SKRIPSI

ANALISIS PEMANFAATAN PLAYWRIGHT UNTUK PENGUJIAN APLIKASI BERBASIS WEB (STUDI KASUS: SISTEM MANAJEMEN JARINGAN)

Disusun Oleh:

N a m a : Anisa Amalia

NIM : 18523157

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

ANALISIS PEMANFAATAN PLAYWRIGHT UNTUK PENGUJIAN APLIKASI BERBASIS WEB (STUDI KASUS: SISTEM MANAJEMEN JARINGAN)

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Anisa Amalia

NIM : 18523157

Yogyakarta, 10 Februari 2022

Pembimbing,

(Andhik Budi Cahyono, S.T., M.T.)

HALAMAN PENGESAHAN DOSEN PENGUJI

ANALISIS PEMANFAATAN PLAYWRIGHT UNTUK PENGUJIAN APLIKASI BERBASIS WEB (STUDI KASUS: SISTEM MANAJEMEN JARINGAN)

TUGAS AKHIR JALUR MAGANG

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

Yogyakarta, 14 Januari 2022

Tim Penguji

Ketua Penguji

Andhik Budi Cahyono, S.T., M.T. _____

Anggota 1

Andhika Giri Persada, S.Kom., M.Eng. _____

Anggota 2

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

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

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

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Anisa Amalia
NIM	:	18523157

Tugas akhir dengan judul:

ANALISIS PEMANFAATAN PLAYWRIGHT UNTUK PENGUJIAN APLIKASI BERBASIS WEB (STUDI KASUS: SISTEM MANAJEMEN JARINGAN)

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, 20 Desember 2021

(Anisa Amalia)

HALAMAN PERSEMBAHAN

Tugas Akhir ini saya persembahkan untuk kedua orangtua yang telah mendukung, membantu baik secara mental dan doa, serta menjadi tempat berkeluh kesah selama masa perkuliahan. Laporan ini juga saya persembahkan untuk diri saya yang telah berusaha dan berjuang hingga saat ini.

HALAMAN MOTO

*“*Karena sesungguhnya sesudah kesulitan itu ada kemudahan”

– QS. Al-Insyirah: 5

“Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya.”

– QS Al Baqarah 286

KATA PENGANTAR

Alhamdulillah, puji syukur ke hadirat Allah SWT yang telah melimpahkan rahmat serta hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Analisis Pemanfaatan Playwright untuk Pengujian Otomatis Aplikasi Berbasis Web (Studi Kasus: Sistem Manajemen Jaringan). Tidak lupa shalawat serta salam kepada Nabi Muhammad SAW yang telah membimbing dan memberikan tuntunan agama yang sempurna bagi seluruh alam.

Laporan ini disusun untuk memenuhi persyaratan tugas akhir penjaluran magang untuk memperoleh gelar sarjana pada Program Studi Informatika. Fakultas Teknologi Industri, Universitas Islam Indonesia. Adapun dalam penyusunan Tugas Akhir, Penulis mendapatkan bimbingan, dukungan, serta bantuan dari banyak pihak. Untuk itu, Penulis ingin mengucapkan terima kasih kepada :

1. Allah SWT yang telah memberikan kesehatan, kemudahan, serta kekuatan.
2. Keluarga khususnya kedua orangtua yang selalu memberikan do'a, dukungan, serta kepercayaan penuh kepada Penulis.
3. Bapak Andhik Budi Cahyono, S.T., M.T. selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk membimbing dan mengarahkan selama proses magang hingga dalam penyelesaian Tugas Akhir.
4. Pimpinan dan staff PT Javan Cipta Solusi yang telah memberikan ilmu, kepercayaan serta bimbingan selama proses magang berlangsung.
5. Para dosen Program Studi Informatika Universitas Islam Indonesia yang telah memberikan bekal ilmu kepada penulis selama perkuliahan
6. Teman seperjuangan kuliah dan magang yang saling membantu, mendukung, dan meluangkan waktu untuk mendengarkan keluh kesah selama perkuliahan.

Atas bantuan berbagai pihak, Penulis dapat menyelesaikan laporan Tugas Akhir. Namun, laporan ini masih jauh dari kata sempurna sehingga Penulis membutuhkan saran guna menyempurnakan laporan ini. Akhir kata, semoga laporan ini dapat bermanfaat bagi pembaca. Terima kasih.

Yogyakarta, 20 Desember 2021

(Anisa Amalia)

SARI

Pengujian menjadi fase penting dalam *System Development Life Cycle* (SDLC) karena di setiap pengembangan perangkat lunak adanya *bug* yang tidak bisa dihindari. Dalam pengujian terdapat ratusan *test case* yang harus dieksekusi, untuk mempermudah pengeksekusiannya dapat menerapkan pengujian otomatis. Saat ini pengujian otomatis menjadi kebutuhan kompetitif yang banyak diimplementasikan oleh berbagai perusahaan, salah satunya yaitu PT Javan Cipta Solusi yang mengimplementasikan pengujian otomatis menggunakan Playwright. Akan tetapi yang perlu diperhatikan dalam pengujian otomatis yaitu pemilihan alat uji yang tepat sesuai dengan kebutuhan dan kondisi sumber daya manusia di dalam proyek tersebut.

Untuk mengetahui kinerja Playwright sebagai alat uji web terbaru maka dilakukan percobaan pengujian pada Sistem Manajemen Jaringan. Selain itu, dilakukan analisis menggunakan metode komparatif untuk mengetahui perbedaan antara alat uji terbaru Playwright dengan alat uji populer Selenium. Hal tersebut membantu untuk memberikan referensi terkait pemilihan alat uji. Disimpulkan bahwa fitur Playwright lebih cocok diimplementasikan oleh penguji yang baru mengenal pengujian otomatis karena kemudahan dalam penggunaannya. Fitur Playwright Codegen sangat membantu dalam pembuatan *script test*. Adapun Selenium WebDrivers akan lebih mudah diimplementasikan oleh penguji yang telah terbiasa dengan pengujian otomatis dan memiliki ilmu dasar mengenai pemrograman.

Kata kunci: Pengujian otomatis, Playwright, Selenium, Sistem Manajemen Jaringan.

GLOSARIUM

Command prompt	baris perintah penerjemah yang tersedia di sistem operasi Windows.
Debug	langkah untuk menelusuri kesalahan kode program.
Script test	file kode pemrograman untuk melakukan pengujian otomatis
Test case	dokumen pengujian yang runtutan dan skenario alur pengujian.
Test runner	fitur Playwright untuk menjalankan pengujian *multiple test*
Pengujian otomatis	pengujian yang dilakukan menggunakan bantuan alat uji

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengujian menjadi fase penting dalam *System Development Life Cycle* (SDLC). Fase tersebut bertujuan untuk mencari sebanyak mungkin kesalahan yang ada serta mengevaluasi kualitas sistem (Jin & Xue, 2011). Pengujian didefinisikan sebagai proses verifikasi serta validasi hasil pengembangan sistem berdasarkan kebutuhan pemangku kepentingan. Verifikasi merupakan aktivitas yang menjamin penerapan perangkat lunak telah sesuai dengan fungsinya. Adapun validasi merupakan aktivitas yang memastikan bahwa perangkat lunak yang dikembangkan telah memenuhi kebutuhan klien (Sulistyanto & SN, 2014). Proses tersebut dapat dilakukan dengan cara mengevaluasi perangkat lunak dari sisi spesifikasi kebutuhan, logika pemrograman, dan *user experience*. Adapun pentingnya pengujian perangkat lunak yaitu untuk meminimalisir adanya *bug* yang berpotensi menimbulkan kerugian biaya, sumber daya manusia, serta menciptakan kepercayaan klien kepada perusahaan.

Pengujian perangkat lunak dapat dilakukan dengan dua cara yaitu pengujian manual dan otomatis. Pengujian manual adalah pengujian yang mengandalkan ketelitian penguji karena dilakukan manual tanpa menggunakan bantuan alat uji. Adapun pengujian otomatis dijalankan menggunakan bantuan *automation testing tools* serta membutuhkan pendokumentasian pemrograman dalam sebuah *script test* (Sharma & Angmo, 2014). Selama proses pengujian terdapat ratusan *test case* yang harus dieksekusi. Jika diuji manual akan membutuhkan waktu yang lama serta *human error* tidak bisa dihindarkan. Permasalahan tersebut dapat diatasi dengan pengujian otomatis.

Salah satu hal yang perlu diperhatikan dalam pengujian otomatis adalah pemilihan *automation testing tools* yang tepat. Pertimbangan dalam pemilihan alat pengujian dapat disesuaikan dengan kebutuhan pengujian serta kondisi sumber daya manusia dalam tim proyek. Tidak hanya itu, kelebihan serta kelemahan dari masing-masing alat uji juga harus dipertimbangkan.

Pada makalah ini akan disampaikan hasil analisis implementasi pengujian otomatis aplikasi berbasis web pada Sistem Manajemen Jaringan. Alat uji yang akan dilakukan pembahasan yaitu Playwright dan Selenium. Hal tersebut dikarenakan Playwright merupakan alat uji web terbaru berbasis lintas browser yang digunakan oleh PT Javan Cipta Solusi dalam melakukan pengujian Sistem Manajemen Jaringan. Adapun Selenium yaitu alat uji web berbasis lintas browser yang

telah populer dan disebutkan menjadi tolak ukur alat uji pada masa mendatang (Saravanan & Prasad, 2016). Perbandingan alat uji tersebut digunakan untuk membandingkan alat uji terbaru dengan alat uji populer yang telah dipercaya oleh banyak pengembang. Informasi tersebut dapat dijadikan sebagai bahan pertimbangan dalam pemilihan alat uji web berbasis *open source*.

1.2 Ruang Lingkup Magang

Pelaksanaan magang di PT Javan Cipta Solusi (Javan) berlangsung selama 6 bulan dimulai dari Maret 2021 – September 2021. Javan merupakan sebuah *software house* yang berfokus pada pengembangan proses bisnis berbasis teknologi. Adapun selama kegiatan magang berlangsung, penulis telah berkontribusi dalam beberapa proyek di lembaga pemerintah maupun swasta, yaitu *Whistleblowing System* dan Sistem Manajemen Jaringan.

Whistleblowing System merupakan aplikasi berbasis web yang digunakan untuk melaporkan tindakan penyalahgunaan atau kecurangan dalam pengadaan barang dan jasa di Indonesia. Rahasia pelapor akan tetap terjaga dikarenakan semua laporan bersifat anonim. Penulis berkontribusi selama 1,5 bulan menjadi *System Analyst* dan *Quality Assurance*. Tanggung jawab dari peran tersebut yaitu melakukan analisis kebutuhan klien dan melakukan pengujian sistem.

Adapun Sistem Manajemen Jaringan merupakan aplikasi berbasis web yang digunakan oleh *vendor* telekomunikasi untuk melakukan permintaan perubahan kapasitas jaringan. Dalam proyek ini, penulis telah berkontribusi selama 4,5 bulan menjadi seorang *Quality Assurance* yang bertanggung jawab terhadap pengujian kualitas sistem. Proyek ini merupakan proyek dengan waktu paling lama yang dikerjakan oleh penulis selama kegiatan magang berlangsung. Oleh karena itu, dalam laporan ini akan fokus membahas pekerjaan selama berkontribusi dalam proyek Sistem Manajemen Jaringan yaitu melakukan pengujian.

Sistem Manajemen Jaringan memiliki cakupan yang luas dalam pengembangannya karena terdiri dari beberapa sub-sistem yang datanya saling terintegrasi yaitu *Create Link, Upgrade & Downgrade, AF Online, Renewal,* serta *Dismantle.* Adapun aktivitas yang dilakukan selama bergabung di proyek Sistem Manajemen Jaringan, yaitu:

a. Membuat dokumen *test case* untuk acuan dalam pengujian
b. Melakukan analisis kebutuhan klien
c. Melakukan pengujian sistem dan API secara manual
d. Melakukan pengujian sistem secara otomatis menggunakan Playwright
e. Melakukan pendokumentasian semua hasil *script test* di Gitlab
f. Melakukan analisis *root cause* dan pelaporan jika terjadi *bug*

Di dalam proyek Sistem Manajemen Jaringan, penulis memiliki tugas utama sebagai *Quality Assurance* yaitu melakukan pengujian otomatis menggunakan Playwright dan mendokumentasikan semua hasil *script test* ke dalam Gitlab untuk proses kolaborasi dengan tim. Adapun dalam laporan Tugas Akhir ini akan berfokus kepada pengimplementasian pengujian otomatis menggunakan Playwright. Hal tersebut dikarenakan aktivitas pengujian otomatis merupakan *task* yang paling banyak dikerjakan dan menjadi tanggung jawab utama selama berkontribusi di dalam proyek Sistem Manajemen Jaringan.

1.3 Tujuan

Laporan Tugas Akhir ini bertujuan untuk melakukan pengujian otomatis menggunakan Playwright pada Sistem Manajemen Jaringan. Selain itu juga akan dilakukan analisis menggunakan metode komparatif terhadap Playwright dan Selenium yang dapat dijadikan referensi dalam pemilihan alat pengujian otomatis berbasis web.

1.4 Manfaat

Manfaat yang diperoleh dari analisis pemanfaatan Playwright dalam pengujian aplikasi berbasis web adalah sebagai berikut :

a. Mengetahui kinerja Playwright sebagai alat pengujian otomatis terbaru melalui percobaan pengujian Sistem Manajemen Jaringan
b. Mengetahui perbedaan Playwright dan Selenium yang dapat dijadikan referensi dalam pemilihan alat pengujian otomatis berbasis web

1.5 Sistematika Penulisan

Sistematika penulisan disusun untuk memberikan gambaran umum isi laporan. Adapun susunan sistematika penulisan sebagai berikut :

a. BAB I: Pendahuluan

   Bab ini membahas mengenai latar belakang, ruang lingkup magang, tujuan, dan manfaat dan sistematika penulisan
b. BAB II: Dasar Teori

   Bab ini berisi ringkasan hasil analisis dari penelitian-penelitian terdahulu dan dasar teori yang berhubungan dengan topik laporan.
c. BAB III: Pelaksanaan Magang

Bab ini berisi gambaran selama magang dan penjelasan ruang lingkup pengujian secara manual maupun otomatis menggunakan Playwright. Dalam bab ini juga membahas hasil analisis Playwright dengan Selenium.

d. BAB IV: Refleksi Pelaksanaan Magang

   Bab ini membahas mengenai refleksi pelaksanaan magang di PT Javan Cipta Solusi dari sisi teknis maupun non teknis.

e. BAB V: Kesimpulan dan Saran

   Bab ini berisi kesimpulan dari hasil pengujian dan hasil analisis Playwright dengan Selenium. Dalam bab ini juga berisi saran bagi peneliti yang akan melakukan penelitian serupa.

BAB II
DASAR TEORI

2.1 Playwright

Playwright merupakan salah satu *automation testing tools* lintas browser terbaru berbasis *open source* yang dikembangkan oleh Microsoft (Playwright, 2021). Playwright dapat melakukan pengujian web dengan menggunakan browser Chromium, Webkit, Firefox dan mendukung 5 bahasa pemrograman yaitu JavaScript, Python, Java, .NET, dan TypeScript. Cara kerja Playwright Codegen yaitu merekam aktivitas penguji pada web yang sedang diuji melalui browser dan melakukan *generate* kode pemrograman dari setiap aktivitas uji. Pengeksekusian script kode dapat dijalankan dengan mode *headless* dan *headful.* Mode *headless* adalah pengeksekusian *script test* tanpa antarmuka (UI) sistem sedangkan mode *headful* akan menampilkan keseluruhan antarmuka sistem yang sedang diuji. Selain itu, terdapat juga Playwright *Test Runner* yang dapat digunakan untuk mengeksekusi *multipe test*.

2.2 Selenium

Selenium adalah seperangkat *automation testing tools* yang mendukung hampir semua web browser dan menyediakan kompatibilitas dengan bahasa pemrograman popular yaitu C#, Java, JavaScript, Ruby, Python, dan PHP. Selenium mendukung semua browser melalui penggunaan WebDriver. Setiap browser memiliki WebDriver tertentu yang disebut dengan *driver*. *Driver* adalah komponen yang bertanggung jawab untuk mendelegasikan ke browser dan menjadi perantara antara Selenium dengan web browser (Raghavendra, 2021). Terdapat 3 jenis Selenium yaitu Selenium IDE yang melakukan perekaman aktifitas pengujian pada sistem, Selenium WebDriver yang menggunakan protokol WebDriver sebagai perantara *script* pengujian dengan web browser dan Selenium Grid yang dapat digunakan bersamaan dengan Selenium WebDriver untuk melakukan pengujian *multiple test case* (Selenium, 2021).

2.3 Pengujian Otomatis

Pengujian menjadi fase penting dalam pembangunan perangkat lunak. Fase ini juga menjadi gerbang akhir sebelum sebuah aplikasi dirilis dan digunakan oleh penggunanya. Seorang penguji harus memiliki ketelitian, cerdas, kesabaran, serta tidak mudah puas dalam melakukan serangkaian pengujian. Hal tersebut dikarenakan penguji harus melakukan pengujian dari bagian terkecil

hingga secara keseluruhan. Oleh karena itu, ketika pengujian permasalahan *human error* tidak bisa dihindarkan.

Pengujian otomatis merupakan salah satu aktivitas pengujian dengan bantuan alat otomatis untuk mengeksekusi *script* pengujian sebelum sistem tersebut dirilis. Alat otomatis akan menjalankan program pengujian dengan mengevaluasi atau membandingkan sistem dengan ekspektasi kasus uji tanpa campur tangan manusia (Lawanna, 2012). Tujuan dari pengujian otomatis adalah untuk menyederhanakan upaya yang dilakukan dalam pengujian sebanyak mungkin sehingga mampu memaksimalkan waktu apalagi jika kasus uji yang dilakukan cukup banyak.

Dijabarkan bahwa pengujian otomatis mengeksekusi kasus uji lebih cepat daripada pengujian manual dan dapat diandalkan dari sisi ketepatan karena mampu menemukan cacat aplikasi lebih banyak (Sharma & Angmo, 2014). Pengujian otomatis juga efektif untuk pengujian regresi yang dilakukan ketika terdapat perbaikan atau pembaruan untuk memastikan bahwa perbaikan tersebut tidak menyebabkan *bug* baru pada sistem. Hal tersebut dikarenakan adanya kecepatan waktu yang didapatkan selama pengujian regresi dikarenakan pengujian tersebut akan menguji semua fitur maupun *flow* sistem sehingga jika dilakukan manual akan memakan waktu yang cukup banyak.

Pengujian otomatis menjadi kebutuhan kompetitif yang banyak diadopsi oleh berbagai perusahan teknologi. Maksud dari kebutuhan kompetitif adalah kebutuhan yang semakin waktu semakin meningkat. Adapun dalam dekade terakhir ini, kebutuhan alat uji telah mendominasi pasar dan alat uji berbasis *open source* paling banyak diadopsi dengan cepat oleh berbagai perusahaan. Salah satu alasan utamanya yaitu penggunaan alat uji otomatis memberikan keuntungan perusahaan untuk meningkatkan produktivitas, memberikan hasil yang akurat, memberikan laporan tepat waktu, dan mengurangi staff penguji sehingga mereka dapat melakukan kegiatan penting lainnya (Saravanan & Prasad, 2016).

Alat pengujian otomatis umumnya terdapat fitur untuk menjalankan skrip pengujian, melaporkan cacat, dan melaporkan status pengujian. Pengujian yang dilakukan dengan alat otomatis dapat dijalankan kapanpun oleh tim proyek. Hal tersebut juga membantu dalam kolaborasi pengujian antar tim penguji.

2.4 Kajian Pustaka

Terdapat beberapa penelitian yang telah dilakukan sebelumnya, terkait implementasi pengujian otomatis aplikasi berbasis web, salah satunya yaitu menggunakan Selenium. Dalam

penelitian (Karuniawati, Sri, & Hakim, 2015), dilakukan analisis implementasi Selenium dalam pengujian perangkat lunak berbasis web. Pada penelitian tersebut, penguji menggunakan metode *Equivalence Partitioning* yang membagi data input menjadi 2 kategori yaitu data valid dan invalid. Kedua kategori data tersebut menjadi data input untuk setiap pengujian. Hasil pengujian menyebutkan bahwa Selenium memiliki performa pengujian yang baik karena mampu menghasilkan *output* pengujian yang sesuai dengan kategori data input dengan persentase lulus uji 100%.

Penelitian yang membahas mengenai perbandingan tiga *automation testing tools* yaitu Selenium, Quick Test Professional, dan TestComplete (Kaur & Gupta, 2013). Dalam studi kasus tersebut, peneliti melakukan perbandingan berdasarkan lisensi, *application support*, bahasa, sistem operasi, *programming skills*, penggunaan, *report generation*, serta *platform dependency*. Hasil dari penelitian tersebut menyebutkan bahwa Selenium dan Quick Test Professional merupakan *automation testing tools* terbaik yang dapat digunakan untuk melakukan pengujian aplikasi berbasis web. Akan tetapi, Selenium unggul dari sisi *open source* sedangkan Quick Test Professional memiliki biaya lisensi dan *maintenance* yang mahal.

Dalam penelitian studi perbandingan automation testing tools lainnya, menjabarkan mengenai Selenium dalam 15 variabel pembanding yaitu *test development platform*, bahasa pemrograman, *programming skills*, kurva pembelajaran, instalasi, pembuatan *script test*, *object storage and maintenance*, gambar pendukung, *devOps integrations*, *test analytics, continuous integration*, dukungan, lisensi, dan biaya (Bhagat, Bhattacharjee, & Ratre, 2020). Disebutkan bahwa Selenium merupakan alat pengujian otomatis *open source* berbasis lintas browser yang telah mendukung bahasa pemrograman populer. Dalam penggunaannya, Selenium juga dapat diintegrasikan dengan banyak *tools* pihak ketiga.

Terdapat penelitian survei pengalaman penggunaan Selenium dari beberapa ahli dan orang-orang yang memiliki minat serta kepentingan terhadapnya. Survei dilakukan melalui Google Forms ke beberapa komunitas Selenium seperti Selenium Conf Slack dan Selenium Reddit (García, Gallego, Gortázar, & Munoz, 2020). Peneliti melakukan survei dalam 7 kategori yaitu informasi umum Selenium, tes pengembangan, sistem unit, frameworks, infrastruktur, dukungan komunitas, dan yang terakhir yaitu pertanyaan terbuka mengenai pengalaman menggunakan Selenium. Salah satu informasi yang didapatkan adalah Selenium WebDriver merupakan jenis Selenium yang paling banyak digunakan dengan persentase kurang lebih 99% (71 dari 72 responden). Selenium juga merupakan automation testing tools yang populer dan memiliki komunitas luas yang tersebar di berbagai negara.

Tidak hanya itu, terdapat penelitian yang menguji Selenium dengan mengintegrasikan dengan *tools* pihak ketiga (Gojare, Joshi, & Gaigaware, 2015). Selenium memiliki batasan tidak bisa melakukan *generate* hasil pengujian yang manfaatnya cukup penting untuk pendokumentasian dan analisis *root cause* jika terjadi kegagalan uji. Oleh karena itu, peneliti melakukan percobaan pengujian dan mengintegrasikan dengan TestNg. Hasil dari pengujian tersebut menyebutkan bahwa laporan pengujian berhasil dibuat dan berjalan dengan baik dalam Selenium. Hal tersebut dapat dijadikan solusi terkait batasan fitur pelaporan di Selenium.

Dari semua penelitian yang telah dipaparkan menunjukkan bahwa Selenium merupakan alat pengujian otomatis yang populer untuk menguji aplikasi berbasis web. Jenis Selenium yang populer dan paling banyak digunakan adalah Selenium WebDrivers karena performa stabil daripada jenis Selenium yang lain (García, Gallego, Gortázar, & Munoz, 2020). Data dan informasi yang disajikan dalam penelitian-penelitian tersebut akan menjadi informasi pendukung yang digunakan untuk membandingkan hasil pengujian dengan salah satu alat pengujian terbaru yaitu Playwright.

BAB III
PELAKSANAAN MAGANG

3.1 *On-Boarding* Magang

Sebelum bergabung ke proyek Sistem Manajemen Jaringan, penulis diharuskan untuk mengikuti *on-boarding* dengan tujuan untuk belajar memahami *flow* sistem, menginstalasi *tools* yang digunakan, serta mempelajari dokumentasi sistem. Dari *on-boarding* ini diharapkan untuk siap dan sudah memahami *flow* sistem secara baik sehingga tidak ada kendala ketika sudah berkontribusi langsung di proyek.

Aktivitas *on-boarding* yang telah dilakukan yaitu berhubungan dengan tugas utama untuk melakukan pengujian, seperti pembuatan *test case*, instalasi Playwright, integrasi dengan Gitlab, serta berlatih untuk membuat *script test* pengujian menggunakan Playwright. Proses berlatih dan beradaptasi dengan Playwright cukup mudah karena banyak diberikan training oleh *Lead* QA.

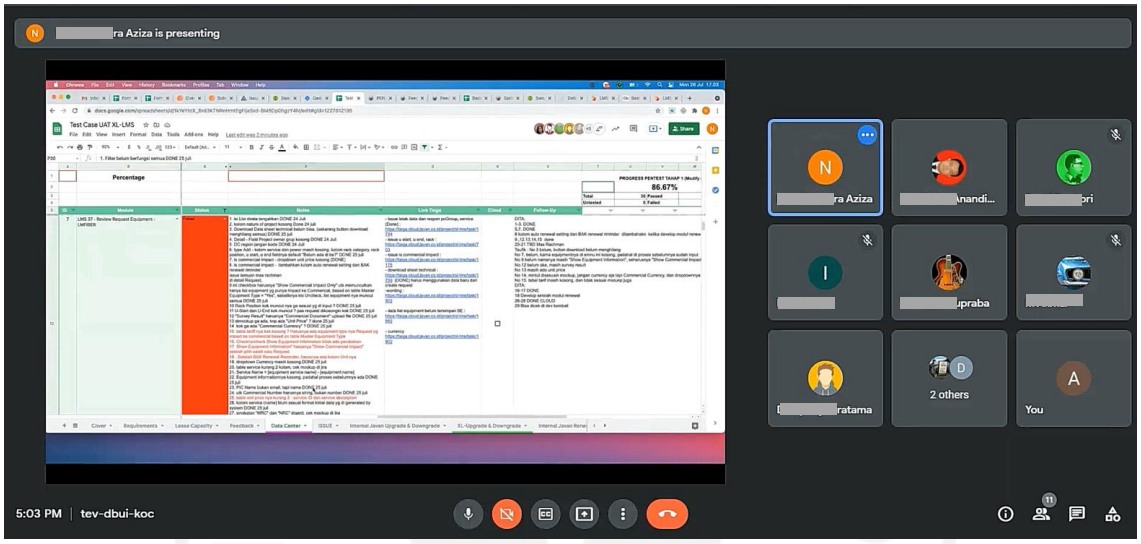
Proses belajar dan beradaptasi dalam menguji sistem menggunakan Playwright cukup singkat tetapi sangat mudah dipahami. Hal tersebut dikarenakan penginstalan dan penggunaan Playwright cukup mudah karena dalam pembuatan *script test* selain dibuat secara manual juga dapat dibuat otomatis menggunakan Playwright Codegen yang berfungsi untuk merekam aktivitas pengguna dalam kode pemrograman.

3.2 Manajemen Proyek

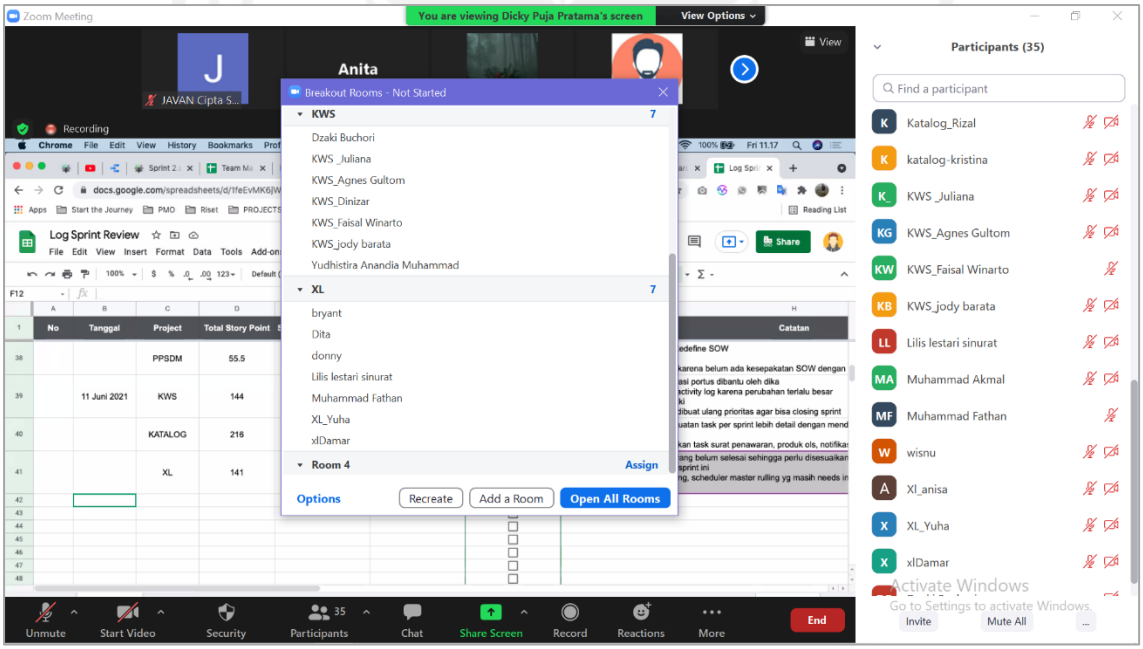
Dalam pengembangan proyek, PT Javan Cipta Solusi menggunakan kerangka kerja *Scrum* dengan jangka waktu setiap *sprint* yaitu 2 minggu. Setiap *sprint* dimulai dengan *Sprint Planning* yang bertujuan untuk menganalisis pekerjaan yang akan masuk ke dalam *sprint* tersebut. Adapun *sprint* akan diakhiri dengan *Sprint Review* yang bertujuan untuk melakukan inspeksi terhadap pekerjaan yang telah diselesaikan. Progress pengembangan harian dipaparkan melalui aktivitas *Daily Scrum* yang dilaksanakan setiap hari selama 15 menit.

Selama magang, penulis berkontribusi untuk memaparkan progress pengembangan sistem kepada klien dalam aktivitas *Sprint Planning, Daily Scrum* maupun *Sprint Review*. Dalam aktivitas tersebut, semua tim internal akan berkontribusi memaparkan hasil lingkup pekerjaanya. Adapun sebagai *Quality Assurance* yang melakukan dokumentasi dan pengerjaan pengujian otomatis, penulis berkewajiban untuk memaparkan hasil pengujian selama *sprint* yang sedang berlangsung. Informasi pengujian yang dipaparkan yaitu fitur yang diuji, hasil pengujian seberapa banyak *test case* yang lulus dan gagal uji, serta hambatan maupun tantangan selama melakukan pengujian. Adapun pada Gambar 3.1 merupakan hasil dokumentasi dari aktivitas Daily Scrum yang langsung

dilanjutkan dengan *Sprint Planning* tim internal, sedangkan pada Gambar 3.2 merupakan hasil dokumentasi dari *Sprint Review* internal yang dilakukan setiap akhir *sprint*.



Gambar 3.1. Dokumentasi pelaksanaan *sprint planning*



Gambar 3.2. Dokumentasi hasil *sprint review*

Taiga merupakan alat manajemen proyek yang digunakan oleh perusahaan untuk membantu mengelola pekerjaan dalam pengembangan sistem. Semua *task* akan didokumentasikan ke dalam Taiga untuk mempermudah pengecekan progress. Setiap progress harus diberikan status *task* yang digunakan sebagai *tracking* status pekerjaan. Adapun setiap *role* pekerjaan mempunyai status *task* tersendiri. Pada Tabel 3.1 menjelaskan mengenai pembagian *role* di dalam proyek Sistem Manajemen Jaringan. Adapun status *task* pada Taiga secara jelas dapat dilihat pada Tabel 3.2.

Tabel 3.1. Pembagian dan deskripsi *role* pekerjaan

Nama	**Role Pekerjaan**	**Deskripsi**
Atep Taopik	*Project Manager*	Bertanggung jawab penuh terhadap tim proyek dan menjadi jembatan komunikasi antar tim internal dengan tim eksternal
Nanditia Fitra	*System Analyst*	Melakukan analisis kebutuhan untuk setiap *sprint* dan memastikan bahwa *sprint* berjalan dengan baik
Dzaki Buchori	*Programmer Back-End & Code Reviewer*	Melakukan pengembangan sistem, pembuatan API, dan melakukan *code review*
Akmal	*Programmer Back-End*	Melakukan pengembangan sistem dan pembuatan API
Damar Huda	*Programmer Front-End*	Melakukan pengembangan sistem antarmuka dan pengintegrasian API
Yuha	*Programmer Front-End*	Melakukan pengembangan sistem antarmuka dan pengintegrasian API
Anisa Amalia	*Quality Assurance*	Melakukan pengujian sistem dan memastikan fungsi telah berjalan dengan baik
Angel	*Quality Assurance*	Melakukan pengujian sistem dan memastikan fungsi telah berjalan dengan baik

Tabel 3.2. Daftar *role* dan status *task* pada Taiga

Assigned	***Label Task***	**Keterangan**
Analyst	*Backlog*	Daftar kebutuhan proyek yang harus dianalisis sebelum diberikan kepada *Programmer*.
	Needs Info	*Task* yang dikembalikan kepada *Analyst* dikarenakan terdapat pertanyaan atau ketidakjelasan dalam proses analisis sehingga membutuhkan informasi dari *Analyst*
Programmer	*To do*	*Task* hasil analisis yang siap harus dikerjakan oleh *programmer*.
	In Progress	*Task* yang sedang dikerjakan oleh *programmer*.
	Feedback	*Task* yang dikembalikan kepada *programmer* karena terdapat *bug/issue* yang ditemukan oleh Tester.
Code Reviewer	*Code Review*	Melakukan *code review* terhadap *merge request* yang diajukan oleh tim.

QA/Tester	*Ready to Test*	*Task* yang sudah dikerjakan oleh *programmer* dan siap diuji oleh tester.
	In Progress	*Task* sedang dilakukan pengujian oleh tester.
	Done	*Task* telah sesuai diuji dan sesuai dengan *acceptance criteria.*

Dalam pengembangan proyek, kerjasama maupun kolaborasi antar tim sangat dibutuhkan untuk mencapai tujuan bersama. Sebagai *Quality Assurance*, penulis melakukan pengujian otomatis bersama rekan QA lainnya. Pembagian tugas selama pengujian yaitu penulis melakukan pengujian untuk modul Sistem Manajemen Jaringan *Update* & *Downgrade* sedangkan rekan QA lainnya melakukan pengujian pada modul terbaru seperti *Dismantle* dan *Renewal*.

3.3 Pengujian Web Menggunakan Playwright

Pengujian web dilakukan menggunakan Playwright pada Sistem Manajemen Jaringan. Sistem Manajemen Jaringan memiliki 7 pengguna dan 20 proses yang saling berhubungan. Dalam proses magang, selama melakukan pengujian Sistem Manajemen Jaringan *Upgrade & Downgrade*, penulis telah membuat script test sebanyak 110 *script* yang telah di dokumentasikan di Gitlab proyek. Akan tetapi, untuk dalam laporan Tugas Akhir ini akan disampaikan hasil pengujian pada proses *Create Link* saja. Setiap proses di dalam modul tersebut saling berhubungan, untuk itu dipilih proses *Create Link* sebagai studi kasus dikarenakan proses penginputan data pertama dilakukan di menu tersebut.

3.3.1 Analisis Sistem Manajemen Jaringan

Sistem Manajemen Jaringan merupakan aplikasi berbasis web dari perusahaan yang bergerak di bidang telekomunikasi. Sistem tersebut digunakan untuk melakukan monitoring maupun manajemen penyewaan tower jaringan. Klien atau *vendor* dapat melakukan permintaan penyewaan jaringan, pemrosesan biaya sewa, pembaharuan, pembongkaran, serta permintaan perubahan kapasitas jaringan. Oleh karena itu, data dalam Sistem Manajemen Jaringan saling terintegrasi dengan beberapa sub-sistem atau modul yang lain, yaitu *Create Link, Upgrade & Downgrade, AF Online, Renewal,* dan *Dismantle.*

Create Link merupakan sistem yang digunakan untuk membuat data *link* baru terkait permintaan pemrosesan tower jaringan. *Upgrade & Downgrade* digunakan untuk melakukan permintaan perubahan kapasitas seperti menaikkan maupun menurunkan kapasitas jaringan. *AF Online* digunakan sebagai sistem yang melakukan pendokumentasian dan pemrosesan

pembayaran. Terakhir yaitu *Renewal* dan *Dismantle* merupakan sistem yang digunakan untuk pembaharuan dan pembongkaran tower jaringan. Adapun untuk studi kasus pengujian otomatis dalam Sistem Manajemen Jaringan menggunakan modul atau sub-sistem *Upgrade* & *Downgrade*. Hal tersebut dikarenakan adanya pembagian tugas antar tim sehingga penulis mendapatkan *task* dan tanggung jawab penuh untuk melakukan pengujian di sistem *Upgrade* & *Downgrade.*

Proses pengujian otomatis menggunakan Playwright diawali dengan memahami proses Sistem Manajemen Jaringan modul *Upgrade* & *Downgrade* yang terdapat pada *technical document* dan *user guide. Technical document* yaitu dokumen yang mendokumentasikan semua kebutuhan dalam pengembangan sistem seperti informasi arsitektur, database, alur proses bisnis BPMN, dokumentasi API, dan kebutuhan fungsional sistem. Adapun *user guide* merupakan dokumentasi panduan pengguna dalam menjalankan sebuah sistem. Kedua dokumen tersebut menjadi acuan dalam memahami proses maupun kebutuhan sistem secara keseluruhan.

Analisis *technical document* dibutuhkan untuk mengidentifikasi setiap fitur termasuk mengetahui kebutuhan data pada setiap fiturnya. Informasi kebutuhan data di setiap fitur bermanfaat untuk proses pengujian dari sisi verifikasi penginputan, contohnya yaitu inputan dengan tipe data *mandatory* harus diisi ketika akan melakukan proses *submit* data, sedangkan tipe data *optional* tidak diharuskan untuk diisi ketika *submit* maupun *save draft* data. Dalam dokumen ini juga diberikan gambaran mengenai setiap fitur dari sisi ekspektasi inputan, ekspektasi keluaran, jalannya sistem, hingga tampilan desain. *Technical document* menjadi acuan dan dokumen penting selama proses pengembangan serta pengujian.

Adapun analisis *user guide* bertujuan untuk mengetahui alur kerja sistem dari proses input awal hingga proses *submit* akhir. Dalam dokumen *user guide*, telah diberikan panduan detail sesuai dengan *role* masing-masing pengguna. Tidak hanya itu, dokumen tersebut juga memberikan panduan *field* atau data apa saja yang harus dimasukkan maupun di proses. Dari hasil analisis memahami *technical document* dan *user guide,* dijabarkan bahwa Sistem Manajemen Jaringan memiliki 7 pengguna dan 20 modul data. Penjelasan mengenai pengguna dapat dilihat pada Tabel 3.3 adapun daftar modul Sistem Manajemen Jaringan dapat dilihat pada Tabel 3.4.

Tabel 3.3. Tabel Deskripsi Setiap Pengguna

No	***Role User***	**Modul**
1	*Lead System*	*Create Link, Update* & *Drop Link, Review* BA L1
2	*Head System*	*Review Link, Review* SP, *Review* BA L2, *Review* BK L2

3	*LM System*	*Review* & *Input* Data, *Update* SP *Review* BK L1
4	*LMHead System*	*Review* SP, *Review* BK L2
5	SPV Mitra	*Review* BA BK sesuai dengan mitranya
6	PIC Mitra	*Review* BA BK sesuai dengan mitranya
7	PMO	*Review* BA PMO

Tabel 3.4 Daftar Modul Sistem Manajemen Jaringan

No	**Modul Sistem Manajemen Jaringan (*Upgrade & Downgrade*)**
1	*Create Link*
2	*Update & Drop Link*
3	*Review Link*
4	*Review & Input Data*
5	*Create SP*
6	*Update SP*
7	*Release SP*
8	*Review SP by Vendor*
9	*Create BA*
10	*Release BA*
11	*Review BA 1*
12	*Review BA 2*
13	*Review BA PMO*
14	*Update BA*
15	*Create BK*
16	*Release BK*
17	*Review BK 1*
18	*Review BK 2*
19	*Monitoring Link*
20	*Monitoring SP*

Alur 20 proses data pada Sistem Manajemen Jaringan saling berhubungan. Data inputan pertama yang terdapat pada modul *Create Link* akan melewati proses modul besar yaitu SP, BA, serta BK yang nantinya akan selesai pada modul Review BK 2. Adapun *Create Link* merupakan

modul inputan pertama yang digunakan untuk melakukan permintaan perubahan kapasitas seperti penurunan dan kenaikan. Terdapat dua pemrosesan dalam *Create Link* yaitu *save draft* dan *submit*. Data *submit* akan masuk ke dalam proses *Review Link* dan dilakukan *review* dengan hasil status *reject* atau *approval*. Sedangkan data *save draft* belum bisa masuk ke *flow* selanjutnya, dalam alur ini *user* dapat melakukan perubahan data sebelum dilakukan *review*.

Data *Create Link* yang telah di-*approve* akan masuk ke dalam proses SP dan memiliki ID SP untuk setiap data yang telah disetujui. Satu ID SP dapat berisi beberapa data *link* yang telah disetujui dengan syarat memiliki *partner* atau mitra yang sama. Pada proses SP, data akan diproses dengan pilihan *approve* atau *reject*. Data yang diproses *approve* akan masuk ke dalam proses BA. Dalam proses BA, data akan dilakukan *review* sama seperti pada modul sebelumnya, akan tetapi yang membedakan yaitu jika dalam proses *Review* BA 1, pengguna diberikan pilihan untuk melakukan *Review* PMO atau tidak. Jika pengguna membutuhkan *Review* PMO maka data akan masuk ke *Review* BA 2 dan *Review* BA PMO. Adapun jika tidak membutuhkan *review* PMO maka data langsung menuju proses BK. Proses BK menjadi *flow* terakhir sebelum semua data yang telah di-*review* akan di proses dan diintegrasikan dengan sistem pembayaran. Adapun menu monitoring dapat digunakan oleh *user* untuk melakukan progress pengecekan terkait data *link* yang telah dibuat. Informasi detail terkait data *link* dapat dilihat pada menu tersebut.

Modul *Create Link* merupakan gerbang inputan awal yang digunakan pengguna untuk memproses permintaan kenaikan atau penurunan kapasitas jaringan. Pada halaman *list* data, terdapat *search bar* dengan 5 *searching parameters* yaitu *link ID (multiple search)*, *link name, vendor, tower ID,* dan *circuit ID* dengan masing-masing *acceptance criteria*. Adapun tipe data dan syarat dari masing-masing parameter, yaitu:

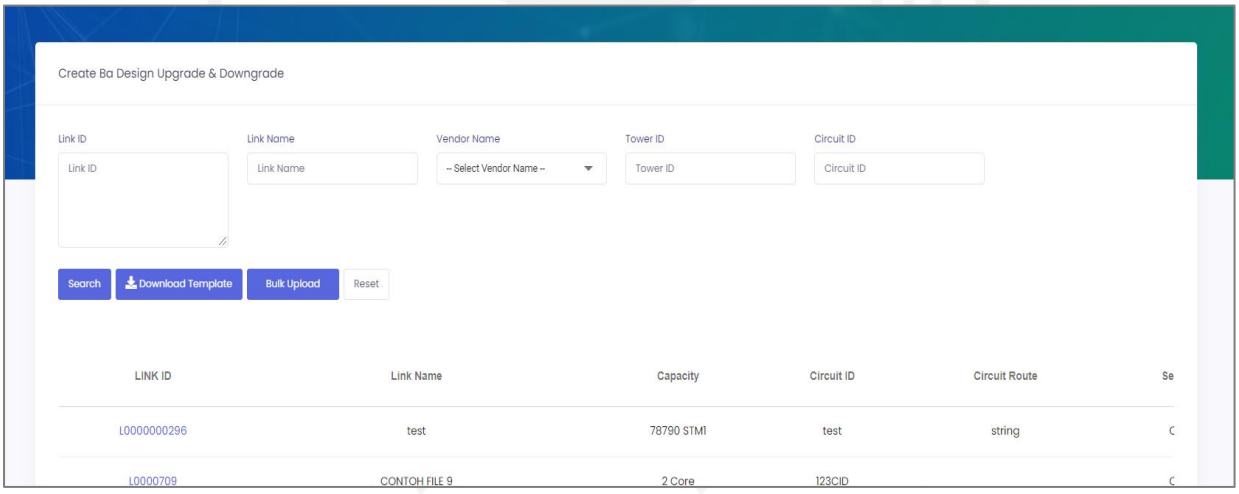
a. *Link* ID: dapat melakukan pencarian *link* ID secara individu maupun banyak data (*multiple search*)
b. *Link name*: untuk melakukan pencarian data berdasarkan *link name* secara individu
c. *Vendor*: untuk melakukan pencarian data berdasarkan *vendor name* atau mitra secara individu
d. Tower ID: untuk melakukan pencarian data berdasarkan tower ID secara individu. Terdapat dua tipe yaitu tower ID NE dan tower ID FE, adapun parameter tower ID berlaku untuk kedua tipe tersebut
e. *Circuit* ID: untuk melakukan pencarian berdasarkan *circuit* ID secara individu

Selain fitur pencarian, dalam modul *Create Link* bagian detail berisi informasi detail dan *form* penginputan data. Pengguna dapat melihat halaman detail data dengan melakukan klik pada *Link* ID yang dipilih. Adapun dalam modul ini terdapat inputan tipe *mandatory* dan *optional*. Tipe *mandatory* merupakan *field* yang harus diinputkan ketika pengguna ingin melakukan *submit* dan

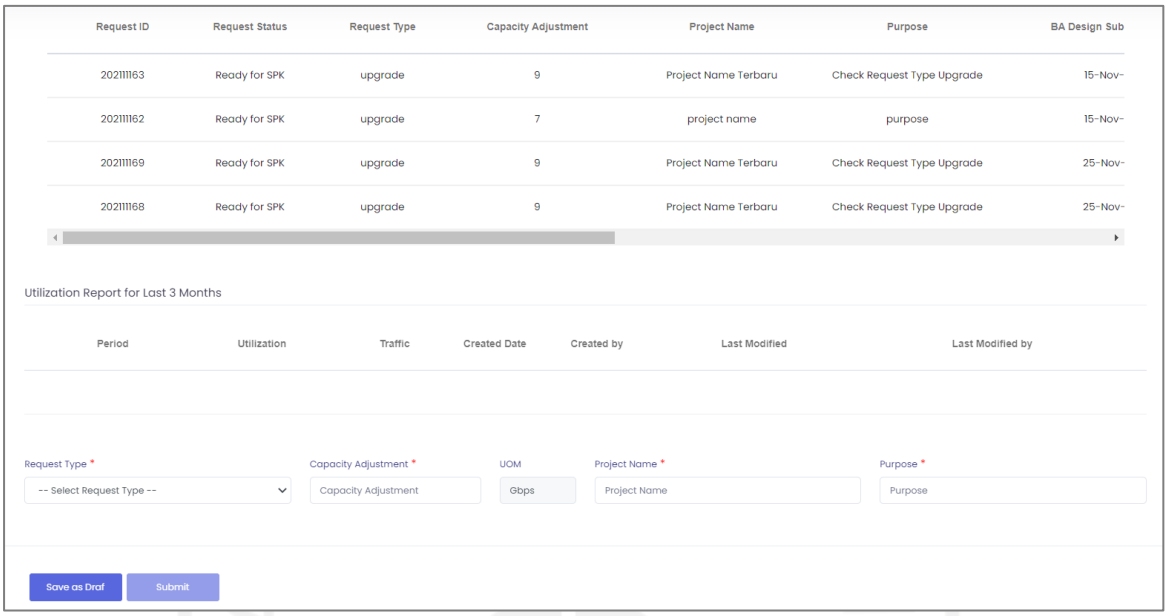
memproses data menuju *flow* berikutnya. Tipe *optional* tidak wajib untuk dilakukan penginputan ketika proses *submit* maupun *save draft*. Adapun *field* yang terdapat pada *form* penginputan dalam halaman detail *Create Link*, yaitu:

a. *Request type* : permintaan untuk kenaikan atau pengurangan kapasitas jaringan dengan pilihan *upgrade* dan *downgrade*. Tipe data *mandatory*
b. *Capacity adjustment* : besarnya kapasitas kenaikan atau pengurangan kapasitas jaringan. Tipe data *integer* dan data *mandatory*
c. UOM : informasi skala pengukuran standar kenaikan atau pengurangan kapasitas jaringan. Nilainya *default* bergantung kepada inputan yang telah diproses sebelumnya dan sistem akan secara otomatis menampilkan datanya
d. *Project name* : nama proyek yang sedang dikembangkan. Tipe data *varchar* dan *mandatory*
e. *Purpose* : tujuan dari proyek ataupun tujuan permintaan kenaikan / pengurangan kapasitas. Tipe data *varchar* dan *mandatory*

Tampilan halaman list index dapat dilihat pada Gambar 3.3 sedangkan tampilan halaman detail link data dapat dilihat pada Gambar 3.4.



Gambar 3.3 Tampilan halaman index *Create Link*



Gambar 3.4 Tampilan halaman detail *Create Link*

3.3.2 Pembuatan Skenario Pengujian

Skenario pengujian atau *test case* merupakan pendokumentasian skenario kasus uji yang mencakup langkah-langkah pengujian, hasil ekspektasi, hasil aktual, dan status pengujian. Dalam pembuatan skenario tes terdapat dua tipe pengujian yaitu pengujian positif dan negatif. Tipe pengujian positif adalah skenario untuk memastikan bahwa sistem melakukan proses yang seharusnya dilakukan sedangkan pengujian negatif yaitu memastikan bahwa sistem tidak melakukan proses yang seharusnya tidak dilakukan. Dokumen skenario penting digunakan sebagai acuan dalam menjalankan pengujian sistem, selain itu dokumen tersebut juga menjadi dokumentasi untuk setiap laporan *bug* sebelum dituliskan dalam Taiga. Adapun dalam pembuatan skenario test, penguji menggunakan *Microsoft Excel*..

Pembuatan skenario pengujian dilakukan ketika semua kebutuhan pengguna telah dilakukan analisis dalam bentuk *task* yang akan diberikan kepada *programmer*. Setiap *task* memiliki deskripsi terkait fungsi dan *acceptance criteria* dari fitur tersebut. Oleh karena itu, penguji dapat membuat skenario pengujian berdasarkan kebutuhan yang telah diidentifikasi oleh Analis. Selain itu, penguji juga dapat menambah pengetahuan terkait fungsi maupun *acceptance criteria* sistem tersebut melalui Jira. Jira merupakan *task management* yang digunakan untuk mendistribusikan *backlog* atau *task* dari klien kepada tim internal. Dalam pengujian

Pengujian *Create Link* tipe *Upgrade* berisi 2 pilihan proses yaitu *save draft* dan *submit*. Proses *submit* dapat dieksekusi jika semua tipe data *mandatory* telah diinputkan, sedangkan proses *save draft* tidak diwajibkan untuk menginputkan tipe data *mandatory* kecuali *request type.* Adapun 4 tipe data *mandatory* yang terdapat pada *Create Link* yaitu *request type, capacity adjustment,*

project, dan *purpose.* Dalam skenario tersebut terdapat 18 skenario dengan 12 tipe pengujian positif dan 6 negatif. Skenario pengujian dibuat dalam bentuk tabel yang mencakup informasi sebagai berikut:

a. ID adalah urutan nomor skenario yang akan diuji. Biasanya disebut juga dengan *test case* ID
b. Sub Modul adalah nama *field* yang akan diuji
c. Deskripsi adalah penjelasan singkat mengenai kasus yang akan diuji
d. Skenario adalah langkah-langkah pengujian yang harus dilakukan oleh penguji
e. Ekspektasi adalah hasil akhir yang diharapkan ketika proses pengujian
f. Hasil aktual adalah hasil yang sebenarnya terjadi ketika proses pengujian
g. Status adalah keterangan akhir setelah proses pengujian selesai. Biasanya berisi *passed* atau *failed.*

Skenario pengujian menu *Create Link* tipe *Upgrade* dapat dilihat pada Tabel 3.5.

Tabel 3.5 Hasil pembuatan skenario tes

ID	Sub Modul	Deskripsi	Skenario Tes	Hasil Ekspektasi	Hasil Aktual	Status
1	*Search by Parameters*	Positif - Melakukan pencarian data menggunakan parameter yang ada	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Input valid *keyword* di search bar 6. Klik *Search*	Dapat menampilkan data sesuai dengan parameters yang dicari	-	*Passed/ failed*
2	*Search by Parameters*	Positif - Melakukan pencarian *multiple* data *Link* ID	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Input multiple data *Link* ID 6. Klik *Search*	Dapat menampilkan multiple data *Link* ID	-	*Passed/ failed*
3	*Search by Parameters*	Positif – *Reset* semua *keyword* di semua *parameters*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Input *keyword* di semua *parameters*	Semua *keyword* akan terhapus	-	*Passed/ failed*

			6. Klik *Reset*			
4	*Search by Parameters*	Negatif - Melakukan pencarian data invalid menggunakan parameter yang ada	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Input invalid *keyword* atau simbol di *search bar* untuk setiap parameters 6. Klik *Search*	Data tidak ditemukan, sistem menampilkan pesan "Data tidak ditemukan"	-	*Passed/ failed*
5	*Upgrade - Submit Data*	Positif - *Submit* data tipe *Upgrade* dengan menginputkan semua *field mandatory*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, masukkan semua *field mandatory* 7. Klik *Submit*	Data sukses terkirim, sistem menampilkan pesan "Data telah sukses terkirim!".	-	*Passed/ failed*
6	*Upgrade - Submit Data*	Negatif - *Submit* data tipe *Upgrade* dengan tidak menginputkan *field mandatory "Capacity Adjustment"*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade,* masukkan semua *field* kecuali *Capacity Adjustment* 7. Klik *Submit*	*Button Submit* tidak aktif, pengguna tidak bisa melakukan proses s*ubmit*	-	*Passed/ failed*
7	*Upgrade - Submit Data*	Negatif - *Submit* data tipe *Upgrade* dengan tidak menginputkan *field mandatory "Project Name*"	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade,* masukkan semua *field* kecuali *Project Name* 7. Klik *Submit*	*Button Submit* tidak aktif, pengguna tidak bisa melakukan proses *submit*	-	*Passed/ failed*

8	*Upgrade - Submit Data*	Negatif - *Submit* data tipe *Upgrade* dengan tidak menginputkan *field mandatory "Purpose"*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Tower* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade,* masukkan semua *field* kecuali *Purpose* 7. Klik *Submit*	*Button Submit* tidak aktif, pengguna tidak bisa melakukan proses s*ubmit*	-	*Passed/ failed*
9	*Upgrade - Submit Data*	Negatif – Input field Capacity Adjustment dengan tipe data varchar	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field capacity adjustment dengan tipe data varchar 7. Klik *Submit*	Tidak bisa melakukan input tipe data varchar (*field disabled*)	-	*Passed/ Failed*
10	*Upgrade - Submit Data*	Positif – Input field project name dengan tipe Integer	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field project name dengan tipe data integer atau float (selain varchar) 7. Klik *Submit*	Tetap dapat melakukan submit data (*field enable*)	-	*Passed/ Failed*
11	*Upgrade - Submit Data*	Positif – Input field purpose dengan tipe Integer	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field purpose dengan tipe data integer atau float 7. Klik *Submit*	Tetap dapat melakukan submit data (*field enable*)	-	*Passed/ Failed*

12	*Upgrade - Save Draft Data*	Positif - *Save draft* data tipe *Upgrade* dengan menginputkan semua *field*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, masukkan semua *field* 7. Klik *Save Draft*	Data sukses terkirim. Sistem menampilkan pesan "Data telah sukses tersimpan!"	-	*Passed/ failed*
13	*Upgrade - Save Draft Data*	Positif - *Save Draft* data tipe *Upgrade* dengan tidak menginputkan *field mandatory "Capacity Adjustment"*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, masukkan semua *field* kecuali *Capacity Adjustment* 7. Klik *Save Draft*	Data sukses terkirim. Sistem menampilkan pesan "Data telah sukses tersimpan!"	-	*Passed/ failed*
14	*Upgrade - Save Draft Data*	Positif - *Save Draft* data tipe *Upgrade* dengan tidak menginputkan *field mandatory "Project Name"*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade,* masukkan semua *field* kecuali *Project Name* 7. Klik *Save Draft*	Data sukses terkirim. Sistem menampilkan pesan "Data telah sukses tersimpan!"	-	*Passed/ failed*
15	*Upgrade - Save Draft Data*	Positif - *Save Draft* data tipe *Upgrade* dengan tidak menginputkan *field mandatory "Purpose"*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, masukkan semua *field* kecuali *Purpose* 7. Klik *Save Draft*	Data sukses terkirim. Sistem menampilkan pesan "Data telah sukses tersimpan!"	-	*Passed/ failed*
16	*Upgrade - Save Draft Data*	Negatif – *Input field Capacity Adjustment*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password*	Tidak bisa melakukan input tipe data	-	*Passed/ Failed*

		dengan tipe data *varchar*	3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field capacity adjustment dengan tipe data *varchar* 7. Klik *Save Draft*	varchar (*field disabled*)		
17	*Upgrade - Save Draft Data*	Positif – Input field project name dengan tipe Integer	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field project name dengan tipe data integer atau float (selain varchar) 7. Klik *Save Draft*	Tetap dapat melakukan submit data (*field enable*)	-	*Passed/ Failed*
18	*Upgrade - Save Draft Data*	Positif – Input field purpose dengan tipe Integer	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field purpose dengan tipe data integer atau float 7. Klik *Save Draft*	Tetap dapat melakukan submit data (*field enable*)	-	*Passed/ Failed*

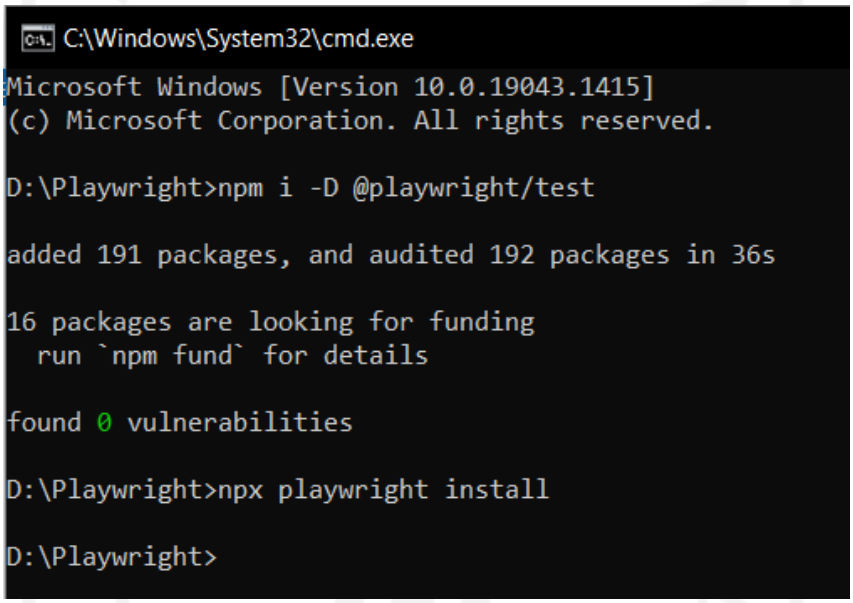
3.3.3 Pembuatan *Script Test*

Pembuatan *script test* dibuat berdasarkan skenario uji yang telah didokumentasikan pada *test case*. Setiap skenario memiliki satu *script test* yang akan dieksekusi. *Script* tersebut akan melakukan pengecekan dengan membandingkan proses aktual yang terjadi dengan ekspektasi uji. Pembuatan *script test* Playwright dapat dilakukan dengan dua cara yaitu dengan penulisan manual maupun penulisan otomatis. Selama magang berlangsung, penulis membuat script test dengan penulisan secara otomatis tanpa pemrograman manual melalui bantuan Playwright Codegen. *Script* tersebut dihasilkan dari rekaman aktivitas pengguna pada proses yang sedang diuji dan akan di-*generate* menjadi kode pemrograman dengan 5 bahasa pemrograman yang dapat dipilih.

Proses instalasi Playwright cukup mudah dengan menjalankan perintah di *Command Prompt* (CMD) atau terminal *Visual Studio Code*. Penguji diharuskan melakukan instalasi *dependency* Playwright dan juga web browser pendukung. Adapun perintah instalasi Playwright dapat dilihat pada Gambar 3.5 sedangkan hasil instalasi Playwright dapat dilihat pada Gambar 3.6.

```
# add dependency
npm i -D @playwright/test

# install supported browsers
npx playwright install
```

Gambar 3.5 Perintah instalasi playwright



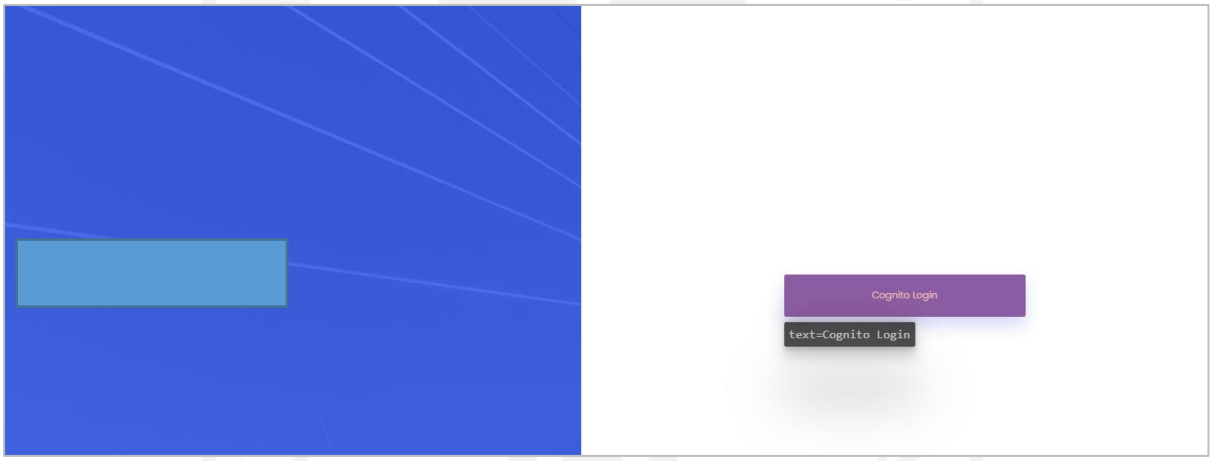
Gambar 3.6 Tampilan hasil instalasi Playwright di CMD

Dalam pembuatan *script test*, dibutuhkan kode editor untuk membantu penguji dalam melakukan pengeditan maupun penambahan kode pemrograman. Langkah pertama yang dilakukan yaitu membuat sebuah folder baru untuk penyimpanan semua *script test*. Folder tersebut dibuka melalui *Visual Studio Code* dan ditambahkan sebuah *workspace* baru untuk pendokumentasian *script*. *Workspace* tersebut berisi file-file yang akan menyimpan semua skenario pengujian. Setelah pembuatan *workspace*, penguji dapat menuliskan perintah perekaman melalui *command prompt* maupun terminal di *Visual Studio Code*. Adapun *url* web sistem yang akan diuji dapat dituliskan di akhir perintah. Perintah perekaman pengujian dapat dilihat pada Gambar 3.7.

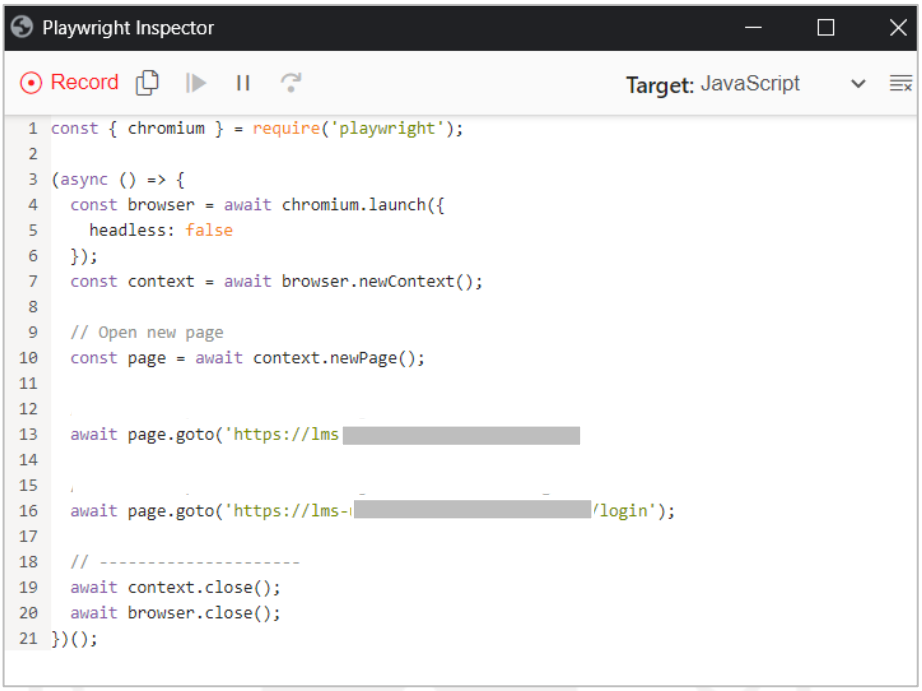
```
#Perintah perekaman web

npx playwright codegen [url web sistem]
```

Gambar 3.7 Perintah perekaman pengujian Playwright Codegen

Setelah perintah perekaman di eksekusi, akan muncul 2 jendela atau halaman baru untuk proses perekaman. Jendela yang pertama yaitu web browser sistem yang akan dilakukan pengujian sedangkan jendela yang lain yaitu *workspace* yang melakukan dokumentasi kode pemrograman pada setiap aktivitas yang dilakukan oleh penguji pada web browser tersebut. Dalam jendela *workspace*, penguji dapat memberhentikan sementara perekaman dengan melakukan klik button *pause* yang terdapat pada *bar* navigasi paling atas. Selain itu, penguji juga dapat dengan mudah memilih bahasa pemrograman yang terletak pada *bar* navigasi tersebut. Adapun 2 jendela yang muncul ketika proses perekaman pengujian dapat dilihat pada Gambar 3.8 dan Gambar 3.9.



Gambar 3.8 Tampilan halaman web browser perekaman



Gambar 3.9 Tampilan halaman *workspace* perekaman

Aktivitas pengujian dapat dilakukan seperti biasa pada jendela web browser yang telah muncul. Setelah proses pengujian selesai, *script* yang telah di-*generate* dapat di salin ke dalam file *workspace* di kode editor. Dalam proses pengujian, untuk menyinkronkan data maka diperlukan perubahan kode. Perubahan kode digunakan untuk mengambil *object* data yang akan diuji. Adapun untuk mempermudah dalam pengeditan *script* dapat memanfaatkan *inspect selector* pada web sistem. Pengambilan object data menggunakan selector dapat dilihat pada Gambar 3.10.

```
// Pengambilan object data secara otomatis

await page.click('tbody > tr.mat-row:nth-child(1) > td:nth-child(2) > a:nth-child(1)');
```

Gambar 3.10 Pengeditan *script object* data

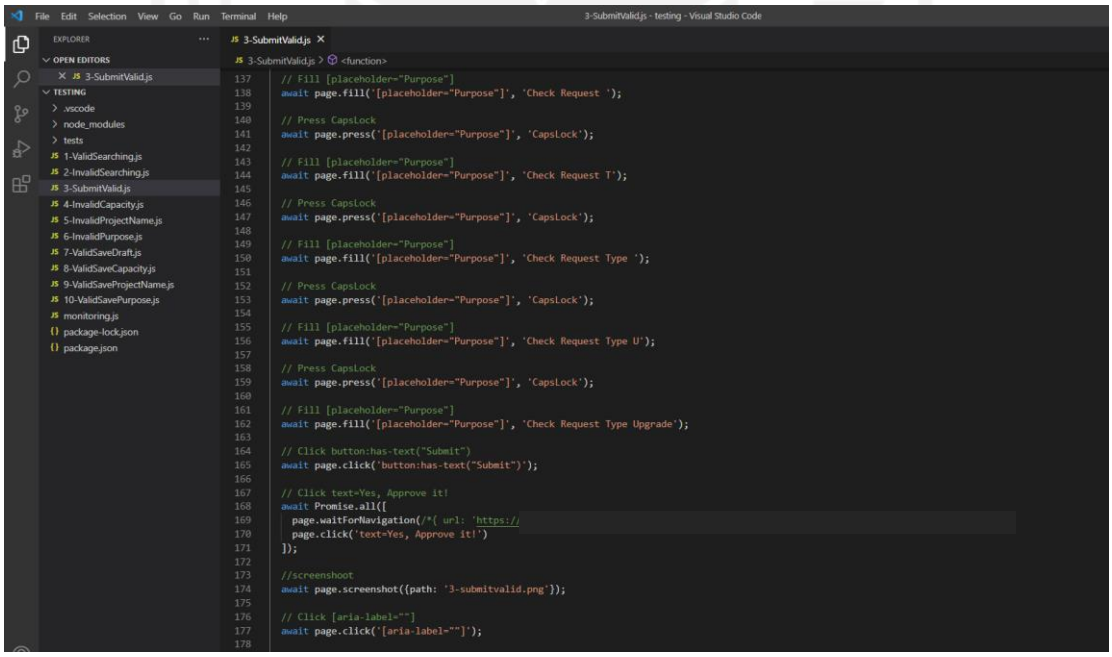
Penguji dapat mengambil tangkapan layar atau gambar untuk mendokumentasikan hasil akhir proses pengujian. Tangkapan layar tersebut dapat digunakan sebagai bukti di pendokumentasian laporan pengujian. Sebelum menambahkan kode pemrograman untuk pengambilan gambar, penguji harus mengetahui runtutan dari *script test*. Hal tersebut bertujuan supaya gambar yang ditangkap sesuai dengan alur atau hasil yang ingin didokumentasikan. Pada percobaan kali ini, penguji mengambil tangkapan layar atau gambar ketika sistem berhasil melakukan *approve submit*

data. Adapun perintah untuk pengambilan gambar dapat dilihat pada Gambar 3.11. Hasil tangkapan layar akan tersimpan pada folder yang sama dengan folder *script test*.

```
// Click button:has-text("Submit")
  await page.click('button:has-text("Submit")');

// Click text=Yes, Approve it!
  await Promise.all([
    page.waitForNavigation
    page.click('text=Yes, Approve it!')
  ]);

//screenshoot
  await page.screenshot({path: '3-submitvalid.png'});
```

Gambar 3.11 Kode penangkapan gambar

Proses pembuatan *script test* telah selesai dengan membuat 18 *script* sesuai skenario pengujian yang telah dibuat. Daftar 18 *script test* yang telah disesuaikan dapat dilihat pada Gambar 3.12.



Gambar 3.12 Tampilan daftar 18 *script test*

3.3.4 Eksekusi *Script Test*

Dalam pengeksekusian *script test* akan menghasilkan kondisi aktual pengujian. Hasil aktual tersebut memiliki 2 status pengujian yaitu *passed* dan *failed.* Status *passed* diberikan ketika hasil aktual sesuai dengan hasil ekspektasi, sebaliknya status *failed* diberikan ketika hasil tidak sesuai dengan hasil ekspektasi.

Playwright memiliki dua mode pengeksekusian yaitu mode *headless* dan *headful*. Mode *headless* mengeksekusi tes tanpa antarmuka (UI) sistem, mode tersebut dieksekusi seperti biasa. Adapun mode *headful* menampilkan keseluruhan antarmuka sistem yang sedang diuji, mode tersebut akan memutar kembali kode perekaman sama persis seperti aktivitas yang dilakukan oleh penguji.

Dalam pengeksekusian, Playwright juga memiliki fitur *debugging* yang dapat digunakan jika penguji melakukan perubahan maupun penambahan dalam *script test*. Hal tersebut bermanfaat untuk memeriksa setiap bagian kode *script* yang dilakukan perubahan sehingga jika terdapat error mampu teratasi dengan mengetahui bagian *script* yang menjadi akar permasalahan. Penguji biasanya menggunakan fitur *debug* untuk mencari letak kesalahan pada bagian *script test* ketika terjadi penambahan maupun perubahan *script*. Pengeksekusian menggunakan *debug* dapat dijalankan menggunakan *command line* Playwright Codegen seperti pada Gambar 3.13 sedangkan Playwright *Test Runner debug* seperti pada Gambar 3.14.

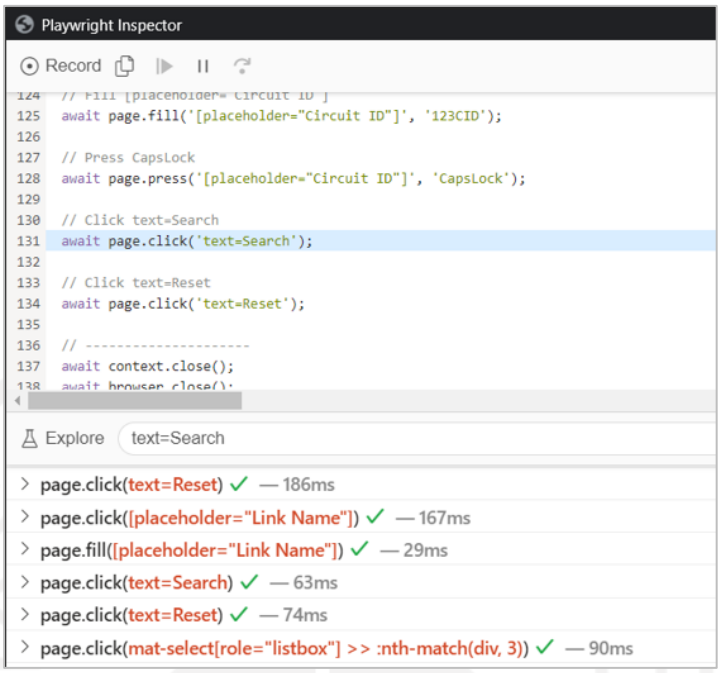
```
#Debug Playwright Codegen

Set PWDEBUG=1
Node (filename).js
```

Gambar 3.13 Perintah *debug* Playwright Codegen

```
#Debug Playwright Test Runner

Npx playwright test --debug
```

Gambar 3.14 Perintah *debug* Playwright *Test Runner*

Dalam pengeksekusian menggunakan *debugging*, akan muncul 2 jendela yaitu jendela web browser dan Playwright *Inspector*. Jendela web browser akan menjalankan seluruh hasil rekaman yang terdapat pada *script test*. Playwright *Inspector* akan memberikan penanda atau *selector* dari setiap alur-alur yang sedang berjalan. Adapun pada Gambar 3.15 menunjukkan jendela Playwright *Inspector debug* yang sedang berjalan.



Gambar 3.15 Tampilan jendela Playwright *Inspector debug*

Selain menggunakan *debug*, Playwright juga dapat mengeksekusi *multiple test* dengan menggunakan Playwright *Test Runner*. Dalam hal ini, 18 *script test* yang telah dibuat juga akan dieksekusi bersamaan menggunakan *test runner*. Daftar semua *script test* dapat dilihat menggunakan perintah *playwright list* seperti pada Gambar 3.16. Adapun untuk proses pengeksekusian 18 *script test* tersebut, dapat menggunakan perintah seperti pada Gambar 3.17 sedangkan penguji juga dapat menjalankan *single test* dengan perintah seperti pada Gambar 3.18.

```
PS D:\Playwright> npx playwright test --list
Listing tests:
  tests.js\10-validSavePurpose\test.js:3:1 › test
  tests.js\7-validSaveDraft\test.js:3:1 › test
  tests.js\8-validSaveCapacity\test.js:3:1 › test
  tests.js\9-ValidSaveProjectName\test.js:3:1 › test
  tests.js\invalidCapacity\test.js:3:1 › test
  tests.js\InvalidCapacityAdjustment\test.js:3:1 › test
  tests.js\invalidProjectName\test.js:3:1 › test
  tests.js\invalidProjectNameInteger\test.js:3:1 › test
  tests.js\invalidPurpose\test.js:3:1 › test
  tests.js\invalidSearching\test.js:3:1 › test
  tests.js\invalidSubmitPurpose\test.js:3:1 › test
  tests.js\multipleSearch\test.js:3:1 › test
  tests.js\resetKeyword\test.js:3:1 › test
  tests.js\submitValid\test.js:3:1 › test
  tests.js\validProjectNameInteger\test.js:3:1 › test
  tests.js\validPurposeInteger\test.js:3:1 › test
  tests.js\validSearching\test.js:3:1 › test
  tests.js\validSubmitProjectNameInt\test.js:3:1 › test
Total: 18 tests in 18 files
PS D:\Playwright>
```

Gambar 3.16 Tampilan daftar *script test*

```
PS D:\Playwright> npx playwright test
```

Gambar 3.17 Perintah eksekusi *multiple test*

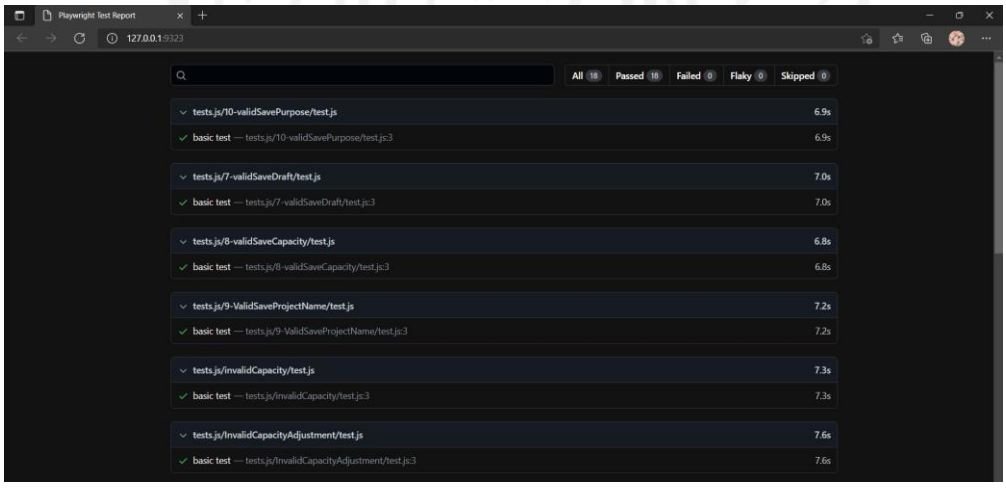
```
Npx playwright test [directory][filename].js
```

Gambar 3.18 Perintah eksekusi *single test* Playwright Test Runner

Untuk membantu proses pendokumentasian pengujian *multiple test*, penguji memanfaatkan fitur HTML *Report* Playwright. Fitur tersebut membantu penguji untuk mencatat semua hasil pengujian status *passed* maupun *failed*. Adapun jika terdapat status yang *failed* maka HTML *Report* akan memberikan informasi penyebab kegagalan uji. Riwayat dokumen laporan tersebut juga akan secara langsung tersimpan dan dapat di download dalam format (.html). Hasil menjalankan *multiple test* dan HTML *Report* dapat dilihat pada Gambar 3.19, sedangkan pada Gambar 3.20 menampilkan hasil dokumen pelaporan.

```
PS D:\Playwright> npx playwright test --reporter=html --timeout=0

Running 18 tests using 6 workers
  Slow test file: tests.js\8-validSaveCapacity\test.js (6m)
  Slow test file: tests.js\9-ValidSaveProjectName\test.js (6m)
  Slow test file: tests.js\invalidCapacity\test.js (5m)
  Slow test file: tests.js\invalidSearching\test.js (5m)
  Slow test file: tests.js\invalidProjectName\test.js (4m)
  Consider splitting slow test files to speed up parallel execution

  18 passed (10m)

To open last HTML report run:

  npx playwright show-report
```

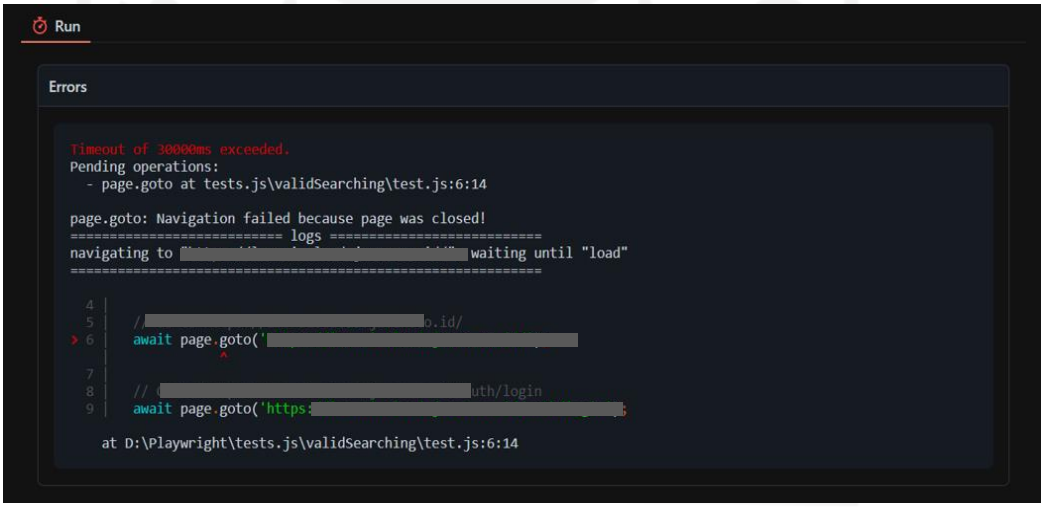
Gambar 3.19 Hasil eksekusi *multiple test* dan HTML *report*



Gambar 3.20 Tampilan laporan pengujian HTML *report*

Selama melakukan pengujian, tidak semua percobaan pengujian menghasilkan status *passed*, terdapat percobaan yang menghasilkan status *failed*. Penyebab dari kegagalan tersebut disebabkan oleh *timeout* yang menyebabkan Playwright tidak bisa mengakses web Sistem Manajemen Jaringan. Solusi dari permasalahan tersebut, penguji melakukan pengecekan jaringan internet karena sepengalaman penguji jika jaringan buruk maka akan menyebabkan *timeout*. Selain itu, penguji juga menambahkan *command line* untuk mencegah *timeout*. Detail penambahan *command line* eksekusi yang digunakan untuk meminimalisir *timeout* dapat dilihat pada Gambar 3.21. Adapun salah satu detail kegagalan uji yang disebabkan oleh *timeout* dapat dilihat pada Gambar 3.22.

Gambar 3.21 Penambahan perintah eksekusi untuk meminimalisir *timeout*



Gambar 3.22 Tampilan pelaporan kegagalan uji *timeout*

Percobaan pengujian otomatis pada Sistem Manajemen Jaringan telah berhasil dilakukan dengan status uji *passed*. Pada Tabel 3.6 menunjukkan hasil pengekseksusian *multiple test* menggunakan Playwright *Test Runner*.

Tabel 3.6 Hasil Pengeksekusian Skenario Tes

ID	Sub Modul	Deskripsi	Skenario Tes	Hasil Ekspektasi	Hasil Aktual	Status
1	*Search by Parameters*	Positif - Melakukan pencarian data menggunakan parameter yang ada	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link*	Dapat menampilkan data sesuai dengan parameters yang dicari	Sesuai dengan hasil ekspektasi	*Passed*

			Upgrade & Downgrade 5. Input valid *keyword* di search bar 6. Klik *Search*			
2	*Search by Parameters*	Positif - Melakukan pencarian *multiple* data *Link* ID	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Input multiple data *Link* ID 6. Klik *Search*	Dapat menampilkan multiple data *Link* ID	Sesuai dengan hasil ekspektasi	*Passed*
3	*Search by Parameters*	Positif – *Reset* semua *keyword* di semua *parameters*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Input *keyword* di semua *parameters* 6. Klik *Reset*	Semua *keyword* akan terhapus	Sesuai dengan hasil ekspektasi	*Passed*
4	*Search by Parameters*	Negatif - Melakukan pencarian data invalid menggunakan parameter yang ada	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Input invalid *keyword* atau simbol di *search bar* untuk setiap parameters 6. Klik *Search*	Data tidak ditemukan, sistem menampilkan pesan "Data tidak ditemukan"	Sesuai dengan hasil ekspektasi	*Passed*
5	*Upgrade - Submit Data*	Positif - *Submit* data tipe *Upgrade* dengan menginputkan semua *field mandatory*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, masukkan semua *field mandatory* 7. Klik *Submit*	Data sukses terkirim, sistem menampilkan pesan "Data telah sukses terkirim!".	Sesuai dengan hasil ekspektasi	*Passed*
6	*Upgrade - Submit Data*	Negatif - *Submit* data tipe *Upgrade* dengan tidak menginputkan *field mandatory*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link*	*Button Submit* tidak aktif, pengguna tidak bisa melakukan	Sesuai dengan hasil ekspektasi	*Passed*

		"Capacity Adjustment"	*Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade,* masukkan semua *field* kecuali *Capacity Adjustment* 7. Klik *Submit*	proses s*ubmit*		
7	*Upgrade - Submit Data*	Negatif - *Submit* data tipe *Upgrade* dengan tidak menginputkan *field mandatory "Project Name*"	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade,* masukkan semua *field* kecuali *Project Name* 7. Klik *Submit*	*Button Submit* tidak aktif, pengguna tidak bisa melakukan proses *submit*	Sesuai dengan hasil ekspektasi	*Passed*
8	*Upgrade - Submit Data*	Negatif - *Submit* data tipe *Upgrade* dengan tidak menginputkan *field mandatory "Purpose"*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Tower* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade,* masukkan semua *field* kecuali *Purpose* 7. Klik *Submit*	*Button Submit* tidak aktif, pengguna tidak bisa melakukan proses s*ubmit*	Sesuai dengan hasil ekspektasi	*Passed*
9	*Upgrade - Submit Data*	Negatif – Input field Capacity Adjustment dengan tipe data varchar	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field capacity adjustment dengan tipe data varchar 7. Klik *Submit*	Tidak bisa melakukan input tipe data varchar (*field disabled*)	Sesuai dengan hasil ekspektasi	*Passed*
10	*Upgrade - Submit Data*	Positif – Input field project name dengan tipe Integer	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field project name dengan tipe data	Tetap dapat melakukan submit data (*field enable*)	Sesuai dengan hasil ekspektasi	*Passed*

			integer atau float (selain varchar) 7. Klik *Submit*			
11	*Upgrade - Submit Data*	Positif – Input field purpose dengan tipe Integer	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field purpose dengan tipe data integer atau float 7. Klik *Submit*	Tetap dapat melakukan submit data (*field enable*)	Sesuai dengan hasil ekspektasi	*Passed*
12	*Upgrade - Save Draft Data*	Positif - *Save draft* data tipe *Upgrade* dengan menginputkan semua *field*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, masukkan semua *field* 7. Klik *Save Draft*	Data sukses terkirim. Sistem menampilkan pesan "Data telah sukses tersimpan!"	Sesuai dengan hasil ekspektasi	*Passed*
13	*Upgrade - Save Draft Data*	Positif - *Save Draft* data tipe *Upgrade* dengan tidak menginputkan *field mandatory "Capacity Adjustment"*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, masukkan semua *field* kecuali *Capacity Adjustment* 7. Klik *Save Draft*	Data sukses terkirim. Sistem menampilkan pesan "Data telah sukses tersimpan!"	Sesuai dengan hasil ekspektasi	*Passed*
14	*Upgrade - Save Draft Data*	Positif - *Save Draft* data tipe *Upgrade* dengan tidak menginputkan *field mandatory "Project Name"*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu Manajemen Jaringan 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade,* masukkan semua *field* kecuali *Project Name* 7. Klik *Save Draft*	Data sukses terkirim. Sistem menampilkan pesan "Data telah sukses tersimpan!"	Sesuai dengan hasil ekspektasi	*Passed*

15	*Upgrade - Save Draft Data*	Positif - *Save Draft* data tipe *Upgrade* dengan tidak menginputkan *field mandatory "Purpose"*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, masukkan semua *field* kecuali *Purpose* 7. Klik *Save Draft*	Data sukses terkirim. Sistem menampilkan pesan "Data telah sukses tersimpan!"	Sesuai dengan hasil ekspektasi	*Passed*
16	*Upgrade - Save Draft Data*	Negatif – *Input field Capacity Adjustment* dengan tipe data *varchar*	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field capacity adjustment dengan tipe data *varchar* 7. Klik *Save Draft*	Tidak bisa melakukan input tipe data varchar (*field disabled*)	Sesuai dengan hasil ekspektasi	*Passed*
17	*Upgrade - Save Draft Data*	Positif – Input field project name dengan tipe Integer	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field project name dengan tipe data integer atau float (selain varchar) 7. Klik *Save Draft*	Tetap dapat melakukan submit data (*field enable*)	Sesuai dengan hasil ekspektasi	*Passed*
18	*Upgrade - Save Draft Data*	Positif – Input field purpose dengan tipe Integer	1. Buka web sistem manajemen jaringan 2. Klik Login, masukkan *username dan password* 3. Klik menu *Manajemen Jaringan* 4. Klik menu *Create Link Upgrade & Downgrade* 5. Klik ID Data yang akan diproses 6. Pilih tipe *Upgrade*, input field purpose dengan tipe data integer atau float 7. Klik *Save Draft*	Tetap dapat melakukan submit data (*field enable*)	Sesuai dengan hasil ekspektasi	*Passed*

3.3.5 Perbandingan Playwright dan Selenium

Playwright dan Selenium WebDrivers merupakan alat uji berbasis lintas browser yang dapat membantu pengembang dalam melakukan pengujian web. Dalam pemilihan alat uji, diperlukan pertimbangan dari segi kebutuhan proyek, keadaan sumber daya manusia (SDM) di dalam proyek, hingga kelebihan serta kelemahan dari alat uji tersebut. Adapun perbandingan antara Playwright dan Selenium WebDrivers dapat dijadikan referensi dalam pemilihan alat uji.

a. Proses Instalasi

Untuk menggunakan Selenium WebDrivers dalam pengujian otomatis, diperlukan instalasi Selenium *libraries* dan *WebDrivers* (Selenium, 2021). Proses instalasi Selenium *libraries* bergantung kepada bahasa pemrograman yang akan digunakan. Proses instalasi selanjutnya yaitu *WebDrivers*, penguji harus melakukan instalasi *driver* yang sesuai dengan browser yang akan digunakan. Selain itu, penguji juga harus melakukan instalasi extension bahasa pemrograman sesuai yang akan digunakan pada kode editor. Daftar *driver* masing-masing browser terdapat pada Tabel 3.7.

Tabel 3.7. Daftar webdrivers

Web Browser	**Driver**
Chromium	Chromedriver
Firefox	Geckodriver
Microsoft Edge	Edgedriver
Internet Explorer	Internet Explorer driver
Safari	Safaridriver

Proses instalasi Selenium cukup mudah tetapi memiliki langkah-langkah yang lebih banyak daripada Playwright. Langkah pertama yang harus dilakukan yaitu instalasi *web driver* sesuai yang digunakan, Adapun penulis menggunakan web browser Chrome sehingga diperlukan instalasi *Chromedriver*. Dalam pemilihan versi Chrome *driver* juga harus disesuaikan dengan versi Chrome yang digunakan. Setelah itu penguji diharuskan untuk menambah *path variables* dari *web driver* tersebut.

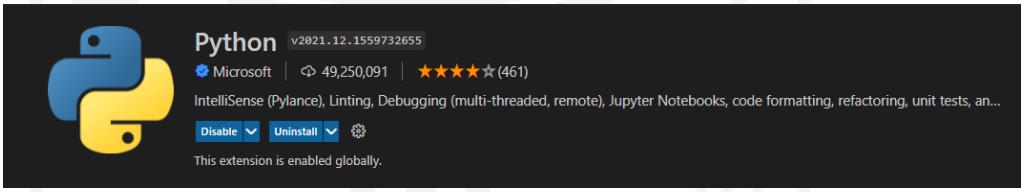
Setelah instalasi *web driver* sesuai dengan browser yang digunakan, diharuskan juga instalasi Selenium sesuai dengan bahasa yang akan digunakan selama pengujian. Dalam hal ini, untuk proses percobaan pengujian menggunakan bahasa Python sehingga dibutuhkan instalasi Selenium *libraries python* dan *extension python*.

Instalasi Selenium *libraries python* dieksekusi pada terminal atau *CMD*. Pada Gambar 3.23 merupakan perintah instalasi selenium dalam bahasa Python.

```
Pip selenium install
Pip install upgrade –upgrade pip #upgrade
```

Gambar 3.23. Perintah install selenium dalam bahasa python

Setelah proses eksekusi berjalan maka langkah selanjutnya yaitu *instalasi extension* bahasa pemrograman di dalam kode editor yang digunakan. Selama pengujian ini, penguji menggunakan kode editor *Visual Studio Code*. Adapun proses instalasi *extension library* harus dipilih sesuai dengan bahasa pemrograman yang akan digunakan. Pada Gambar 3.24 mendokumentasikan hasil instalasi *extension python*. Setelah semua terinstalasi dengan baik maka pengujian dengan alat bantu Selenium Web Drivers telah dapat digunakan.



Gambar 3.24. Instalasi extension python

Berbeda dengan Selenium WebDrivers yang harus menginstalasi setiap bahasa pemrograman dan *drivers* yang sesuai dengan browser yang akan digunakan, *command* instalasi Playwright sudah mendukung penginstalan semuanya sehingga tidak diperlukan instalasi satu persatu. Dalam perubahan web browser juga tidak diperlukan instalasi lagi, hal tersebut dikarenakan dapat diubah secara langsung dalam kode pemrograman atau *script test.*

```
1  const { chromium } = require('playwright');
2
3  (async () => {
4    const browser = await chromium.launch({
5      headless: false
6    });
7    const context = await browser.newContext();
8
```

Gambar 3.25. Perubahan web browser pada Playwright

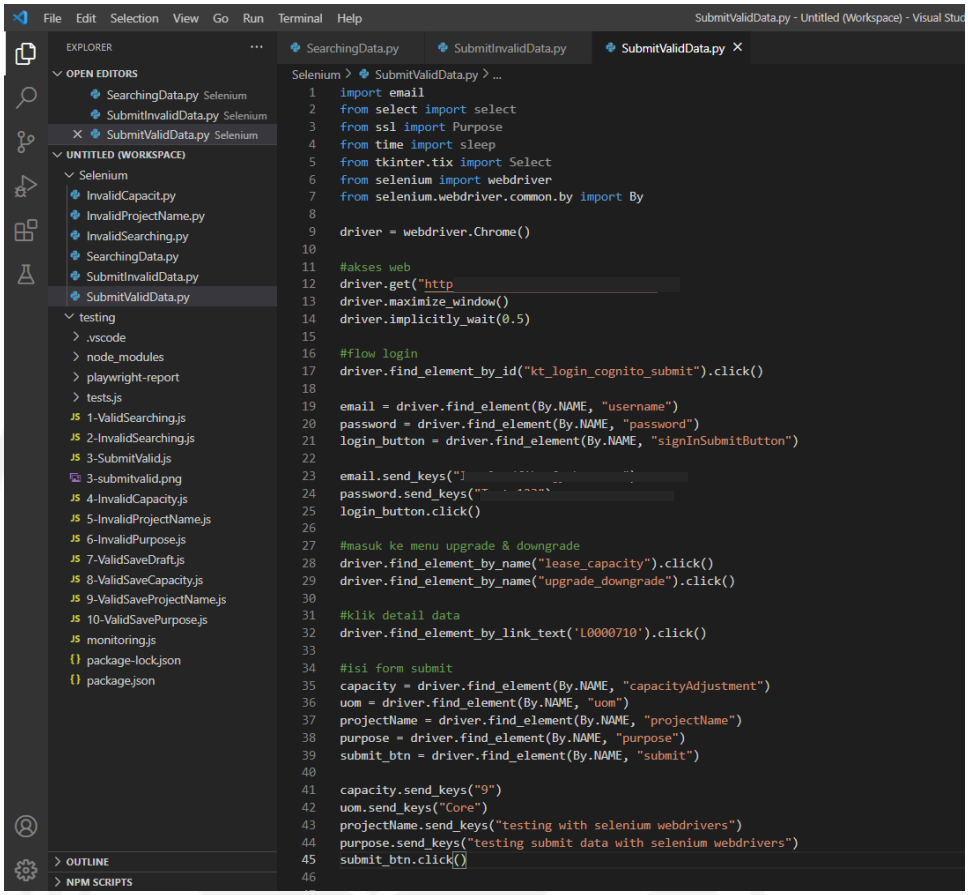
b. Pembuatan dan pengeksekusian *Script Test*

Percobaan pembuatan *script test* menggunakan Selenium Web Drivers telah dilakukan. Selama proses percobaan pembuatan *script test* diperlukan proses adaptasi untuk membuat skenario dengan kode pemrograman manual. Adapun proses adaptasi dalam penggunaan dan pembuatan *script test* Selenium lebih lama daripada proses adaptasi menggunakan Playwright. Hal tersebut dikarenakan penggunaan dan pembuatan Playwright cukup mudah untuk dilakukan. Selain itu, terdapat fitur Playwright Codegen yang membuat *script test* dengan perekaman aktivitas penguji dan akan langsung di-*generate* dalam bentuk kode pemrograman dengan bahasa pemrograman yang dapat dipilih sesuai dengan keinginan penguji. Kode yang telah di-*generate* juga diberikan penjelasan maupun petunjuk singkat terkait proses yang dituliskan di dalam *script* tersebut. Hal tersebut dapat membantu penguji untuk melakukan pembuatan dan perubahan *script test* secara manual.

Dari sisi pembuatan script test, terdapat kesamaan dalam pengambilan elemen atau objek data di Selenium maupun Playwright. Selenium dapat mengidentifikasi atau mengambil objek berdasarkan *key Id* atau *key name* dari objek tersebut. Hampir sama dengan Selenium, Playwright juga mengambil elemen atau objek berdasarkan struktur tersebut. Kesamaan tersebut memudahkan penulis dalam pembuatan *script test* menggunakan Selenium. Pada Gambar 3.26 merupakan contoh pengambilan elemen atau objek pada Selenium dan Playwright. Adapun pada Gambar 3.27 merupakan hasil dokumentasi percobaan pembuatan *script test* menggunakan Selenium WebDrivers.

```
#locator element akses login pada Selenium
Driver.find_element_by_id("kt_login_cognito_submit").click()

#locator element akses login pada Playwright
Await page.click("text=Cognito Login");
```

Gambar 3.26. Locator elemen pada Playwright dan Selenium



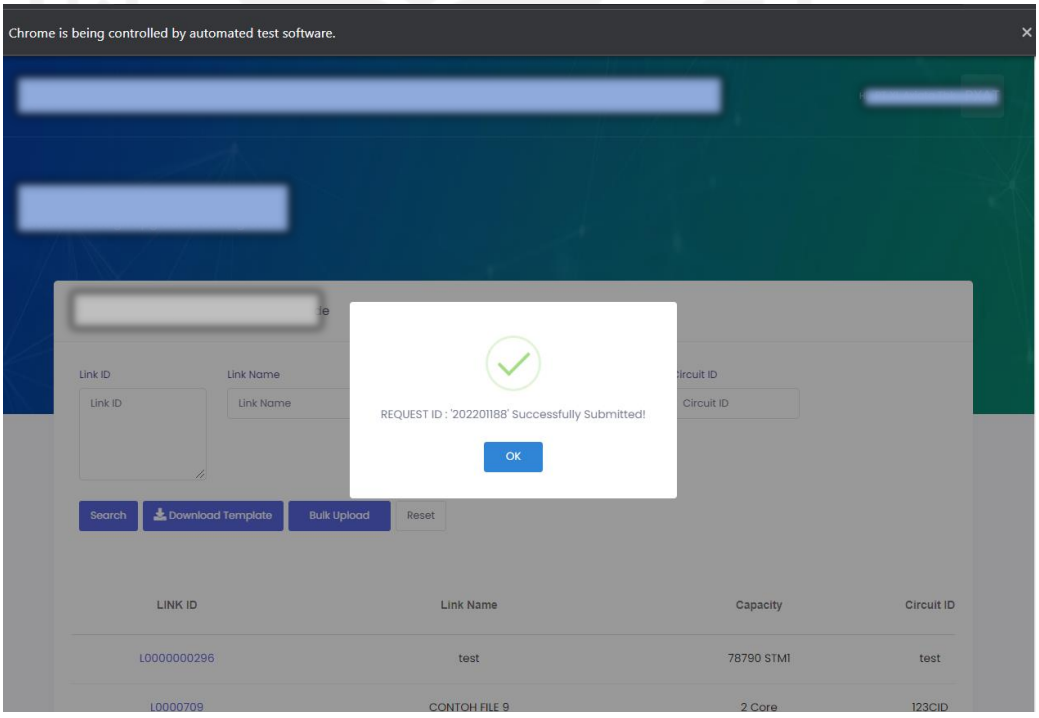
Gambar 3.27. Pembuatan *script test* dengan Selenium WebDriver

Adapun dari hasil pembuatan *script test*, *script* Selenium WebDrivers lebih singkat dari *script test* yang dihasilkan oleh Playwright Codegen. Semua aktivitas yang dilakukan penguji pada web akan di-*generate* oleh Playwright ke dalam bahasa pemrograman yang telah dipilih. Hal tersebut membuat *script test* yang dihasilkan Playwright Codegen cukup panjang. Untuk mempersingkatnya, penulis melakukan perubahan *script* Playwright Codegen dengan mengambil alur dan proses yang penting. Adapun dalam pembuatan *script* Selenium, penulis membutuhkan waktu untuk beradaptasi dalam mempelajari penggunaan serta penulisan *script test*. Akan tetapi skenario yang dibuat cukup sederhana sehingga selama pembuatan *script* tidak terdapat kendala yang besar.

Dari sisi pembuatan *script test*, terdapat sebuah penelitian yang melakukan survei mengenai Selenium dalam kategori pengembangan tes (García, Gallego, Gortázar, & Munoz, 2020). Survei disebarkan ke dalam komunitas besar seperti Selenium Conf Slack dan Selenium Reddit. Salah satu hasil dari survei kategori pengembangan test menyebutkan bahwa sebanyak 36,11% responden mengalami permasalahan dalam pengujian dan pengembangan *script test.* Jawaban responden meliputi sulit untuk

dibuat, sulit untuk ditulis, sulit untuk ditiru sebagai pengguna, dan sulit untuk digunakan kembali. Penguji pemula yang belum memiliki pengalaman *automation test* maupun *skills* pemrograman yang baik merasa kesulitan ketika mencoba membangun pengujian tes Selenium WebDrivers.

Selain dari sisi pembuatan *script test*, dilakukan percobaan pengeksekusian Playwright dan Selenium menggunakan mode *headfull* UI. Mode *headfull* merupakan mode pengeksekusian yang dijalankan dengan menampilkan antarmuka (UI) dari sistem yang sedang diuji. Pada percobaan ini, Playwright dan Selenium menjalankan skenario tes yang sama dan mengeksekusi pada web browser yang sama yaitu Chrome. Adapun hasil dari pengeksekusiannya adalah Playwright menjalankan pengujian lebih cepat daripada Selenium. Hal tersebut berdasarkan pada waktu yang dibutuhkan untuk mengeksekusi kasus uji. Playwright membutuhkan waktu 1 menit 17 detik sedangkan Selenium dengan menjalankan kasus uji yang sama membutuhkan waktu 1 menit 25 detik. Pada Gambar 3.28 merupakan hasil dokumentasi dari pengeksekusian *script test* menggunakan mode *headfull* Selenium.



Gambar 3.28. Pengeksekusian tes menggunakan mode *headfull* Selenium

Adapun dari sisi pelaporan pengujian, secara *default* Selenium tidak memiliki fitur laporan pengujian. Laporan pengujian penting untuk mendokumentasikan semua hasil uji yang telah dieksekusi. Laporan ini tidak hanya untuk mendokumentasikan jumlah hasil *failed* dan *passed* tetapi juga membantu proses pencarian serta penyelesaian *root*

cause dari suatu *bug*. Dalam hal ini, jika dibutuhkan pelaporan pengujian Selenium membutuhkan aplikasi pihak ketiga untuk membantu menyediakan fitur tersebut. Adapun dari sisi Playwright, secara default alat uji Playwright telah mempunyai fitur laporan pengujian salah satunya yaitu HTML *Report*. Penggunaan fitur tersebut dapat dilakukan bersamaan dengan mengeksekusi Playwright *Test Runner* selain itu penggunaannya juga sangat mudah.

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Teknis

4.1.1 Pengujian Otomatis Playwright

Magang memberikan pengalaman dan pengetahuan baru mengenai pengujian terutama pengujian otomatis. Apalagi dalam perkuliahan, belum pernah merasakan secara langsung melakukan pengujian sistem dengan bantuan alat uji. Baik pengujian manual maupun otomatis sama-sama membutuhkan ketelitian dalam pengujiannya. Selain ketelitian, penguji juga diharuskan untuk sabar dan tidak cepat puas ketika berhasil menemukan *bug* karena pasti akan ada *bug* lain yang belum ditemukan. Oleh karena itu, aktivitas pengujian harus dilakukan berkali-kali dan diulangi setiap ada perubahan dalam pengembangan.

Pengujian otomatis dijalankan ketika setiap komponen sudah saling terhubung dan semua fitur dalam modul tersebut telah diselesaikan. Pengujian otomatis juga menjadi syarat akhir *definition of done* dari setiap pengembangan modul. Setiap pengembangan modul dapat di kategorikan selesai di *environment cloud* mauapun *environment development* jika modul tersebut telah dieksekusi menggunakan pengujian otomatis dan memiliki *script* yang di dokumentasikan dalam Gitlab proyek.

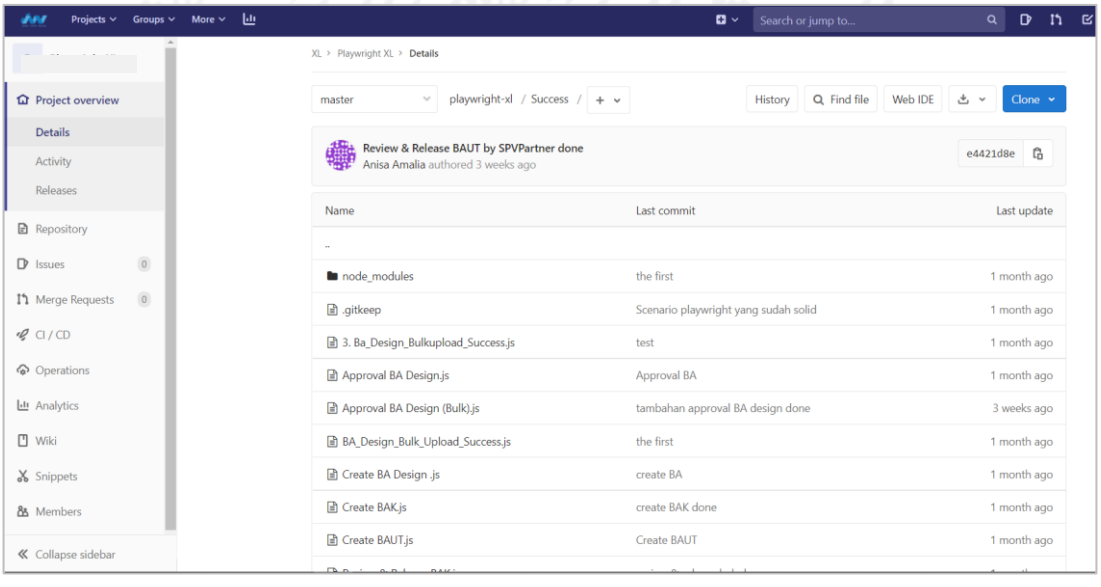
Adapun tidak semua proyek yang dikembangkan telah memanfaatkan pengujian otomatis menggunakan Playwright. Selama magang, dari semua proyek yang dikembangkan, hanya 2 proyek yang mengimplementasikan pengujian otomatis Playwright yaitu pada proyek Sistem Manajemen Jaringan dan *E-Commerce*. Keduanya merupakan sistem berbasis web yang memiliki cakupan pengembangan yang cukup luas. Pengimplementasian Playwright untuk memudahkan pengujian fungsional mungkin bisa diterapkan ke beberapa proyek yang lain. Selain penggunaannya yang mudah, pengujian otomatis juga dapat membantu dalam mempercepat pengujiaan yang berulang-ulang karena penguji hanya perlu menjalankan *script test* tanpa harus melakukannya secara manual. Selain itu, dalam pengujian otomatis juga terdapat bukti laporan yang dapat digunakan sebagai dokumentasi semua hasil uji.

Selama bergabung dalam Proyek Sistem Manajemen Jaringan, terdapat beberapa fitur Playwright yang belum digunakan secara maksimal. Salah satu fitur penting yang belum digunakan yaitu laporan pengujian. Laporan pengujian menjadi hal penting untuk mendokumentasikan semua hasil uji terutama untuk pengujian *multiple test*. Tidak hanya untuk mendokumentasikan jumlah hasil uji *passed* dan *failed* tetapi dalam laporan pengujian Playwright juga mendokumentasikan penyebab error yang terjadi jika status uji tersebut *failed*. Hal tersebut

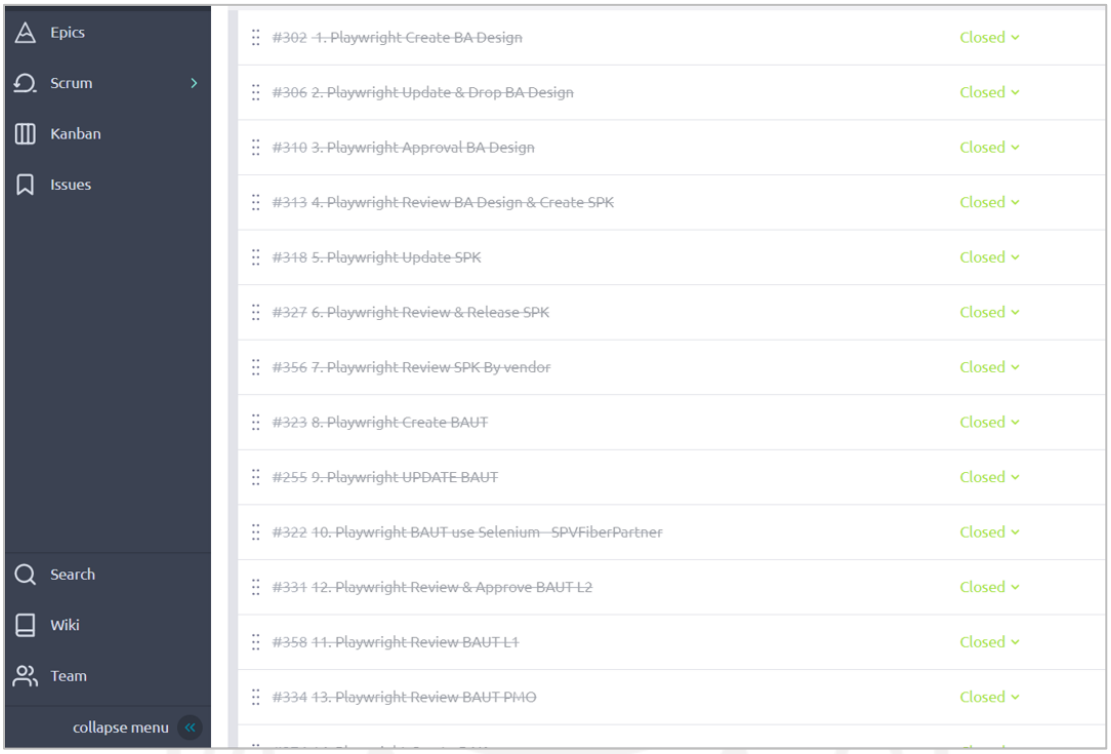
dapat membantu penguji maupun *programmer* untuk melakukan analisis *root cause* terkait gagalnya kasus uji.

Dalam pengimplementasian pengujian otomatis terdapat beberapa hambatan dan tantangan. Salah satu hambatannya yaitu jika sistem belum sepenuhnya selesai dikembangkan dan masih berupa unit-unit yang belum terintegrasi maka akan sulit untuk penggunaan pengujian otomatis. Hal tersebut berkaitan dengan proses adaptasi penggunaan alat uji karena setiap perubahan pada sistem maka penguji juga harus mengubah kode pemrograman tersebut secara manual dalam beberapa *script test*. Setiap *flow* saling berhubungan sehingga diharuskan untuk memperbaiki semua *script test* yang berkaitan. Selain itu, selama mengeksekusi *script test* dalam mode *headfull* yang menjalankan pengujian dengan antarmuka (UI) web, sering kali Playwright tidak bisa langsung menjalankan ke langkah uji selanjutnya atau *stuck* di langkah tersebut. Permasalahan tersebut dapat diatasi dengan bantuan klik secara manual yang dilakukan oleh penguji. Oleh karena itu, penguji juga harus fokus dan teliti jika menjalankan dalam mode *headfull*.

Kolaborasi dan kerjasama tim yang dilakukan selama pengujian menggunakan Playwright juga sangat membantu selama pengujian. Dalam proyek Sistem Manajemen Jaringan terdapat 2 orang *Quality Assurance* yang saling berkolaborasi dalam pembuatan *script test*. Setiap *script test* akan di dokumentasikan menggunakan Gitlab proyek untuk memudahkan proses kerjasama tim. Selain itu, *Project Manager* juga akan mudah untuk memantau progress pembuatan *script test.* Hal tersebut dikarenakan *script test* atau pengujian Playwright ini merupakan *definition of done* dari sebuah modul yang nantinya akan diberikan kepada klien sebagai *document delivery*. Adapun hasil pendokumentasian *script test* pada Gitlab dapat dilihat pada Gambar 4.1 sedangkan daftar *task* pengujian menggunakan Playwright dapat di lihat pada Gambar 4.2.



Gambar 4.1 Tampilan pendokumentasian *script test* di Gitlab



Gambar 4.2 Daftar *task* pengujian otomatis yang telah diselesaikan

4.1.2 Pemilihan Alat Uji

Salah satu hal yang perlu diperhatikan dalam pengujian otomatis adalah pemilihan alat uji yang tepat. Pertimbangan dalam pemilihan alat pengujian dapat disesuaikan dengan kebutuhan pengujian serta kondisi sumber daya manusia dalam tim proyek. Tidak hanya itu, kelebihan serta kelemahan dari masing-masing alat uji juga harus dipertimbangkan. Kemudahannya dalam penggunaan, adanya fitur laporan pengujian, serta kecepatan dalam setiap eksekusi dapat menjadi bahan untuk pertimbangan pemilihan alat uji.

Keputusan pihak berwenang perusahaan dalam memilih alat uji Playwright untuk pengujian otomatis pasti telah dipertimbangkan dengan sebaik-baiknya. Pengimplementasian pengujian otomatis menggunakan alat uji Playwright baru dijalankan pada awal tahun 2020. Playwright merupakan alat uji otomatis terbaru yang dapat mendukung dan mempermudah dalam proses pengujian. Adapun Selenium merupakan alat uji populer yang telah banyak digunakan *developers* untuk proses pengujian web. Keduanya sama-sama alat uji berbasis *open source* yang mendukung pengujian lintas browser.

Proses pembuatan *script test* dapat dibuat dengan manual maupun perekaman. Proses manual dilakukan dengan penulisan *script test* secara mandiri tanpa *generate* kode sedangkan proses perekaman dapat dilakukan dengan bantuan Playwright Codegen yang berfungsi dalam *generate* aktivitas uji ke dalam bahasa pemrogaman. Kemudahan Playwright dalam pembuatan *script test* melalui fitur Playwright Codegen sangat membantu penguji dalam membuat skenario uji. Fitur

tersebut membantu mempercepat dalam pembuatan *script test* dengan banyak bahasa pemrograman tergantung dari pilihan penguji.

Adapun selama aktivitas magang, penulis belum pernah menggunakan Selenium dalam proses pengujiannya. Akan tetapi dalam penyusunan laporan akhir, penulis telah melakukan percobaan pengujian Sistem Manajemen Jaringan *Upgrade* & *Downgrade* menggunakan alat uji Selenium WebDrivers. Alat uji tersebut memiliki beberapa persamaan dengan Playwright yaitu sama-sama mendukung pengujian web, gratis *open source*, dan pengujian berbasis lintas browser. Selain itu, dalam penulisan *script test* pada Selenium juga memiliki kemiripan dengan Playwright. Selenium dapat mengidentifikasi atau mengambil objek berdasarkan *key Id* atau *key name* dari objek tersebut. Hampir sama dengan Selenium, Playwright juga mengambil objek berdasarkan struktur objek tersebut.

4.2 Non Teknis

4.2.1 Komunikasi

Terdapat dua tipe komunikasi yaitu secara lisan maupun tulisan, keduanya merupakan keterampilan yang harus dimiliki oleh setiap orang. Dalam dunia kerja, komunikasi yang jelas sangat dibutuhkan supaya tidak adanya kesalahan arti komunikasi antara pengirim dan penerima informasi. Komunikasi tidak hanya sekadar jelas, tetapi juga harus memerhatikan aturan dan kesopanan. Selama magang, penulis belajar berkomunikasi dan menyampaikan pendapat yang baik ketika bekerja di lingkungan perusahaan. Berkomunikasi seperti menyampaikan kendala dan menyampaikan pendapat merupakan hal penting yang dilakukan untuk menjaga hubungan baik antar tim guna mencapai tujuan bersama.

Magang sebagai *Quality Assurance* diharuskan untuk berkomunikasi dengan tim proyek dan klien eksternal. Komunikasi di dalam tim proyek dilakukan untuk melaporkan setiap progress pengujian dan kendala yang terjadi ketika pembuatan *script test* maupun jika terdapat error yang belum diketahui penyebabnya. Penulis juga sering di bantu dan bekerja sama dengan Sistem Analis dan Programmer dalam mencari serta menyelesaikan solusi terkait *bug* yang ditemukan. Selain itu, penulis juga melakukan komunikasi dengan *Manager* atau *Lead* QA jika terdapat kendala teknis selama penggunaan Playwright maupun untuk saling berdiskusi mengenai strategi dalam pengimplementasian pengujian otomatis menggunakan Playwright.

Adapun selama magang berlangsung, komunikasi antar tim QA di dalam proyek juga cukup penting apalagi untuk proses koordinasi dalam pembagian dan monitoring *task* pengujian. Pembagian task pengujian otomatis modul Sistem Manajemen Jaringan dibagi menjadi 2 orang sehingga diperlukan komunikasi satu sama lain. Setiap terdapat kendala dalam pengujian maupun pemahaman *acceptance criteria* dari setiap *task* maka penulis selalu berdiskusi dengan *partner*

QA internal. Biasanya proses diskusi langsung dilakukan menggunakan *Google Meet* supaya lebih mudah untuk untuk proses berdiskusi secara sinkron.

Komunikasi terkait pengujian tidak hanya dilakukan dalam lingkup internal tetapi juga lingkup eksternal bersama dengan klien. Setiap hari terdapat sesi *Daily Scrum* selama 15 menit yang dilakukan untuk melaporkan aktivitas yang dilakukan kemarin, aktivitas yang akan dilakukan pada hari itu, dan hasil maupun hambatan dari aktivitas yang telah dilakukan kemarin. Selain itu, setiap minggunya juga terdapat agenda inspeksi untuk melaporkan dan mempresentasikan semua hasil pengujian pada *sprint* yang sedang berjalan. Proses tersebut sangat membantu penulis untuk mempelajari cara berkomunikasi yang baik, menyampaikan pendapat, menghargai pendapat orang lain, serta berdiskusi.

4.2.2 Proses Beradaptasi Menggunakan Alat Uji

Pemilihan alat uji Playwright sebagai alat bantu dalam pengujian otomatis pasti telah dipertimbangkan dengan baik oleh manager maupun pihak berwenang. Sebelum penggunaan Playwright, *Lead* QA memberikan *training* bersama seluruh divisi selama 1 jam. Dalam waktu singkat tersebut, semua peserta training sudah dapat membuat dan menjalankan pengujian sederhana. Dari hal tersebut dapat dijelaskan bahwa penggunaan Playwright sangat mudah untuk dipelajari dalam waktu yang singkat.

Adapun selama penyusunan laporan ini, penulis telah mencoba membuat *script test* dan mengeksekusinya menggunakan Selenium. Penulis berpendapat bahwa dalam pembuatan *script test* antara Selenium dan Playwright memiliki persamaan dalam pengambilan objek. Penulis juga telah mampu membuat beberapa skenario sederhana dalam menguji modul Sistem Manajemen Jaringan. Hanya saja jika belum terbiasa dan belum memiliki dasar pengetahuan mengenai pemrograman maka membutuhkan waktu lebih untuk mempelajarinya. Tidak hanya itu, dalam proses instalasi Selenium juga memiliki banyak langkah yang harus dilakukan dari mulai menginstalasi *web drivers* sesuai dengan browser yang digunakan, menginstalasi *extension* sesuai dengan bahasa pemrograma, serta merubah *path environment*. Proses tersebut cukup lama jika dibandingkan dengan Playwright yang hanya menjalankan dua *command line* di CMD dan dapat langsung menginstalasi Playwright.

Dari perbedaan dan persamaan tersebut dapat menggambarkan bahwa proses pemilihan alat uji dapat disesuaikan dengan kebutuhan proyek maupun sumber daya tim yang ada. Penulis berpendapat bahwa pemilihan Playwright sebagai alat uji Sistem Manajemen Jaringan sudah cukup tepat.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Percobaan pengujian Sistem Manajemen Jaringan dalam proses *Create Link* berhasil dijalankan menggunakan Playwright dengan persentase lulus uji 100%. Kemudahan Playwright Codegen dalam pembuatan *script test* yang terotomatis secara langsung ketika melakukan perekaman kasus uji sangat membantu penguji. Adapun baik Playwright maupun Selenium merupakan *automation testing tools* yang memiliki performa bagus untuk pengujian web. Masing-masing alat uji memiliki kelebihan dan kelemahan. Oleh karena itu, dalam pemilihan alat uji dapat disesuaikan dengan kebutuhan proyek dan ketersedian sumber daya manusia di dalam proyek tersebut.

Selenium WebDrivers akan lebih mudah diimplementasikan dan digunakan oleh penguji yang sudah terbiasa dalam pengujian otomatis dan memiliki pengetahuan dasar mengenai pemrograman. Pengetahuan tersebut digunakan untuk proses pengujian seperti pembuatan *script test* maupun pengintegrasian dengan alat pihak ketiga. Adapun Playwright cocok diimplementasikan dan digunakan oleh penguji yang baru mengenal *automation testing* karena kemudahannya dalam pengujian. Alat ini dapat digunakan dari pemula hingga profesional, apalagi intensitas tim *development* Playwright yang selalu melakukan pembaharuan dan inovasi akan membuat Playwright besar di masa mendatang.

5.2 Saran

Terdapat saran untuk penelitian yang selanjutnya yaitu melakukan pengukuran kecepatan Playwright dalam mengeksekusi *multiple test* untuk proses pengujian regresi. Hasil dari pengeksekusian uji regresi tersebut dapat dibandingkan dengan alat uji yang lainnya seperti Selenium. Selain itu, terdapat juga saran untuk melakukan integrasi *tools* pihak ketiga dalam pendokumentasian laporan pengujian menggunakan *Allure Report* dikarenakan pada versi yang terbaru, Playwright telah didukung oleh tools pihak ketiga untuk menyediakan laporan pengujian selain menggunakan HTML *Report*.

DAFTAR PUSTAKA

Bhagat, B., Bhattacharjee, S., & Ratre, S. (2020). Software testing techniques & automation tools. *Mukt Shabd Journal,* , 5957–5962.

Chatterjee, A. (2018). COST-BENEFIT ANALYSIS OF AUTOMATION. *Dspace Delhi Technological University*.

García, B., Gallego, M., Gortázar, F., & Munoz, O. M. (2020). A survey of the selenium ecosystem. *Electronics*.

Gojare, S., Joshi, R., & Gaigaware, D. (2015). Analysis and design of selenium webdriver automation testing framework. *Procedia Computer Science*, 341-346.

Jin, J., & Xue, F. (2011). Rethinking software testing based on software architecture. *2011 Seventh International Conference on Semantics, Knowledge and Grids*, 148-151.

Karuniawati, S., Sri, W., & Hakim, I. L. (2015). Implementasi Metode Cause Effect Graphing (CEG) dalam Pengujian Requirement Perangkat Lunak (Stud iKasus: Aplikasi G-College). *e-Proceeding of Engineering*, 6475-6480.

Kaur, H., & Gupta, G. (2013). Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete. *Int. Journal of Engineering Research and Applications* , 1739-1743.

Lawanna, A. (2012). The Theory of Software Testing. *AU Journal of Technology*, 35-40.

Playwright. (2021, November 10). Diambil kembali dari Playwright Dev: https://playwright.dev/

Raghavendra, S. (2021). Introduction to Selenium. In Python Testing with Selenium. *Apress, Berkeley, CA.*, 1-14.

Saravanan, K., & Prasad, E. P. (2016). Open source software test automation tools : A competitive necessity. *Scholedge International Journal of Management & Development, 3(6)*, 103-110.

Selenium. (2021, November 11). Diambil kembali dari Selenium dev: https://www.selenium.dev/

Sharma, M., & Angmo, R. (2014). Web Based Automation Testing and Tools. *(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (1) , 2014, 908-9*, 908-912.

Sulistyanto, H., & SN, A. (2014). URGENSI PENGUJIAN PADA KEMAJEMUKAN PERANGKAT LUNAK DALAM MULTI PERSPEKTIF. *Komuniti : Jurnal Komunikasi dan Teknologi Informasi*, 65-74.

LAMPIRAN