icon-size:contains(${size})`).should('have.class', 'text-white
bg-purple-700 border-purple-700 shadow-primary');
        cy.get(`.btn-icon-size:not(:contains(${size}))`).should('have.class',
'text-gray-500 hover:bg-purple-100 hover:border-purple-700 hover:text-purple-700');
      });
    });
```

```
  });
```

Gambar 4.26 Memilih *icon size*

h. Memilih *stroke size*

```
describe('Icon Stroke Size Selection Test', () => {
    it('Should apply the correct stroke size when selecting each stroke size', ()
=> {
      // Visit the page with the icon stroke size options
      cy.visit('/');

      // Define an array of stroke sizes
      const strokeSizes = [0.5, 1, 1.5, 2, 2.5, 3];

      // Iterate over each stroke size
      strokeSizes.forEach(stroke => {
        // Click on the icon stroke size button
        cy.get(`.btn-icon-stroke:contains(${stroke})`).click({ multiple: true });

        // Assert that the selected icon stroke size has the correct size applied
        cy.get(`.btn-icon-stroke:contains(${stroke})`).should('have.class', 'text-
white bg-purple-700 border-purple-700 shadow-primary');
        cy.get(`.btn-icon-stroke:not(:contains(${stroke}))`).should('have.class',
'text-gray-500 hover:bg-purple-100 hover:border-purple-700 hover:text-purple-700');
      });
    });
  });
```

Gambar 4.27 Memilih *stroke size*

i. Memilih *icon*

```
describe('Choosing icon', () => {
    it.only('should assert choose icon and count', () => {
      const selectors = [
        'div:nth-child(1) > .relative > .absolute > .btn-select',
        'div:nth-child(2) > .relative > .absolute > .btn-select',
        'div:nth-child(3) > .relative > .absolute > .btn-select',
        'div:nth-child(4) > .relative > .absolute > .btn-select',
        'div:nth-child(5) > .relative > .absolute > .btn-select'
      ];

      // Click each element and assert the selected icon count
      selectors.forEach(async (selector, index) => {
        cy.get(selector).click();
```

```
        cy.get('span[x-text="selectedLine.length === 0 ?\'\' :
selectedLine.length"]').should('contain.text', (index + 1).toString());
      });
    });
  });
```

Gambar 4.28 Memilih *icon*

j. Mengunduh *icon*

```
describe('Download icon with login first', () => {
    it('should assert the selected icon count and download icons', () => {
      //Sebelum melakukan download diperlukan login ke akun terlebih dahulu

      cy.get('.pr-2').click();
      cy.get('#email').type('cobacoba4@gmail.com');
      cy.get('#password').type('cobacoba123');
      cy.get(':nth-child(8) > .btn').click();
      cy.get('.btn > .text-black').should('contain.text', 'cobatestTA');

      const selectors = [
        'div:nth-child(1) > .relative > .absolute > .btn-select',
        'div:nth-child(2) > .relative > .absolute > .btn-select',
        'div:nth-child(3) > .relative > .absolute > .btn-select',
        'div:nth-child(4) > .relative > .absolute > .btn-select',
        'div:nth-child(5) > .relative > .absolute > .btn-select'
      ];

      // Click each element and assert the selected icon count
      selectors.forEach(async (selector, index) => {
        cy.get(selector).click();
        cy.get('span[x-text="selectedLine.length === 0 ?\'\' :
selectedLine.length"]').should('contain.text', (index + 1).toString());
      });

      // Click the download button
      cy.get('button').contains('Download 5 Icons').click();
      cy.get('.grid > .block').click();


      // Specify the path to save the downloaded file
      const savePath = '/Users/masgilang/Downloads/';

      // Assert that the file has been downloaded
      cy.contains('div', 'Thanks for downloading Iconhub!').should('exist');
    });
```

```
  });
```

Gambar 4.29 Mengunduh *icon*

k. Memilih *icon* berdasarkan kategori

```
describe('Selecting icon from dropdown', () => {
    it('selects "Brand" from dropdown and asserts the appearance of Disney Plus
icon', () => {
      // Pilih opsi "Brand" dari dropdown
      cy.get('#dropdownCategory').select('Brand');

      // Tunggu hingga icon Disney Plus muncul
      cy.get('[data-icon="disney-plus"]').should('be.visible');

      // Lakukan asersi untuk memastikan bahwa icon Disney Plus muncul
      cy.get('[data-icon="disney-plus"]').should('exist');
    });
  });
```

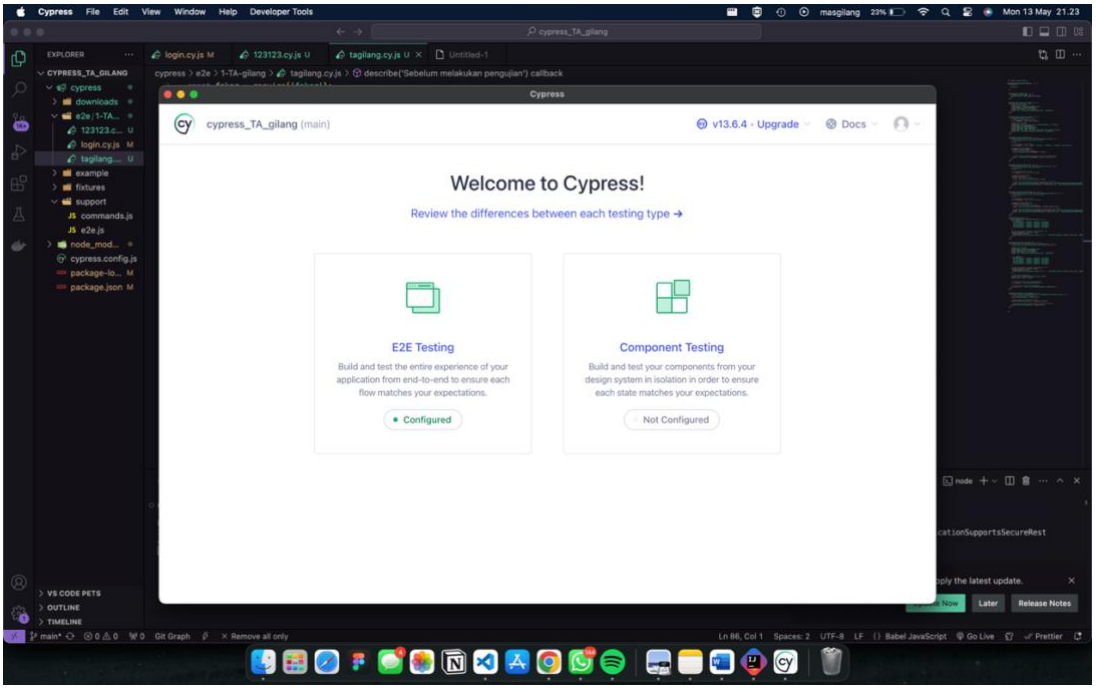
Gambar 4.30 Memilih *icon* berdasarkan kategori

4.2.3 Menjalankan *test* dan hasil *test*

Untuk menjalankan pengujian dan melihat hasil *test* penulis perlu membuka terminal pada vscode dengan code pada gambar 4.31 berikut dan akan tertampil halaman.

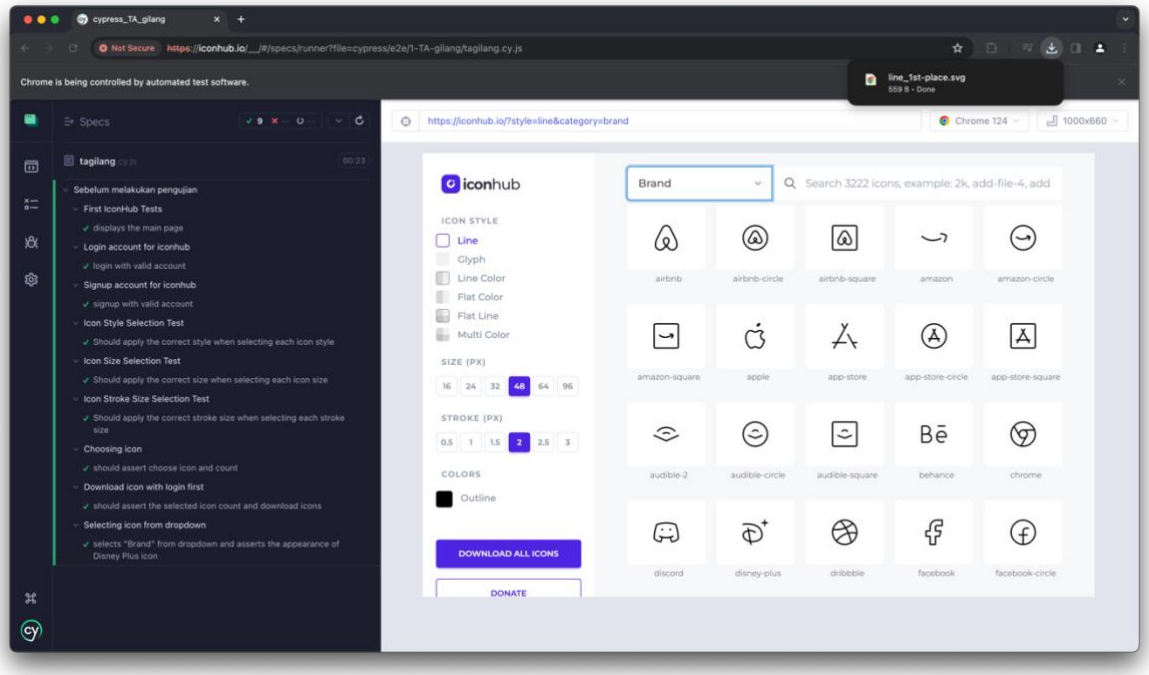
```
Npx cypress open
```

Gambar 4.31 Menjalankan *test*



Gambar 4.32 Tampilan awal cypress *test*

Pada halaman ini penulis hanya perlu memilih E2E Testing karena yang selaras dengan penelitian ini yakni UI *testing* yaitu E2E *Testing* pada Cypress.



Gambar 4.33 Tampilan hasil *test* cypress

Setelah memilih *project* mana yang akan dilakukan pengujian maka pengujian akan dimulai secara otomatis dan terlihat detail dari *test* nya di bagian kiri serta hasilnya. Hasil yang didapat di pengujian pada Cypress ini adalah 23 detik dengan semua *test case* berhasil dijalankan.

4.3 Pengujian UI dengan Playwright

Untuk melakukan pengujian menggunakan Playwright, penulis harus melakukan *setup* terlebih dahulu pada *framework* Playwright. Playwright bisa dijalankan dengan beberapa bahasa seperti JavaScript/TypeScript, Python, Java, dan .NET, penulis memilih menggunakan bahasa TypeScript karena sebelumnya penulis mendapat hambatan yakni *error* pada proses pengujian dan saat mencoba dengan Bahasa TypeScript penulis berhasil melakukan pengujian.

4.3.1 Setup *Framework* Playwright

Hal yang dilakukan pertama kali penulis yaitu meng-*install* nya. Meng-*install* Playwright cukup mudah, bisa dilakukan dengan *install* via NPM melalui Terminal.

```
CD /your/project/path
```

Gambar 4.34 Contoh menavigasi ke direktori *project*.

```
npm init playwright@latest
```

Gambar 4.35 Contoh meng-*install* Playwright ke direktori *project*.

Setelah melakukan seperti hal diatas, maka akan ter-*install* secara otomatis dan tidak ada lagi *setup* yang perlu penulis lakukan, serta bisa langsung menulis *test script* nya.

4.3.2 Menulis *Script Test*

Penulis menulis kode atau *script* untuk menjalankan pengujian fungsional otomatis berdasarkan *test case* yang telah dibuat.

a. Fungsi beforeEach

```
test.beforeEach(async ({ page }) => {
  await page.goto('https://iconhub.io/');
});
```

Gambar 4.36 Fungsi *beforeEach*

b. Menampilkan Halaman Utama

```
test('displays the main page', async ({ page }) => {
```

```
  await expect(page.locator('.text-lg > img')).toBeVisible();
});
```

Gambar 4.37 Menampilkan halaman utama

c. Membuat Akun baru

```
test('signup with valid account', async ({ page }) => {
  // Generate random email using faker
  const randomEmail = faker.internet.email();

  await page.locator('.pr-2').click();
  await page.locator('body > div > div > div > form > div:nth-child(10) >
a').click();
  await page.locator('#email').type(randomEmail);
  await page.locator('#password').type('cobacoba123');
  await page.locator('#fullname').type('cobacobain');
  await page.locator('body > div > div > div > form > div.w-full.my-2 >
button').click();
  await expect(page.locator('.btn > .text-
black')).toContainText('cobacobain');
});
```

Gambar 4.38 Membuat akun baru

d. Masuk ke akun berhasil

```
test('login with valid account', async ({ page }) => {
  await page.locator('.pr-2').click();
  await page.locator('#email').type('cobacoba4@gmail.com');
  await page.locator('#password').type('cobacoba123');
  await page.locator(':nth-child(8) > .btn').click();
  await expect(page.locator('.btn > .text-
black')).toContainText('cobatestTA');
});
```

Gambar 4.39 Masuk ke akun berhasil

e. Gagal masuk ke akun

```
test('login with valid account', async ({ page }) => {
  await page.locator('.pr-2').click();
  await page.locator('#email').type('cobacoba444@gmail.com');
  await page.locator('#password').type('cobacoba12345');
```

```
  await page.locator(':nth-child(8) > .btn').click();
  await expect(page.locator('.btn > .text-
black')).toContainText('cobatestTA');
});
```

Gambar 4.40 Gagal masuk ke akun

f. Memilih *icon style*

```
test('selecting icon style applies the correct style', async ({ page }) =>
{
  // Get a list of all icon style buttons
  const styleButtons = await page.locator('.change-style').all();

  // Iterate over each style button
  for (const button of styleButtons) {
    // Click on the icon style button
    await button.click();

    // Get the name of the icon style
    const styleName = await button.getAttribute('data-name');

    // Wait for the icon to update with the new style and assert the style
    await page.waitForSelector(`.icon[data-style="${styleName}"]:visible`);

  }
});
```

Gambar 4.41 Memilih *icon style*

g. Memilih *icon size*

```
test('selecting icon size applies the correct size', async ({ page }) => {
  // Simulate clicking to change the size to 16
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(1) > div.flex.flex-wrap > div:nth-child(1) > div').click();
  // Wait for the icon to update with the new size
  await page.waitForSelector('svg[data-icon-name="1st-
place"][width="16"][height="16"]', { state: 'visible' });

  // Simulate clicking to change the size to 24
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(1) > div.flex.flex-wrap > div:nth-child(2) > div').click();
  // Wait for the icon to update with the new size
  await page.waitForSelector('svg[data-icon-name="1st-
place"][width="24"][height="24"]', { state: 'visible' });

  // Simulate clicking to change the size to 32
```

```
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(1) > div.flex.flex-wrap > div:nth-child(3) > div').click();
  // Wait for the icon to update with the new size
  await page.waitForSelector('svg[data-icon-name="1st-
place"][width="32"][height="32"]', { state: 'visible' });

  // Simulate clicking to change the size to 48
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(1) > div.flex.flex-wrap > div:nth-child(4) > div').click();
  // Wait for the icon to update with the new size
  await page.waitForSelector('svg[data-icon-name="1st-
place"][width="48"][height="48"]', { state: 'visible' });

  // Simulate clicking to change the size to 64
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(1) > div.flex.flex-wrap > div:nth-child(5) > div').click();
  // Wait for the icon to update with the new size
  await page.waitForSelector('svg[data-icon-name="1st-
place"][width="64"][height="64"]', { state: 'visible' });

  // Simulate clicking to change the size to 96
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(1) > div.flex.flex-wrap > div:nth-child(6) > div').click();
  // Wait for the icon to update with the new size
  await page.waitForSelector('svg[data-icon-name="1st-
place"][width="96"][height="96"]', { state: 'visible' });
});
```

Gambar 4.42 Memilih *icon size*

h. Memilih *stroke size*

```
test('selecting icon stroke size applies the correct stroke size', async ({
page }) => {

  // Simulate clicking to change the size to 24
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(1) > div.flex.flex-wrap > div:nth-child(2) > div').click();
  // Wait for the icon to update with the new size
  await page.waitForSelector('svg[data-icon-name="1st-
place"][width="24"][height="24"]', { state: 'visible' });

  // Simulate clicking to change the size to 0.5
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(2) > div.flex.flex-wrap > div:nth-child(1) > div').click();

  // Simulate clicking to change the size to 1
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(2) > div.flex.flex-wrap > div:nth-child(2) > div').click();

  // Simulate clicking to change the size to 1.5
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(2) > div.flex.flex-wrap > div:nth-child(3) > div').click();

  // Simulate clicking to change the size to 2
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(2) > div.flex.flex-wrap > div:nth-child(4) > div').click();

  // Simulate clicking to change the size to 2.5
```

```
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(2) > div.flex.flex-wrap > div:nth-child(5) > div').click();

  // Simulate clicking to change the size to 3
  await page.locator('#sidebar > div > div:nth-child(3) > div > div:nth-
child(2) > div.flex.flex-wrap > div:nth-child(6) > div').click();
});
```

Gambar 4.43 Memilih *stroke size*

i. Memilih *icon*

```
test('asserting the selected icon count after clicking multiple elements',
async ({ page }) => {
  const selectors = [
    'div:nth-child(1) > .relative > .absolute > .btn-select',
    'div:nth-child(2) > .relative > .absolute > .btn-select',
    'div:nth-child(3) > .relative > .absolute > .btn-select',
    'div:nth-child(4) > .relative > .absolute > .btn-select',
    'div:nth-child(5) > .relative > .absolute > .btn-select'
  ];

  for (const selector of selectors) {
    // Click the element
    await page.locator(selector).click();

    // Wait for the target element to be visible
    await page.waitForSelector('span[x-text="selectedLine.length === 0
?\'\' : selectedLine.length"]', { state: 'visible' });

    // Get the text content of the target element
    const element = await page.locator('span[x-text="selectedLine.length
=== 0 ?\'\' : selectedLine.length"]');
    const textContent = await element.innerText();

    // Assert that the text content matches the expected value (which is
the index of the loop plus 1)
    expect(textContent.trim()).toBe((selectors.indexOf(selector) +
1).toString());
  }
});
```

Gambar 4.44 Memilih *icon*

j. Mengunduh *icon*

```
test('asserting the selected icon count and downloading icons', async ({
page }) => {

  await page.locator('.pr-2').click();
  await page.locator('#email').type('cobacoba4@gmail.com');
  await page.locator('#password').type('cobacoba123');
  await page.locator(':nth-child(8) > .btn').click();
  await expect(page.locator('.btn > .text-
black')).toContainText('cobatestTA');


  const selectors = [
    'div:nth-child(1) > .relative > .absolute > .btn-select',
```

```
    'div:nth-child(2) > .relative > .absolute > .btn-select',
    'div:nth-child(3) > .relative > .absolute > .btn-select',
    'div:nth-child(4) > .relative > .absolute > .btn-select',
    'div:nth-child(5) > .relative > .absolute > .btn-select'
  ];

  // Click each element and assert the selected icon count
  for (const selector of selectors) {
    await page.locator(selector).click();
    await page.waitForSelector('span[x-text="selectedLine.length === 0
?\'\' : selectedLine.length"]', { state: 'visible' });
    const element = await page.locator('span[x-text="selectedLine.length
=== 0 ?\'\' : selectedLine.length"]');
    const textContent = await element.innerText();
    expect(textContent.trim()).toBe((selectors.indexOf(selector) +
1).toString());
  }

  // Start waiting for download before clicking. Note no await.
  const downloadPromise = page.waitForEvent('download');

  // Click the download button
  await page.getByRole('button', { name: 'Download 5 Icons' }).click();
  await page.getByRole('button', { name: 'SVG', exact: true }).click();


  // Wait for the download to complete
  const download = await downloadPromise;

  // Specify the path to save the downloaded file
  const savePath = '/Users/masgilang/Downloads/';

  // Save the downloaded file
  await download.saveAs(savePath + download.suggestedFilename());
});
```

Gambar 4.45 Mengunduh *icon*

k. Memilih *icon* berdasarkan kategori

```
test('search icon in the search bar', async ({ page }) => {
  // Pilih opsi "Brand" dari dropdown
  await page.selectOption('#dropdownCategory', 'Brand');

  // Tunggu hingga icon Disney Plus muncul
  await page.waitForSelector('[data-icon="disney-plus"]', { state:
'visible' });

  // Lakukan asersi untuk memastikan bahwa icon Disney Plus muncul
  await expect(page.locator('[data-icon="disney-plus"]')).toBeVisible();
});
```

Gambar 4.46 Memilih *icon* berdasarkan kategori

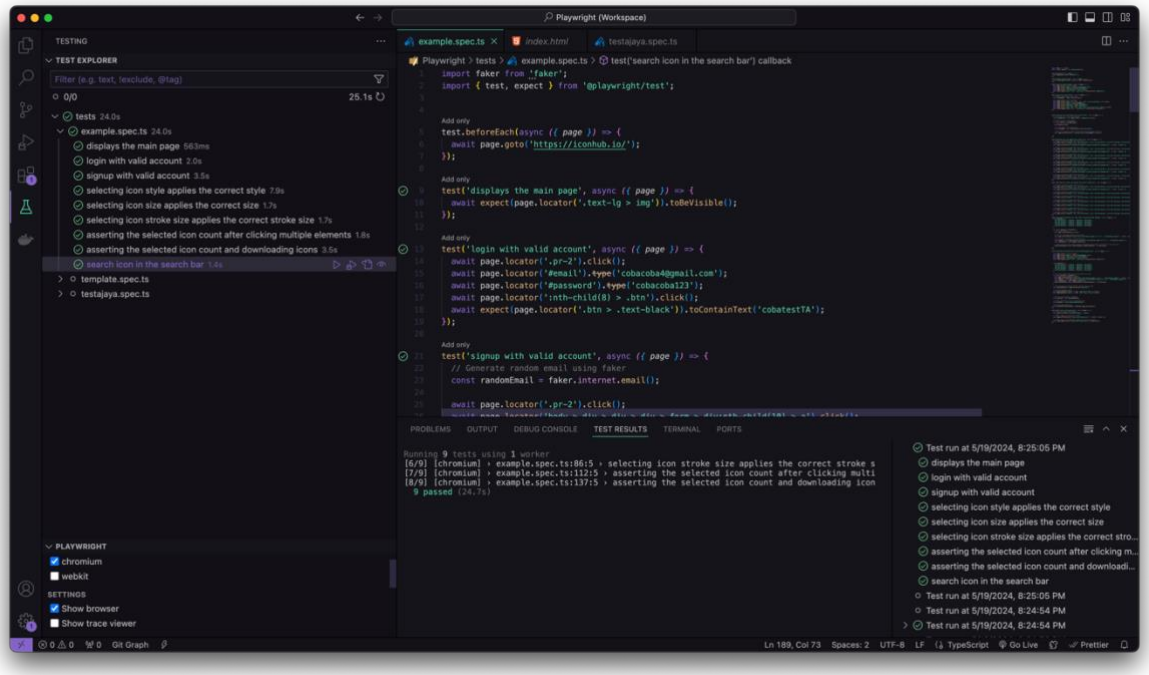
4.3.3 Menjalankan *test* dan hasil *test*

Untuk menjalankan pengujian dan melihat hasil *test* penulis bisa dengan dua cara, pertama membuka terminal pada vscode dengan *code* pada gambar 4.47 dan akan tertampil halaman.

```
Npx playwright test
```

Gambar 4.47 Menjalankan *test*

Lalu cara yang kedua dengan langsung menjalankannya pada menu *Testing* di vscode seperti gambar 4.48 ini.



Gambar 4.48 Tampilan saat pengujian pada playwright

Dari hasil pengujian berikut terdapat bahwa *test case* berhasil dijalankan dengan waktu 25 detik.

4.4 Penerapan Metode DIA Sebagai Pembanding

Setelah menyelesaikan pengujian UI fungsional otomatis pada ketiga *test automation framework* yang dipilih, langkah selanjutnya adalah pengumpulan data data yang diperlukan baik yang sudah dilakukan maupun yang akan dilakukan dan menganalisis hasil pengujian tersebut serta menerapkan Metode *The Distance to The Ideal Alternative* (DIA). Metode DIA digunakan untuk mengevaluasi dan membandingkan kinerja dari masing-masing *framework*

berdasarkan parameter-parameter yang telah ditentukan sebelumnya. Dengan menerapkan metode ini, penulis dapat menentukan *framework* mana yang paling mendekati kondisi ideal berdasarkan kriteria yang telah ditetapkan, sehingga memungkinkan pemilihan *framework* yang paling optimal dari segi kinerja untuk pengujian UI.

4.4.1 Pengumpulan Data

Berikut pada tabel 4.1 merupakan data hasil pengujian otomatis yang telah dilakukan oleh penulis pada ketiga *test automation framework*.

Tabel 4.1 Hasil pengujian pada setiap *framework*

Nama *Framework*	**Waktu Hasil Eksekusi**	***Script Creation Time***	***Easy of Setup***	***Scripting Language***	***Cost***
Selenium	28 detik	7 jam	Sulit	Java, JS, TS, Python	*Free*
Cypress	23 detik	2 jam	Mudah	Javascript	*Free*
Playwright	25 detik	4 jam	Mudah	Java, JS, C#, Python, Ruby	*Free*

Tabel 4. 2 Hasil pengujian pada setiap *framework*

Nama *Framework*	***Product Support***	**Dokumentasi hasil *report***	***Inspection Element***	***Usability***	***Portability***
Selenium	Ada (1)	Tidak ada (0)	Tidak ada (0)	48,13	*Cross Platform*
Cypress	Ada (1)	Ada (1)	Ada (1)	72,29	*Cross Platform*
Playwright	Ada (1)	Ada (1)	Ada (1)	57,29	*Cross Platform*

Hasil Pengumpulan data Usability

Berdasarkan data pada tabel 4.2 pada kolom *Usability* data tersebut didapatkan dari hasil perhitungan SUS *Score*. Pada proses penghitungan SUS *Score* terdapat 10 pertanyaan yang diberikan pada responden lalu rentang penilaian skor nya adalah 1 untuk sangat tidak setuju dan 3 adalah ragu ragu serta 5 untuk sangat setuju, dengan cara perhitungan setiap pertanyaan ganjil (1,3,5,7,9) adalah jumlah skor dikurang 1 dan setiap pertanyaan genap (2,4,6,8,10)

adalah 5 dikurang jumlah skor. Lalu untuk perhitungan akhir untuk mengetahui skor akhir nya dengan cara menjumlahkan semua total skor per responden lalu dikali dengan 2,5 (Brooke, 2013). Dengan begitu SUS *Score* pada sub parameter *Usability* didapatkan. Berikut merupakan contoh pengambilan SUS *Score* pada *Framework* Selenium.

Data responden Selenium

Responden	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
R1	5	3	5	2	5	3	5	2	3	5
R2	4	3	4	3	4	2	3	2	3	5
R3	4	2	4	2	5	2	3	2	4	5
R4	2	2	3	2	3	2	3	2	3	2
R5	4	3	3	2	4	2	3	3	3	4
R6	4	3	4	5	3	4	1	3	2	5
R7	5	2	4	3	4	3	4	3	4	5
R8	5	3	4	2	4	1	4	2	4	2
R9	2	2	3	2	2	3	2	3	2	3
R10	1	3	2	4	2	5	1	5	2	4
R11	3	5	2	5	4	4	4	5	1	4

Gambar 4. 49 Hasil data responden pada *framework* Selenium

Pada gambar 4.49 terdapat hasil data dari tiap responden yang memberikan nilai pada 10 pertanyaan SUS. Berikut ini merupakan hasil penghitungan SUS *Score* pada *Framework* Selenium.

Data perhitungan SUS Selenium

Responden	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total skor
R1	4	2	4	3	4	2	4	3	2	0	70
R2	3	2	3	2	3	3	2	3	2	0	57,5
R3	3	3	3	3	4	3	2	3	3	0	67,5
R4	1	3	2	3	2	3	2	3	2	3	60
R5	3	2	2	3	3	3	2	2	2	1	57,5
R6	3	2	3	0	2	1	0	2	1	0	35
R7	4	3	3	2	3	2	3	2	3	0	62,5
R8	4	2	3	3	3	4	3	3	3	3	77,5
R9	1	3	2	3	1	2	1	2	1	2	45
R10	0	2	1	1	1	0	0	0	1	1	17,5
R11	2	0	1	0	3	1	3	0	0	1	27,5
											0
											48,13

Gambar 4. 50 Hasil perhitungan SUS *Score* pada *framework* Selenium

Pada gambar 4.50 terdapat hasil perhitungan yang telah dilakukan sehingga mendapatkan nilai akhir dengan mengambil nilai rata-rata dari tiap total skor tiap responden dengan hasil 48,13.

4.4.2 Pembentukan Matriks Keputusan

Berdasarkan data yang telah dikumpulkan pada bab sebelumnya tentang pemilihan parameter dan pembobotan parameter serta sub parameter, berikut penulis membuat matriks keputusan untuk parameter dan sub parameter yang telah ditentukan.

Matriks Keputusan Parameter *Technical Economic View*

Berdasarkan data yang tercantum pada Tabel 3.2 dan bobot yang telah ditetapkan, berikut pada tabel 4.3 ini adalah hasilnya.

Tabel 4.3 Hasil matriks keputusan parameter *technical ecnomic view*

Alternatif	Sub Parameter		
	Scripting Language	*Cost*	*Product Support*
Selenium	1	1	1
Cypress	0	1	1
Playwright	0.75	1	1

Matriks Keputusan Parameter *Testing Process View*

Berdasarkan data yang tercantum pada Tabel 3.6 dan bobot yang telah ditetapkan, berikut ini adalah hasilnya.

Tabel 4.4 Hasil matriks keputusan parameter *testing process view*

Alternatif	Sub Parameter				
	Easy of Setup	*Script Creation Time*	Dokumentasi hasil report	Waktu hasil eksekusi	*Inspection Element*
Selenium	0	0	0	0	0
Cypress	1	1	1	1	1
Playwright	1	0.5	1	0.5	1

Matriks Keputusan Parameter *Quality View*

Berdasarkan data yang tercantum pada Tabel 3.12 dan bobot yang telah ditetapkan, berikut ini adalah hasilnya.

Tabel 4.5 Hasil matriks keputusan parameter *quality view*

Alternatif	Sub Parameter	
	Usability	*Portability*
Selenium	0	1
Cypress	0.75	1
Playwright	0.25	1

4.4.3 Normalisasi Matriks Keputusan

Pada tahap normalisasi matriks dilakukan menggunakan persamaan (2,2) sehingga menghasilkan data yang telah ternormalisasi untuk masing masing parameter.

Normalisasi Matriks Keputusan Parameter *Technical Economic View*

Berdasarkan persamaan (2,2), penulis melakukan normalisasi matriks keputusan menggunakan Tabel 4.3, dan untuk perhitungan pada *framework* Selenium pada sub parameter *Technical Economic View*, hasil yang diperoleh adalah sebagai berikut.

$$\frac{1}{\sqrt{1^2 + 0^2 + 0{,}75^2}} = 0{,}8$$

Untuk melihat hasil lengkap dari perhitungan normalisasi matriks keputusan pada setiap *Scripting Language* di masing-masing sub parameter, data tersebut disajikan dalam tabel 4.6

Tabel 4.6 Hasil perhitungan normalisasi matriks *technical economic view*

Alternatif	Sub Parameter		
	Scripting Language	*Cost*	*Product Support*
Selenium	0,80	0,58	0,58
Cypress	0	0,58	0,58
Playwright	0.60	0,58	0,58

Normalisasi Matriks Keputusan Parameter *Testing Process View*

Berdasarkan persamaan (2,2), untuk normalisasi matriks Keputusan, hasil yang diperoleh adalah sebagai berikut.

Tabel 4.7 Hasil perhitungan normalisasi matriks *testing process view*

Alternatif	Sub Parameter				
	Easy of Setup	*Script Creation Time*	Dokumentasi hasil report	Waktu hasil eksekusi	*Inspection Element*
Selenium	0	0	0	0	0
Cypress	0,71	0,89	0,71	0,89	0,71
Playwright	0,71	0.45	0,71	0.45	0,71

Normalisasi Matriks Keputusan Parameter *Quality View*

Berdasarkan persamaan (2,2), untuk normalisasi matriks Keputusan, hasil yang diperoleh adalah sebagai berikut.

Tabel 4.8 Hasil perhitungan normalisasi matriks *quality view*

Alternatif	Sub Parameter	
	Usability	*Portability*
Selenium	0	0,58
Cypress	0,96	0,58
Playwright	0.32	0,58

4.4.4 Pembobotan Matriks Normalisasi

Pada tahap ini matriks keputusan yang telah di normalisasi dikalkulasikan dengan bobot parameter atau sub parameter sesuai dengan persamaan (2,4) dan (2,5)

Pembobotan Matriks Normalisasi Parameter *Technical Economic View*

Berdasarkan persamaan (2,4) dan (2,5), penulis melakukan pembobotan matriks normalisasi menggunakan Tabel 4.6 dengan bobot (w) pada Tabel 3.1, dan untuk perhitungan pada *framework* Selenium pada sub parameter *Scipting Language,* hasil yang diperoleh adalah sebagai berikut.

$$(0{,}80 \times 0{,}46) = 0{,}368$$

Untuk melihat hasil lengkap dari perhitungan pembobotan matriks normalisasi pada setiap *test automation framework* di masing-masing sub parameter, data tersebut disajikan dalam Tabel 4.9

Tabel 4.9 Hasil perhitungan pembobotan matriks *technical economic view*

Alternatif	Sub Parameter		
	Scripting Language	*Cost*	*Product Support*
Selenium	0,368	0,209	0,104
Cypress	0	0,209	0,104
Playwright	0.276	0,209	0,104

Pembobotan Matriks Normalisasi Parameter *Testing Process View*

Berdasarkan persamaan (2,4), dan (2,5) untuk pembobotan matriks normalisasi, hasil yang diperoleh adalah sebagai berikut.

Tabel 4.10 Hasil perhitungan pembobotan matriks *testing process view*

Alternatif	Sub Parameter				
	Easy of Setup	*Script Creation Time*	Dokumentasi hasil report	Waktu hasil eksekusi	*Inspection Element*
Selenium	0	0	0	0	0
Cypress	0,192	0,081	0,327	0,081	0,064
Playwright	0,192	0.041	0,327	0.041	0,064

Pembobotan Matriks Normalisasi Parameter *Quality View*

Berdasarkan persamaan (2,4), dan (2,5) untuk pembobotan matriks normalisasi, hasil yang diperoleh adalah sebagai berikut.

Tabel 4.11 Hasil perhitungan pembobotan matriks *quality view*

Alternatif	Sub Parameter	
	Usability	*Portability*
Selenium	0	0,052
Cypress	0,873	0,052
Playwright	0.291	0,052

4.4.5 Penentuan Solusi Ideal Positif dan Solusi Ideal Negatif Parameter *Technical Economic View*

Dalam langkah ini, penulis menentukan alternatif positif dan negatif untuk setiap sub parameter berdasarkan nilai yang terdapat pada Tabel 4.9 sesuai dengan persamaan (2,6) dan (2,7). Solusi ideal positif ditentukan dari nilai terbesar, sedangkan Solusi ideal negatif ditentukan dari nilai terkecil dalam Tabel 4.9 untuk masing-masing sub parameter.

Tabel 4.12 Hasil ideal positif *technical economic view*

Alternatif Positif	Sub Parameter	$A^+ = maxV_{ij} = [V_1^+, V_2^+ \cdots V_n^+]$
V_1^+	***Scripting Language***	0,368
V_2^+	***Cost***	0,209
V_3^+	***Product Support***	0,104

Tabel 4.13 Hasil ideal negatif *technical economic view*

Alternatif Negatif	Sub Parameter	$A^- = maxV_{ij} = [V_1^-, V_2^- \cdots V_n^-]$
V_1^-	***Scripting Language***	0
V_2^-	***Cost***	0,209
V_3^-	***Product Support***	0,104

4.4.6 Penentuan Solusi Ideal Positif dan Solusi Ideal Negatif Parameter *Testing Process View*

Dalam langkah ini, penulis menentukan alternatif positif dan negatif untuk setiap sub parameter berdasarkan nilai yang terdapat pada Tabel 4.10 sesuai dengan persamaan (2,6) dan (2,7). Solusi ideal positif ditentukan dari nilai terbesar, sedangkan Solusi ideal negatif ditentukan dari nilai terkecil dalam Tabel 4.10 untuk masing-masing sub parameter.

Tabel 4.14 Hasil ideal positif *testing process view*

Alternatif Positif	Sub Parameter	$A^+ = maxV_{ij} = [V_1^+, V_2^+ \cdots V_n^+]$
V_1^+	***Easy of Setup***	0,192
V_2^+	***Script Creation Time***	0,081
V_3^+	**Dokumentasi Hasil Report**	0,327
V_4^+	**Waktu Hasil Eksekusi**	0,081
V_5^+	***Inspection Element***	0,064

Tabel 4.15 Hasil ideal negatif *testing process view*

Alternatif Negatif	Sub Parameter	$A^- = maxV_{ij} = [V_1^-, V_2^- \cdots V_n^-]$
V_1^-	***Easy of Setup***	0
V_2^-	***Script Creation Time***	0
V_3^-	**Dokumentasi Hasil Report**	0
V_4^-	**Waktu Hasil Eksekusi**	0
V_5^-	***Inspection Element***	0

4.4.7 Penentuan Solusi Ideal Positif dan Solusi Ideal Negatif Parameter *Quality View*

Dalam langkah ini, penulis menentukan alternatif positif dan negatif untuk setiap sub parameter berdasarkan nilai yang terdapat pada Tabel 4.11 sesuai dengan persamaan (2,6) dan (2,7). Solusi ideal positif ditentukan dari nilai terbesar, sedangkan Solusi ideal negatif ditentukan dari nilai terkecil dalam Tabel 4.11 untuk masing-masing sub parameter.

Tabel 4.16 Hasil ideal positif *quality view*

Alternatif Positif	Sub Parameter	$A^+ = maxV_{ij} = [V_1^+, V_2^+ \cdots V_n^+]$
V_1^+	***Usability***	0,873
V_2^+	***Portability***	0,052

Tabel 4.17 Hasil ideal negatif *quality view*

Alternatif Negatif	Sub Parameter	$A^- = maxV_{ij} = [V_1^-, V_2^- \cdots V_n^-]$
V_1^-	***Usability***	0
V_2^-	***Portability***	0,052

4.4.8 Perhitungan Jarak Mahnhattan

Pada langkah ini penulis menentukan hasil perhitungan Jarak Manhattan Maksimum dan Minimum sesuai dengan persamaan (2,8) dan (2,9). dalam perhitungan Jarak Manhattan

menggunakan nilai alternatif positif dan negatif sebagai acuan. Sehingga menghasilkan nilai yang tercantum pada tabel-tabel berikut ini.

Jarak Manhattan Parameter *Technical Economic View*

Berdasarkan persamaan (2,8), untuk mendapatkan hasil perhitungan jarak Manhattan maksimum untuk *framework* Selenium pada sub parameter *Scipting Language*, hasil yang diperoleh adalah sebagai berikut.

$$D_1^+ = \left[V_{1,1} - a_1^+\right] + \left[V_{2,1} - a_2^+\right] + \left[V_{3,1} - a_3^+\right]$$
$$= (0{,}368 - 0{,}368) + (0{,}209 - 0{,}209) + (0{,}104 - 0{,}104)$$
$$= 0$$

Untuk melihat hasil lengkap dari perhitungan pada setiap alternatif di masing-masing sub parameter, data tersebut disajikan dalam Tabel 4.18

Tabel 4.18 Hasil jarak manhattan maksimum *technical economic view*

Alternatif	*Manhattan Maximum*	Nilai
Selenium	D_1^+	0
Cypress	D_2^+	0,368
Playwright	D_3^+	0,092

Selanjutnya adalah menghitung Jarak Manhattan Minimum. Hasil lengkap dari perhitungan setiap alternatif di masing-masing sub parameter, data tersebut disajikan dalam Tabel 4.19

Tabel 4.19 Hasil jarak manhattan minimum *technical economic view*

Alternatif	*Manhattan Minimum*	Nilai
Selenium	D_1^-	0,368
Cypress	D_2^-	0
Playwright	D_3^-	0,276

Jarak Manhattan Parameter *Testing Process View*

Berdasarkan persamaan (2,8), dan (2,9), berikut adalah hasil perhitungan Jarak Manhattan Maksimum dan Minimum setiap alternatif di masing-masing sub parameter, data tersebut disajikan dalam Tabel 4.20 dan Tabel 4.21

Tabel 4.20 Hasil jarak manhattan maksimum testing process view

Alternatif	*Manhattan Maximum*	Nilai
Selenium	D_1^+	0,745
Cypress	D_2^+	0
Playwright	D_3^+	0,08

Tabel 4.21 Hasil jarak manhattan minimum testing process view

Alternatif	*Manhattan Minimum*	Nilai
Selenium	D_1^-	0
Cypress	D_2^-	0,745
Playwright	D_3^-	0,665

Jarak Manhattan Parameter *Quality View*

Berdasarkan persamaan (2,8), dan (2,9), berikut adalah hasil perhitungan Jarak Manhattan Maksimum dan Minimum setiap alternatif di masing-masing sub parameter, data tersebut disajikan dalam Tabel 4.22 dan Tabel 4.23.

Tabel 4.22 Hasil jarak manhattan maksimum quality view

Alternatif	*Manhattan Maximum*	Nilai
Selenium	D_1^+	0,874
Cypress	D_2^+	0
Playwright	D_3^+	0,582

Tabel 4. 23 Hasil jarak manhattan minimum quality view

Alternatif	*Manhattan Maximum*	Nilai
Selenium	D_1^-	0
Cypress	D_2^-	0,874
Playwright	D_3^-	0,291

4.4.9 Penentuan *Positive Ideal Alternative (PIA)*

Mengacu pada persamaan (2.10), nilai *Positive Ideal Alternative* (PIA) untuk setiap parameter dapat ditentukan dengan mencari nilai terkecil pada Jarak Manhattan maksimum dan nilai terbesar pada Jarak Manhattan Minimum. Nilai PIA untuk setiap parameter adalah sebagai berikut.

Untuk parameter *Technical Economic View*, nilai PIA adalah :

PIA = 0 ; 0.368

Lalu untuk parameter *Testing Process View*, nilai PIA adalah :

PIA = 0 ; 0.745

Sedangkan untuk parameter *Quality View*, nilai PIA adalah :

PIA = 0 ; 0.874

4.4.10 Identifikasi Peringkat

Tahap ini adalah langkah terakhir yang digunakan untuk menentukan peringkat berdasarkan nilai terendah, yang menunjukkan jarak terpendek dari alternatif yang ideal menggunakan persamaan (2,11). Hal ini menghasilkan nilai R_i untuk masing-masing parameter.

Framework Terbaik pada Parameter *Technical Economic View*

Dengan menggunakan persamaan (2,11), hasil perhitungan nilai R_i untuk *framework* Selenium adalah sebagai berikut.

$$\sqrt{(0-0)^2 + (0{,}368 - 0{,}368)^2} = \sqrt{0+0} = 0$$

Untuk melihat hasil lengkap dari perhitungan pada setiap *framework*, data tersebut disajikan dalam Tabel 4.24, Tabel 4.25 dan Tabel 4.26.

Tabel 4.24 Hasil peringkat parameter *technical economic view*

Nama *Framework*	R_i	Peringkat
Selenium	0	1
Cypress	0,520	3
Playwright	0,130	2

Framework Terbaik pada Parameter *Testing Process View*

Berikut merupakan hasil peringkat *framework* pada parameter *Testing Process View*:

Tabel 4.25 Hasil peringkat parameter *testing process view*

Nama *Framework*	R_i	Peringkat
Selenium	1.236	3
Cypress	0	1
Playwright	0,823	2

Framework* Terbaik pada Parameter *Quality View

Berikut merupakan hasil peringkat *framework* pada parameter *Quality View*:

Tabel 4.26 Hasil peringkat parameter

Nama *Framework*	R_i	**Peringkat**
Selenium	0,747	3
Cypress	0	1
Playwright	0,245	2

4.5 Analisis Perbandingan

Pada tahap ini, penulis akan melakukan proses analisis perbandingan. Setelah mendapat hasil nilai R_i dari ketiga parameter yang telah ditetapkan, selanjutnya penulis akan melakukan analisis kembali untuk membandingkan nilai dari ketiga parameter. Menurut (Ayan dkk., 2023; Ezell dkk., 2021) menggunakan perhitungan *Weighted Scoring Model* ini mempermudah dan menguantifikasi proses pengambilan keputusan yang kompleks dengan mempertimbangkan berbagai faktor penting, terutama untuk mendapatkan hasil akhir yang lebih *valid* agar perbandingan kinerja *test automation framework* bisa terukur secara keseluruhan. Berikut merupakan tahap-tahap pada *Weighted Scoring Model.*

4.5.1 Peringkat untuk setiap Parameter

Berdasarkan hasil peringkat yang penulis peroleh, berikut merupakan tabel peringkat beserta bobot (w) dari data pada Tabel 4.27 untuk setiap *framework.*

Tabel 4.27 Peringkat dari setiap parameter dan bobot

Parameter	Selenium	Cypress	Playwright	Bobot
Technical Economic View	1	3	2	0,27
Testing Process View	3	1	2	0,64
Quality View	3	1	2	0,09

4.5.2 Menghitung Skor Tertimbang (*Weighted Scoring*)

Berikut merupakan contoh perhitungan Skor tertimbang pada *framework* Selenium.

$$\text{Skor} = (1 \times 0.27) + (3 \times 0.64) + (3 \times 0.09) = 0.27 + 1.92 + 0.27 = 2.46$$

Untuk melihat hasil lengkap dari perhitungan pada setiap *framework*, data tersebut disajikan dalam Tabel 4.28

Tabel 4.28 Hasil perhitungan skor tertimbang

Nama *Framework*	Total Skor	Peringkat
Selenium	2,46	3
Cypress	1,54	1
Playwright	2,00	2

4.5.3 Alternatif Terbaik

Berdasarkan perhitungan di atas, alternatif terbaik adalah Cypress dengan total skor terkecil (1.54). Cypress menempati peringkat pertama, diikuti oleh Playwright di peringkat kedua, dan Selenium di peringkat ketiga.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini bertujuan untuk membandingkan kinerja tiga *framework* otomatisasi pengujian, yaitu Selenium, Cypress, dan Playwright, dalam konteks pengujian UI (*User Interface*) atau *functional test* pada platform *website*. Penelitian ini dilakukan untuk menjawab rumusan masalah yang telah ditentukan, yaitu merancang *test case* untuk fitur-fitur pada *website* dan membandingkan hasil pengujian UI dari masing-masing *framework* berdasarkan parameter *Technical and Economic View, Testing Process View, dan Quality View* dengan metode *Distance to the Ideal Alternative* (DIA). Dari hasil penelitian dan analisis yang telah dilakukan, dapat disimpulkan bahwa:

1. Cara Menganalisis *Framework*: Dalam penelitian ini, ketiga *framework*, yaitu Selenium, Cypress, dan Playwright, Dengan memilih 3 parameter/10 sub parameter, menentukan bobot, membuat *test case*s, mengeksekusi *test*, dan mencatat hasil eksekusi. Selanjutnya dilakukan penghitungan dengan metode DIA untuk memilih *framework* terbaik. Hasilnya menunjukkan bahwa Cypress memiliki dokumentasi dan kemudahan penggunaan yang lebih baik dibandingkan dengan Selenium dan Playwright. Ketiga *framework* diuji dalam hal kecepatan dan efisiensi saat menjalankan tes. Cypress menunjukkan waktu eksekusi yang lebih cepat dan lebih sedikit kegagalan tes dibandingkan dengan Selenium dan Playwright. Cypress juga unggul dalam hal penyediaan laporan tes yang komprehensif dan mudah dipahami oleh penguji, dibandingkan dengan Selenium dan Playwright.
2. Hasil Analisis Perbandingan: Pada parameter *Technical Economic View*, berdasarkan analisis menggunakan metode DIA, Cypress memiliki skor terbaik dalam hal efisiensi teknis dan biaya operasional. Pada parameter *Testing Process View*, Cypress menunjukkan proses pengujian yang lebih efisien dan stabil dibandingkan dengan Selenium dan Playwright, berdasarkan analisis *test case*. Pada parameter *Quality View*, Cypress juga memberikan hasil pengujian dengan tingkat akurasi dan konsistensi yang lebih baik, sesuai dengan parameter kualitas yang diukur dalam penelitian ini.

Penelitian ini berhasil menjawab rumusan masalah yang telah ditentukan, yaitu menganalisis penggunaan masing-masing *framework* dalam menulis skrip tes, menjalankan tes, dan menghasilkan laporan tes, serta membandingkan kinerja ketiga *framework* berdasarkan

parameter *Technical and Economic View, Testing Process View*, dan *Quality View* menggunakan metode *Distance to the Ideal Alternative* (DIA). Dari hasil penelitian, dapat disimpulkan bahwa Cypress adalah *framework* terbaik untuk pengujian UI pada *platform website*, dengan keunggulan dalam kemudahan penggunaan, efisiensi, dan kualitas hasil pengujian.

5.2 Saran

Dari serangkaian tahapan yang telah dilakukan dalam penelitian ini, terdapat beberapa kendala dan batasan yang dihadapi, terutama terkait dengan pengujian menggunakan Selenium. Berikut beberapa saran yang diberikan untuk penelitian selanjutnya:

1. Memantau Pembaruan: Dalam pengujian menggunakan Selenium, terjadi beberapa kendala akibat *bug* yang muncul setelah *update* dari ChromeDriver dan Google Chrome. Disarankan untuk selalu memantau pembaruan dari alat pengujian yang digunakan dan mencari solusi alternatif jika terjadi masalah serupa di masa mendatang.
2. Penggunaan Komunitas: Lebih aktif dalam mencari solusi di komunitas *online* seperti GitHub, Stackoverflow dan Google Groups untuk mendapatkan informasi terbaru dan solusi dari masalah yang dihadapi.
3. Pengembangan Fitur: Menambahkan fitur-fitur baru yang dapat meningkatkan efisiensi dan efektivitas pengujian, seperti integrasi dengan alat CI/CD (*Continuous Integration/Continuous Deployment*).
4. Pengujian Lebih Lanjut: Melakukan pengujian lebih lanjut dengan skenario pengujian yang lebih kompleks dan jumlah *test case* yang lebih banyak untuk mendapatkan hasil yang lebih representatif.

DAFTAR PUSTAKA

Abdulwareth, A. J., & Al-Shargabi, A. A. (2021). Toward a Multi-Criteria Framework for Selecting Software Testing Tools. *IEEE Access*, *9*, 158872–158891. https://doi.org/10.1109/ACCESS.2021.3128071

Ayan, B., Abacıoğlu, S., & Basilio, M. P. (2023). A Comprehensive Review of the Novel Weighting Methods for Multi-Criteria Decision-Making. *MDPI Information*, *14*(5). https://doi.org/10.3390/info14050285

Badkar A. (2023, August 27). *Best Test Automation Frameworks (Updated 2023)*. Https://Www.Browserstack.Com/Guide/Best-Test-Automation-Frameworks.

Banerjee, I., Nguyen, B., Garousi, V., & Memon, A. (2013). Graphical user interface (GUI) testing: Systematic mapping and repository. *Information and Software Technology*, *55*(10), 1679–1694. https://doi.org/10.1016/j.infsof.2013.03.004

Barus, A. C., & Siburian, L. (2019). *STUDI PERBANDINGAN ALAT PENGUJIAN OTOMATIS UNTUK APLIKASI ANDROID*. *6*(6), 645–654. https://doi.org/10.25126/jtiik.20196953

Brahmbhatt, K. H. (2023). Comparative analysis of selecting a test automation framework for an e-commerce website. *Tallinn University of Technology*.

Brooke, J. (2013). SUS: a retrospective. *JUS Journal of Usability Studies*, *8*, 29–40.

Dias, T., Batista, A., Maia, E., & Praça, I. (2023). *TestLab: An Intelligent Automated Software Testing Framework*.

Ezell, B., Lynch, C. J., & Hester, P. T. (2021). Methods for Weighting Decisions to Assist Modelers and Decision Analysts: A Review of Ratio Assignment and Approximate Techniques. *MDPI Applied Sciences*, *11*(21). https://doi.org/10.3390/app112110397

Gamido, H. V., & Gamido, M. V. (2019). Comparative review of the features of automated software testing tools. *International Journal of Electrical and Computer Engineering*, *9*(5), 4473–4478. https://doi.org/10.11591/ijece.v9i5.pp4473-4478

Khaliq, Z., Farooq, S. U., & Khan, D. A. (2022). A deep learning-based automated framework for functional User Interface testing. *Information and Software Technology*, *150*, 106969. https://doi.org/10.1016/j.infsof.2022.106969

Melia, S., & Putra, P. F. (2023). Analisis Perbandingan Tools Pengujian Otomatis pada GUI Aplikasi Berbasis WEB. *SENTIMAS : Seminar Nasional Penelitian Dan Pengabdian Masyarakat*, *PROSIDING 2023*, 267–273.

Merina, C., Anggraini, N., Afrizal, S. H., & Hakiem, N. (2018). *A Comparative Analysis of Test Automation Frameworks Performance for Functional Testing in Android-Based Applications using the Distance to the Ideal Alternative Method*.

Mobaraya, F., & Ali, S. (2019). *Technical Analysis of Selenium and Cypress as Functional Automation Framework for Modern Web Application Testing*. 27–46. https://doi.org/10.5121/csit.2019.91803

Mwaura, W. (2021). *End-to-End Web Testing with Cypress: Explore techniques for automated frontend web testing with Cypress and JavaScript*. Packt Publishing. http://ieeexplore.ieee.org/document/10162884

Nagabushanam, D., S, S., Vijayasree, D., Sai Roopa, N., & Arun, A. (2022). *A Review on the Process of Automated Software Testing*. https://doi.org/10.48550/arXiv.2209.03069

Nguyen, A., Le, B., & Nguyen, V. (2019). Prioritizing automated user interface tests using reinforcement learning. *ACM International Conference Proceeding Series*, 56–65. https://doi.org/10.1145/3345629.3345636

Pelivani, E., Besimi, A., & Cico, B. (2021). A Comparative Study of UI Testing Framework. *2021 10th Mediterranean Conference on Embedded Computing, MECO 2021*. https://doi.org/10.1109/MECO55406.2022.9797165

Raihansyah, Yuyun Dwi Lestari, & Yessi Fitri Anissa Lubis. (2022). Sistem Pendukung Keputusan Pemilihan Tempat Kegiatan Olahraga di Medan Dengan Metode The Distance To The Ideal Alternative (DIA). *Jurnal Komputer Teknologi Informasi Sistem Komputer*, *1*(2), 1–9.

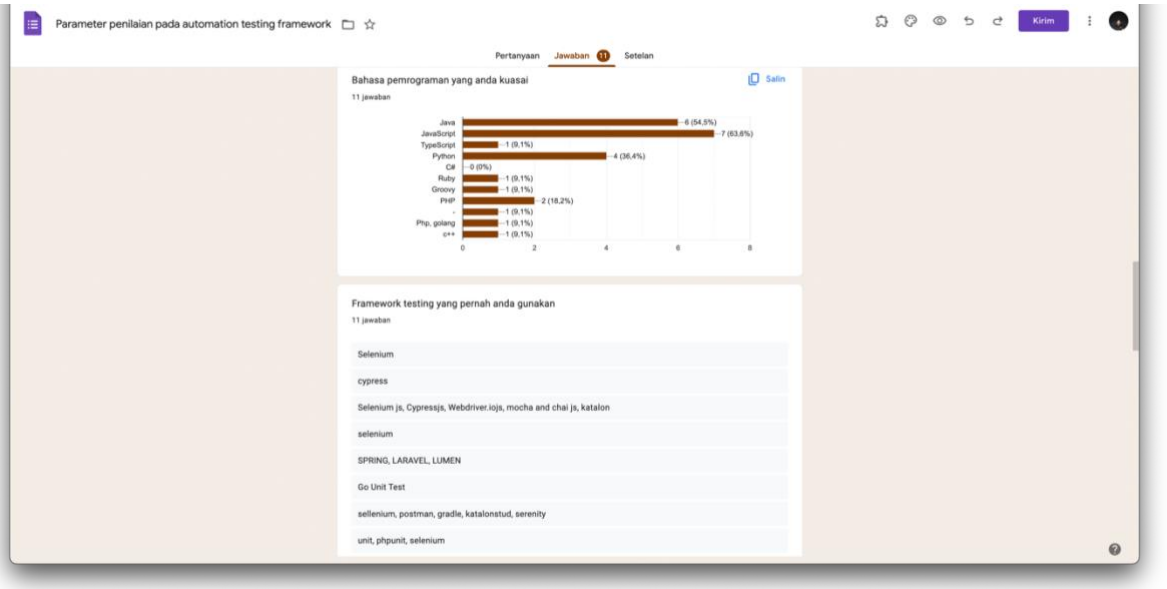
Rijoly, M., Sapulette, N. T., Tomasouw, B. P., & Patty, D. (2023). Penerapan Metode The Distance To The Ideal Alternative (DIA) Untuk Menyelesaikan Pegawai Di PT. Fast Food Indonesia (KFC Indonesia) Kakialy Tanah Tinggi, Ambon. *Tensor: Pure and Applied Mathematics Journal*, *4*(1), 13–20. https://doi.org/10.30598/tensorvol4iss1pp13-20

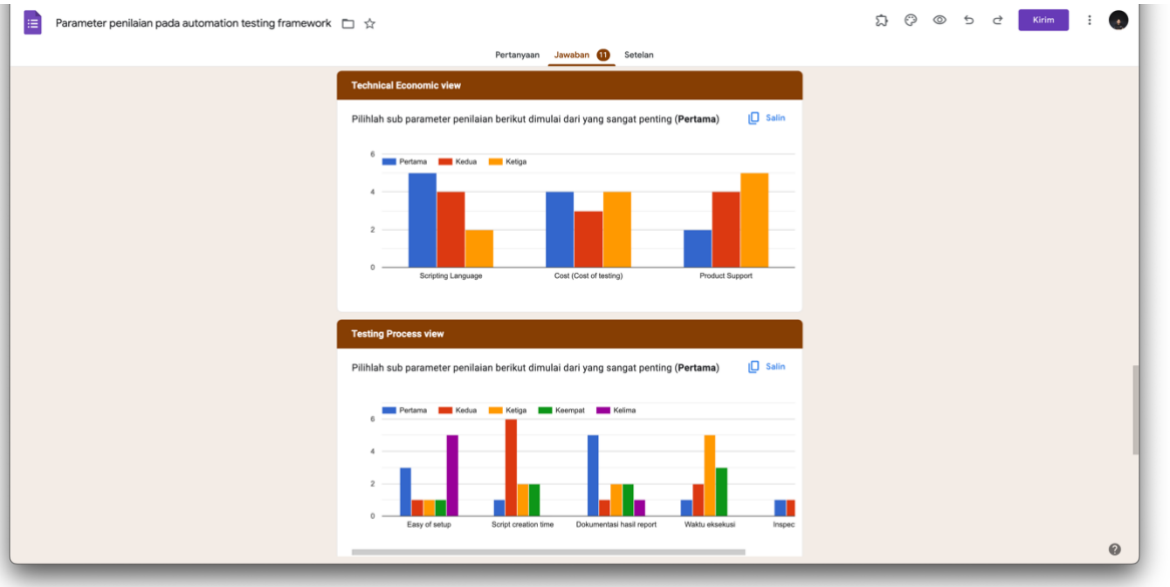
Saputra, B. D., & Stefanie, A. (2023). Automation Testing Api, Android, dan Website Menggunakan Serenity Bdd Pada Software Sistem Manajemen Rumah Sakit. *Jurnal Ilmiah Wahana Pendidikan, Mei*, *2023*(10), 114–126. https://doi.org/10.5281/zenodo.7983405

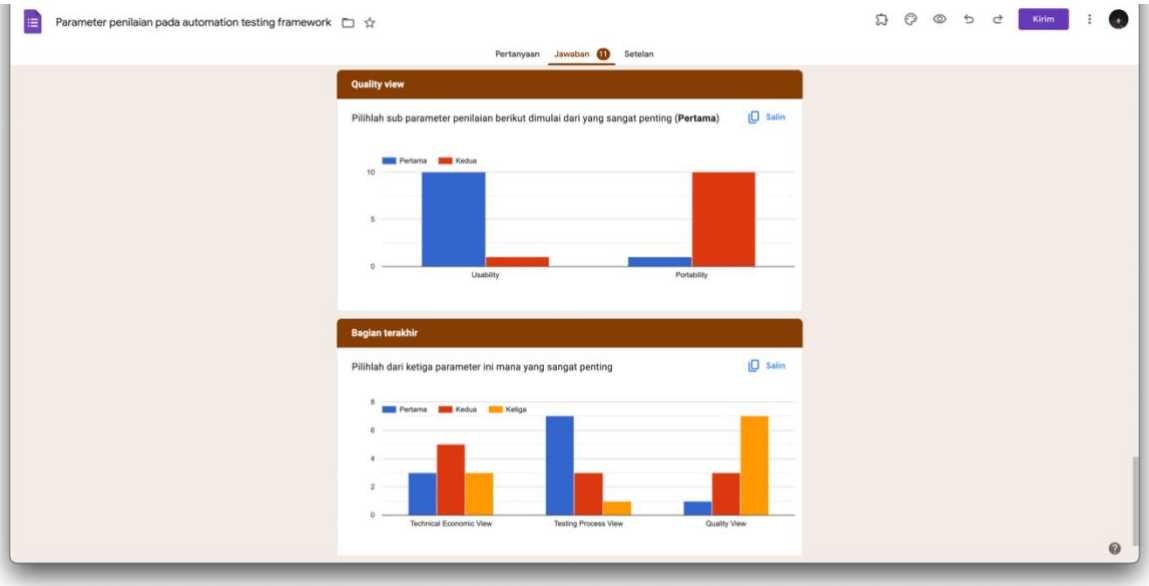
Thooriqoh, H. A., Annisa, T. N., & Yuhana, U. L. (2021). Selenium Framework for Web Automation Testing: A Systematic Literature Review. *Jurnal Ilmiah Teknologi Informasi*, *19*, 65–76.

LAMPIRAN

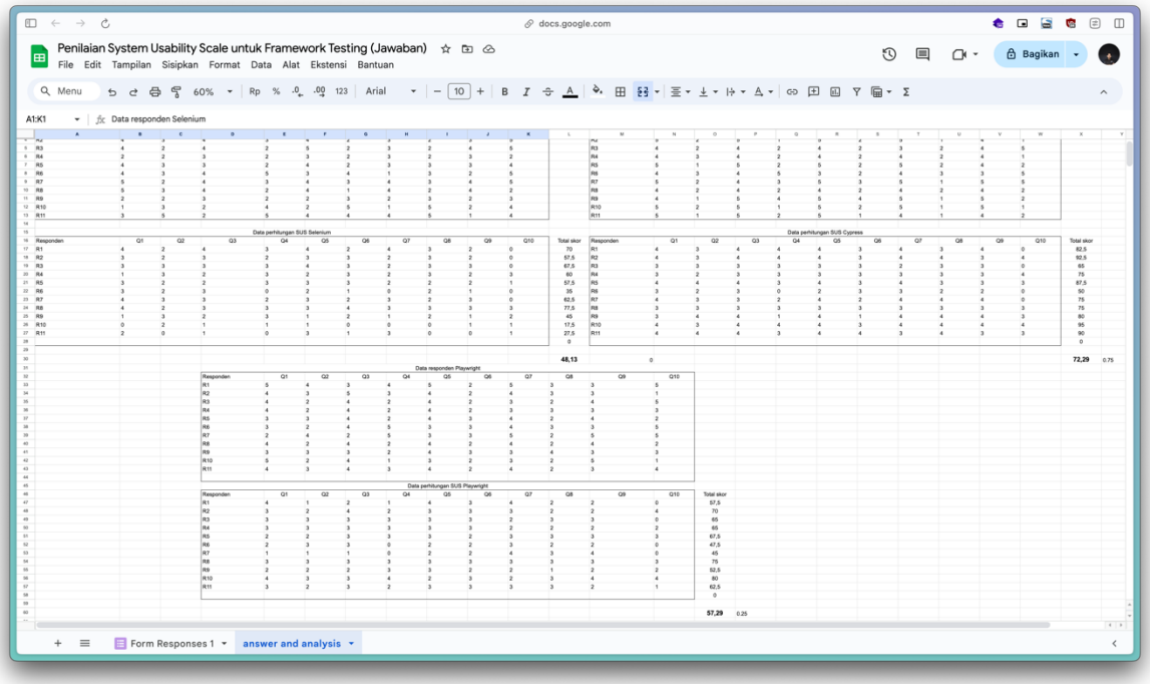
1. Proses Pembobotan Parameter dan Nilai

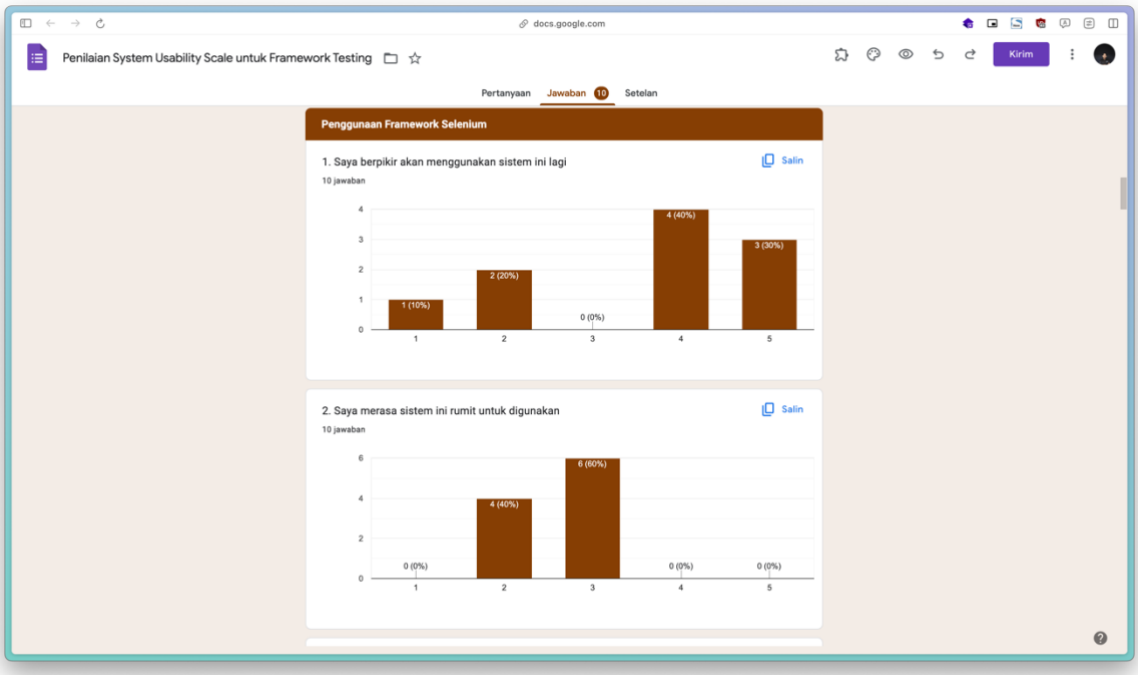


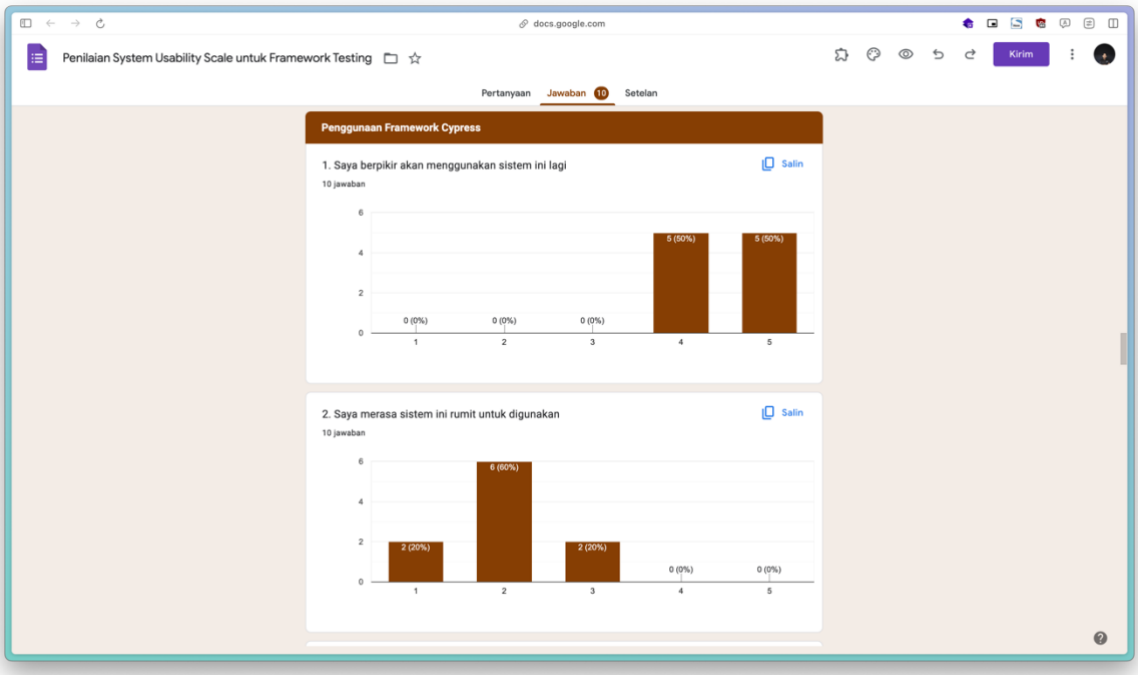




2. Proses penghitungan SUS




docs.google.com
Penilaian System Usability Scale untuk Framework Testing
Kirim
Pertanyaan
Jawaban 10
Setelan
Penggunaan Framework Selenium
1. Saya berpikir akan menggunakan sistem ini lagi
Salin
10 jawaban
1 (10%)
2 (20%)
0 (0%)
4 (40%)
3 (30%)
2. Saya merasa sistem ini rumit untuk digunakan
Salin
10 jawaban
0 (0%)
4 (40%)
6 (60%)
0 (0%)
0 (0%)


docs.google.com
Penilaian System Usability Scale untuk Framework Testing
Kirim
Pertanyaan
Jawaban 10
Setelan
Penggunaan Framework Cypress
1. Saya berpikir akan menggunakan sistem ini lagi
Salin
10 jawaban
0 (0%)
0 (0%)
0 (0%)
5 (50%)
5 (50%)
2. Saya merasa sistem ini rumit untuk digunakan
Salin
10 jawaban
2 (20%)
6 (60%)
2 (20%)
0 (0%)
0 (0%)

3. Log pengerjaan testing pada *framework*

Pengujian Tugas Akhir

Tanggal	Backlog	Keterangan	Jam Mulai	Jam Selesai	Status
23-Feb	Setup Cypress, konfigurasi awal, latihan membuat case		14:00	16:00	Done
24-Feb	Inspection Element, input code for automation test		19:00	21:00	Done
25-Feb	retest, memastikan aman		20:00	20.30	Done
30-Mar	Setup Playwright, konfigurasi awal, latihan membuat case		10:00	15:00	Done
31-Mar	Inspection Element, input code for automation test		20:00	0:00	Done
1-Apr	retest, memastikan aman		9:00	11:00	Done
12-Apr	Setup Selenium, konfigurasi awal, latihan membuat case	Kendala pada konfigurasi	8:00	16:00	Done
13-Apr	Setup Selenium, konfigurasi awal, latihan membuat case		8:00	16:00	Done
14-Apr	Inspection Element, input code for automation test		9:00	12:00	Done
15-Apr	Inspection Element, input code for automation test		13:00	18:00	Done
16-Apr	retest, memastikan aman		9:00	10:00	Done