

MODUL
PRAKTIKUM KUALITAS PERANGKAT LUNAK
MODUL KATALON



Oleh
TEAM TI 4B
Dosen Pembimbing : Musta'inul Abdi, SST., M.Kom.

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER
TAHUN 2022

KATA PENGANTAR

Puji serta syukur penulis ucapkan kepada Allah SWT yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan laporan hasil Magang Industri ini dilakukan dalam rangka memenuhi salah satu syarat untuk mendapatkan nilai Mata Kuliah Praktikum Kualitas Perangkat Lunak pada semester VII (Tujuh) Jurusan Teknologi Informasi dan Komputer Program Studi D-IV Teknik Informatika.

Katalon adalah software testing yang digunakan untuk menguji kualitas dan fungsi dari aplikasi yang telah diproduksi. Diluncurkan pada September 2016, Katalon sukses menembus hingga 9% penetrasi pasar untuk UI (user interface) test automation hanya dalam jangka waktu dua tahun.

Pada kesempatan ini penulis ingin mengucapkan terima kasih kepada :

1. Bapak Musta'inul Abdi, SST., M.Kom. selaku pembimbing Mata Kuliah Praktikum Kualitas Perangkat Lunak.
2. Bapak Muhammad Arhami, S.Si, M.Kom selaku Ketua Jurusan Teknologi Informasi dan Komputer.
3. Bapak Salahuddin, ST, M.Cs selaku Ketua Program Studi D-IV Teknik Informatika.
4. Bapak Ir. Rizal Syahyadi, ST., M.Eng. Sc selaku Direktur Politeknik Negeri Lhokseumawe.

Penulis menyadari masih banyak kekurangan dan kelemahan dalam pelaksanaan dan penyusunan Modul ini. Namun, penulis berharap semoga Modul ini dapat bermanfaat bagi pembaca. Dengan demikian, segala kritik dan saran yang

membangun dari para pembaca akan penulis terima sehingga dapat menjadi sebuah pelajaran agar dapat membuat dengan lebih baik lagi.

Lhokseumawe, 2022

Penulis

TEAM TI 4B

DAFTAR ISI

	HAL
KATA PENGANTAR	ii
DAFTAR ISI	iv
DAFTAR GAMBAR.....	vi
DAFTAR TABEL	xii
BAB 1 PENDAHULUAN	1
1.1. Tujuan	1
1.2. Dasar Teori.....	1
1.3. Percobaan.....	2
BAB 2 MENGAMBIL DATA CSV.....	11
2.1. Tujuan	11
2.2. Dasar Teori.....	11
2.3. Percobaan.....	11
BAB 3 RESULT AND REPORTING	17
3.1. Tujuan	17
3.2. Dasar Teori.....	17
3.3. Percobaan.....	17
3.3.1. Test Execution Reports	17
3.3.2. Basic Reports Plugins.....	22
3.3.3. How To Email Results	26
3.3.4. How To Use Katalon Analytics Step by Step	32
BAB 4 MENJALANKAN TEST MENGGUNAKAN COMMANDLINE DAN PLUGIN JENKINS PADA KATALON STUDIO	36
4.1. Tujuan	36
4.2. Dasar Teori.....	36
4.3. Percobaan.....	37
BAB 5 INTEGRATION WITH GIT AND JENKINS	42
5.1. Tujuan	42
5.2. Dasar Teori.....	42
5.3. Percobaan Integrasi Dengan GIT	42
5.4. Percobaan Integrasi Dengan Jenkins	51
BAB 6 API TESTING	56
6.1. Tujuan	56

6.2. Dasar Teori.....	56
6.3. Percobaan API Testing	56
DAFTAR PUSTAKA.....	89

DAFTAR GAMBAR

Gambar 1.1 <i>test suite</i>	1
Gambar 1.2 <i>test suite paraller</i>	2
Gambar 1.3 <i>Tampilan pemilihan langkah pertama</i>	2
Gambar 1.4 <i>Tampilan atur nama test suite</i>	3
Gambar 1.5 <i>Tampilan hasil setelah membuat test suite</i>	3
Gambar 1.6 <i>Tampilan test case browser</i>	3
Gambar 1.7 <i>Tampilan hasil setelah masuk test case ke test suite</i>	4
Gambar 1.8 <i>Tampilan browser</i>	4
Gambar 1.9 <i>Tampilan proses berjalan</i>	4
Gambar 1.10 <i>Tampilan website orangehrmlive</i>	4
Gambar 1.11 <i>Tampilan hasil test suite</i>	5
Gambar 1.12 <i>Tampilan detail test suite</i>	5
Gambar 1.13 <i>Tampilan hasil test suite di folder</i>	5
Gambar 1.14 <i>Tampilan hasil laporan test suite</i>	5
Gambar 1.15 <i>Tampilan daftar laporan</i>	6
Gambar 1.16 <i>Tampilan menambahkan test case pada test suite</i>	6
Gambar 1.17 <i>Tampilan file sample test ke dalam folder login</i>	6
Gambar 1.18 <i>Tampilan test suite collection</i>	7
Gambar 1.19 <i>Tampilan create new test suite</i>	7
Gambar 1.20 <i>Tampilan hasil test suite collection</i>	7
Gambar 1.21 <i>Tampilan test suite collection browser</i>	8
Gambar 1.22 <i>Tampilan hasil setelah test suite collection dirun</i>	8
Gambar 1.23 <i>Tampilan proses berjalan</i>	8
Gambar 1.24 <i>Tampilan hasil browser</i>	9
Gambar 1.25 <i>Tampilan isi dari Log Viewer</i>	9
Gambar 1.26 <i>Tampilan pengujian menggunakan Test Suite Collection</i>	9
Gambar 1.27 <i>Tampilan isi dari file laporan test suite collection</i>	10
Gambar 1.28 <i>Tampilan show detail dibrowser chrome</i>	10
Gambar 1.29 <i>Tampilan show detail dibrowser firebox</i>	10
Gambar 2.1 File Excel	11
Gambar 2.2 Add Variabels.....	11
Gambar 2.3 Hasil Variables	12
Gambar 2.4 Mengubah Value	12
Gambar 2.5 Object Password	12
Gambar 2.6 Save	12
Gambar 2.7 Create New Suite	13
Gambar 2.8 Add Test 2	13
Gambar 2.9 Show Data Binding.....	13
Gambar 2.10 Variabel Binding	14
Gambar 2.11 <i>New Test Data</i>	14
Gambar 2.12 <i>Test Explore</i>	14
Gambar 2.13 Tahap Browser	15

Gambar 2.14 File xlsx	15
Gambar 2.15 Test Data	15
Gambar 2.16 Proses Variabel Binding	16
Gambar 2.17 Proses menjalankan	16
Gambar 2.18 Hasil Pengujian.....	16
Gambar 3.1 <i>Menu log view pada katalon</i>	17
Gambar 3.2 <i>Tes yang dijalankan</i>	18
Gambar 3.3 <i>Menu filter log</i>	18
Gambar 3.4 <i>Tampilan hirarki log</i>	18
Gambar 3.5 <i>Pengaturan format report</i>	19
Gambar 3.6 <i>Pilih menu report</i>	19
Gambar 3.7 <i>Report yang sudah dipilih</i>	19
Gambar 3.8 <i>Tampilan jendela report</i>	20
Gambar 3.9 <i>Tampilan test case table</i>	20
Gambar 3.10 <i>Tampilan tab summary</i>	20
Gambar 3.11 <i>Tampilan tab execution setting</i>	21
Gambar 3.12 <i>Tampilan tab execution environtment</i>	21
Gambar 3.13 <i>Menu test case detail</i>	21
Gambar 3.14 <i>Test case dialog</i>	22
Gambar 3.15 <i>Tampilan test case suite</i>	22
Gambar 3.16 <i>Detail test case suite</i>	22
Gambar 3.17 <i>Tampilan menu project</i>	23
Gambar 3.18 <i>Menu pada project setting</i>	23
Gambar 3.19 <i>List pengujian yang telah dibuat</i>	23
Gambar 3.20 <i>Open containing folder</i>	24
Gambar 3.21 <i>Lokasi folder disimpan</i>	24
Gambar 3.22 <i>Hasil pengujian</i>	24
Gambar 3.23 <i>Laporan pengujian dalam format HTML</i>	25
Gambar 3.24 <i>Laporan pengujian dalam format CSV</i>	25
Gambar 3.25 <i>Laporan di ekspor dalam format PDF</i>	26
Gambar 3.26 <i>Pilih lokasi penyimpanan file</i>	26
Gambar 3.27 <i>Pilih menu project setting pada toolbar</i>	27
Gambar 3.28 <i>Pilih template email</i>	27
Gambar 3.29 <i>Tampilan mail server setting</i>	27
Gambar 3.30 <i>Pengisian Email pada mail server setting</i>	28
Gambar 3.31 <i>Pilih apply</i>	28
Gambar 3.32 <i>Tampilan saat proses berjalan</i>	28
Gambar 3.33 <i>Tampilan proses gagal (error)</i>	29
Gambar 3.34 <i>Tampilan menu security</i>	29
Gambar 3.35 <i>Tampilan jendela app password</i>	29
Gambar 3.36 <i>Pilih generate</i>	30
Gambar 3.37 <i>Tampilan halaman App password</i>	30
Gambar 3.38 <i>Proses berhasil</i>	30
Gambar 3.39 <i>Tampilan report format</i>	31
Gambar 3.40 <i>Mengubah template test Suite</i>	31
Gambar 3.41 <i>Halaman Execution Infromation</i>	31
Gambar 3.42 <i>Notifikasi file PDF masuk pada email</i>	32

Gambar 3.43 <i>Pilihan untuk memulai project baru</i>	32
Gambar 3.44 <i>Open project</i>	33
Gambar 3.45 <i>Halaman test activities</i>	33
Gambar 3.46 <i>Pilihan test Suite</i>	34
Gambar 3.47 <i>Test Suite yang sudah ditambahkan</i>	34
Gambar 3.48 <i>Test Suite dijalankan menggunakan Browser</i>	34
Gambar 3.49 <i>Test Suite sedang di eksekusi</i>	35
Gambar 3.50 <i>Tampilan hasil detail report</i>	35
Gambar 4.1 Melihat Versi Java Pada CMD	37
Gambar 4.2 Mengenerate Command Pada Aplikasi Katalon	38
Gambar 4.3 Copy to Clipboard Generated Command	39
Gambar 4.4 Path Instalasi Folder Pada CMD	39
Gambar 4.5 Menempel Perintah Generated Command	39
Gambar 4.6 Mengganti Path Ke File Runtime Engine	40
Gambar 4.7 Testing Website Otomatis Pada Browser Chrome	40
Gambar 4.8 Output Testing Pada CMD	41
Gambar 5.1 Membuat <i>repository</i> baru	43
Gambar 5.2 Pengaturan <i>repository</i>	43
Gambar 5.3 Membuat projek di <i>Dashboard</i> Katalon	43
Gambar 5.4 Memilih projek tim	44
Gambar 5.5 Memberi nama projek	44
Gambar 5.6 Hasil pembuatan projek	44
Gambar 5.7 Koneksi dengan <i>repository</i> GIT	44
Gambar 5.8 Meng-copy link untuk <i>clone</i> repo	45
Gambar 5.9 <i>Copy API token</i>	45
Gambar 5.10 Membuka projek di Katalon Studio	45
Gambar 5.11 Memilih projek dengan API token	46
Gambar 5.12 Melakukan <i>commit</i> projek	46
Gambar 5.13 <i>Commit</i> projek Katalon Studio	46
Gambar 5.14 Terdapat <i>Pull Request</i> di Repo GitHub	47
Gambar 5.15 Membuat <i>test case</i>	47
Gambar 5.16 <i>Commit test case</i>	47
Gambar 5.17 Hasil pada repo GitHub	48
Gambar 5.18 Isi folder Test Case	48
Gambar 5.19 Koneksi Katalon dengan GitHub	49
Gambar 5.20 <i>Open Project</i>	49
Gambar 5.21 Hasil <i>Open Project</i>	49
Gambar 5.22 Membuat <i>test case</i> baru	50
Gambar 5.23 <i>Commit</i> perubahan projek	50
Gambar 5.24 Hasil perubahan repo GitHub	50
Gambar 5.25 Isi folder <i>Test Case</i>	51
Gambar 5.26 Isi <i>test case 1 by tom</i>	51
Gambar 5.27 isi <i>test case 2 by ranran</i>	51
Gambar 5.28 Membuat <i>Test Suite</i>	52
Gambar 5.29 Memberi nama <i>Test Suite</i>	52
Gambar 5.30 Memasukkan <i>test case</i> ke dalam <i>test suite</i>	52
Gambar 5.31 Hasil setelah menambahkan <i>test case</i>	53

Gambar 5.32 Jendela <i>Generate Command for Console Mode</i>	53
Gambar 5.33 <i>Copy</i> teks ke <i>Clipboard</i>	53
Gambar 5.34 <i>Download GIT</i>	54
Gambar 5.35 <i>Installer GIT</i>	54
Gambar 5.36 Menyalin link <i>clone repository</i>	54
Gambar 5.37 Melakukan <i>pull repository</i>	55
Gambar 5.38 Melakukan <i>push repository</i>	55
Gambar 6.1 Membuat Folder Baru.....	56
Gambar 6.2 Pilih Web Service Request	57
Gambar 6.3 Membuat Web Service Request.....	57
Gambar 6.4 URL API Pada Web	57
Gambar 6.5 Tampilan File	58
Gambar 6.6 Tampilan Response	58
Gambar 6.7 Membuat Folder Baru.....	59
Gambar 6.8 Pilih Web Service Request	59
Gambar 6.9 Membuat Web Service Request.....	59
Gambar 6.10 Searching Website di Google	60
Gambar 6.11 Tampilan Website.....	60
Gambar 6.12 Tampilan File	60
Gambar 6.13 Tampilan Service Function.....	61
Gambar 6.14 Membuat Folder User	61
Gambar 6.15 Folder User	61
Gambar 6.16 Form Test Case.....	62
Gambar 6.17 Web Service Keyword	62
Gambar 6.18 Send Request	62
Gambar 6.19 Get User Details	63
Gambar 6.20 Response Pada Kolom Output	63
Gambar 6.21 Verify Response Status Code	63
Gambar 6.22 Verify Response Status Code	63
Gambar 6.23 Property Value.....	64
Gambar 6.24 Verify Element Property Value	64
Gambar 6.25 Klik Icon Play.....	64
Gambar 6.26 Melihat Hasil di Log Viewer	64
Gambar 6.27 Mengganti isi dari Verify Element Property Value.....	65
Gambar 6.28 Hasil Error	65
Gambar 6.29 Penyebab Error	65
Gambar 6.30 Membuat File UserValidation	66
Gambar 6.31 Tampilan dari File UserValidation	66
Gambar 6.32 Centang Validate User Details.....	66
Gambar 6.33 Klik Icon Play.....	66
Gambar 6.34 Hasil dari Log Viewer	67
Gambar 6.35 Folder Reports Melihat Hasil Pengujian	67
Gambar 6.36 Hasil Pengujian File Json	67
Gambar 6.37 Contoh SOAP API.....	68
Gambar 6.38 Script SOAP API.....	68
Gambar 6.39 Membuat Folder	69
Gambar 6.40 Folder Countryinfoservices	69

Gambar 6.41 Membuat Web Service Request.....	69
Gambar 6.42 Tambahkan Ekstensi Wizdler pada Chrome	70
Gambar 6.43 Ikon dari Ekstensi Wizdler	70
Gambar 6.44 List Request.....	70
Gambar 6.45 Request dari ListOfCountryNamesByName.....	70
Gambar 6.46 Web Service Request dengan nama ListCountries.....	71
Gambar 6.47 List Countries Sudah Dibuat.....	71
Gambar 6.48 Load Service Function.....	72
Gambar 6.49 Service Function.....	72
Gambar 6.50 Copy Script.....	72
Gambar 6.51 Save Script.....	73
Gambar 6.52 Request Script.....	73
Gambar 6.53 List Nama dari Berbagai Negara	73
Gambar 6.54 Web Service Request SOAP.....	74
Gambar 6.55 Server Function	74
Gambar 6.56 Capital City	74
Gambar 6.57 Script CapitalCity	75
Gambar 6.58 Save Script.....	75
Gambar 6.59 Hasil Request.....	75
Gambar 6.60 Mengambil Request Berdasarkan Variabel Tertentu	76
Gambar 6.61 Menu Variabel	76
Gambar 6.62 Isi Tabel Variabel	77
Gambar 6.63 Isi Value	77
Gambar 6.64 Hasil Request yang Sudah Dimodifikasi	78
Gambar 6.65 Pengujian Test Case dari Kedua Web Service Request	78
Gambar 6.66 Variabel Response1	78
Gambar 6.67 Script dari Variabel Response1.....	79
Gambar 6.68 Mengambil Nilai Tertentu dari Respon	79
Gambar 6.69 List Cuplikan	80
Gambar 6.70 Verifikasi Nilai XML	80
Gambar 6.71 Variabel xml1	81
Gambar 6.72 Parsing Nilai Response	81
Gambar 6.73 Code dari Variabel Countrycode	81
Gambar 6.74 Variabel Countrycode disimpan pada Variabel GlobalVariable	81
Gambar 6.75 Panggil API GetCapital	82
Gambar 6.76 Simpan dan Jalankan API GetCapital	82
Gambar 6.77 Hasil Verifikasi.....	82
Gambar 6.78 Ekstrak Country Code	83
Gambar 6.79 Hasil Country Code yang Diekstrak	83
Gambar 6.80 Validasi Country Code	83
Gambar 6.81 Memperlihatkan Hasil Country Code	84
Gambar 6.82 Code Verifikasi.....	84
Gambar 6.83 Memverifikasi Request.....	85
Gambar 6.84 Simpan Script dan Jalankan Test Case	85
Gambar 6.85 Verifikasi yang Sudah Berhasil	85
Gambar 6.86 Sample Rest API	86
Gambar 6.87 URL Reqres.....	86

Gambar 6.88 Salin URL.....	87
Gambar 6.89 Respon dari Teks URL reqres.in.....	87
Gambar 6.90 Web Service Request pada reqres.in.....	87
Gambar 6.91 Contoh API.....	88
Gambar 6.92 Hasil Pengujian API-CHAINING	88

DAFTAR TABEL

No table of figures entries found.

BAB 1

PENDAHULUAN

1.1. Tujuan

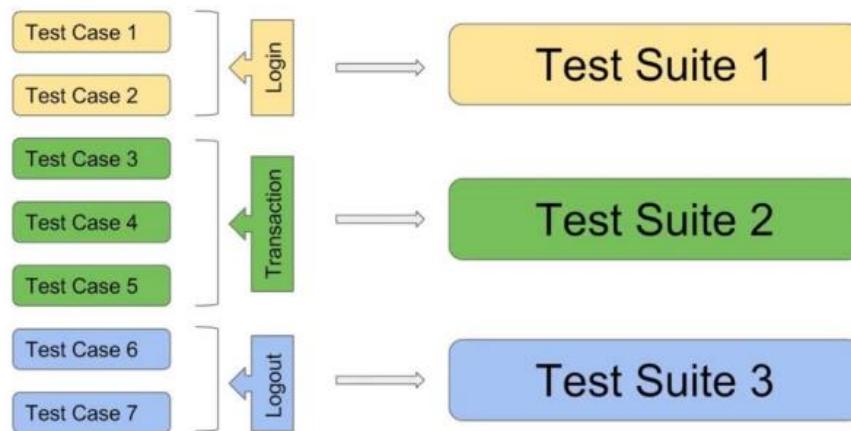
Pada bagian ini akan membahas tentang Test Suite (TS) dan Test Suite Parallel pada katalon studio, Diharapkan pembaca dapat :

1. Mahasiswa mampu memahami Test Suite dan Test Suite Parallel.
2. Mahasiswa mampu memahami langkah-langkah untuk membuat Test lSuite dan Test Suite Paraller.

1.2. Dasar Teori

A. Apa Itu Test Suite?

Test Suite (TS) adalah kumpulan kasus uji. Test Suite juga dapat dikatakan sebagai Test yang di dalamnya terdapat koleksi dari banyak Test Case. Sebuah Test Case dapat menjadi bagian dari Test Suite yang berbeda. Tiap Test Case mempunyai peran masing-masing. Berikut adalah gambar sebuah Test Suite dapat mempunyai lebih dari satu Test Case.



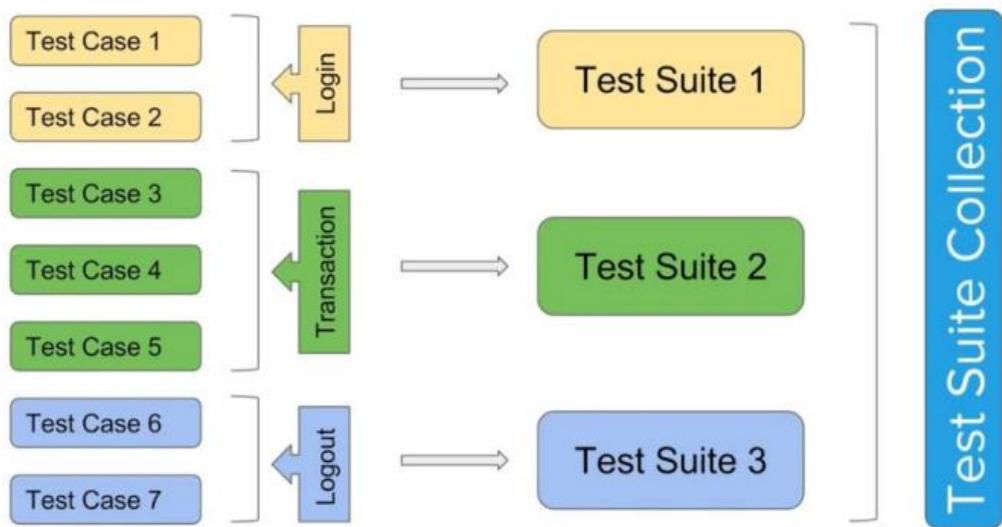
Gambar 1.1 *test suite*

B. Apa itu Test Suite Parallel?

Test Suite Parallel adalah test yang digunakan oleh user untuk menjalankan beberapa pengujian secara otomatis. perbedaan pengujian secara paralel menggunakan Test Suite dan pengujian hanya dengan Test Suite yaitu terletak pada

bagian pengujian yang dilakukan. Pengujian secara Test Suite biasa akan menguji satu persatu dari Test Case yang ada.

Akan tetapi, pengujian menggunakan paralel dapat meminimalkan waktu yang dibutuhkan untuk pengujian, yaitu dengan cara melakukan pengujian bersamaan sekali jalan. Oleh karena itu, pengujian secara paralel dikatakan sebagai pengujian yang dapat mengoptimalkan waktu dengan tepat.

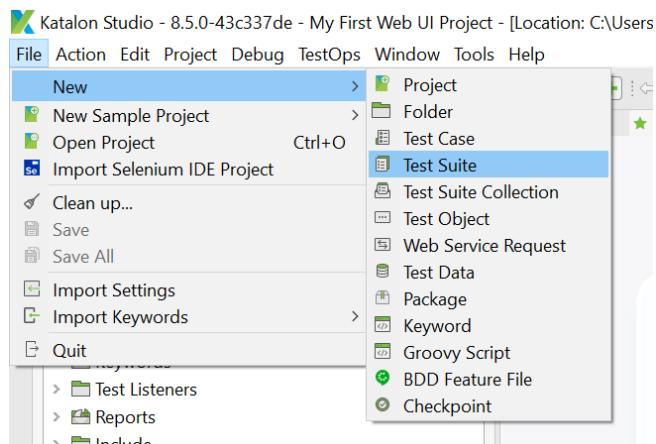


Gambar 1.2 *test suite paraller*

1.3. Percobaan

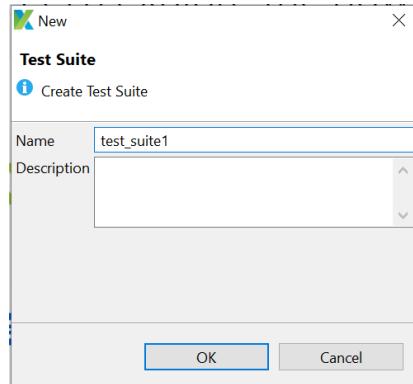
- A. Berikut ini adalah Langkah-langkah untuk membuat Test Suite :

 - Untuk membuat Test Suite, pilih File > New > Test Suite



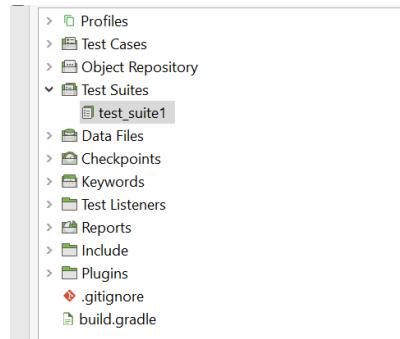
Gambar 1.3 Tampilan pemilihan langkah pertama

- Atur nama Test Suite dengan test_suite1



Gambar 1.4 Tampilan atur nama test suite

- Hasil setelah membuat Test Suite



Gambar 1.5 Tampilan hasil setelah membuat test suite

- Untuk menambahkan Test Case ke dalam Test Suite, yaitu dengan cara klik Add dan centang pada Test Case yang akan dimasukkan, kemudian klik OK.



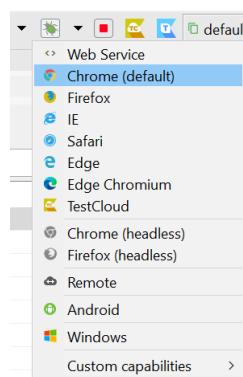
Gambar 1.6 Tampilan test case browser

- Hasil setelah memasukkan Test Case ke dalam Test Suite, kemudian tambahkan centang di sample_test_case pada bagian run

Add Move Up		Show Data Binding	
Enter text to search...			
No.	ID	Description	Run
1	Test Cases/sample test_case		<input checked="" type="checkbox"/>

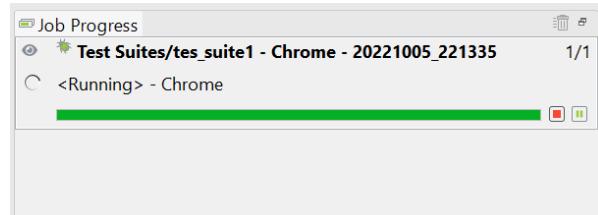
Gambar 1.7 Tampilan hasil setelah masuk test case ke test suite

- Kemudian jalankan pada browser, dan lihat hasilnya



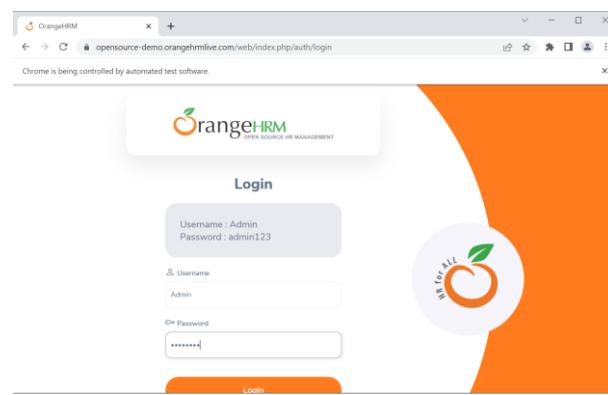
Gambar 1.8 Tampilan browser

- Tunggu hingga proses selesai berjalan

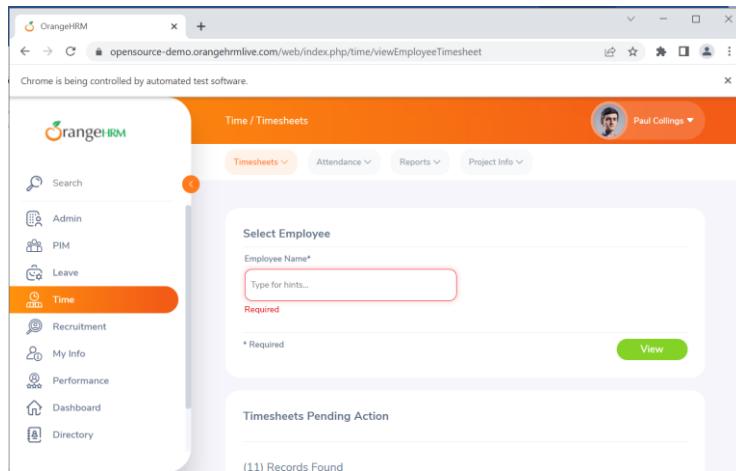


Gambar 1.9 Tampilan proses berjalan

- Tampilan Test Suite setelah dijalankan



Gambar 1.10 Tampilan website orangehrmlive



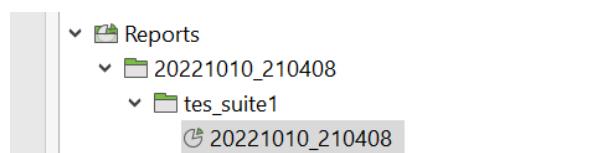
Gambar 1.11 Tampilan hasil test suite

- Berikut adalah detail dari Test Suite yang telah dijalankan

Item	Object	Input
✗ 1 - Open Browser		""
✗ 2 - Navigate To Url		"https://opensource-demo.orangehrmlive.com/web/index.php/auth/login"
✗ 3 - Set Text	input_Username_username	"Admin"
✗ 4 - Set Encrypted Text	input_Password_password	"hUKwJTbofgPU9eVlw/CnDQ=="
✗ 5 - Click	button_Login	
✗ 6 - Click	a_Time	
✗ 7 - Click	input	
✗ 8 - Click	button_View	
✗ 9 - Click	input	
✗ 10 - Click	button_View	
✗ 11 - Click	button_View	
✗ 12 - Double Click	button_View	
✗ 13 - Verify Element Present	span_Required	0

Gambar 1.12 Tampilan detail test suite

- Setelah menjalankan Test Suite, pada folder Reports akan mendapatkan data berupa rekapan data dari tes yang telah dilakukan



Gambar 1.13 Tampilan hasil test suite di folder

- Isi laporan Test Suite

Test Cases Table	
<input checked="" type="checkbox"/>	Passed
<input checked="" type="checkbox"/> Failed	
<input checked="" type="checkbox"/>	Error
<input checked="" type="checkbox"/>	Incomplete
<input checked="" type="checkbox"/>	Skipped
Search here...	
No.	Name
1	sample test_case (22.270s)
Resolution	

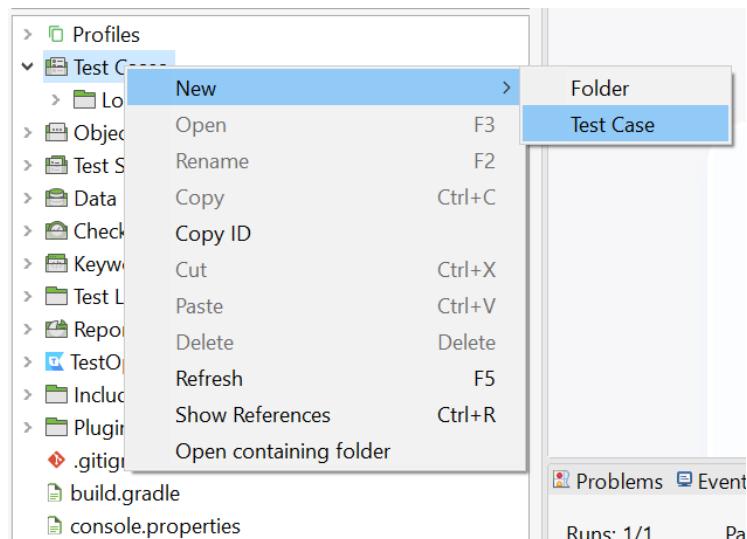
Gambar 1.14 Tampilan hasil laporan test suite

- Pada bagian bawah dari daftar laporan, terdapat detail lengkap dari tes yang dilakukan.

Summary		Execution Settings		Execution Environment	
Test Suite ID	Test Suites/tes suite1				
Host name	ASUS - DESKTOP-DMTQ8BF			Local OS	Windows 10 64bit
Katalon version	8.5.0.208			Platform	Chrome 105.0.0.0
Start	2022-10-10 21:04:33			End	2022-10-10 21:04:56
Elapsed	22.868s				
Total TC	1				
Passed	1			Failed	0
Error	0			Skip	0
Incomplete	0				

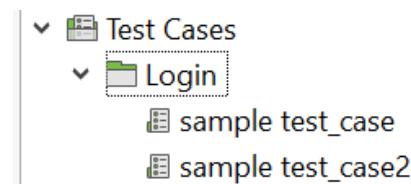
Gambar 1.15 Tampilan daftar laporan

- Agar lebih memudahkan dalam menambahkan Test Case pada Test Suite, maka lebih baik menempatkan file Test Case ke dalam folder tertentu menurut proses yang dilakukan, caranya adalah klik kanan pada Test Cases >New>Folder



Gambar 1.16 Tampilan menambahkan test case pada test suite

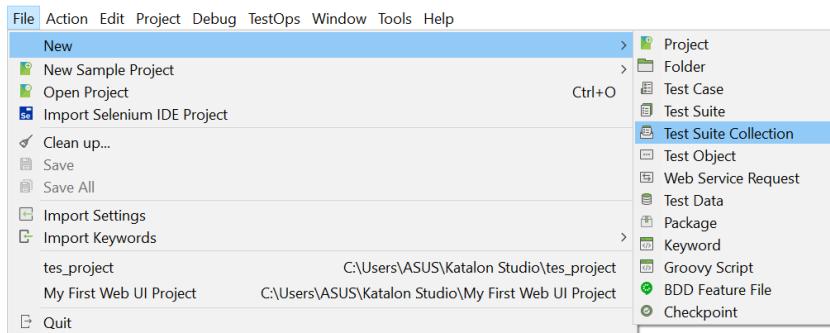
- Kemudian pindahkan file sample test ke dalam folder login dan inilah hasil setelah membuat folder Login.



Gambar 1.17 Tampilan file sample test ke dalam folder login

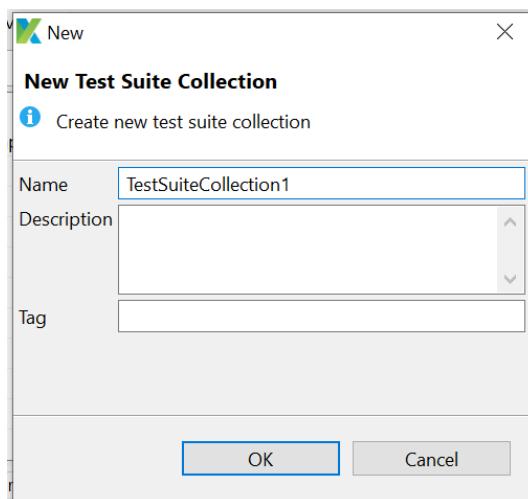
B. Berikut ini adalah langkah-langkah untuk menjalankan Test Suite secara parallel dengan menggunakan Test Suite Collection :

- Untuk membuat Test Suite Collection klik File>New>Test Suite Collection



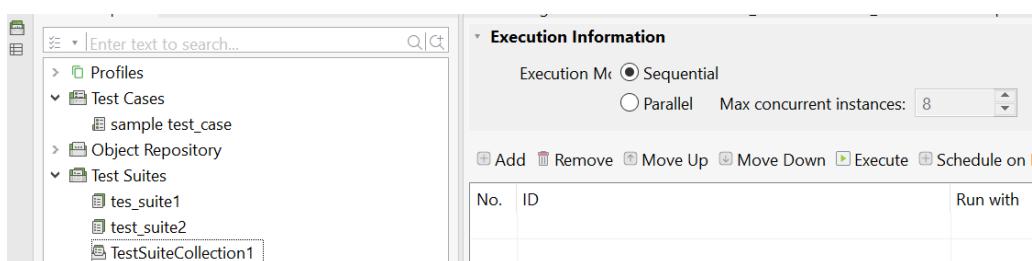
Gambar 1.18 Tampilan test suite collection

- Atur nama dengan Test Suite Collection1 dan OK



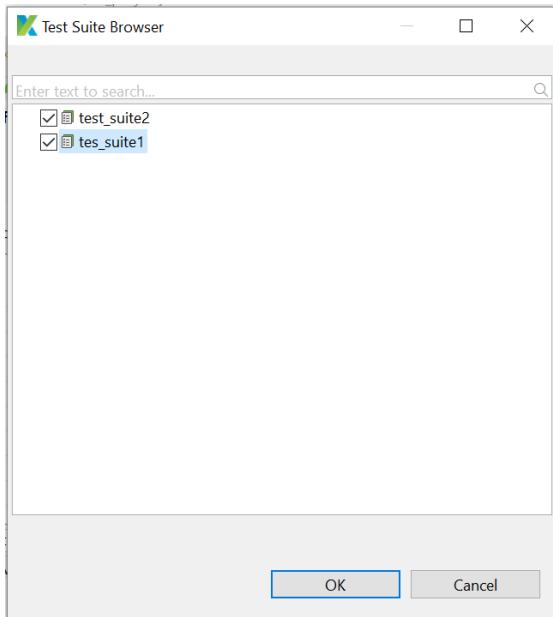
Gambar 1.19 Tampilan create new test suite

- Berikut adalah tampilan dari Test Suite Collection



Gambar 1.20 Tampilan hasil test suite collection

- Tekan tombol Add, kemudian centang semua Test Suite yang telah dibuat sebelumnya



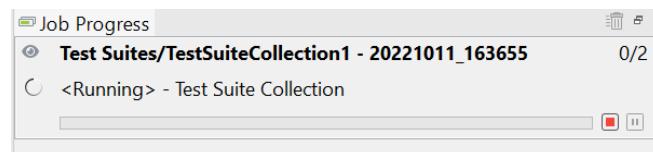
Gambar 1.21 Tampilan test suite collection browser

- Hasil setelah melakukan penambahan Test Suite ke Test Suite Collection. Pada kolom run with dapat diatur menggunakan browser apa saja. Pada bagian Run With ini, web browser yang dipilih boleh berbeda. Maka dari itu, dapat dijalankan dibeberapa browser sekaligus. Pada bagian Execution Information atur Execution Mode ke Parallel.

Execution Information					
<input type="radio"/> Sequential <input checked="" type="radio"/> Parallel Max concurrent instances: 8 Delay between instances (in seconds): 0					
<input type="checkbox"/> Add <input type="checkbox"/> Remove <input type="checkbox"/> Move Up <input type="checkbox"/> Move Down <input type="checkbox"/> Execute <input type="checkbox"/> Schedule on Katalon TestOps					
No.	ID	Run with	Run Configuration	Profile	Run
1	Test Suites/test_suite2	<input checked="" type="radio"/> Firefox	<input checked="" type="radio"/> default	<input type="checkbox"/>	
2	Test Suites/tes_suite1	<input checked="" type="radio"/> Chrome	<input checked="" type="radio"/> default	<input type="checkbox"/>	

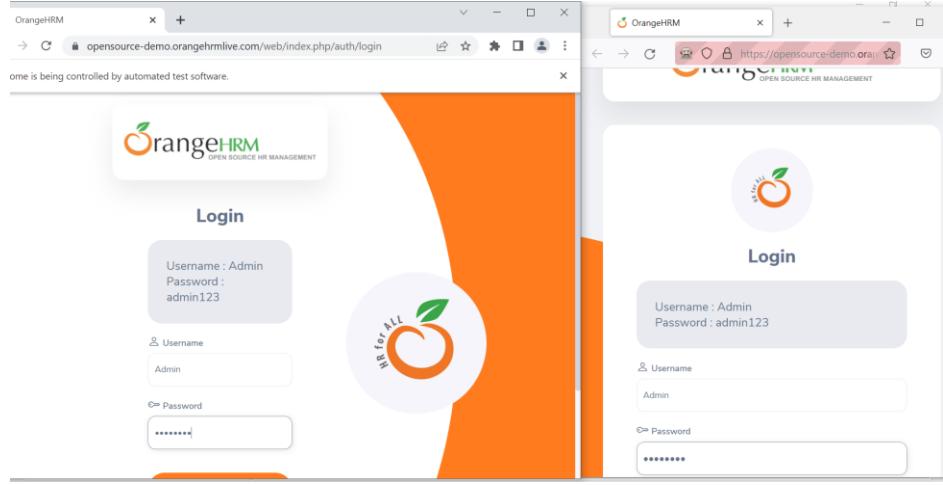
Gambar 1.22 Tampilan hasil setelah test suite collection dirun

- Tunggu proses testing memulai pengujian hingga membuka browser dan menuju link yang telah ditentukan sebelumnya



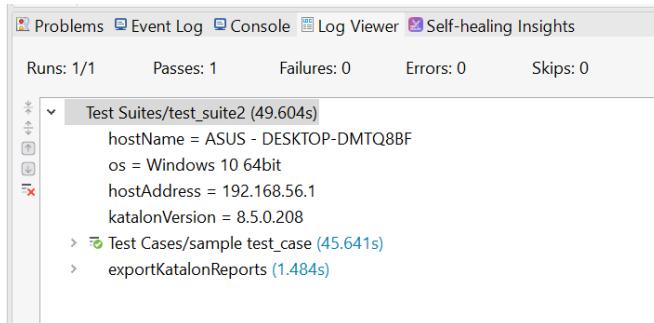
Gambar 1.23 Tampilan proses berjalan

- Berikut adalah cuplikan layar ketika terjadi pengujian secara paralel, yang berarti pengujian dilakukan secara bersamaan (sebelah kiri dengan browser chrome dan sebelah kanan dengan browser firefox).



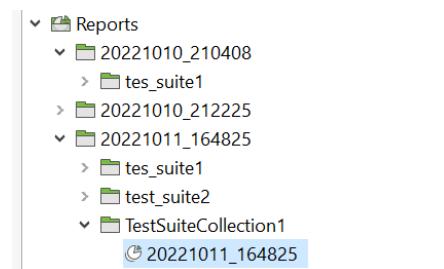
Gambar 1.24 Tampilan hasil browser

- Berikut adalah tampilan isi dari Log Viewer dengan menghasilkan seluruh pengujian berhasil dilakukan tanpa terjadi Failure maupun Error.



Gambar 1.25 Tampilan isi dari Log Viewer

- Setelah melakukan pengujian menggunakan Test Suite Collection, pada bagian Reports akan ada secara otomatis laporan hasil dari pengujian.



Gambar 1.26 Tampilan pengujian menggunakan Test Suite Collection

- Berikut isi dari file laporan yang ada. Pada laporan ini dapat diketahui bahwa pengujian dilakukan terhadap dua Test Suite yaitu test_suite1 dan Test_Suite2 yang dijalankan pada browser chrome dan tidak memiliki Failed/Error. Adapun bagian Show Details berfungsi untuk melihat detail dari tiap-tiap pengujian. Detail yang diberikan berupa Test Case apa saja yang digunakan dan berapa lama waktu yang diperlukan untuk menjalankan Test Case tersebut.

No.	ID	Environm...	Profile	Status	Failed Tests / Total	
1	Test Suites/test_suite2	Firefox	default	Complete	0/1	Show details
2	Test Suites/tes_suite1	Chrome	default	Complete	0/1	Show details

Gambar 1.27 Tampilan isi dari file laporan test suite collection

- Hasil dari menekan show detail dengan browser chrome.

Test Cases Table		Execution Environment	
<input checked="" type="checkbox"/> Passed	<input checked="" type="checkbox"/> Failed	<input checked="" type="checkbox"/> Error	<input checked="" type="checkbox"/> Incomplete
<input checked="" type="checkbox"/> Skipped			
<input type="text" value="Search here..."/> Export report Katalon TestOps Show Test Case Details			
No.	Name	Resolution	Video
> 1	sample test_case (23.589s)		
Summary Execution Settings Execution Environment			
Test Suite ID	Test Suites/tes_suite1		
Host name	ASUS - DESKTOP-DMTQ8BF	Local OS	Windows 10 64bit
Katalon version	8.5.0.208	Platform	Chrome 105.0.0.0
Start	2022-10-11 16:55:07	End	2022-10-11 16:56:12
Elapsed	1m - 4.447s		
Total TC	1		
Passed	1	Failed	0
Error	0	Skip	0
Incomplete	0		

Gambar 1.28 Tampilan show detail dibrowser chrome

- Hasil dari menekan show detail dengan browser firefox.

Test Cases Table		Execution Environment	
<input checked="" type="checkbox"/> Passed	<input checked="" type="checkbox"/> Failed	<input checked="" type="checkbox"/> Error	<input checked="" type="checkbox"/> Incomplete
<input checked="" type="checkbox"/> Skipped			
<input type="text" value="Search here..."/> Export report Katalon TestOps Show Test Case Details			
No.	Name	Resolution	Video
> 1	sample test_case (45.641s)		
Summary Execution Settings Execution Environment			
Test Suite ID	Test Suites/test_suite2		
Host name	ASUS - DESKTOP-DMTQ8BF	Local OS	Windows 10 64bit
Katalon version	8.5.0.208	Platform	Firefox 105.0
Start	2022-10-11 16:54:54	End	2022-10-11 16:55:41
Elapsed	46.344s		
Total TC	1		
Passed	1	Failed	0
Error	0	Skip	0
Incomplete	0		

Gambar 1.29 Tampilan show detail dibrowser firebox

BAB 2

MENGAMBIL DATA CSV

2.1. Tujuan

Pada bagian ini akan membahas tentang cara mengambil data CSV pada katalon studio, Diharapkan pembaca dapat :

3. Mampu menggunakan komponen step CSV file input untuk membaca file teks tipe CSV.
4. Mampu menggunakan komponen *step fixed file* input untuk membaca file teks berisi data *fixed width*.

2.2. Dasar Teori

CSV adalah tipe file yang dapat dibuat atau diedit pada Excel. CSV menyimpan informasi yang dipisahkan oleh koma, bukan dalam kolom. Saat teks dan angka disimpan dalam file CSV, menjadi mudah dipindahkan dari satu program ke program lain.

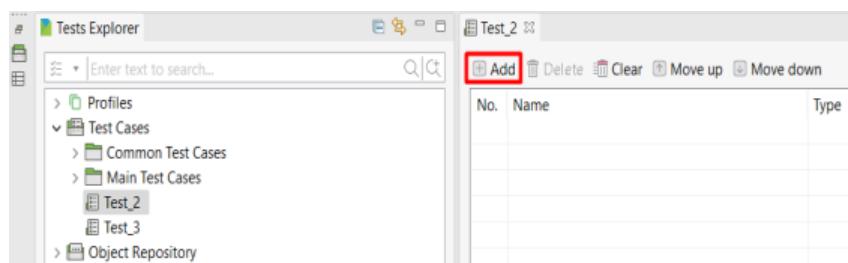
2.3. Percobaan

1. Pertama, buatlah file excel kemudian isikan datanya seperti gambar berikut:

	A	B
1	Username	Password
2	Admin	admin123
3	Admin2	admin1234

Gambar 2.1 File Excel

2. Buka test case yang sudah pernah dibuat pada sebelumnya, kemudian masuk ke tab Variables lalu klik Add.



Gambar 2.2 Add Variabels

3. Selanjutnya create 2 variabel seperti di gambar bawah berikut.

No.	Name	Type	Default value
1	Username	String	""
2	Password	String	""

Gambar 2.3 Hasil Variables

4. Klik pada object username kemudian ubah *Value Type* menjadi *Variable* dan *Value* menjadi *Username*.

Item	Object	Input	Output
-> 1 - Open Browser		"https://opensource-demo.or	
-> 2 - Set Text	input_Username_username	"Admin"	
-> 3 - Set Text	input_Password_password	"admin123"	

Input

No.	Param Name	Param Type	Value Type	Value
1	text	String	Variable	Username

Gambar 2.4 Mengubah Value

5. Lakukan juga pada object password

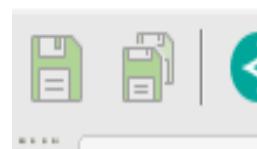
Item	Object	Input	Output
-> 1 - Open Browser		"https://opensource-demo.or	
-> 2 - Set Text	input_Username_username	Username	
-> 3 - Set Text	input_Password_password	"admin123"	

Input

No.	Param Name	Param Type	Value Type	Value
1	text	String	Variable	Password

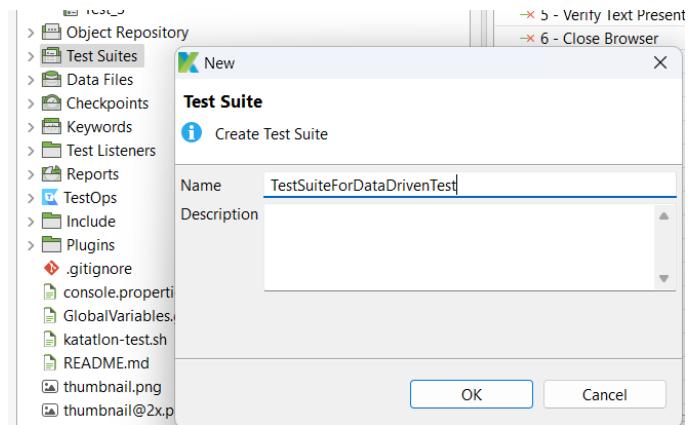
Gambar 2.5 Object Password

6. Kemudian save.



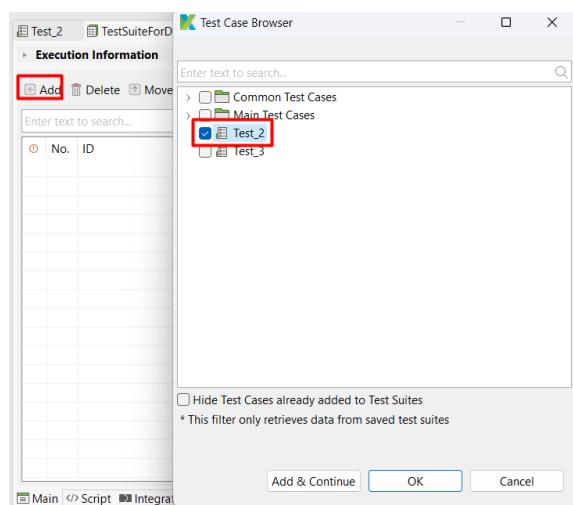
Gambar 2.6 Save

7. Setelah itu Create Test Suite.



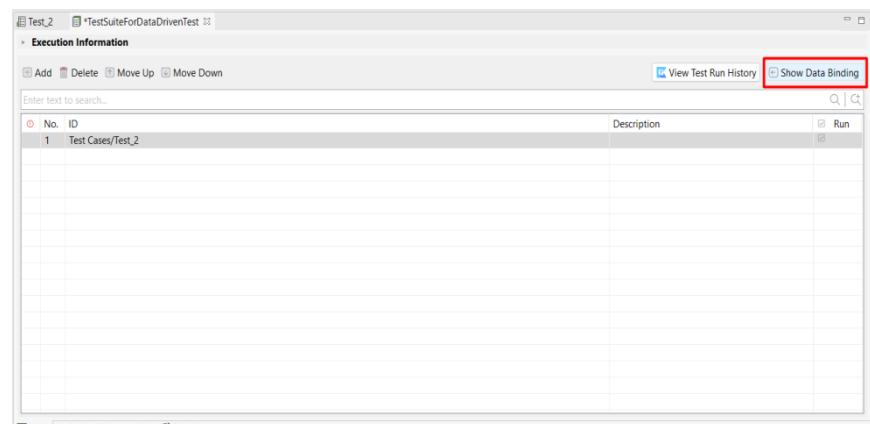
Gambar 2.7 Create New Suite

8. Tambahkan test case yang telah ada sebelumnya.



Gambar 2.8 Add Test 2

9. Kemudian klik *Show Data Binding*.



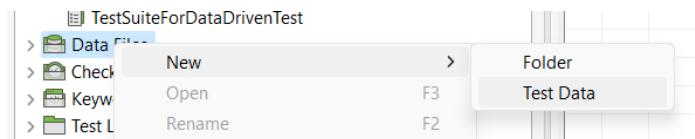
Gambar 2.9 Show Data Binding

10. Pastikan pada *Variable Binding* terdapat variabel username dan password.

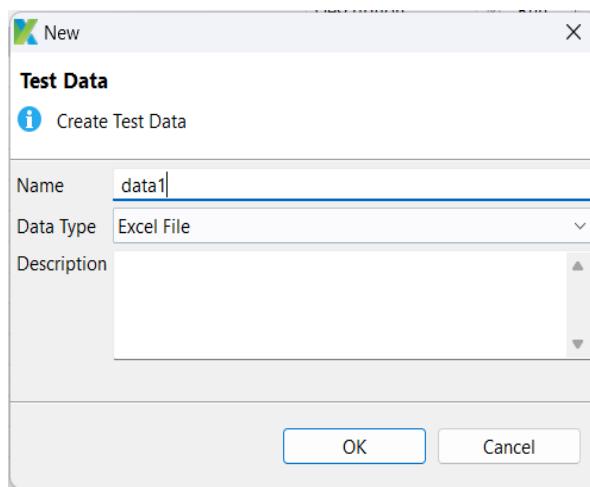
The screenshot shows the 'Variable Binding' configuration window. At the top, there's a section titled 'Select Data Binding option:' with two radio button options: 'Use Variables and Binding at Test Case' (unchecked) and 'Use Variables and Binding at Suite Test Case' (checked). Below this is a 'Test Data' table with columns: No., ID, Data Iteration, and Type. The table is currently empty. At the bottom is a 'Variable Binding' table with columns: No., Name, Default va..., Type, Test Data, and Value. This table contains two entries: '1 Username' with 'Default' type and '2 Password' with 'Default' type.

Gambar 2.10 Variabel Binding

11. Pada Data Files di Tests Explorer, tambahkan data excel yang tadi telah dibuat.

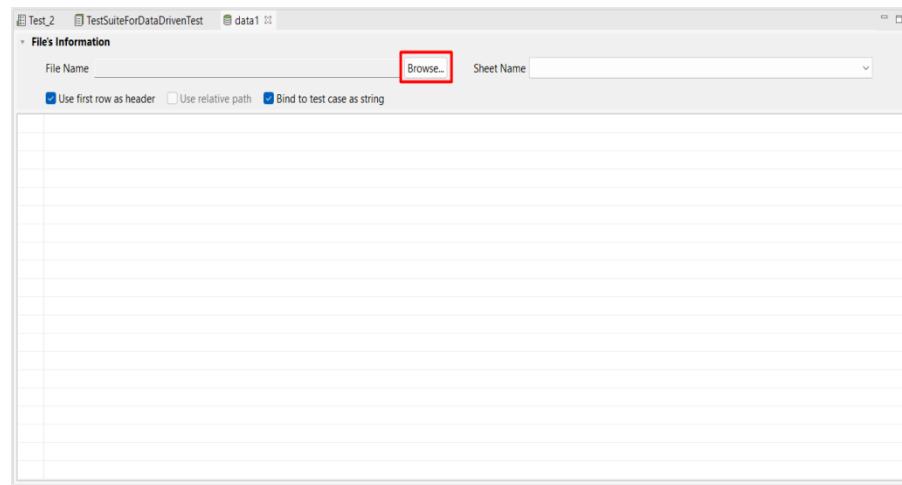


Gambar 2.11 New Test Data



Gambar 2.12 Test Explore

12. Pada datafiles klik *Browse*.



Gambar 2.13 Tahap Browser

13. Kemudian pilih file xlsx yang telah dibuat tadi, lalu save.

No.	Username	Password
1	Admin	admin123
2	Admin2	admin1234

Gambar 2.14 File xlsx

14. Kembali lagi ke Test Suites, pada *Test Data* klik add lalu pilih data file yang tadi telah dibuat

A screenshot of the Test Suite interface. The title bar shows 'Test_2', 'TestSuiteForDataDrivenTest', and 'data1'. Under the 'Execution Information' tab, there is a table with one row: '1 Test Cases/Test_2'. On the right side of the screen, there is a 'Select Data Binding option:' dropdown with two options: 'Use Variables and Binding at Test Case' (radio button) and 'Use Variables and Binding at Suite Test Case' (radio button, checked). Below this is the 'Test Data' section, which contains a table with one row: 'Test Cases/Test_2'. The bottom right corner shows a small preview of the 'data1' file from the previous step.

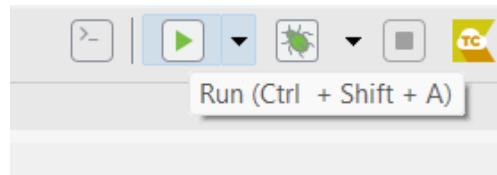
Gambar 2.15 Test Data

15. Pada Variable Binding ubah kedua tipe variabel menjadi *Data Column* kemudian pilih test datanya-nya, dan ubah valuenya sesuai dengan variable masing-masing

Variable Binding						
	No.	Name	Default va...	Type	Test Data	Value
①	1	Username	"	Data Column	1 - Data Files/data1	Username
	2	Password	"	Data Column	1 - Data Files/data1	Password

Gambar 2.16 Proses Variabel Binding

16. Kemudian coba jalankan.



Gambar 2.17 Proses menjalankan

17. Berikut adalah hasil dari pengujian. Apabila mencoba login dengan password yang salah maka akan keluar error.

```

> sampleBeforeTestSuite (0.120s)
> ⚡ Test Cases/Test_2 (204.363s)
  < ✗ Test Cases/Test_2 (165.797s)
    Username = Admin2
    Password = admin1234
    > ⚡ 1 - openBrowser("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login") (121.454s)
      ⚡ 2 - setText(findTestObject("Page_OrangeHRM/input_Username_username"), Username) (8.050s)
      ⚡ 3 - setText(findTestObject("Page_OrangeHRM/input_Password_password"), Password) (0.719s)
      ⚡ 4 - click(findTestObject("Page_OrangeHRM/button_Login")) (0.385s)
      ✗ 5 - verifyTextPresent("Paul Collings", false) (33.026s)
    >   6 - Video (0.001s)
  < ✗ Test Cases/Test_2 (128.521s)
    Username = Admin2
    Password = admin1234
    > ⚡ 1 - openBrowser("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login") (95.944s)
      ⚡ 2 - setText(findTestObject("Page_OrangeHRM/input_Username_username"), Username) (5.840s)
      ⚡ 3 - setText(findTestObject("Page_OrangeHRM/input_Password_password"), Password) (0.724s)
      ⚡ 4 - click(findTestObject("Page_OrangeHRM/button_Login")) (0.444s)
      ✗ 5 - verifyTextPresent("Paul Collings", false) (23.413s)

```

Gambar 2.18 Hasil Pengujian

BAB 3

RESULT AND REPORTING

3.1. Tujuan

1. Mengetahui apa itu test execution reports pada katalon.
2. Mengetahui bagaimana cara melihat hasil test execution pada katalon.
3. Mengetahui bagaimana cara melihat hasil report pengujian dalam format tertentu.
4. Mengetahui bagaimana cara melihat hasil report pengujian pada email.
5. Mengetahui apa itu analytic katalon pada kataon studio.

3.2. Dasar Teori

Test execution report merupakan cara untuk melihat hasil pengujian yang telah dilakukan, dimana hasil pengujian ini akan ditampilkan dalam bentuk laporan.

3.3. Percobaan

3.3.1. Test Execution Reports

Didalam katalon studio terdapat dua fitur yang menyajikan hasil eksekusi dari setiap test yang dijalankan.

- Log : untuk menyajikan hasil eksekusi dari test case, test suite, dan test suite collection.
- Report : untuk menyajikan hasil eksekusi dari test suite dan test suite collection.

A. Log Viewer

1. Kita dapat menggunakan tools log viewer untuk melihat log dari setiap test case.

All	Level	Time	Message
Info	PASSED	2022-10-13 07:58:34.082	Browser is opened with url: ''
Passed	PASSED	2022-10-13 07:58:43.383	Navigate to 'https://opensource-demo.orangehrmlive.com'
Failed	PASSED	2022-10-13 07:58:48.807	Text 'Admin' is set on object 'Object Repository/

Gambar 3.1 Menu log view pada katalon

2. Dalam jendela log viewer kita dapat melihat tes yang dijalankan, tes yang lulus uji, kesalahan, error, dan test yang dilewati.

Runs: 2/2			
Passes: 2			
Failures: 0			
Errors: 0			
Skips: 0			
All	Level	Time	Message
Info	PASSED	2022-10-13 07:58:34.082	Browser is opened with url: "
Passed	PASSED	2022-10-13 07:58:43.383	Navigate to 'https://opensource-demo.orangehrm.com/web/index.php/auth/login'
Failed	PASSED	2022-10-13 07:58:48.807	Text 'Admin' is set on object 'Object Repository/test2/Page_OrangeHRM/input_Username_username'
Error	PASSED	2022-10-13 07:58:49.629	Text ***** has been set on object 'Object Repository/test2/Page_OrangeHRM/input_Password_password'
Warning	PASSED	2022-10-13 07:58:50.208	Object: 'Object Repository/test2/Page_OrangeHRM/button_Login'
PASSED		2022-10-13 07:58:51.004	Test completed successfully

Gambar 3.2 Tes yang dijalankan

3. Jika ingin memfilter log berdasarkan hasil kita dapat menggunakan menu pada jendela log sebelah kiri.

Runs: 2/2			
Passes: 2			
Failures: 0			
All	Level	Time	
Info	START	2022-10-13 07:58:54.106	
Passed	PASSED	2022-10-13 07:59:00.118	
Failed	END	2022-10-13 07:59:00.120	
Error	START	2022-10-13 07:59:00.120	
Warning	PASSED	2022-10-13 07:59:03.628	
Not Run	END	2022-10-13 07:59:03.630	
	START	2022-10-13 07:59:03.630	
	PASSED	2022-10-13 07:59:04.303	
	END	2022-10-13 07:59:04.305	
	START	2022-10-13 07:59:04.305	

Gambar 3.3 Menu filter log

4. Katalon juga menyediakan tampilan hirarki yang didalamnya terdapat jendela log dan jendela detail dari setiap tahapan pengujian

Katalon Studio - 8.5.0-43c37de - git-katalon.git - [Location: C:\Users\Windows X\Katalon Studio\git-katalon.git]

File Action Edit Project Debug TestOps Window Tools Help

Problems Event Log Console Log Viewer Self-healing Insights

Runs: 2/2 Passes: 2 Failures: 0 Errors: 0 Skips: 0

Test Suites/TestSuiteOne (45.301s)
hostname = WINDOWS X - DESKTOP-JR2AETD
os = Windows 10 64bit
hostAddress = 192.168.150.1
katalonVersion = 8.5.0.208

Test Cases/Second Test Case (27.913s)
1 - openBrowser("") (9.464s)
2 - navigateToUrl("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login") (9.921s)
3 - setText(findTestObject("test2/Page_OrangeHRM/input_Username_username"), "Admin") (4.804s)
4 - setEncryptedText(findTestObject("test2/Page_OrangeHRM/input_Password_password"), "nUKwTbofgf")
5 - click(findTestObject("test2/Page_OrangeHRM/button_Login")) (0.581s)
6 - closeBrowser() (0.870s)

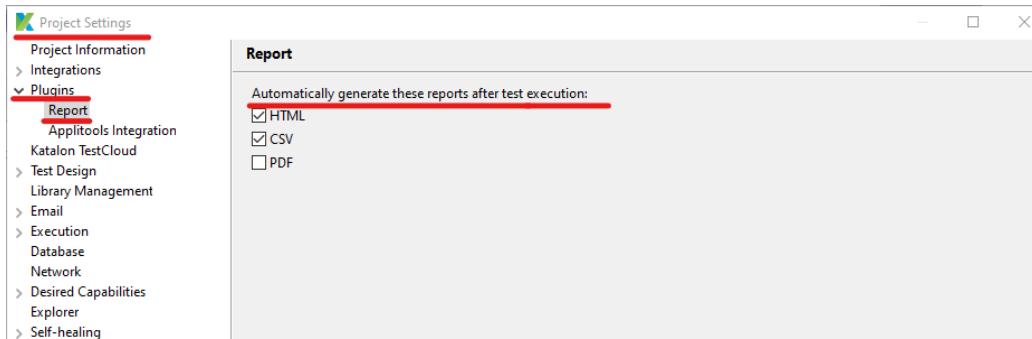
Test Cases/Third Test Case (14.455s)
exportKatalonReports (1.666s)

10-13-2022 07:58:44 AM setText(findTestObject("test2/Page_OrangeHRM/input_Username_username"), "Admin")
Elapsed time: 4.804s
Text 'Admin' is set on object 'Object Repository/test2/Page_OrangeHRM/input_Username_username'

Gambar 3.4 Tampilan hirarki log

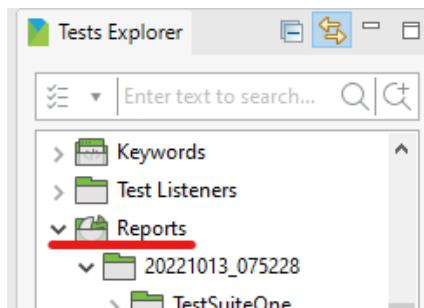
B. Report

1. Dalam katalon studio kita dapat mengatur format report yang digunakan untuk menyajikan report melalui menu project > setting > plugins > report Ada tiga jenis format report yang disajikan yaitu html, csv, dan pdf.



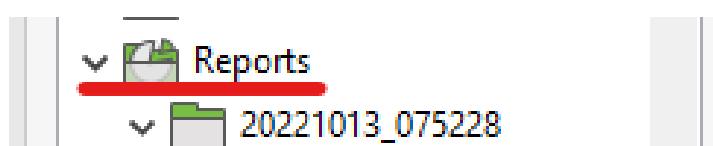
Gambar 3.5 Pengaturan format report

2. Untuk melihat report kita dapat menuju jendela explorer dan memilih menu report.



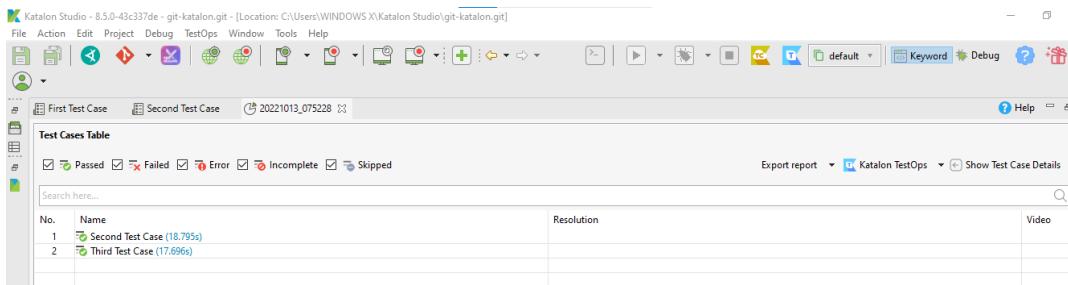
Gambar 3.6 Pilih menu report

3. Report yang disajikan sesuai dengan jumlah eksekusi yang dilakukan terhadap test suite ataupun test suite collection. Penamaan report secara default diawali oleh tahun, bulan, dan tanggal atau sesuai dengan waktu eksekusi



Gambar 3.7 Report yang sudah dipilih

4. Berikut ini adalah tampilan jendela report secara keseluruhan.



Gambar 3.8 Tampilan jendela report

5. Terdapat test case table yang akan menampilkan test case yang diuji beserta dengan status kelulusan, kita juga dapat memfilter tampilan dengan menggunakan checkbox di bagian atas jendela test case table.

Test Cases Table	
<input checked="" type="checkbox"/> Passed <input checked="" type="checkbox"/> Failed <input checked="" type="checkbox"/> Error <input checked="" type="checkbox"/> Incomplete <input checked="" type="checkbox"/> Skipped	
Search here...	
No.	Name
1	Second Test Case (18.795s)
2	Third Test Case (17.696s)

Gambar 3.9 Tampilan test case table

6. Di bagian bawah jendela report juga terdapat ringkasan pengujian yang terletak pada tab summary.

Summary		Execution Settings		Execution Environment	
Test Suite ID	Test Suites/TestSuiteOne				
Host name	WINDOWS X - DESKTOP-JR2AETD			Local OS	Windows 10 64bit
Katalon version	8.5.0.208			Platform	Chrome 106.0.0.0
Start	2022-10-13 07:52:59			End	2022-10-13 07:53:36
Elapsed	37.152s				
Total TC	2				
Passed	2			Failed	0
Error	0			Skip	0
Incomplete	0				

Gambar 3.10 Tampilan tab summary

7. Kita juga dapat melihat seluruh setting saat eksekusi melalui tab Execution setting

Name	Value
autoApplyNeighborXpaths	false
ignorePageLoadTimeoutException	false
timeCapsuleEnabled	false
executionProfile	default
excludeKeywords	[verifyElementPresent, verifyElementNotPresent]
xpathPriority	[[left=x:path@attributes, right=true], [left=xpathIdRelative, right=true], [left= dom:name, right=true], [left=xpath:link, ...]]
timeout	30.0
actionDelay	0.0
methodsPriorityOrder	[[left=XPATH, right=true], [left=BASIC, right=true], [left=CSS, right=true], [left=IMAGE, right=true]]
proxy	("proxyOption": "NO_PROXY", "proxyServerType": "HTTP", "username": "", "password": "", "proxyServerAddress": "")
defaultFailureHandling	CONTINUE_ON_FAILURE
terminateDriverAfterTestCase	false
defaultPageLoadTimeout	30.0
report	{videoRecorderOption={enable=false, useBrowserRecorder=true, videoFormat=AVI, videoQuality=LOW, recordAllTests}}
enablePageLoadTimeout	false
terminateDriverAfterTestSuite	true
useActionDelayInSecond	SECONDS
testDataInfo	()
selfHealingEnabled	true
WebUI	()

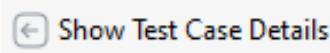
Gambar 3.11 Tampilan tab execution setting

8. Kita juga dapat melihat lingkungan pengujian melalui tab Execution Environtment

Name	Value
hostName	WINDOWS X - DESKTOP-JR2AETD
os	Windows 10 64bit
katalonVersion	8.5.0.208
browser	Chrome 106.0.0.0
hostAddress	192.168.150.1
sessionId	7Sec0030e41df4c6017f904d89cbdb8
seleniumVersion	3.141.59
proxyInformation	ProxyInformation { proxyOption=NO_PROXY, proxyServerType=HTTP, username=, password=***** , proxyServerAd...}
platform	Windows 10

Gambar 3.12 Tampilan tab execution environtment

9. Untuk melihat tampilan detail dari pengujian kita dapat menggunakan menu test case detail



Gambar 3.13 Menu test case detail

10. Berikut adalah tampilan dari detail setiap pengujian, yang didalamnya terdapat tahapan pengujian, deskripsi, dan waktu pengujian.

Test Case's Log		
Test Log		
<input checked="" type="checkbox"/> Info	<input checked="" type="checkbox"/> Passed	<input checked="" type="checkbox"/> Failed
<input type="checkbox"/> Error	<input checked="" type="checkbox"/> Incomplete	<input checked="" type="checkbox"/> Warning
<input type="checkbox"/> Not Run		
Search here...		
Item	Description	Elapsed
> 1. openBrowser("")		8.137s
> 2. navigateToUrl("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")		5.621s
> 3. setText(findTestObject("test2/Page_OrangeHRM/input_Username_username"), "Admin")		1.762s
> 4. setEncryptedText(findTestObject("test2/Page_OrangeHRM/input_Password_password"), "hUKwJTbogPU9eVlw/CnDQ==")		0.677s
> 5. click(findTestObject("test2/Page_OrangeHRM/button_Login"))		0.495s
> 6. closeBrowser()		0.894s

Gambar 3.14 Test case dialog

3.3.2. Basic Reports Plugins

- Pada tahapan sebelumnya kita sudah banyak membuat test case, test suit, maupun test suit collection pada katalon. Salah satunya test case seperti yang terlihat pada gambar di bawah ini.

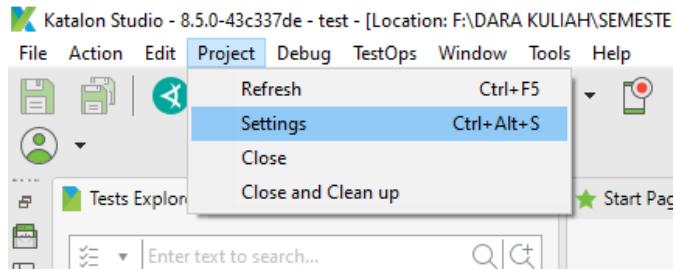
TestCase_1			
Add	Recent keywords	Delete	Move up
Item	Object	Input	Output
→ 1 - Open Browser		""	
→ 2 - Navigate To Url		"https://opensource-demo.orangehrmlive.com/web/index.php/auth/login"	
→ 3 - Set Text	input_Username_username	"Admin"	
→ 4 - Set Encrypted Text	input_Password_password	"hUKwJTbogPU9eVlw/CnDQ=="	
→ 5 - Click	button_Login		
→ 6 - Click	button_Search		
→ 7 - Click	a_Time		
→ 8 - Verify Element Present	a_Time	0	
→ 9 - Close Browser			

Gambar 3.15 Tampilan test case suite

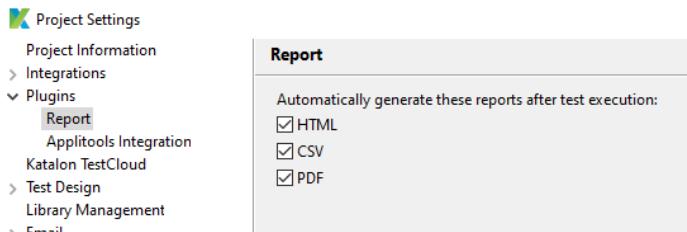
Passed		Failed		Error		Incomplete		Skipped		Export						
Search here...																
No.	Name															
1	Test1 (34.378s)															
2	TestCase_2 (24.477s)															

Gambar 3.16 Detail test case suite

- Sebelum kita mulai mengekspor laporan hasil pengujian dalam format HTML, CSV, atau PDF, terlebih dahulu kita melakukan setting pada menu projek, dimana untuk semua plugins report ceklis untuk semua format pilihan yang ada.

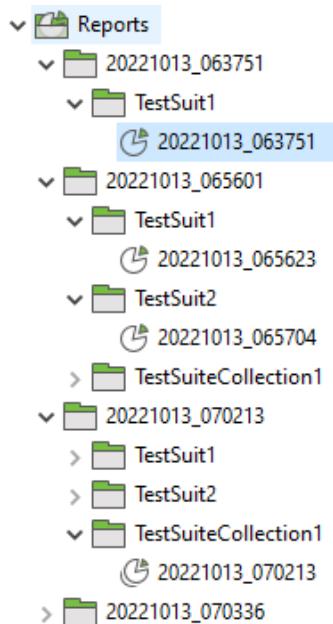


Gambar 3.17 Tampilan menu project



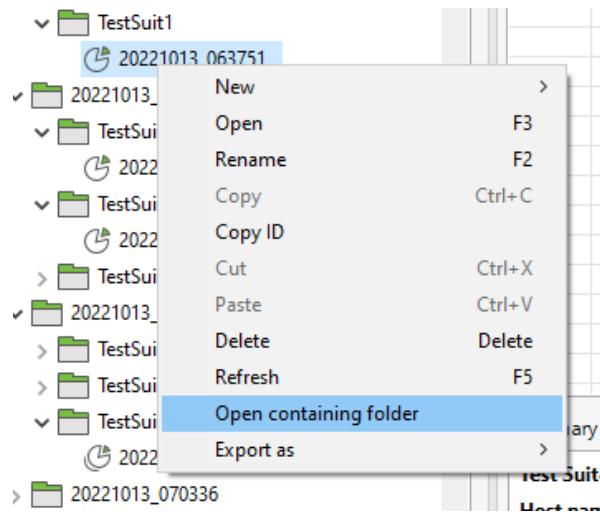
Gambar 3.18 Menu pada project setting

3. Setelah itu pada menu reports, dapat kita lihat berbagai pengujian yang telah kita buat. Untuk kasus ini kita melakukan export terhadap pengujian test case yang sudah dilakukan penambahan pada test suit1.



Gambar 3.19 List pengujian yang telah dibuat

4. Untuk melihat hasil report dalam format HTML, klik kanan pada report kemudian pilih **open containing folder**.



Gambar 3.20 *Open containing folder*

5. Selanjutnya user akan di arahkan ke halaman lokasi tempat penyimpanan folder.

is PC > MASTER (F:) > DARA KULIAH > SEMESTER 7 > PL > TUGAS > PRAKTIKUM > Healthcare Sample > Reports			
Name	Date modified	Type	Size
20221013_063751	10/13/2022 6:39 AM	File folder	

Gambar 3.21 *Lokasi folder disimpan*

6. Ketika folder dibuka, user dapat melihat hasil report pengujian sudah ada dalam format HTML dan CSV, ini dapat terjadi karena sebelumnya kita telah melakukan setting terhadap report plugin untuk mengaktifkan ceklis terhadap berbagai pilihan format plugin yang tersedia.

20221013_063751.csv	10/13/2022 6:39 AM	Microsoft Office E...	3 KB
20221013_063751.html	10/13/2022 6:39 AM	Chrome HTML Do...	202 KB

Gambar 3.22 *Hasil pengujian*

7. Berikut report hasil pengujian dalam format HTML dan CSV.

TestSuit1 Test Log

Execution Environment

Host name: ACER - DARA-MELISA-PC
 Local OS: Windows 10 64bit
 Katalon version: 8.5.0.208
 Browser: Chrome 106.0.0.0

Test Execution Log

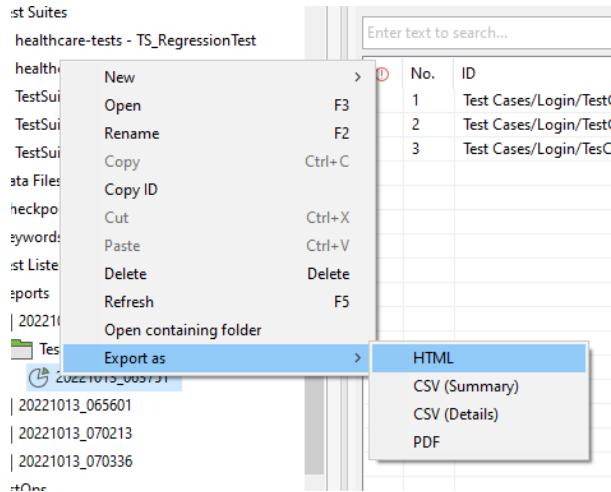
- TEST SUITE: TestSuit1
 - Full Name: TestSuit1
 - Start / End / Elapsed: 2022-10-13 06:38:26.583 / 2022-10-13 06:39:26.409 / 00:00:59.826
 - Status: 2 test total, 2 passed, 0 failed, 0 error, 0 incomplete, 0 skipped
- TEST CASE: Test Cases/Test1
 - Full Name: TestSuit1/Test Cases/Test1
 - Start / End / Elapsed: 2022-10-13 06:38:27.534 / 2022-10-13 06:39:01.912 / 00:00:34.378
 - Status: PASSED
 - TEST STEP: openBrowser("")
 - TEST STEP: navigateToUrl("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")
 - TEST STEP: setText(findTestObject("Object Repository/Test2/Page_OrangeHRM/input_Username_username"), "Admin")
 - TEST STEP: setEncryptedText(findTestObject("Object Repository/Test2/Page_OrangeHRM/input_Password_password"), "hUKwJTbofgPU9eVlw/CnDQ==")
 - TEST STEP: click(findTestObject("Object Repository/Test2/Page_OrangeHRM/button_Login"))
 - TEST STEP: click(findTestObject("Object Repository/Test2/Page_OrangeHRM/button_Search"))
 - TEST STEP: click(findTestObject("Object Repository/Test2/Page_OrangeHRM/a_Time"))
 - TEST STEP: verifyElementPresent(findTestObject("Object Repository/Test2/Page_OrangeHRM/a_Time"), 0)
 - TEST STEP: closeBrowser()

Gambar 3.23 Laporan pengujian dalam format HTML

Suite/Test/Step Name										
A	B	C	D	E	F	G	H	I	J	K
1 Suite/Test	Browser	Description	Tag	Start time	End time	Duration	Status			
2 TestSuit1	Chrome 106.0.0.0			#####	#####	59.826s	PASSED			
3										
4 Test Case	Chrome 106.0.0.0			#####	#####	34.378s	PASSED			
5 openBrow	Chrome 106.0.0.0			#####	#####	10.253s	PASSED			
6 navigateT	Chrome 106.0.0.0			#####	#####	6.128s	PASSED			
7 setText(f	Chrome 106.0.0.0			#####	#####	2.376s	PASSED			
8 setEncryp	Chrome 106.0.0.0			#####	#####	0.860s	PASSED			
9 click(find1	Chrome 106.0.0.0			#####	#####	0.730s	PASSED			
10 click(find1	Chrome 106.0.0.0			#####	#####	5.336s	PASSED			
11 click(find1	Chrome 106.0.0.0			#####	#####	3.779s	PASSED			
12 verifyElen	Chrome 106.0.0.0			#####	#####	0.434s	PASSED			
13 closeBrow	Chrome 106.0.0.0			#####	#####	0.947s	PASSED			
14 Video	Chrome 106.0.0.0			#####	#####	0.003s	PASSED			
15										
16 Test Case	Chrome 106.0.0.0			#####	#####	24.477s	PASSED			
17 openBrow	Chrome 106.0.0.0			#####	#####	8.099s	PASSED			
18 setText(f	Chrome 106.0.0.0			#####	#####	2.614s	PASSED			
19 setText(f	Chrome 106.0.0.0			#####	#####	0.984s	PASSED			
20 click(find1	Chrome 106.0.0.0			#####	#####	0.517s	PASSED			
21 click(find1	Chrome 106.0.0.0			#####	#####	5.388s	PASSED			
22 click(find1	Chrome 106.0.0.0			#####	#####	2.760s	PASSED			
23 closeBrow	Chrome 106.0.0.0			#####	#####	0.940s	PASSED			
24 Video	Chrome 106.0.0.0			#####	#####	0.003s	PASSED			
25										

Gambar 3.24 Laporan pengujian dalam format CSV

8. Untuk melihat report hasil pengujian dalam format PDF, anda bisa klik kanan pada report, kemudian pilih **exports as**, lalu pilih format pdf.



Gambar 3.25 Laporan di ekspor dalam format PDF

9. Pilih tempat penyimpanan file, kemudian save. Berikut reprt hasil pengujian dalam format PDF.

#	ID	Description	Status
1	Test Cases/Test1		PASSED
2	Test Cases/TestCase_2		PASSED

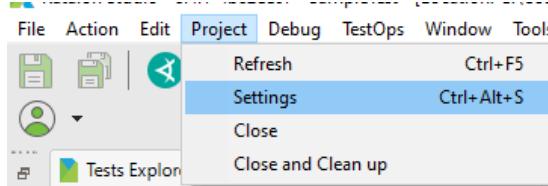
Gambar 3.26 Pilih lokasi penyimpanan file

3.3.3. How To Email Results

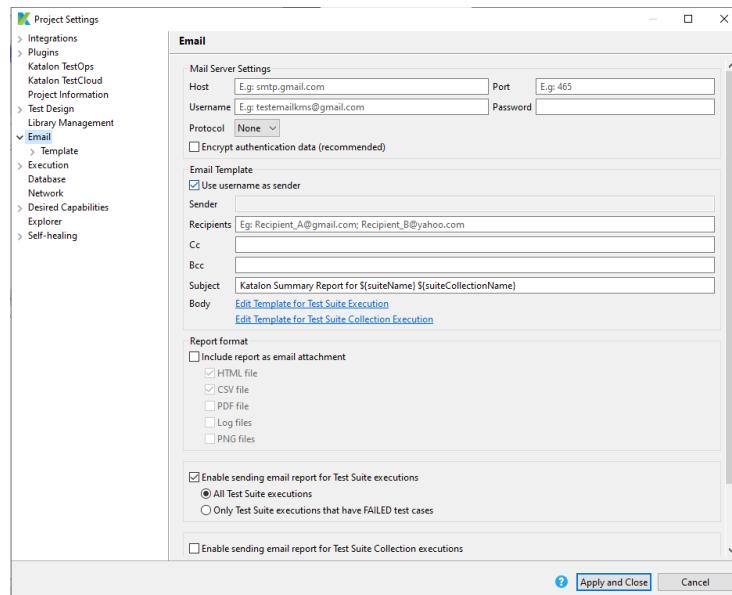
Setelah melakukan pengujian perangkat lunak, terkadang seorang dari tim ingin hasil pengujian tersebut dikirim ke email secara otomatis agar menghemat

waktu dalam kolaborasi sesama tim. Adapun langkah-langkah yang diperlukan untuk mengirim hasil pengujian ke email sebagai berikut :

1. Bukan bagian Project>Settings>Email



Gambar 3.27 Pilih menu project setting pada toolbar



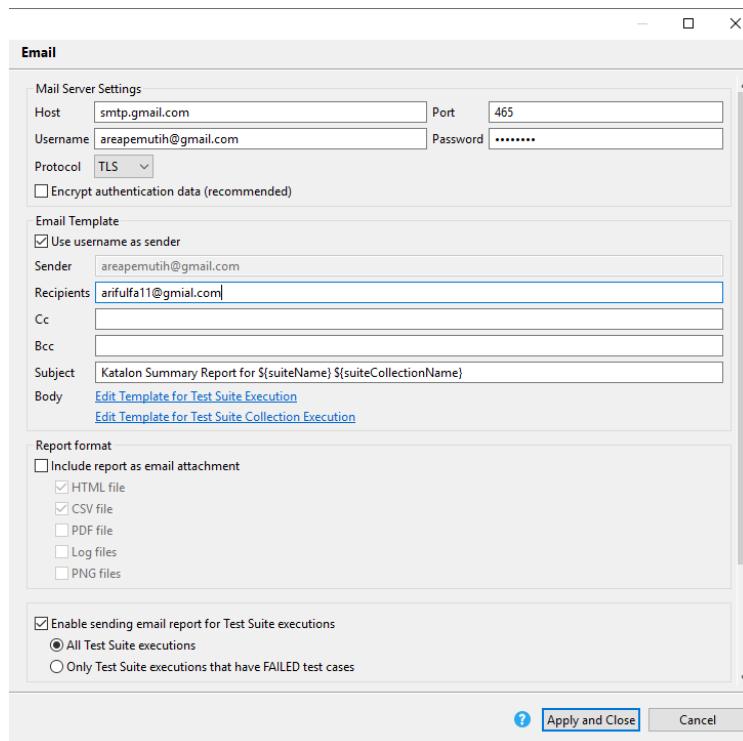
Gambar 3.28 Pilih template email

2. Sebelum mengisi kolom email, pahami dulu yang dimaksud dengan *Mail Server Settings*

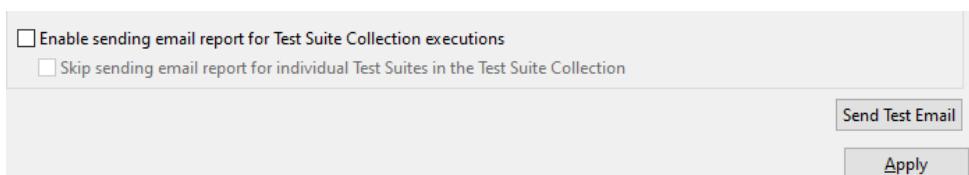
Email sever	Host	Port	Reference
Gmail	smtp.gmail.com	465 or 587	Check Gmail through other email platforms
Outlook	smtp.office365.com	587 or 25	How to set up a multifunction device or application to send email using Microsoft 365 or Office 365
Yahoo! Mail	smtp.mail.yahoo.com	465	POP access settings and instructions for Yahoo Mail

Gambar 3.29 Tampilan mail server setting

3. Setelah memahami isi dari *Mail Server Settings*. Selanjutnya isikan pada *email* seperti berikut ini, sesuaikan dengan email anda. Kemudian tekan *Send Test Email*, *email* akan dikirim kepada *email* yang terdapat pada kolom *Recipient*.



Gambar 3.30 Pengisian Email pada mail server setting

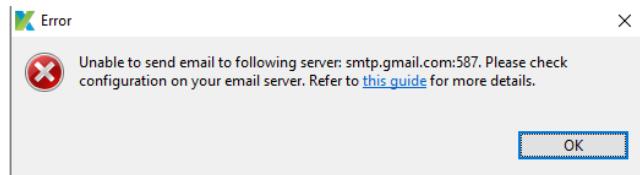


Gambar 3.31 Pilih apply

4. Tunggu proses berjalan, dan jika terjadi *error*, ikuti langkah selanjutnya

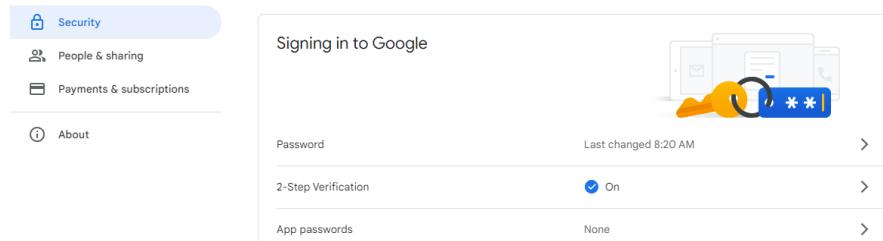


Gambar 3.32 Tampilan saat proses berjalan



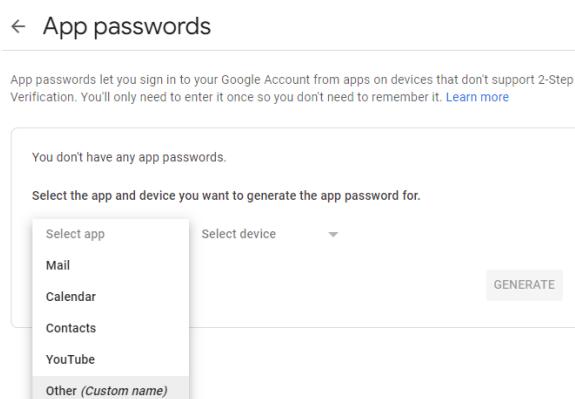
Gambar 3.33 Tampilan proses gagal (error)

5. Adapun *error* terjadi dikarenakan kita menggunakan *protocol smtp* untuk mengirim *email*, dan *protocol* ini adalah sebuah *protocol* pihak ketiga. Oleh karena itu, untuk membuat *email* dapat terkirim dengan berhasil, lakukan langkah berikut untuk memberikan akses *email* kepada pihak ketiga dengan cara pilih akun gmail anda, kemudian pilih *Manage your Google Account*
6. Masuk ke bagian menu *Security*, kemudian scroll pada bagian *Signin in to Google*, pada bagian tersebut aktifkan *2-Step Verification* dengan cara menginput nomor *handphone* yang *valid*, setelah berhasil, maka akan terdapat menu *App passwords*, buka menu tersebut



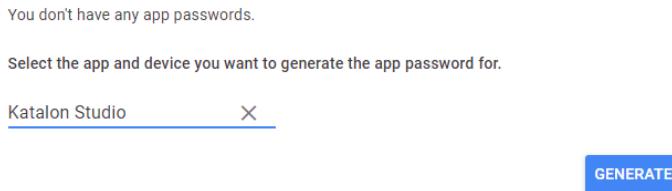
Gambar 3.34 Tampilan menu security

7. Pada bagian *Select App*, pilih *Other (Custom name)*



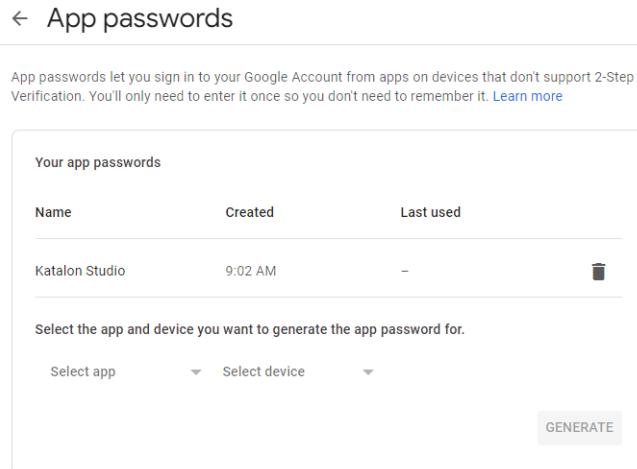
Gambar 3.35 Tampilan jendela app password

8. Isikan dengan Katalon Studio atau nama yang anda inginkan. Kemudian klik *Generate*



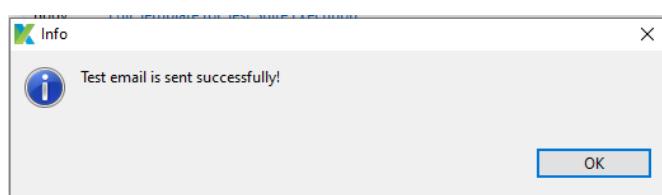
Gambar 3.36 Pilih generate

9. Kemudian anda akan diberikan *password* otomatis, kemudian klik *DONE*
10. Secara otomatis muncul daftar *App Password*



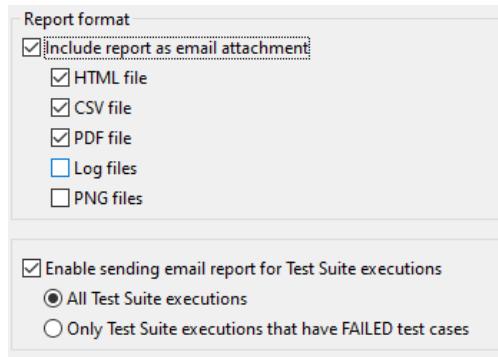
Gambar 3.37 Tampilan halaman App password

11. *Copy-kan password* sebelumnya, dan simpan pada bagian kolom *Password*, kemudian lakukan *Send Test Email* ulang.
Jika gagal menggunakan *port* dan *protocol* di atas, coba ganti ke *port* dan *protocol* seperti gambar di bawah.
12. Maka *email* akan berhasil dikirimkan. Jika gagal, ganti *Port* yang lainnya



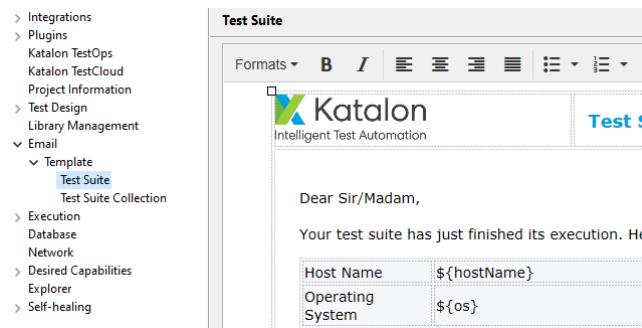
Gambar 3.38 Proses berhasil

13. Jika tidak terdapat pada Kotak Masuk, kemungkinan akan dianggap sebagai *spam*
14. Langkah berikutnya, atur pada bagian *Report format* dengan berikut, dan klik *Apply* setelah melakukan konfigurasi



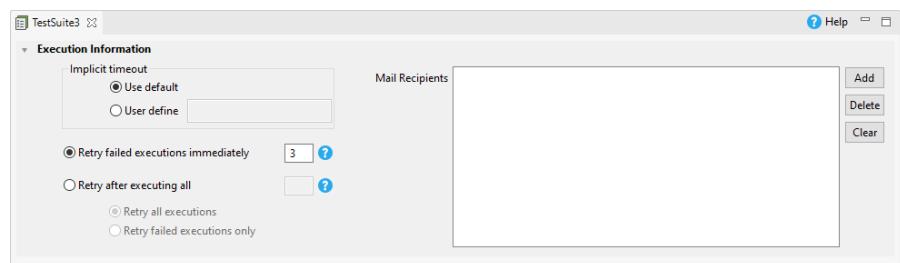
Gambar 3.39 Tampilan report format

15. Adapun untuk mengubah *template* dari *email*, anda dapat melakukannya pada bagian Email>Template>Test Suite atau Test Suite Collection.



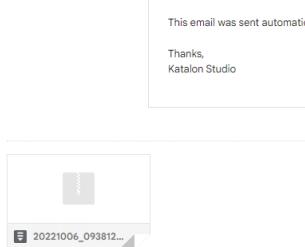
Gambar 3.40 Mengubah template test Suite

16. Setelah melakukan konfigurasi *email*, selanjutnya buka file *Test Suite* anda, lakukan *expand* terhadap *Execution Information*. Pada jendela tersebut terdapat informasi mengenai *Mail Recipients*. Masukkan daftar nama *email* yang akan menerima pemberitahuan mengenai tes yang dilakukan



Gambar 3.41 Halaman Execution Infromation

Jika sebelumnya anda mencentang bagian *include report as email attachment*, maka pada notifikasi *email* anda akan mendapat file berupa seperti .csv, .html, .pdf, dan lain sebagainya.



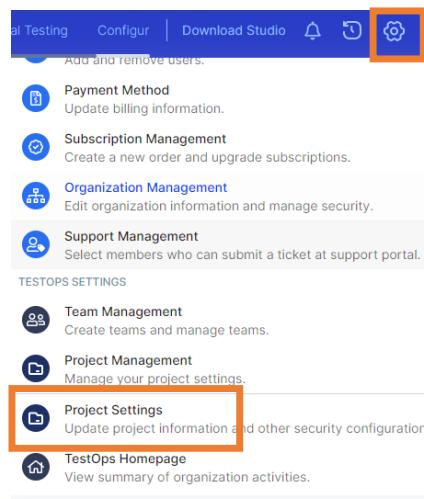
Gambar 3.42 Notifikasi file PDF masuk pada email

Setelah mempelajari modul ini, mahasiswa diharapkan sudah dapat melakukan pengujian beserta mengirim *email* hasil pengujian dengan berhasil.

3.3.4. How To Use Katalon Analytics Step by Step

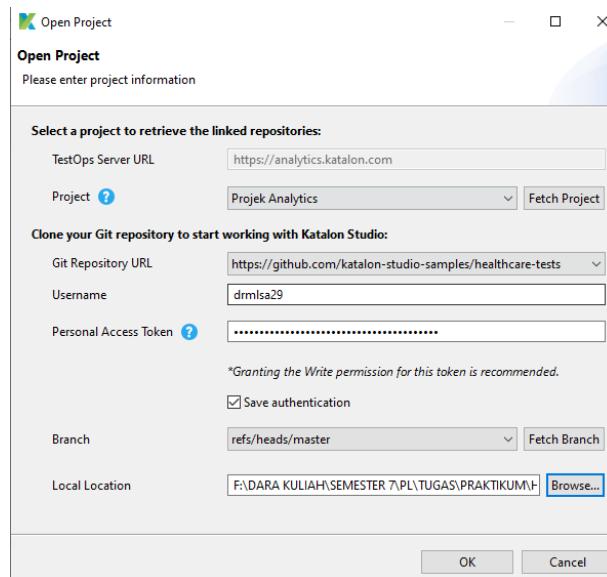
Pada modul ini kita akan mempelajari bagaimana cara menggunakan katalon analytics. Katalon analytic merupakan suatu proses yang berfungsi untuk melihat, menyimpan, dan menganalisis hasil atau laporan dari pengujian yang dilakukan pada katalon. Berikut ini merupakan tahapan menggunakan katalon analytics.

1. Langkah pertama yang kita lakukan adalah, kita bisa membuat projek baru untuk katalon. Projek baru dapat dibuat dengan masuk ke web katalon, kemudian klik setting, lalu pilih new projek, dan projek baru pun telah dibuat.



Gambar 3.43 Pilihan untuk memulai project baru

2. Selanjutnya, buka aplikasi katalon kemudian klik file lalu open projek. Masukkan username dan token dari akun github anda, lalu pada bagian projek pilih nama projek yang sudah dibuat pada web katalon analytics sebelumnya, pada kasus ini saya membuat projek dengan nama projek analytics. Pada proses ini kita sudah berhasil menghubungkan langsung test projek yang kita lakukan di aplikasi katalon ke web katalon analytics.

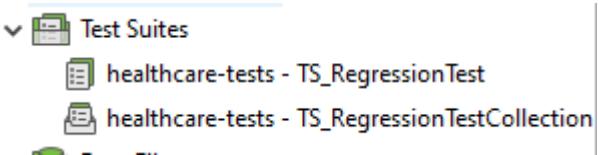


Gambar 3.44 Open project

3. Jika kita lihat pada halaman test activities pada web katalon analytics, belum ada test activities apapun yang terekam karena memang belum ada pengujian yang dilakukan.

Gambar 3.45 Halaman test activities

4. Selanjutnya kita akan menjalankan test suite sebagai bentuk percobaannya. Test suit ini merupakan test case yang sudah disediakan oleh katalon sebagai sampel pengujian. Untuk kasus ini saya memilih test case TR_RegressioonTest.



Gambar 3.46 Pilihan test Suite

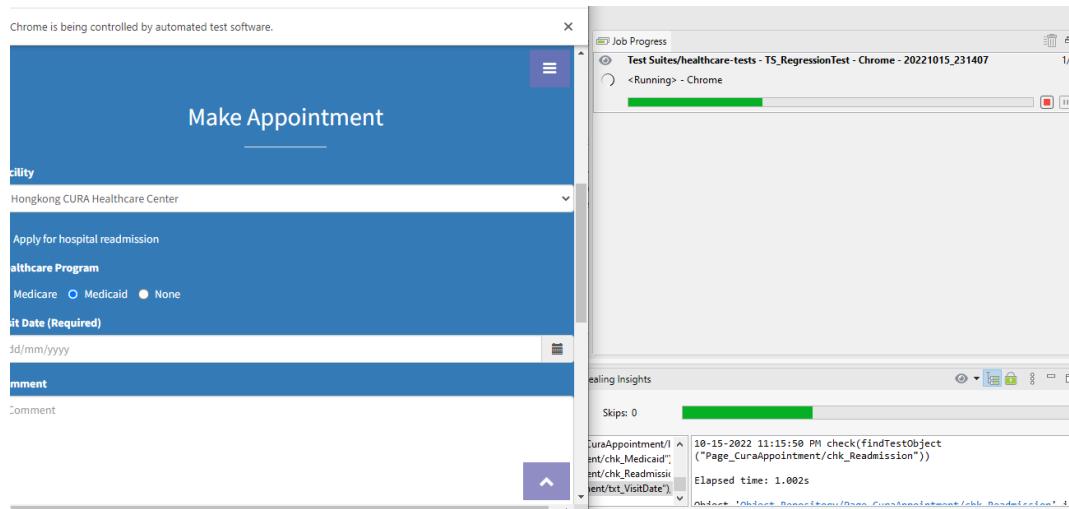
Seperti yang dapat dilihat pada gambar di bawah ini, pada test suit tersebut sudah ditambahkan beberapa test case.

	No.	ID
1	1	Test Cases/Main Test Cases/TC1_Verify Successful Login
2	2	Test Cases/Main Test Cases/TC2_Verify Successful Appointment
3	3	Test Cases/Main Test Cases/TC3_Visual Testing Example

Gambar 3.47 Test Suite yang sudah ditambahkan

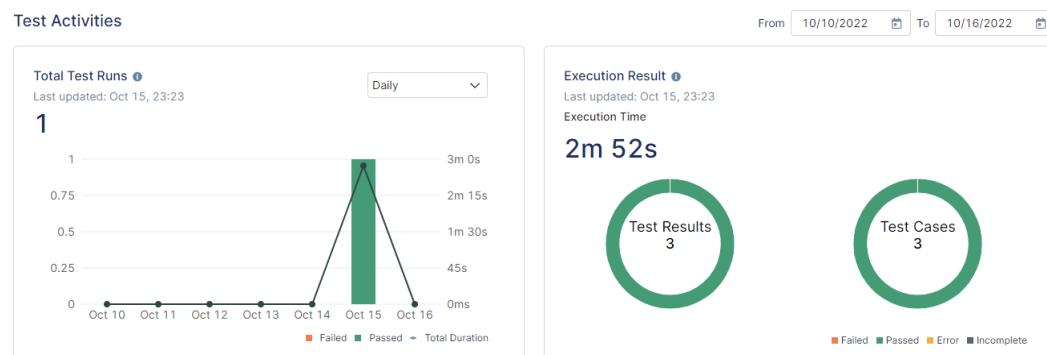
5. Jalankan test suit dengan browser pilihan anda, dan tunggu sampai proses eksekusi selesai. Berikut ini gambar tampilan test suit yang sedang dieksekusi.

Gambar 3.48 Test Suite dijalankan menggunakan Browser



Gambar 3.49 Test Suite sedang di eksekusi

- Setelah proses eksekusi selesai, kita bisa melihat hasil report pengujian pada web analytics katalon. Terlihat pada test activities sudah ada 1 proses pengujian yang sudah dilakukan.



Gambar 3.50 Tampilan hasil detail report

Dapat dilihat seperti pada gambar di atas, hasil detail report menunjukkan 1 passed, dan 0 error untuk pengujian yang dilakukan.

BAB 4

MENJALANKAN TEST MENGGUNAKAN COMMANDLINE DAN PLUGIN JENKINS PADA KATALON STUDIO

4.1. Tujuan

Bagian ini akan membahas dasar – dasar bagaimana cara menjalankan test melalui commandline katalon studio.

Pada akhir pembahasan, diharapkan pembaca dapat :

1. Mengetahui kenapa menjalankan test di commandline itu penting.
2. Mempelajari cara menjalankan test melalui command line.
3. Mengetahui kelebihan menjalankan test melalui commandline.
4. Mempelajari cara menginstall Jenkins & menjalankan Jenkins.
5. Mengetahui cara membuat sebuah project di Jenkins.
6. Mengetahui cara menambahkan command pada aplikasi katalon studio.
7. Mempelajari cara menjalankan test.

4.2. Dasar Teori

Katalon Studio merupakan salah satu software yang digunakan untuk automation testing untuk aplikasi berbasis web dan mobile. Katalon studio dapat dijalankan di semua sistem operasi yaitu windows, linux, dan mac os.

Jenkins adalah server otomatisasi open source mandiri yang dapat digunakan untuk mengotomatiskan semua jenis tugas yang terkait dengan pembuatan, pengujian, dan pengiriman atau penerapan perangkat lunak. Jenkins dapat diinstal melalui paket sistem asli, Docker, atau bahkan dijalankan secara mandiri oleh mesin apa pun dengan Java Runtime Environment (JRE) yang diinstal.

Beberapa kemungkinan langkah yang dapat dilakukan menggunakan Jenkins adalah:

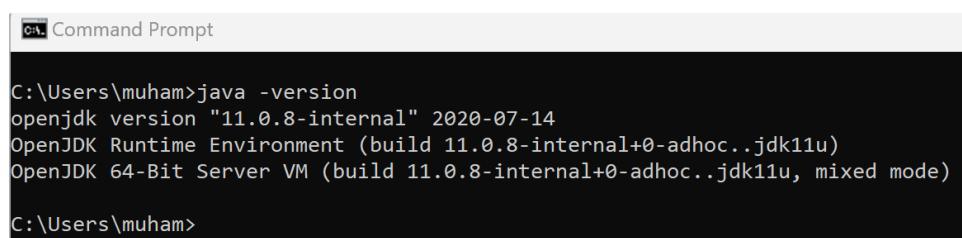
1. Build software menggunakan sistem build seperti Gradle, Maven, dan lainnya.
2. Pengujian otomasi menggunakan kerangka kerja pengujian seperti Nose2, PyTest, Robot, Selenium, dan banyak lagi.
3. Jalankan skrip pengujian (menggunakan terminal Windows, shell Linux, dll).

4. Dapatkan hasil pengujian dan lakukan tindakan posting seperti mencetak laporan pengujian, dan banyak lagi.
5. Jalankan skenario pengujian terhadap kombinasi input yang berbeda untuk mendapatkan cakupan pengujian yang lebih baik.
6. Continuous Integration (CI) tempat artefak dibuat dan diuji secara otomatis. Ini membantu dalam mengidentifikasi masalah dalam produk pada tahap awal pengembangan.

Alasan mengapa menggunakan Jenkins adalah untuk membuat dan menguji produk software secara berkelanjutan, sehingga pengembang dapat terus mengintegrasikan perubahan ke dalam build. Jenkins adalah alat CI / CD open source yang paling populer di pasaran saat ini dan digunakan untuk mendukung DevOps, bersama dengan alat native cloud lainnya. Dalam hampir semua diskusi tentang integrasi berkelanjutan sumber terbuka atau alat pengiriman berkelanjutan (CI / CD), Jenkins pasti akan diangkat. Otomatisasi (termasuk otomatisasi pengujian) adalah salah satu praktik utama yang memungkinkan tim DevOps memberikan solusi teknologi yang "lebih cepat, lebih baik, lebih murah". Jenkins telah menjadi teknologi pendukung utama yang semakin membantu praktik DevOps mendapatkan adopsi yang luas di banyak organisasi di seluruh dunia.

4.3. Percobaan

Syarat yang perlu diperhatikan pada tahapan utama adalah menginstall java, atau jika sudah terinstall di perangkat bisa dilihat dengan menggunakan perintah sebagai berikut:



```
C:\Users\muham>java -version
openjdk version "11.0.8-internal" 2020-07-14
OpenJDK Runtime Environment (build 11.0.8-internal+0-adhoc..jdk11u)
OpenJDK 64-Bit Server VM (build 11.0.8-internal+0-adhoc..jdk11u, mixed mode)

C:\Users\muham>
```

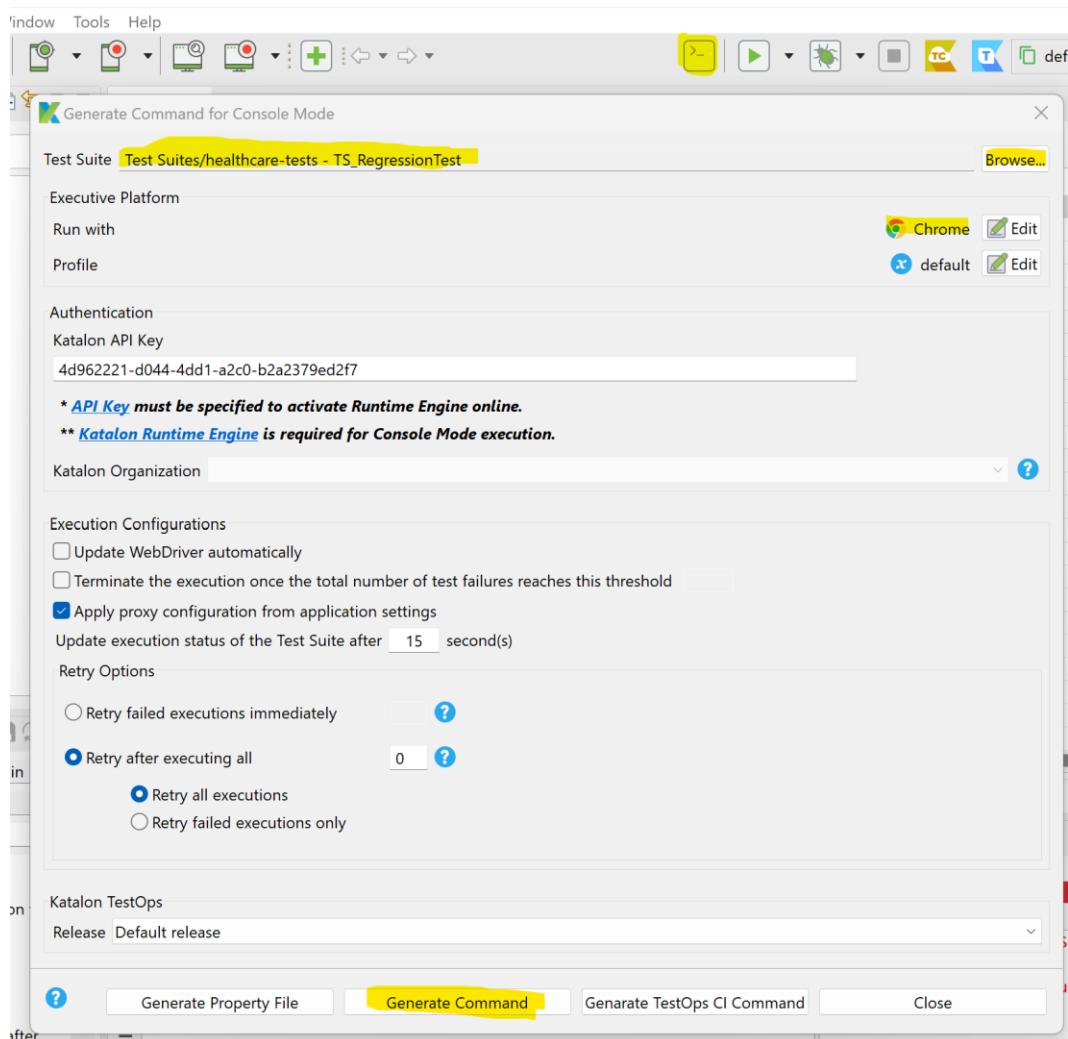
Gambar 4.1 Melihat Versi Java Pada CMD

Dapat dilihat pada gambar diatas terdapat list java yang sudah terinstall setelah menggunakan perintah `java -version` pada commandline. Jika list java atau `java`

belum terinstall di perangkat, download dan install java melalui link berikut ini:
<https://www.oracle.com/java/technologies/downloads/>

Tahapan selanjutnya adalah mengikuti tatacara mengeksekusi test melalui commandline pada halaman dokumentasi katalon studio sebagai pedoman, menggunakan link berikut ini: <https://docs.katalon.com/docs/legacy/katalon-runtime-engine/command-syntax-command-lineconsole-mode-execution>

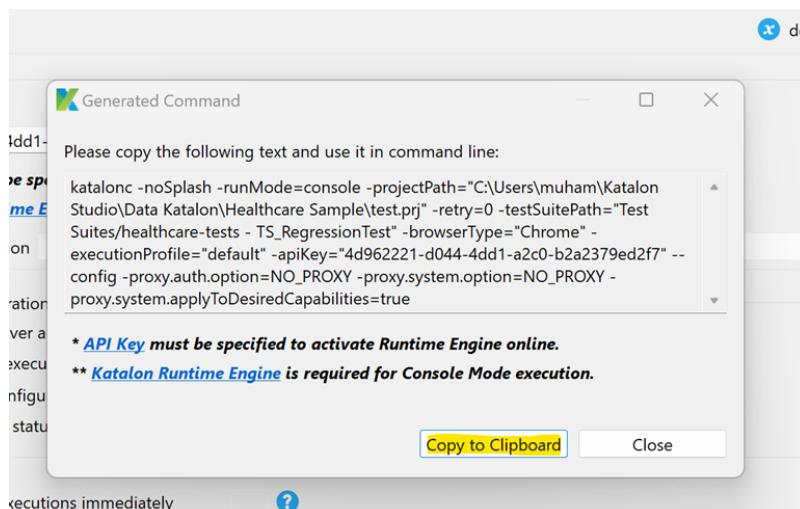
Langkah selanjutnya adalah membuka aplikasi testing catalon studio untuk mempersiapkan command dengan cara mengenerate command seperti gambar di bawah ini:



Gambar 4.2 Mengenerate Command Pada Aplikasi Katalon

Pada gambar diatas dapat dilihat tahapan pertama untuk mengenerate command adalah dengan cara:

- 1 Klik pada icon terminal.
- 2 Kemudian muncul modal / popup menu, pilih Test suite dengan cara klik browse.
- 3 Selanjutnya pilih platform untuk menjalankan test, pada contoh diatas adalah dengan menggunakan google chrome.
- 4 Kemudian klik tombol generate command dan copy hasil generate tersebut seperti gambar di bawah ini:



Gambar 4.3 Copy to Clipboard Generated Command

- 5 Setelah mengenerate command, tahapan selanjutnya adalah buka terminal pada laptop dan arahkan path kedalam folder instalasi atau folder aplikasi katalon tersimpan, seperti gambar di bawah ini:

```
C:\Users\muham>D:  
D:\>cd D:\KULIAH\KPL\Katalon_Studio_PE_Windows_64-8.5.0  
D:\KULIAH\KPL\Katalon_Studio_PE_Windows_64-8.5.0>
```

Gambar 4.4 Path Instalasi Folder Pada CMD

- 6 Tahapan selanjutnya adalah copy command dan paste di dalam terminal laptop yang sudah diarahkan ke path instalasi aplikasi katalon, kemudian klik ENTER seperti gambar di bawah ini:

```
D:\KULIAH\KPL\Katalon_Studio_PE_Windows_64-8.5.0>katalonc -noSplash -runMode=console -projectPath="C:\Users\muham\Katalon_Studio\Data Katalon\Healthcare Sample\test.prj" -retry=0 -testSuitePath="Test Suites/healthcare-tests - TS_RegressionTest" -browserType="Chrome" -executionProfile="default" -apiKey="4d962221-d044-4dd1-a2c0-b2a2379ed2f7" --config -proxy.auth.option=NO_PROXY -proxy.system.option=NO_PROXY -proxy.system.applyToDesiredCapabilities=true
```

Gambar 4.5 Menempel Perintah Generated Command

- 7 Kemudian akan muncul error seperti di bawah ini, penyebabnya adalah katalon tidak bisa di jalankan di terminal windows, solusinya adalah dengan menggunakan aplikasi tambahan yaitu Katalon Runtime Engine, yang bisa di download menggunakan tautan berikut ini
- https://testops.katalon.io/api/v1/katalon/download?platform=win_64&type=re
- 8 Tahapan berikutnya adalah ganti path di terminal windows kearah dimana runtime engine, copy command yang sudah tergenerate dan kemudian jalankan seperti gambar di bawah ini:

```

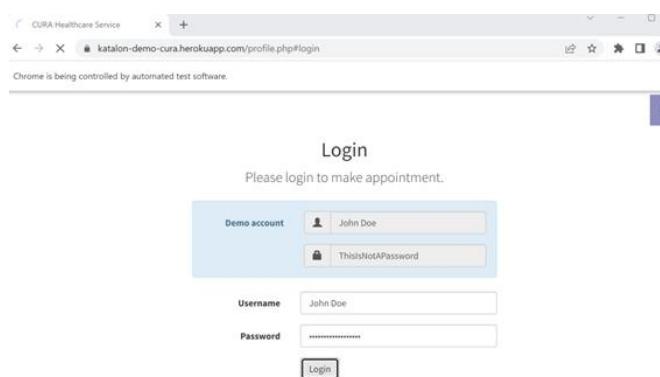
INFO: Katalon Version: 8.5.1
INFO: Katalon Edition: Platform
INFO: Command-line arguments: -runMode=console -projectPath=C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\test.prj -retry=0 -testSuitePath=Test Suites/healthcare-tests - TS_RegressionTest -browserType=Chrome -executionProfile=default -apiKey=***** --config -proxy.auth.option=NO_PROXY -proxy.system.option=NO_PROXY -proxy.system.applyToDesiredCapabilities=true
INFO: User working dir: D:\KULIAH\KPL\Katalon_Studio_Engine_Windows_64-8.5.1
INFO: Error log: C:/Users/muham/AppData/Local/Temp/session-fe5298f7/.metadata/.log
INFO: Katalon KatOne server URL: https://admin.katalon.com
INFO: Katalon TestOps server URL: https://testops.katalon.io
INFO: Katalon Store server URL: https://store.katalon.com
INFO: User home: C:\Users\muham
INFO: Java vendor: Azul Systems, Inc.
INFO: Java version: 1.8.0_282
INFO: Local OS: Windows 10 64bit
INFO: CPU load: 6%
INFO: Total memory: 16062 MB
INFO: Free memory: 2896 MB
INFO: Machine ID: f80c9d08ff0ebe722e51118dc139aa4a

Delete folder: bin
Delete folder: Libs
Cleaning up workspace
Opening project file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\test.prj
Generating global variables...
Project 'test' opened
Start reloading plugins...

```

Gambar 4.6 Mengganti Path Ke File Runtime Engine

- 9 Selanjutnya akan otomatis terbuka browser chrome yang menjalankan website untuk testing secara otomatis seperti gambar di bawah ini:



Gambar 4.7 Testing Website Otomatis Pada Browser Chrome

10 Berikut adalah beberapa hasil atau output dari testing menggunakan commandline:

```
uploading log files of test suite
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\1665379394281.png
-----
Test Suites/healthcare-tests - TS_RegressionTest - Chrome - 20221010_122231.....
.....3/3(100%)
-----
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\1665379476109.png
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\20221010_122231.csv
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\20221010_122231.log
-----
Test Suites/healthcare-tests - TS_RegressionTest - Chrome - 20221010_122231.....
.....3/3(100%)
-----
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\execution.properties
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\execution.uuid
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\execution0.log
-----
Test Suites/healthcare-tests - TS_RegressionTest - Chrome - 20221010_122231.....
.....3/3(100%)
-----
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\JUnit_Report.xml
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\keyes\keyes-appointment page.png
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\keyes\keyes-booked appointment.png
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\keyes\keyes-login page.png
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\testCaseBinding
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\tsc_id.txt
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\videos\screen_recording_1_0.avi
-----
Test Suites/healthcare-tests - TS_RegressionTest - Chrome - 20221010_122231.....
.....3/3(100%)
-----
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\videos\screen_recording_1_0.srt
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\videos\screen_recording_2_0.avi
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\videos\screen_recording_2_0.srt
Sending file: C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\Reports\20221010_122231\healthcare-tests - TS_RegressionTest\20221010_122231\videos\screen_recording_3_0.avi
-----
Test Suites/healthcare-tests - TS_RegressionTest - Chrome - 20221010_122231.....
.....3/3(100%)
```

Gambar 4.8 Output Testing Pada CMD

BAB 5

INTEGRATION WITH GIT AND JENKINS

5.1. Tujuan

1. Mengetahui cara menambahkan projek ke dalam GIT
2. Mengetahui cara meng-Clone Projek
3. Mengetahui cara commit, pull, dan push

5.2. Dasar Teori

A. GIT

Git adalah sistem kontrol versi perangkat lunak yang gratis. Git dapat digunakan untuk menyimpan dan mengelola proyek TestComplete, git juga dapat bekerja dengan repositori lokal (terletak di mesin Anda), serta dengan repositori jarak jauh (terletak di jaringan).

B. Integration with Git

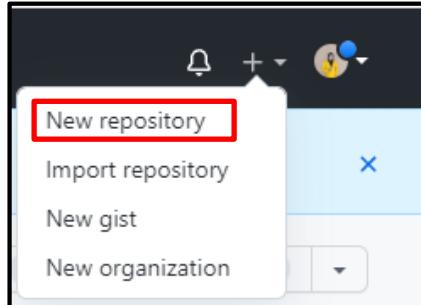
Integration with Git berarti dapat mengintegrasikan TestComplete dengan Git dan bekerja dengan repositori langsung dari antarmuka pengguna TestComplete. Tanpa meninggalkan IDE TestComplete, Anda dapat menambahkan file proyek ke repositori, melakukan perubahan, membatalkan perubahan dan memasukkan ke repositori, dan lainnya. Tindakan ini dapat dilakukan untuk masing-masing item proyek dan elemen turunannya, tidak hanya untuk file project dan project suite. TestComplete menyertakan dialog bawaan khusus yang menyederhanakan pelaksanaan berbagai perintah pada file dan repositori, seperti melihat riwayat file, membuat dan menggabungkan cabang, dan sebagainya.

Disini kita akan membuat remote repository di github, membuat project di katalon studio kemudian meng-clone repository di katalon studio.

5.3. Percobaan Integrasi Dengan GIT

1. Pastikan di katalon studio telah terhubung GIT 
2. Buatlah sebuah repository baru di GitHub,
Caranya sebagai berikut:
 1. Buka situs github di <https://github.com>, buat akun github kemudian login dengan akun yang telah dibuat.

2. Klik tanda plus(+) pada pojok kanan atas kemudian pilih *new repository*



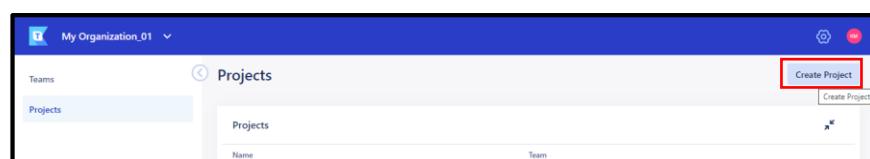
Gambar 5.1 Membuat *repository* baru

3. Buatlah nama repository yang anda inginkan, kemudian pilih public agar semua orang dapat melihat repository anda dan beri centang di add a readme file. Lalu klik create repository

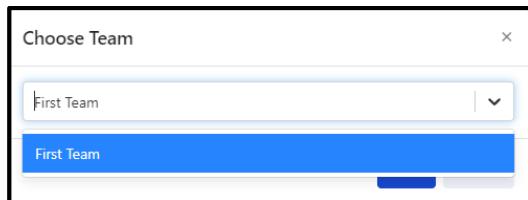
Gambar 5.2 Pengaturan *repository*

3. Buatlah sebuah projek baru di katalon studio Caranya sebagai berikut:

1. Buka situs katalon di <https://katalon.com/>, daftarkan akun di web tersebut.
2. Kemudian masuk dengan akun yang telah didaftar lalu akan diarahkan pada dashboard web katalon studio.
3. Klik create project, lalu pilih team yang diinginkan atau membuat team yang baru.



Gambar 5.3 Membuat projek di *Dashboard* Katalon



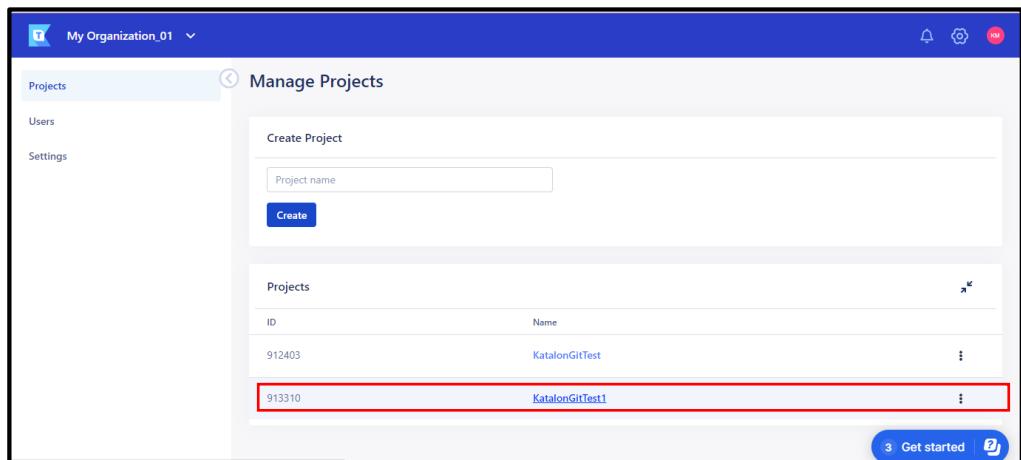
Gambar 5.4 Memilih projek tim

4. Masukkan nama project yang diinginkan



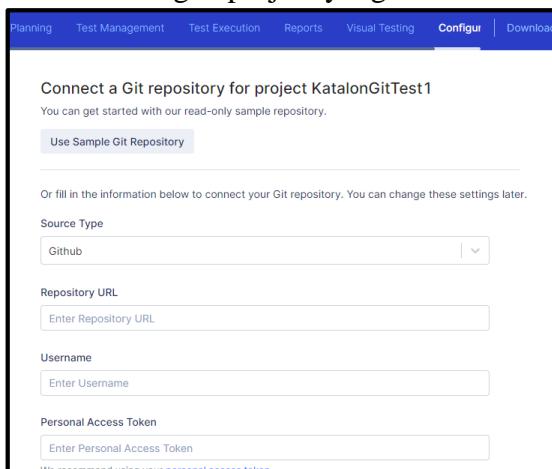
Gambar 5.5 Memberi nama projek

Maka projek yang telah dibuat akan tampil seperti pada gambar berikut



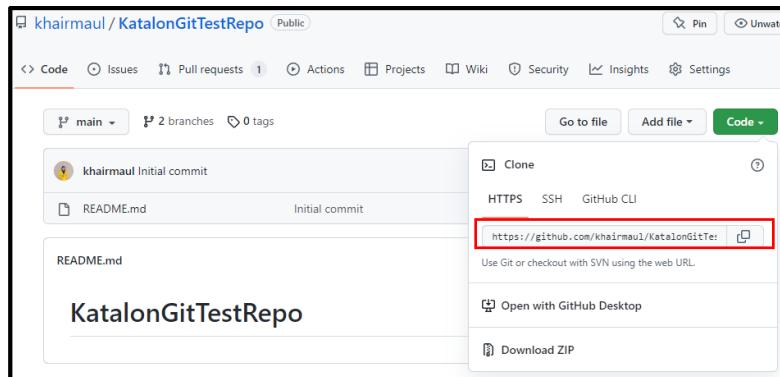
Gambar 5.6 Hasil pembuatan projek

5. Klik pada nama project tersebut, kemudian akan diarahkan untuk mengkoneksi Git dengan project yang telah dibuat.



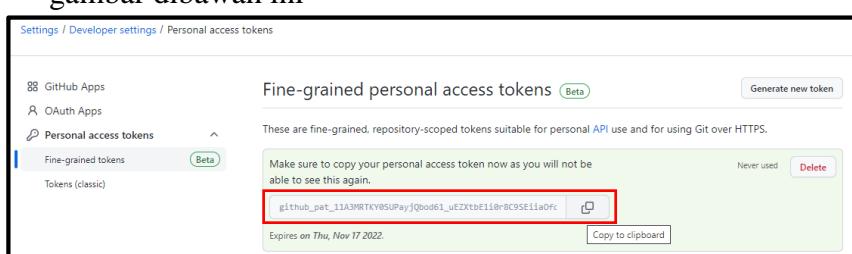
Gambar 5.7 Koneksi dengan *repository* GIT

- Untuk mendapatkan repository URL, kembali ke github klik code pada repository yang telah dibuat, dan copi url clone httpsnya, tampak seperti pada gambar dibawah ini



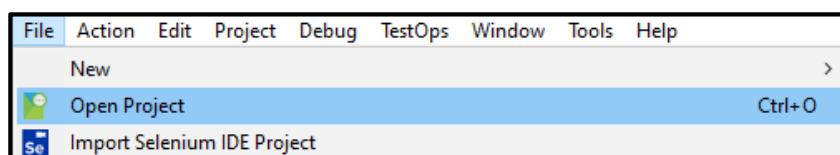
Gambar 5.8 Meng-copy link untuk *clone* repo

- Untuk mendapatkan personal acces token, klik segitiga yang ada pada pojok kanan atas pilih settings → [Developer settings](#) → [Personal access tokens](#) → [Generate new token](#) → dimintakan untuk memasukkan password akun github → masukkan nama token → lalu klik generate, maka akan diperoleh token seperti pada gambar dibawah ini



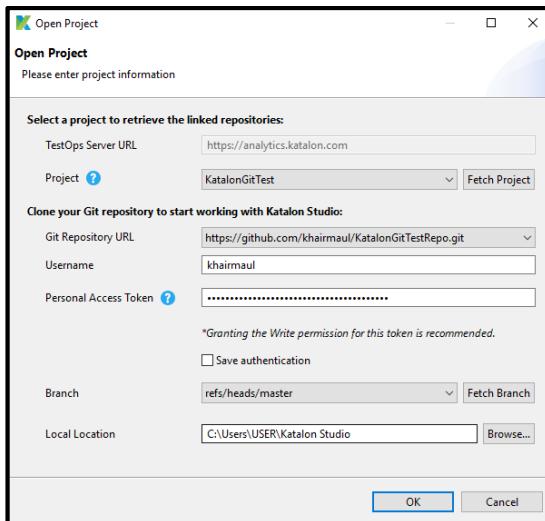
Gambar 5.9 Copy API token

4. Open dan clone clone project



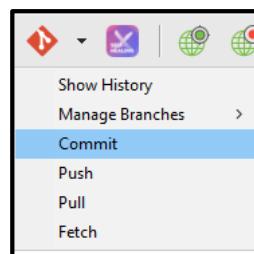
Gambar 5.10 Membuka projek di Katalon Studio

Pilih project yang ingin di open, masukkan username dan acces token dan pilih branch master, seperti dibawah ini



Gambar 5.11 Memilih projek dengan API token

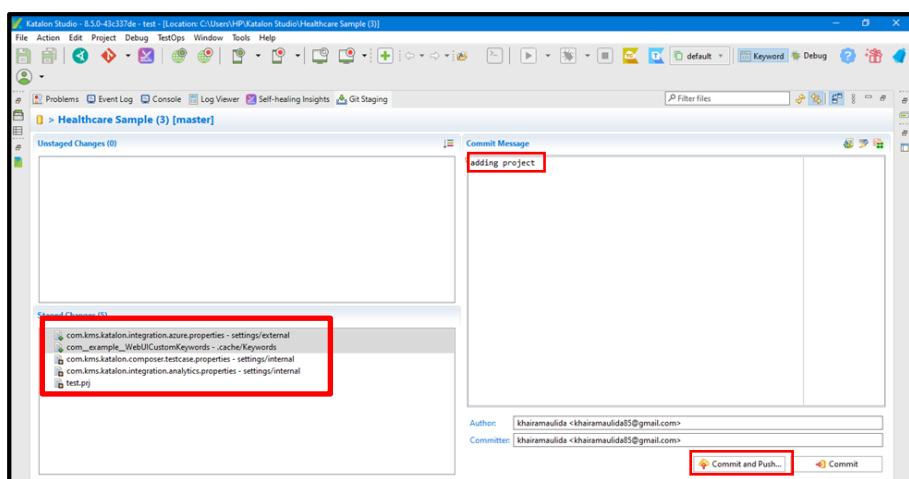
5. Commit project



Gambar 5.12 Melakukan *commit* projek

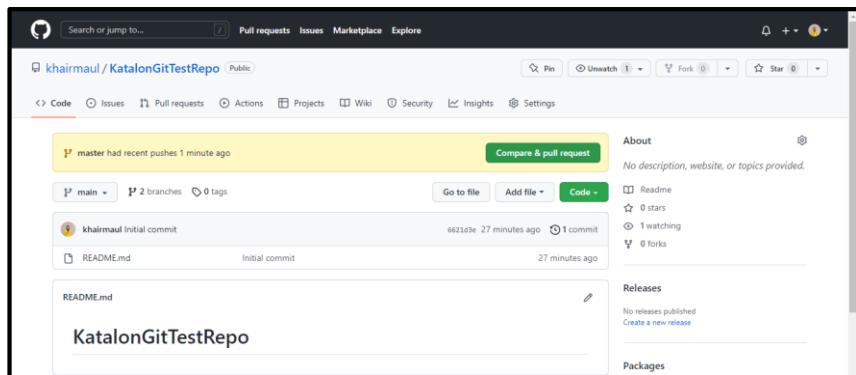
Kemudian lakukan langkah-langkah dibawah ini:

1. Pindahkan isi yang ada di unstaged changes ke dalam staged changes
2. Isikan text dalam jendela commit message kemudian klik klik commit and push.



Gambar 5.13 *Commit* projek Katalon Studio

Hasilnya dapat dilihat di akun github, seperti pada gambar dibawah ini, kemudian refresh terlebih dahulu.

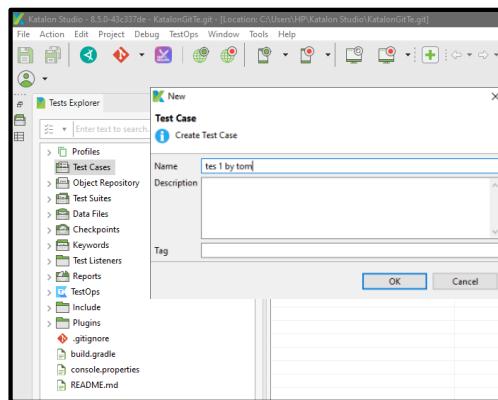


Gambar 5.14 Terdapat *Pull Request* di Repo GitHub

Dalam projek ini dapat dibuat perubahan apa pun dan kapan pun yang diinginkan, juga dapat melakukan tes literasi lengkap dan push di test case baru.

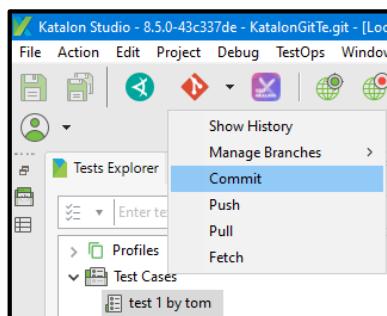
Contoh 1:

- Saya akan menambahkan test case dan memberi nama test 1 by tom
Dapat dilihat seperti pada gambar dibawah ini



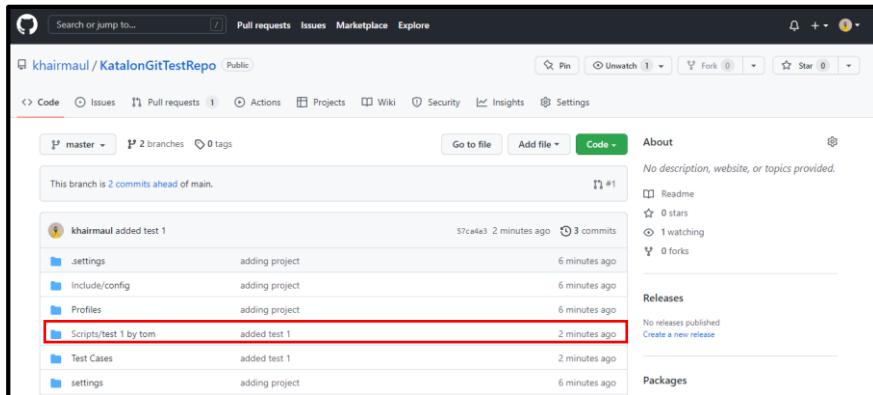
Gambar 5.15 Membuat *test case*

Kemudian klik icon github dan pilih commit



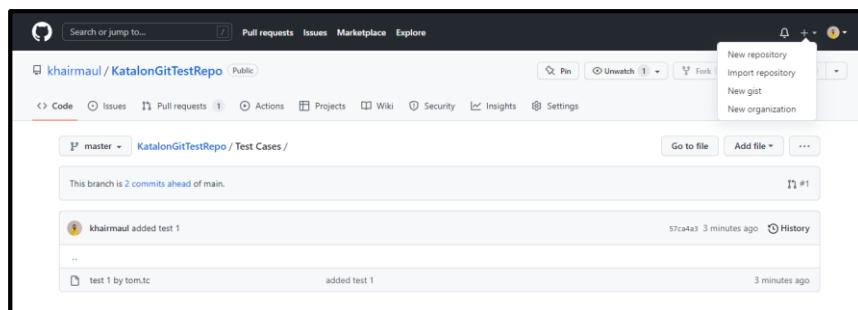
Gambar 5.16 Commit *test case*

Ulangi langkah-langkah seperti pada saat commit project sebelumnya, dan hasilnya dapat dilihat di akun github setelah direfresh seperti pada gambar di bawah ini. Test case “test 1 by tom telah masuk ke dalam repo github”



Gambar 5.17 Hasil pada repo GitHub

Ketika folder dibuka maka akan ditampilkan file testcase tersebut.



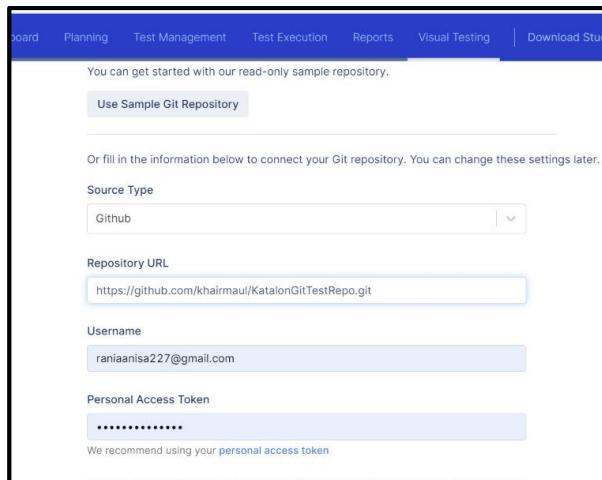
Gambar 5.18 Isi folder Test Case

Contoh 2:

- Ran akan mengkloning proyek yang dibuat oleh tom ke dalam katalon studio miliknya.

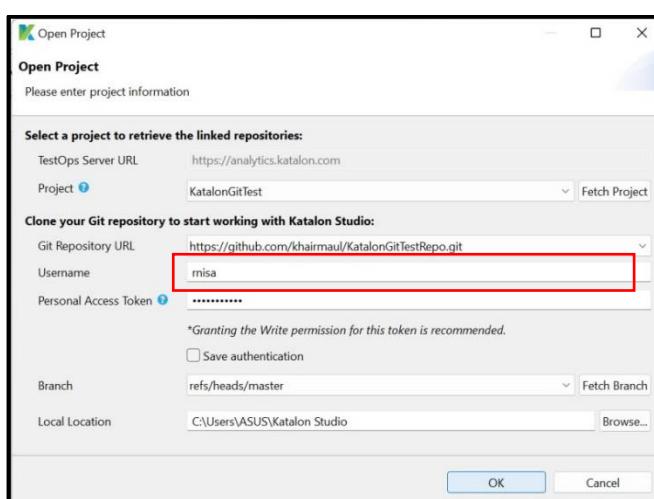
Berikut langkah-langkahnya:

1. Buat sebuah project di katalon studio dengan nama yang sama dengan tom kemudian koneksikan dengan git.



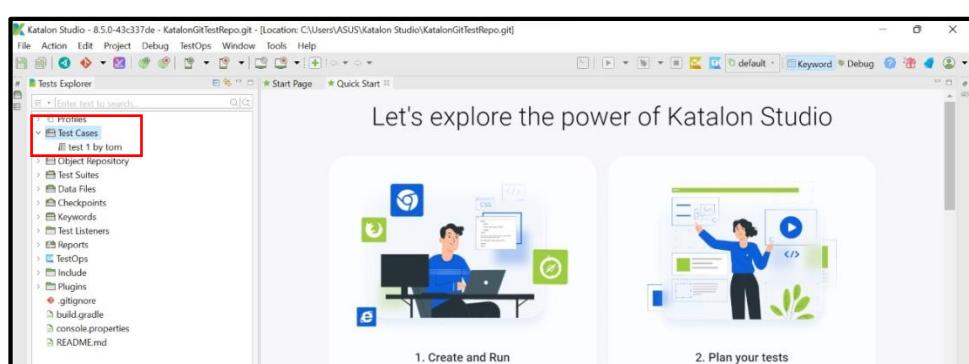
Gambar 5.19 Koneksi Katalon dengan GitHub

2. Open project dan clone repository tom



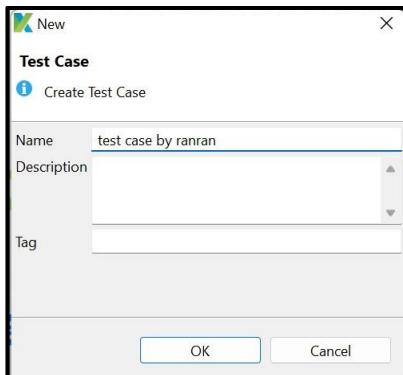
Gambar 5.20 Open Project

Maka akan terlihat test case yang telah dibuat oleh tom seperti pada gambar dibawah.



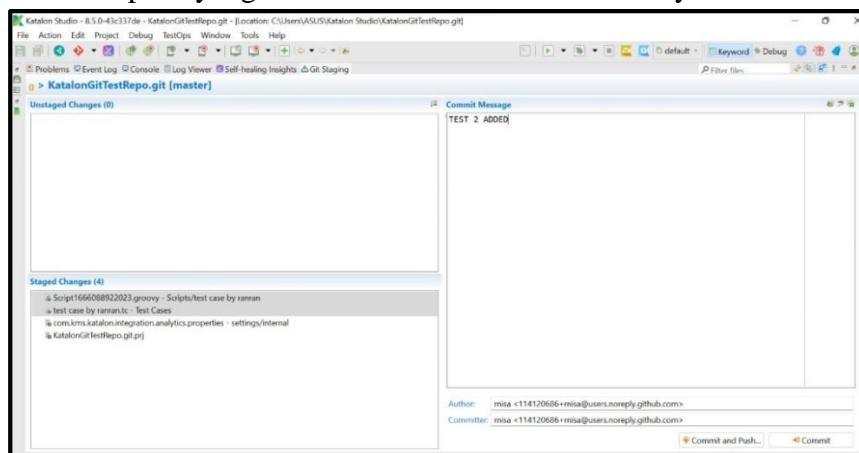
Gambar 5.21 Hasil Open Project

- Kemudian ran akan membuat perubahan pada repository tom yaitu membuat testcase baru



Gambar 5.22 Membuat *test case* baru

- Kemudian meng-commit testcase yang telah dibuat. Caranya sama seperti yang telah dilakukan diatas sebelumnya.



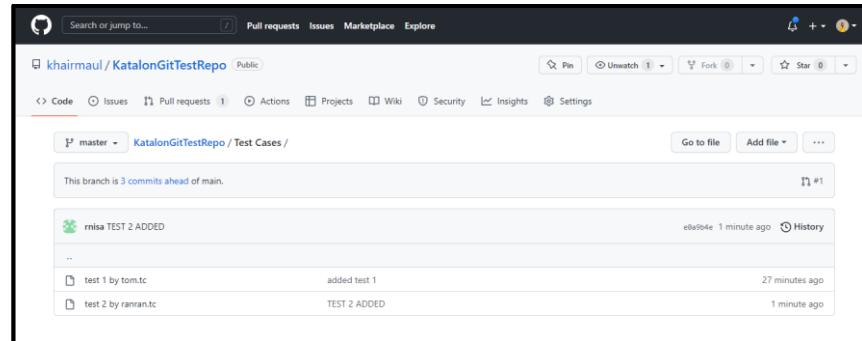
Gambar 5.23 Commit perubahan projek

- Sekarang dapat dilihat pada repository github terdapat folder Test dan Cases commit yaitu MANAGE TEST 2 ADDED

rmisa TEST 2 ADDED		
<input type="checkbox"/> .settings	adding project	31 minutes ago
<input type="checkbox"/> Include/config	adding project	31 minutes ago
<input type="checkbox"/> Profiles	adding project	31 minutes ago
<input type="checkbox"/> Scripts	TEST 2 ADDED	24 seconds ago
<input type="checkbox"/> Test Cases	TEST 2 ADDED	24 seconds ago
<input type="checkbox"/> settings	TEST 2 ADDED	24 seconds ago
<input type="checkbox"/> .gitignore	adding project	31 minutes ago
<input type="checkbox"/> KatalonGitTestRepo.git.prj	TEST 2 ADDED	24 seconds ago
<input type="checkbox"/> README.md	Initial commit	1 hour ago
<input type="checkbox"/> build.gradle	adding project	31 minutes ago
<input type="checkbox"/> console.properties	adding project	31 minutes ago

Gambar 5.24 Hasil perubahan repo GitHub

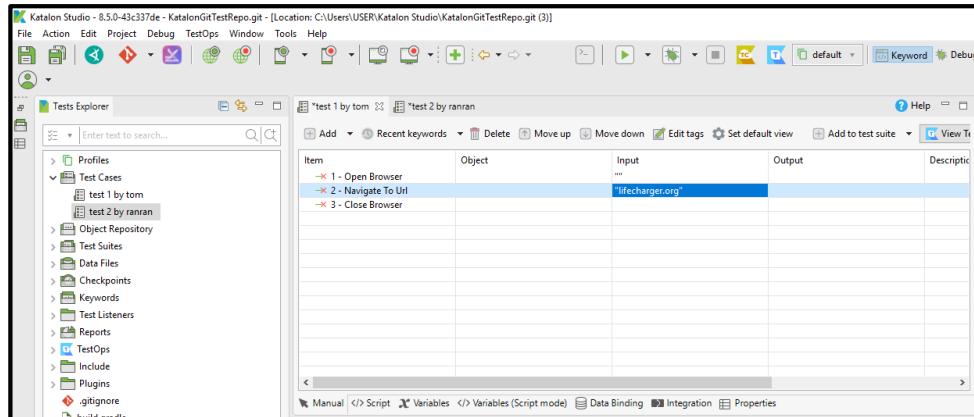
Ketika folder tersebut di klik maka akan menampilkan test case yang telah dibuat oleh ran



Gambar 5.25 Isi folder *Test Case*

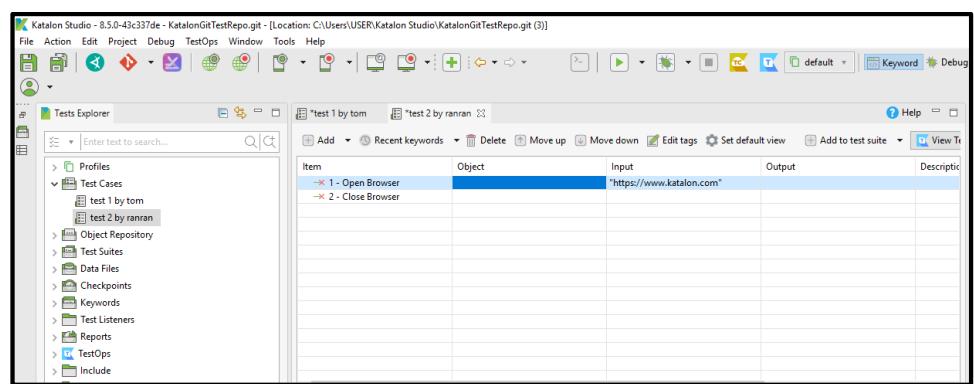
5.4. Percobaan Integrasi Dengan Jenkins

1. Buka katalon studio, pilih *test cases* ‘*test 1 by tom*’ kemudian klik *add* dan pilih *item-item* seperti gambar dibawah.



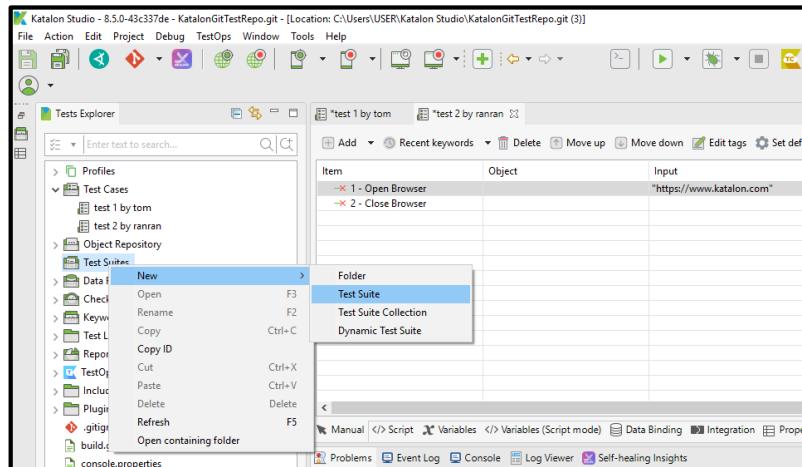
Gambar 5.26 Isi *test case 1 by tom*

2. Kemudian pada *tes 2 by ranran* klik *Add* pilih *item-item* seperti gambar dibawah.



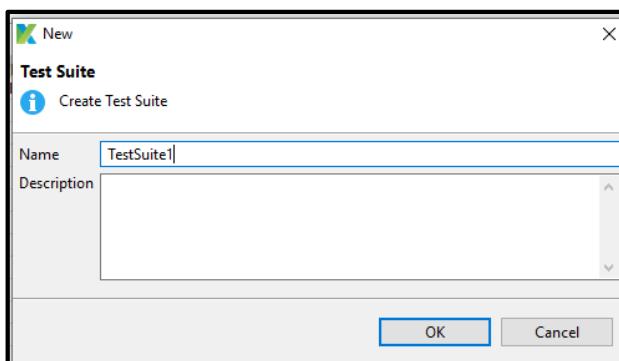
Gambar 5.27 isi *test case 2 by ranran*

3. Klik kanan pada *test suite* pilih *New* kemudian pilih *Test Suite*



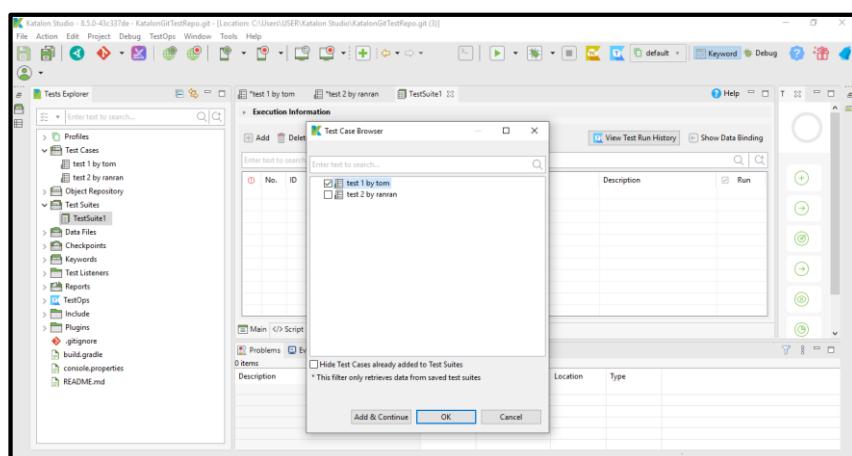
Gambar 5.28 Membuat *Test Suite*

4. Membuat test suite baru dengan memasukkan nama *TestSuite* yang diinginkan.



Gambar 5.29 Memberi nama *Test Suite*

5. Kemudian klik add dan pilih *test 1 by tom*



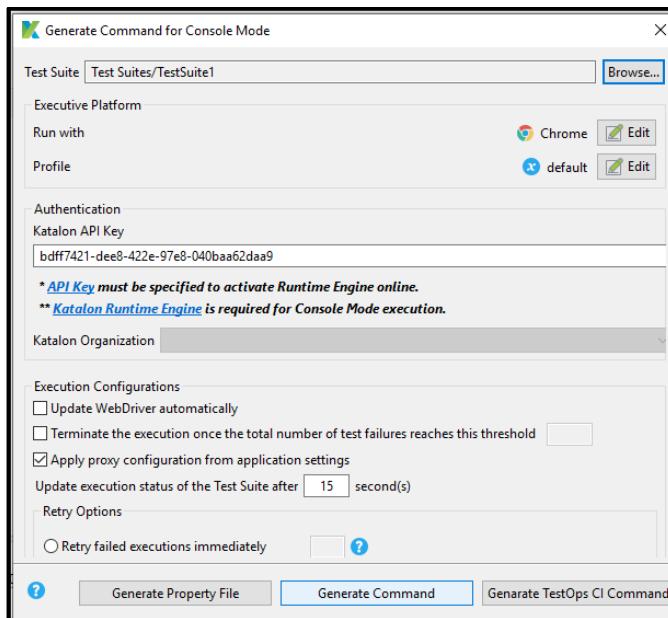
Gambar 5.30 Memasukkan *test case* ke dalam *test suite*

6. Maka hasilnya seperti berikut

No.	ID	Description	Run
1	Test Cases/test 1 by tom		<input checked="" type="checkbox"/>

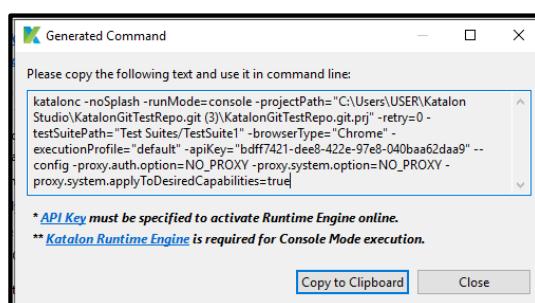
Gambar 5.31 Hasil setelah menambahkan *test case*

7. Kemudian klik build CMD , pada *Browser* pilih Test suites lalu klik generate command



Gambar 5.32 Jendela *Generate Command for Console Mode*

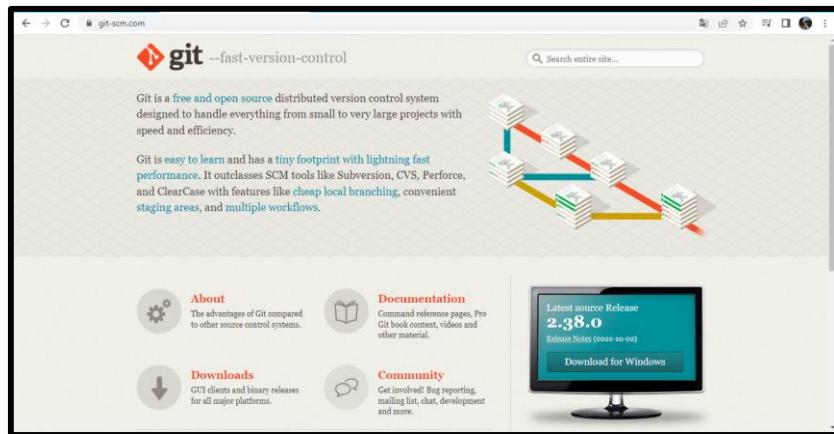
8. Lalu klik pada tombol Copy to Clipboard



Gambar 5.33 *Copy* teks ke *Clipboard*

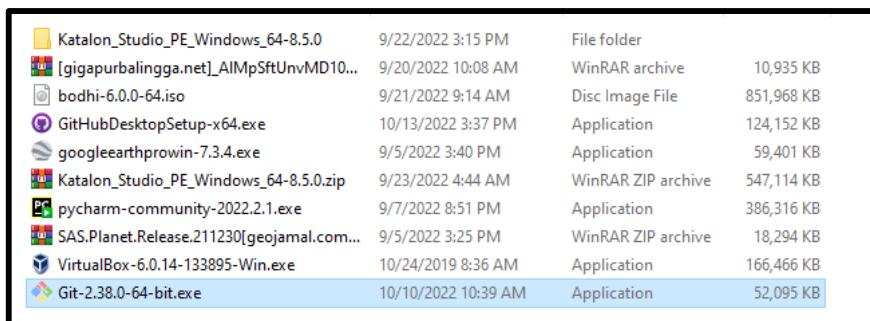
Tahap 2 Install Git

1. Buka git melalui link <https://git-scm.com/>
2. Download git sesuai operasi anda



Gambar 5.34 Download GIT

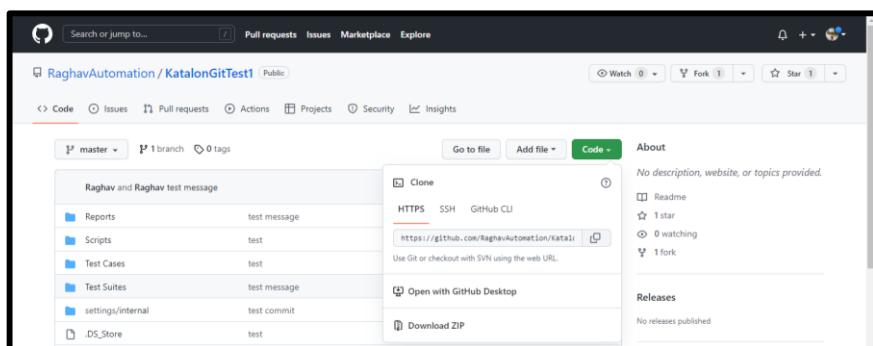
3. Kemudian install



Gambar 5.35 Installer GIT

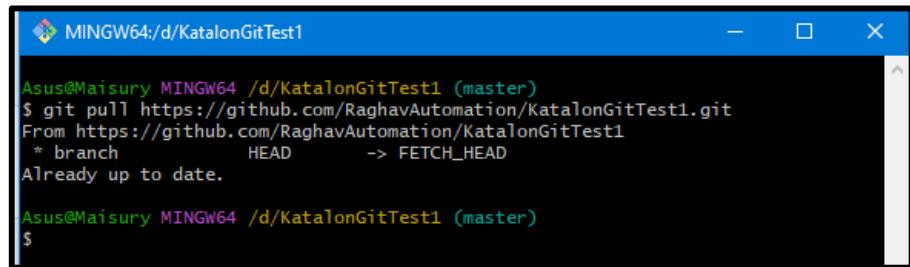
Tahap 3 Uji perintah Git

1. Buka terminal cmd
2. Masuk ke github salin link



Gambar 5.36 Menyalin link clone repository

3. Setelah salin link pada github, paste kan pada git bash

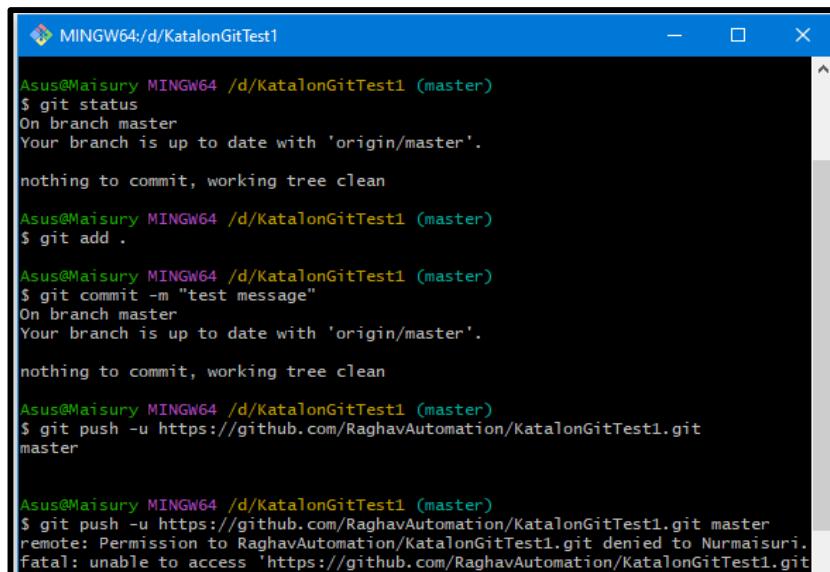


```
Asus@Maisury MINGW64 /d/KatalonGitTest1 (master)
$ git pull https://github.com/RaghavAutomation/KatalonGitTest1.git
From https://github.com/RaghavAutomation/KatalonGitTest1
 * branch            HEAD      -> FETCH_HEAD
Already up to date.

Asus@Maisury MINGW64 /d/KatalonGitTest1 (master)
$
```

Gambar 5.37 Melakukan *pull repository*

4. Gunakan perintah Git Push untuk memasukkan file ke repository pada github.



```
Asus@Maisury MINGW64 /d/KatalonGitTest1 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

Asus@Maisury MINGW64 /d/KatalonGitTest1 (master)
$ git add .

Asus@Maisury MINGW64 /d/KatalonGitTest1 (master)
$ git commit -m "test message"
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

Asus@Maisury MINGW64 /d/KatalonGitTest1 (master)
$ git push -u https://github.com/RaghavAutomation/KatalonGitTest1.git master
remote: Permission to RaghavAutomation/KatalonGitTest1.git denied to Nurmaisuri.
fatal: unable to access 'https://github.com/RaghavAutomation/KatalonGitTest1.git'
```

Gambar 5.38 Melakukan *push repository*

BAB 6

API TESTING

6.1. Tujuan

1. Mengetahui apa itu API
2. Mengetahui bagaimana cara API Testing pada katalon
3. Mengetahui bagaimana cara mengirim nilai dari satu api ke api lainnya dengan xml.
4. Mengetahui bagaimana cara mengirim nilai dari satu api ke api lainnya dengan json.

6.2. Dasar Teori

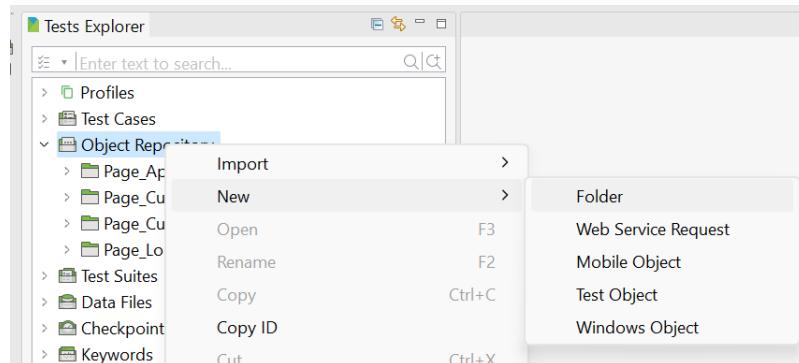
Api merupakan aplikasi yang memungkinkan para developer untuk mengintegrasikan bagian aplikasi dengan bagian aplikasi lainnya.

6.3. Percobaan API Testing

a. How To Do Api Testing With Katalon

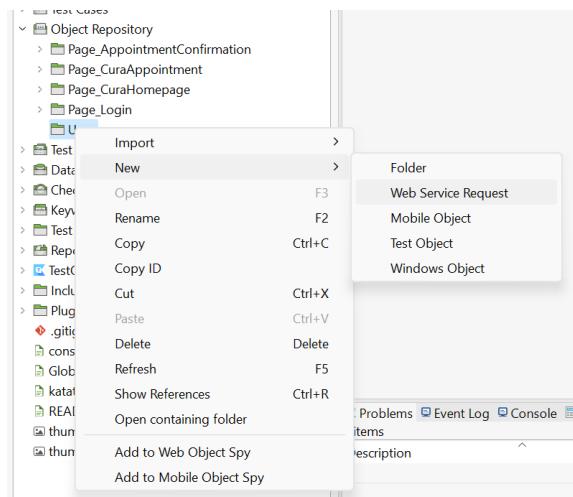
Katalon juga dapat digunakan sebagai aplikasi untuk melakukan pengujian terhadap API. Berikut langkah-langkah yang dapat dilakukan dalam menguji API pada katalon.

1. Buka project pada katalon, pada bagian Test Explorer klik kanan pada folder Object Respository untuk membuat folder baru.



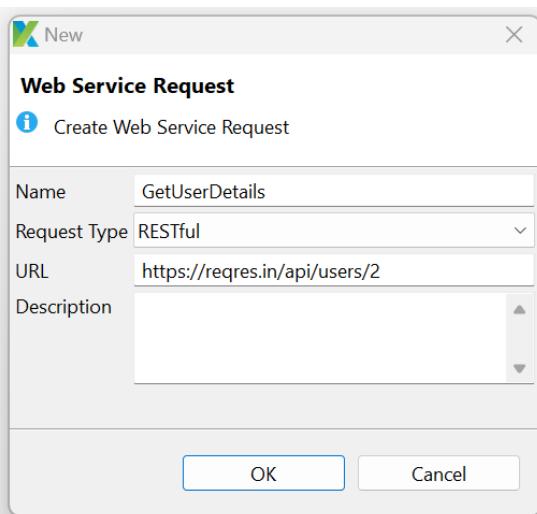
Gambar 6.1 Membuat Folder Baru

2. Setelah folder berhasil dibuat klik kanan, pilih web service Request



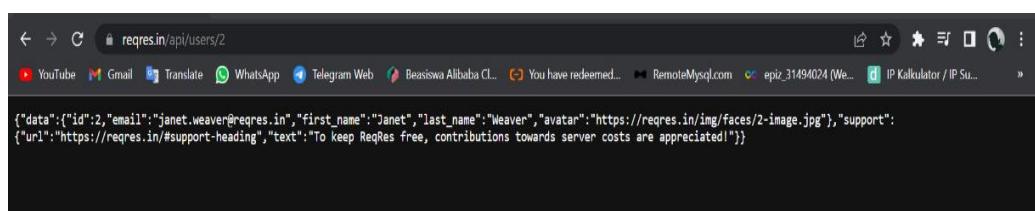
Gambar 6.2 Pilih Web Service Request

3. Kemudian isikan name untuk file yang akan dibuat, pilih RESTful sebagai request type, dan masukkan URL API, klik Ok.



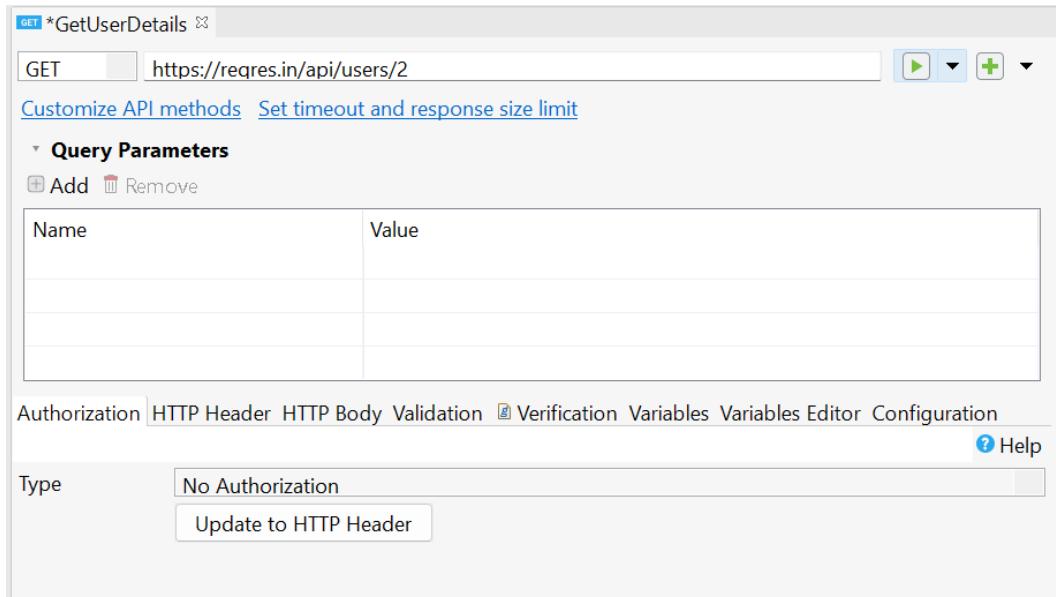
Gambar 6.3 Membuat Web Service Request

4. URL API tersebut jika dibuka melalui web akan tampil seperti berikut



Gambar 6.4 URL API Pada Web

5. Berikut tampilan file yang dibuat, selanjutnya klik icon play.



Gambar 6.5 Tampilan File

6. Berikut tampilan hasilnya, yaitu GET user dengan id 2 berhasil dilakukan, ditandai dari Responsenya yaitu status 200 OK

The screenshot shows the Postman Response panel with a status of 200 OK, elapsed time of 1131 ms, and size of 644 bytes. The response body is displayed in pretty JSON format, showing a user object with properties like id, email, first_name, last_name, avatar, support, and url. The JSON response is as follows:

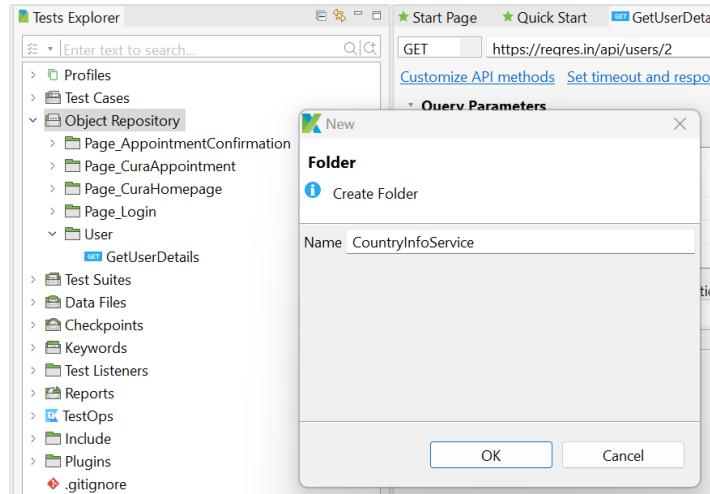
```
1 {  
2   "data": {  
3     "id": 2,  
4     "email": "janet.weaver@reqres.in",  
5     "first_name": "Janet",  
6     "last_name": "Weaver",  
7     "avatar": "https://reqres.in/img/faces/2-image.jpg"  
8   },  
9   "support": {  
10    "url": "https://reqres.in/#support-link"  
11  }  
12 }  
13 
```

Select JSON or XML response data and press Ctrl/Command + K to add verification scripts directly.

JSON XML HTML JavaScript Wrap Lin

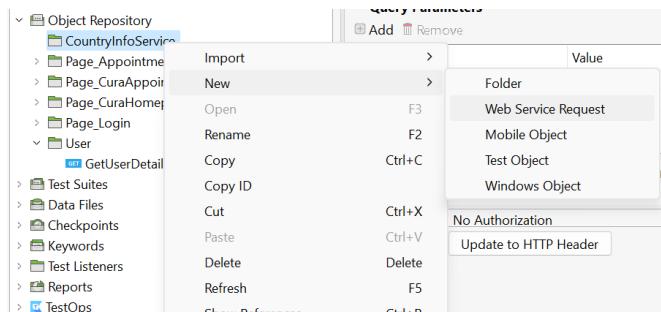
Gambar 6.6 Tampilan Response

- Selanjutnya buat folder baru dengan nama “CountryInfoService” pada Object Repository



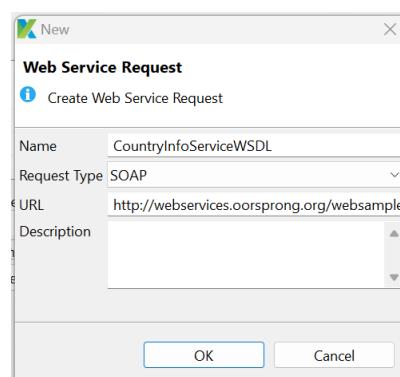
Gambar 6.7 Membuat Folder Baru

- Klik kanan pada folder “CountryInfoService”, pilih new dan klik Web Service Request



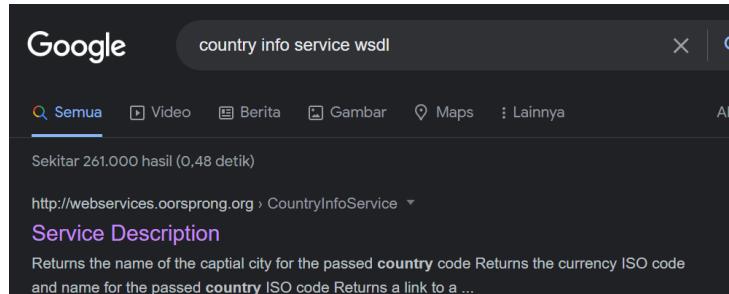
Gambar 6.8 Pilih Web Service Request

- Kemudian isikan name sebagai berikut, pilih SOAP sebagai Request Type dan tempelkan url berikut.



Gambar 6.9 Membuat Web Service Request

10. Untuk mendapatkan URL diatas dapat di searching di google, buka website berikut.



Gambar 6.10 Searching Website di Google

11. Berikut tampilan website, lalu copy link addressnya.

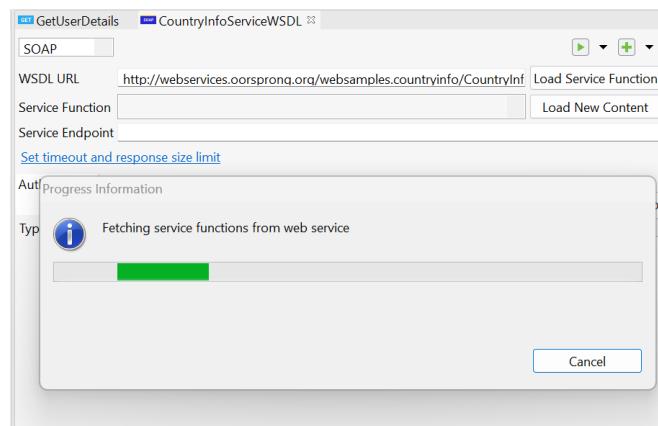
```

<?xml version="1.0"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12" xmlns:tns="http://www.orsprong.org/websamples.countryinfo" name="CountryInfoService"
  targetNamespace="http://www.orsprong.org/websamples.countryinfo">
  <types>
    <xss:schema elementFormDefault="qualified" targetNamespace="http://www.orsprong.org/websamples.countryinfo">
      <xss:complexType name="tContinent">
        <xss:sequence>
          <xss:element name="sCode" type="xs:string"/>
          <xss:element name="sName" type="xs:string"/>
        </xss:sequence>
      </xss:complexType>
      <xss:complexType name="tCurrency">
        <xss:sequence>
          <xss:element name="sISOCode" type="xs:string"/>
          <xss:element name="sName" type="xs:string"/>
        </xss:sequence>
      </xss:complexType>
      <xss:complexType name="tCountryCodeAndName">
        <xss:sequence>
          <xss:element name="sISOCode" type="xs:string"/>
          <xss:element name="sName" type="xs:string"/>
        </xss:sequence>
      </xss:complexType>
      <xss:complexType name="tCountryCodeAndNameGroupedByContinent">
        <xss:sequence>
          <xss:element name="Continent" type="tns:tContinent"/>
          <xss:element name="CountryCodesAndNames" type="tns:ArrayOfCountryCodeAndName"/>
        </xss:sequence>
      </xss:complexType>
      <xss:complexType name="tCountryInfo">
        <xss:sequence>
          <xss:element name="sISOCode" type="xs:string"/>
          <xss:element name="sName" type="xs:string"/>
          <xss:element name="sCapitalCity" type="xs:string"/>
          <xss:element name="sPhoneCode" type="xs:string"/>
          <xss:element name="sContinentCode" type="xs:string"/>
          <xss:element name="sCurrencyISOCode" type="xs:string"/>
        </xss:sequence>
      </xss:complexType>
    </xss:schema>
  </types>
  <message name="GetContinent">
    <part name="parameters" type="tns:tContinent"/>
  </message>
  <message name="GetCountryCodeAndName">
    <part name="parameters" type="tns:tCountryCodeAndNameGroupedByContinent"/>
  </message>
  <message name="GetCountryInfo">
    <part name="parameters" type="tns:tCountryInfo"/>
  </message>
  <port name="CountryInfoPort" binding="tns:CountryInfoPortBinding">
    <soap:address location="http://webservices.orsprong.org/websamples.countryinfo/CountryInfoService?wsdl" />
  </port>
</definitions>

```

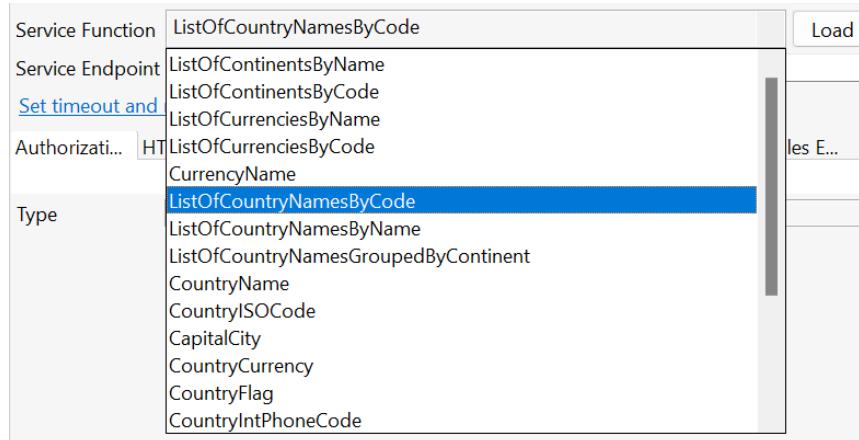
Gambar 6.11 Tampilan Website

12. Berikut Tampilan File yang telah kita buat, klik “Load Service Function”.



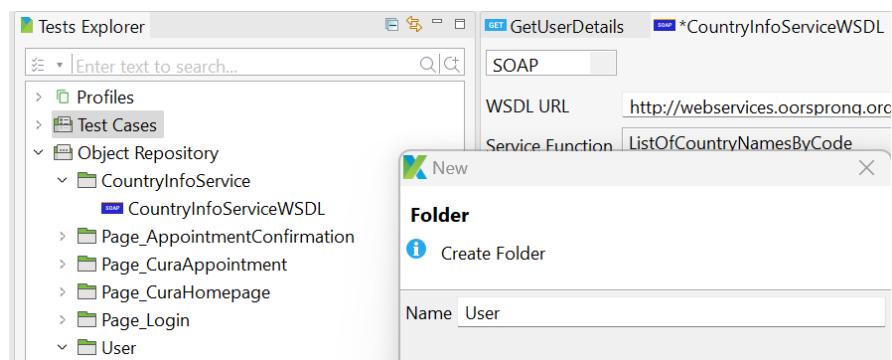
Gambar 6.12 Tampilan File

13. Pada Service Function pilih “ListOfCountryNamesByCode”



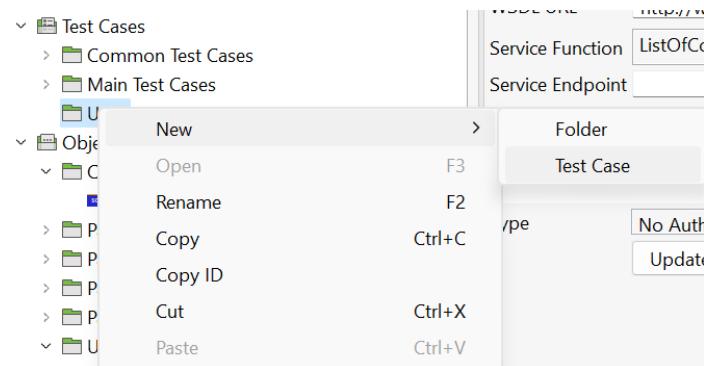
Gambar 6.13 Tampilan Service Function

14. Langkah selanjutnya pada Test Explorer, buat folder “User” didalam Test Case



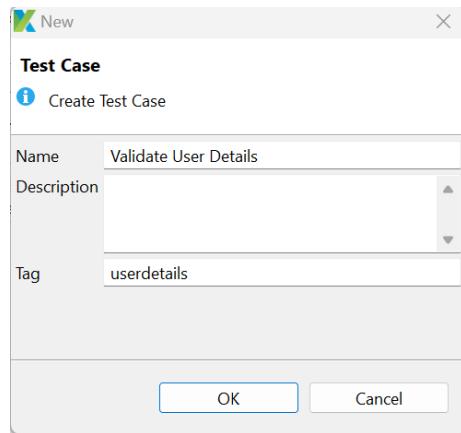
Gambar 6.14 Membuat Folder User

15. Pada folder “User” klik kanan pilih New dan klik Test Case



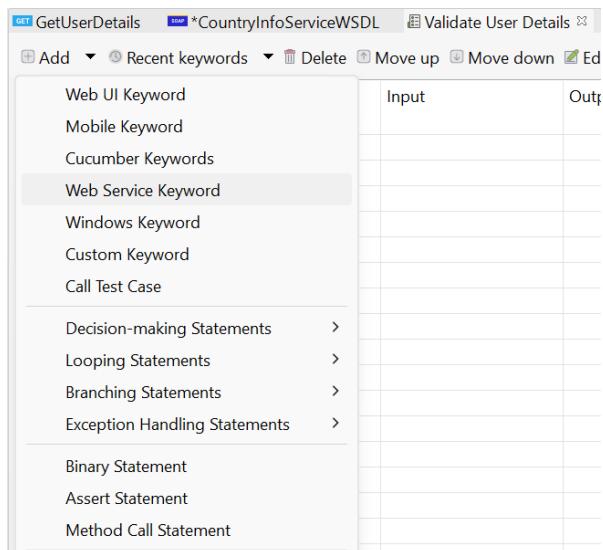
Gambar 6.15 Folder User

16. Isikan form seperti berikut, klik Ok



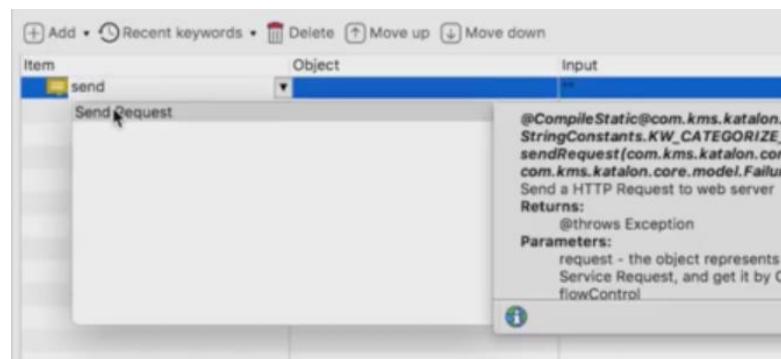
Gambar 6.16 Form Test Case

17. Selanjutnya klik dropdown Add pilih Web Service Keyword



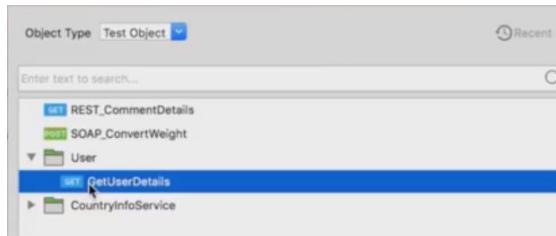
Gambar 6.17 Web Service Keyword

18. Gantikan Namanya menjadi “Send Request”



Gambar 6.18 Send Request

19. Pada Object yang berisi null, klik lalu pilih “GetUserDetails”



Gambar 6.19 Get User Details

20. Pada Output isikan “response”

Item	Object	Input	Output	Description
✖ 1 - Send Request	GetUserDetails		response	

Gambar 6.20 Response Pada Kolom Output

21. Tambahan “Verify Response Status Code”

A screenshot of the Katalon Studio keyword editor. The table shows a single row with 'Item' '1 - Send Request', 'Object' 'GetUserDetails', 'Input' '"; 0', and 'Output' 'response'. To the right of the table, a tooltip provides detailed information about the 'Verify Response Status Code' step, including its Java code implementation and parameters.

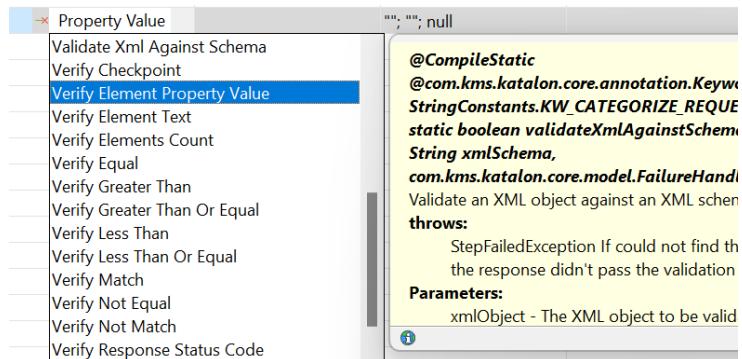
Gambar 6.21 Verify Response Status Code

22. Ubah input dari “Verify Response Status Code” seperti berikut

A screenshot of the 'Input' dialog box. It contains a table with two rows. Row 1 has 'Param Name' 'responseObject', 'Param Type' 'ResponseObject', 'Value Type' 'Variable', and 'Value' 'response'. Row 2 has 'Param Name' 'expectedStatusCode', 'Param Type' 'Integer', 'Value Type' 'Number', and 'Value' '200'. At the bottom are 'OK' and 'Cancel' buttons.

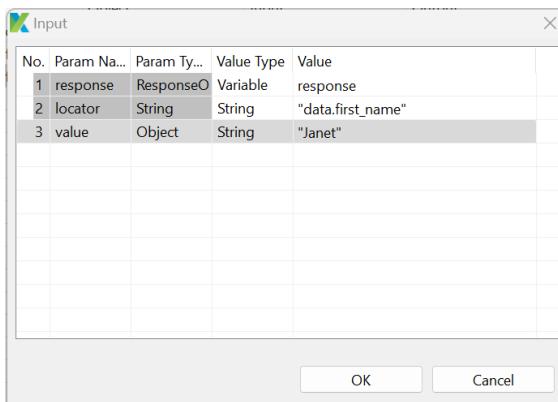
Gambar 6.22 Verify Response Status Code

23. Kemudian tambahkan “Verify Element Property Value”



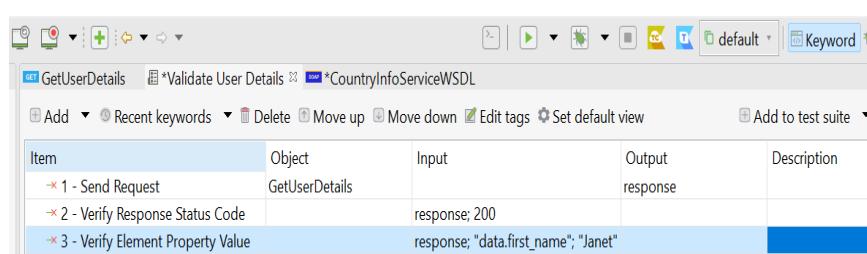
Gambar 6.23 Property Value

24. Ubah input dari “Verify Element Property Value” seperti berikut



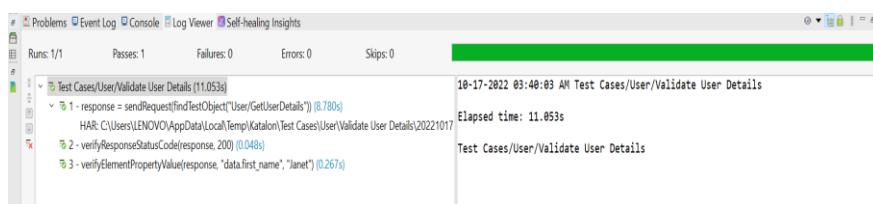
Gambar 6.24 Verify Element Property Value

25. Sehingga menjadi seperti berikut, selanjutnya klik icon play



Gambar 6.25 Klik Icon Play

26. Berikut hasilnya, dapat dilihat lewat Log Viewer, hasilnya tidak ada errornya



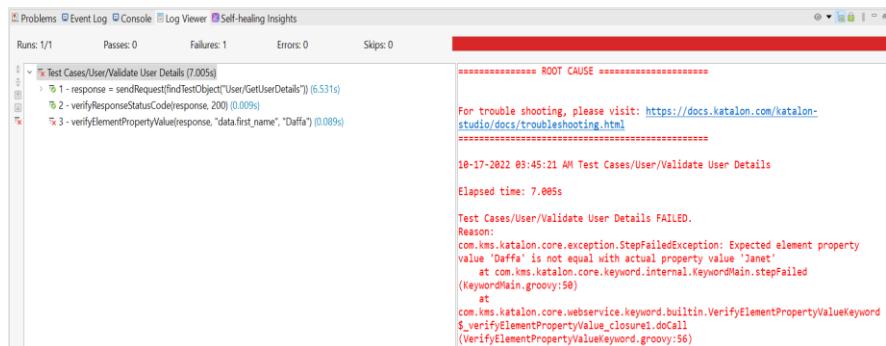
Gambar 6.26 Melihat Hasil di Log Viewer

27. Namun ketika kita ganti inputan pada “Verify Element Property Value” seperti berikut

No.	Param Na...	Param Ty...	Value Type	Value
1	response	ResponseO	Variable	response
2	locator	String	String	"data.first_name"
3	value	Object	String	"Daffa"

Gambar 6.27 Mengganti isi dari Verify Element Property Value

28. Maka hasilnya akan error, seperti berikut



Gambar 6.28 Hasil Error

29. Penyebab Error dikarenakan pada Object Repository terdapat folder User yang didalamnya memiliki file GetUserDetails yang berisi sebagai berikut. Oleh karena itu inputan pada “Verify Element Property Value” hanya menerima “Janet” sebagai string selain itu akan error.

Response

Status: 200 OK Elapsed: 1505 ms Size: 640 bytes

Body Header Verification Log Validation Log

pretty raw preview

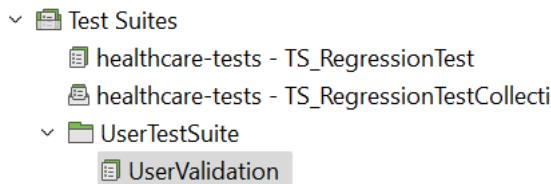
```

1  {
2  	"data": {
3  		"id": 2,
4
5  		"email": "janet.weaver@reqres
6 .in",
7  		"first_name": "Janet",
8  		"last_name": "Weaver",
9
10 	"avatar": "https://reqres.in/
11 img/faces/2-image.jpg"
12 	},
13 	"support": {
14 	support.url: https://reqres.in/#su
15 }
16
17 Select JSON or XML response data and press
18 Ctrl/Command + K to add verification scripts directly.
19
20  JSON  XML  HTML  JavaScript  Wrap Lin

```

Gambar 6.29 Penyebab Error

30. Buatkan test suite didalam folder UserTestSuite berikan nama “UserValidation”



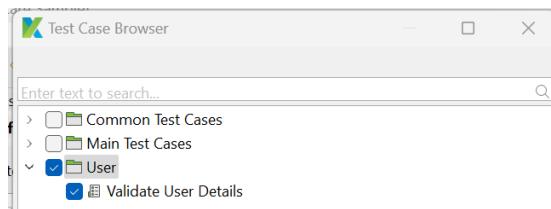
Gambar 6.30 Membuat File UserValidation

31. Berikut tampilannya, klik add

No.	ID	Description	Run

Gambar 6.31 Tampilan dari File UserValidation

32. Centang Validate User Details



Gambar 6.32 Centang Validate User Details

33. Berikut setelah di add, klik icon play

No.	ID	Run
1	Test Cases/User/Validate User Details	

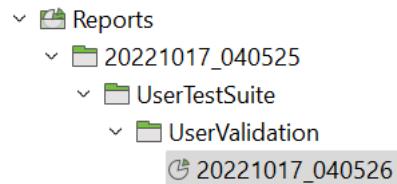
Gambar 6.33 Klik Icon Play

34. Berikut hasil dari Log Viewer

```
10-17-2022 04:05:33 AM Test Suites/UserTestSuite/UserValidation
Elapsed time: 9.756s
```

Gambar 6.34 Hasil dari Log Viewer

35. Selanjutnya kita buka Reports untuk melihat hasil pengujian



Gambar 6.35 Folder Reports Melihat Hasil Pengujian

36. Berikut hasil report dari pengujian file json yang telah dilakukan

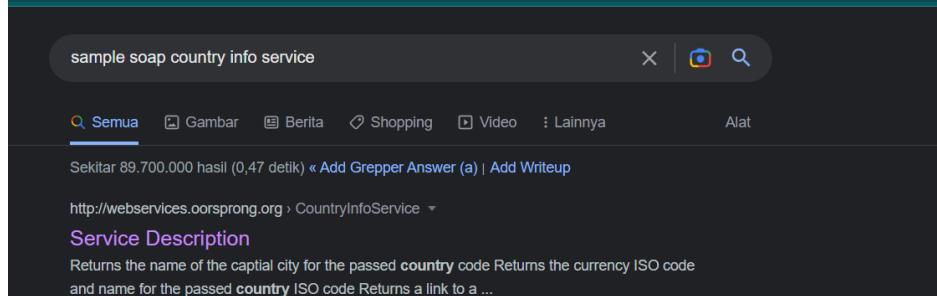
Test Cases Table		Resolution
<input checked="" type="checkbox"/> Passed	<input checked="" type="checkbox"/> Failed	<input checked="" type="checkbox"/> Error
<input checked="" type="checkbox"/> Incomplete		
<input checked="" type="checkbox"/> Skipped		
Export report Katalon TestOps		
Search here...		
No.	Name	Resolution
1	Validate User Details (8.439s)	
Summary Execution Settings Execution Environment		
Test Suite ID	Test Suites/UserTestSuite/UserValidation	
Host name	LENOVO - LAPTOP-M8U7JU6R	Local OS Windows 10 64bit
Katalon version	8.5.0.208	Platform
Start	2022-10-17 04:05:33	End 2022-10-17 04:05:43
Elapsed	9.756s	
Total TC	1	
Passed	1	Failed 0
Error	0	Skip 0
Incomplete	0	

Gambar 6.36 Hasil Pengujian File Json

b. API CHAINING | SOAP-XML | HOW TO SEND VALUE FROM ONE API TO OTHER

Katalon juga dapat melakukan chaining API. Chaining API adalah suatu proses untuk mengekstrak nilai dari respon suatu API dan memberikan nilai ke permintaan API berikutnya. Pada tahapan ini kita akan melakukan chaining API pada API SOAP yang memiliki respon XML. Berikut tahapannya.

1. Cari satu contoh SOAP API, seperti gambar di bawah ini.



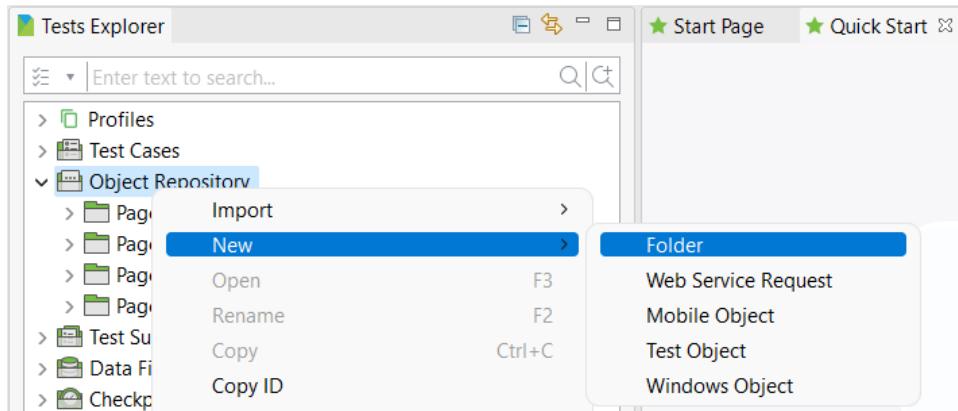
Gambar 6.37 Contoh SOAP API

2. Berikut merupakan contoh script soap api dalam format xml.

```
<?xml version="1.0"?>
<definitions xmlns="http://schemas.xmlsoap.org/wSDL/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wSDL/soap/" xmlns:soap12="http://schemas.xmlsoap.org/wSDL/soap12/" xmlns:tns="http://www.orsprong.org/websamples.countryinfo" targetNamespace="http://www.orsprong.org/websamples.countryinfo">
  <types>
    <xsd:schema elementFormDefault="qualified" targetNamespace="http://www.orsprong.org/websamples.countryinfo">
      <xsd:complexType name="tContinent">
        <xsd:sequence>
          <xsd:element name="sName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
  </types>
  <message name="CountryInfoService" type="tns:CountryInfoService">
    <part name="parameters" type="tns:tCountryCodeAndNameGroupedByContinent" />
    <part name="request" type="tns:tCountryCodeAndName" />
    <part name="response" type="tns:tCountryInfo" />
  </message>
  <binding name="CountryInfoService" type="tns:CountryInfoService">
    <operation name="GetContinentCodeByName" type="tns:tGetContinentCodeByName" />
    <operation name="GetCountryCodeAndName" type="tns:tGetCountryCodeAndName" />
    <operation name="GetCountryInfo" type="tns:tGetCountryInfo" />
  </binding>
  <service name="CountryInfoService" type="tns:CountryInfoService" />
</definitions>
```

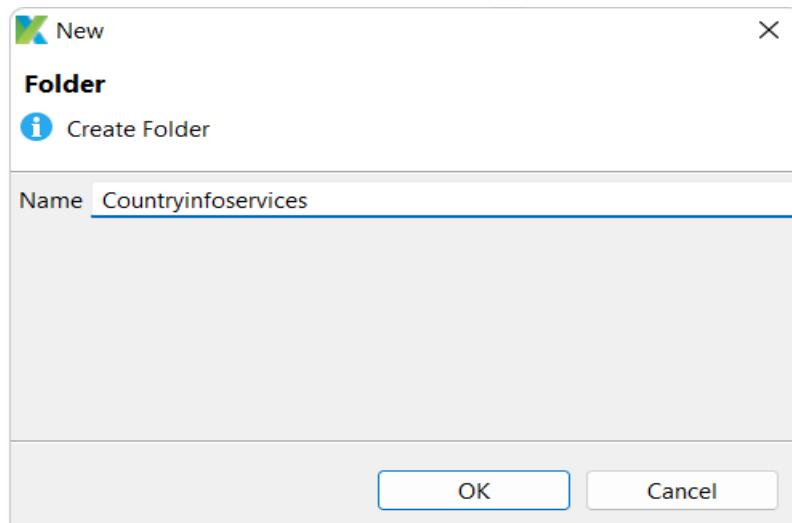
Gambar 6.38 Script SOAP API

3. Selanjutnya buat 1 folder baru.



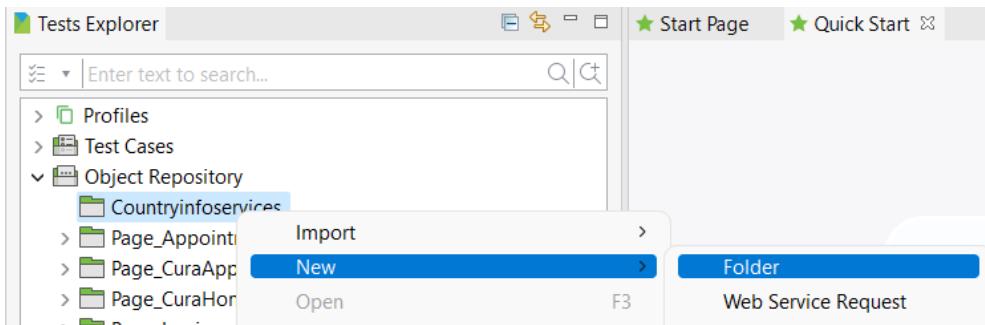
Gambar 6.39 Membuat Folder

4. Kemudian, beri nama Countrinfoservices, lalu klik ok.



Gambar 6.40 Folder Countryinfoservices

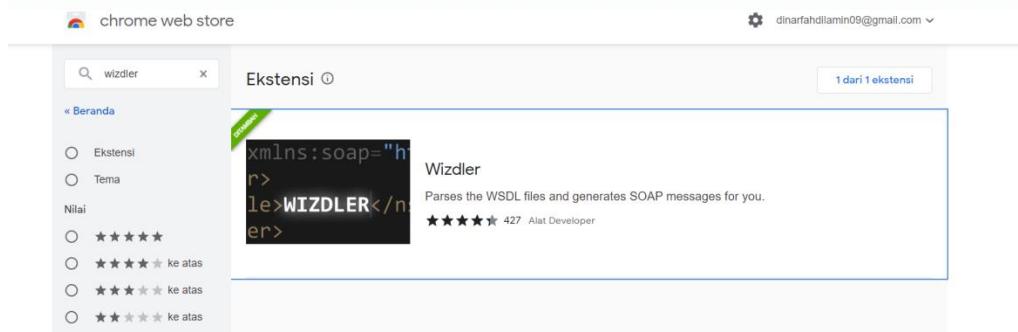
5. Pada folder Countrinfoservice, buat 1 web service request.



Gambar 6.41 Membuat Web Service Request

6. Karena kita akan mengambil beberapa permintaan dari script xml tadi, maka tambahkan extansi wizdler pada chrome. Extansi ini berguna untuk melewati

dokumen vestal seperti forma, sehingga permintaan yang di request tadi dapat diterima.



Gambar 6.42 Tambahkan Ekstansi Wizdler pada Chrome

7. Jika extensi sudah ditambahkan, maka akan muncul ikon seperti gambar dibawah ini.



Gambar 6.43 Ikon dari Ekstansi Wizdler

8. Berikut ini merupakan list request yang tersedia.



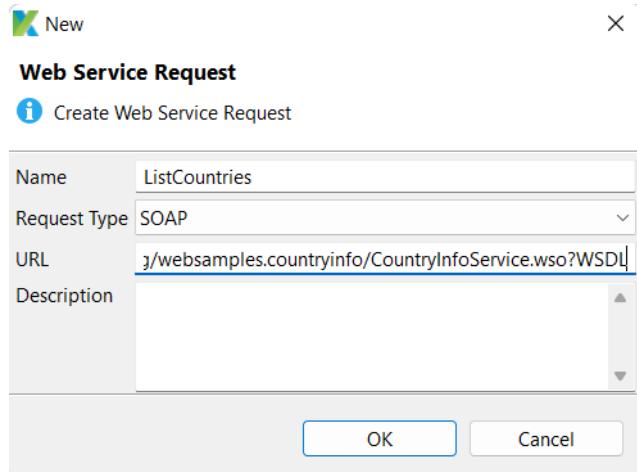
Gambar 6.44 List Request

9. Selanjutnya kita akan mengambil request dari ListOfCountryNamesByName, berikut tampilannya.



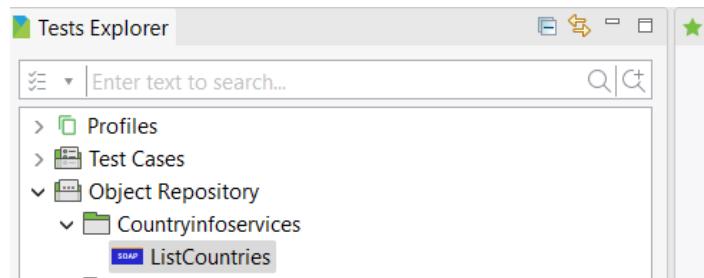
Gambar 6.45 Request dari ListOfCountryNamesByName

10. Buat web service request dengan nama ListCountries, dengan pilihan requestType adalah SOAP, dan masukan URL dari sample SOAP API tadi.



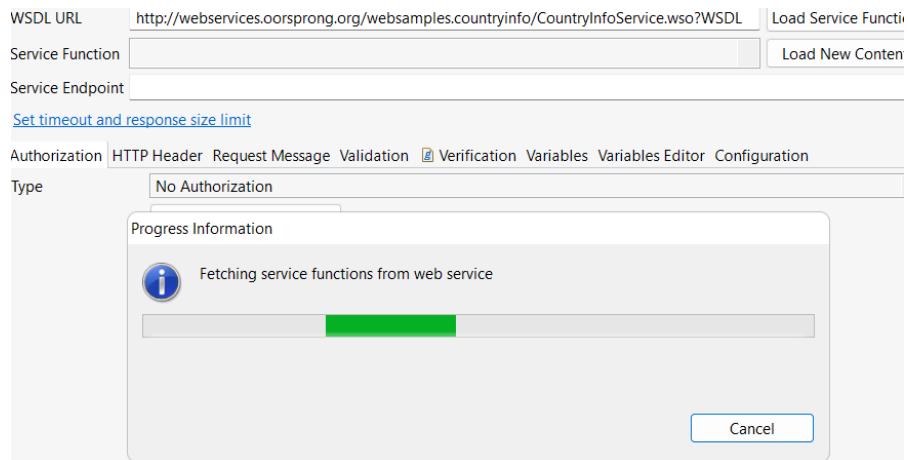
Gambar 6.46 Web Service Request dengan nama ListCountries

11. Dapat kita lihat web service request sudah terbentuk.



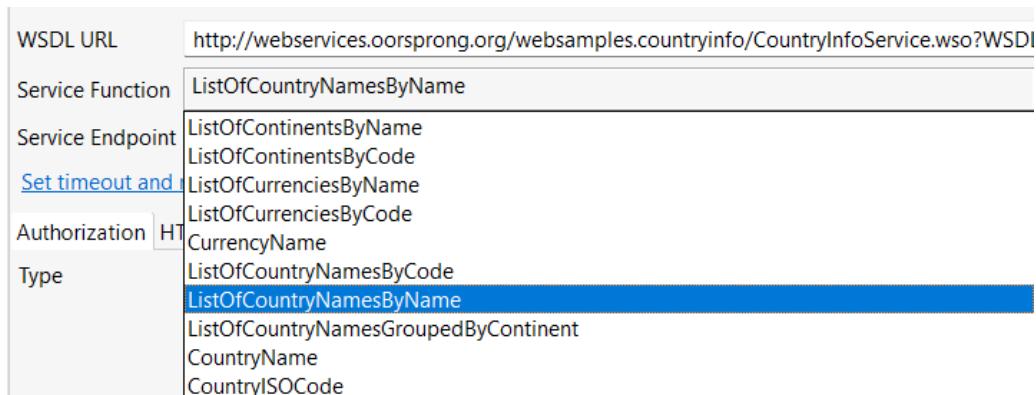
Gambar 6.47 List Countries Sudah Dibuat

12. Masukkan service function, sebelumnya klik load service function terlebih dahulu untuk melakukan pencarian terhadap service function yang tersedia.



Gambar 6.48 Load Service Function

13. Pilih ListOfCountryNamesByName untuk service functionnya.



Gambar 6.49 Service Function

14. Kemudian pada request message copi script yang ada pada url ListOfCountryNamesByName tadi.

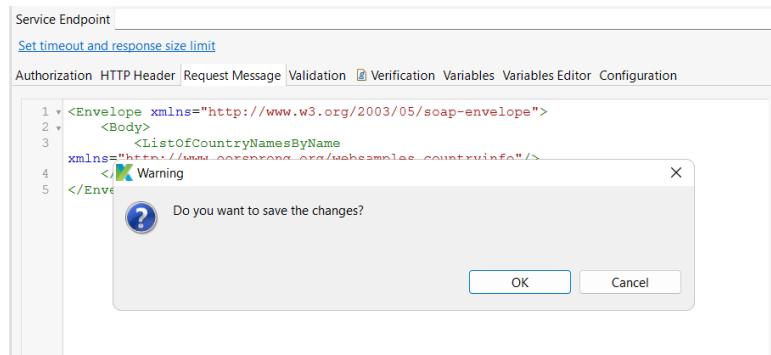
A screenshot of a code editor window titled 'Request Message'. The code is a SOAP envelope with the following XML:

```
1 <Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
2   <Body>
3     <ListOfCountryNamesByName
4       xmlns="http://www.orsprong.org/websamples.countryinfo"/>
5   </Body>
</Envelope>
```

The code is color-coded for syntax highlighting, with tags in green and attributes in blue.

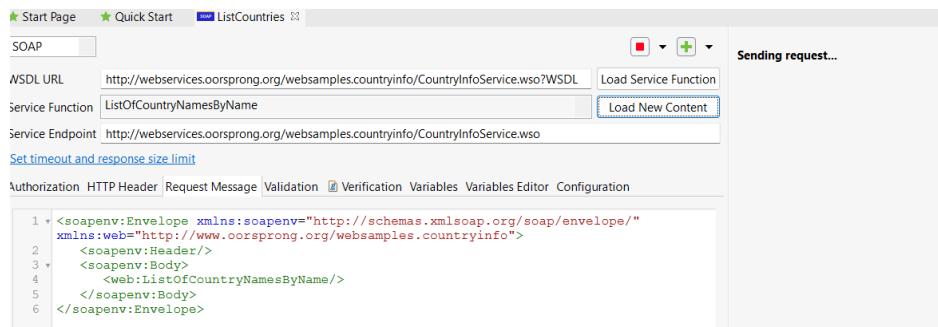
Gambar 6.50 Copy Script

15. Kemudian save script.



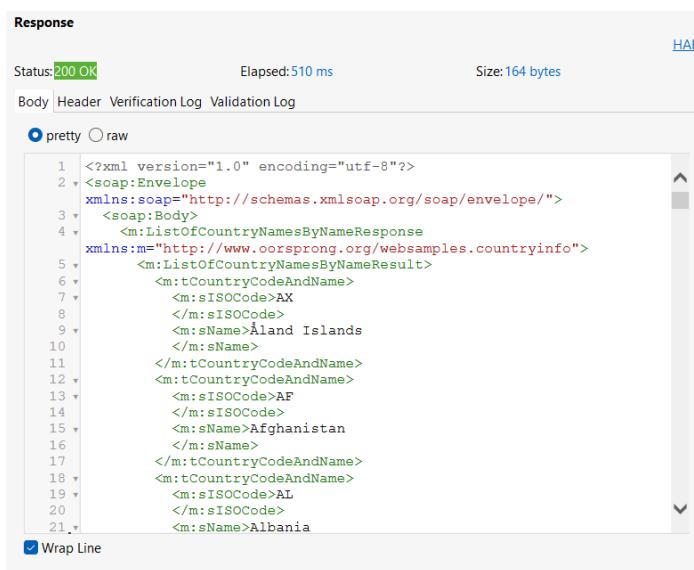
Gambar 6.51 Save Script

16. Selanjutnya jalankan script untuk melakukan request/permintaan.



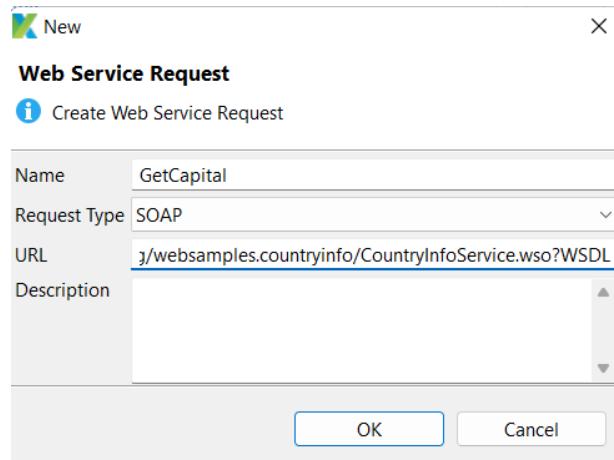
Gambar 6.52 Request Script

17. Berikut ini hasil dari request yang diberikan, dimana berisi list nama-nama dari berbagai negara.



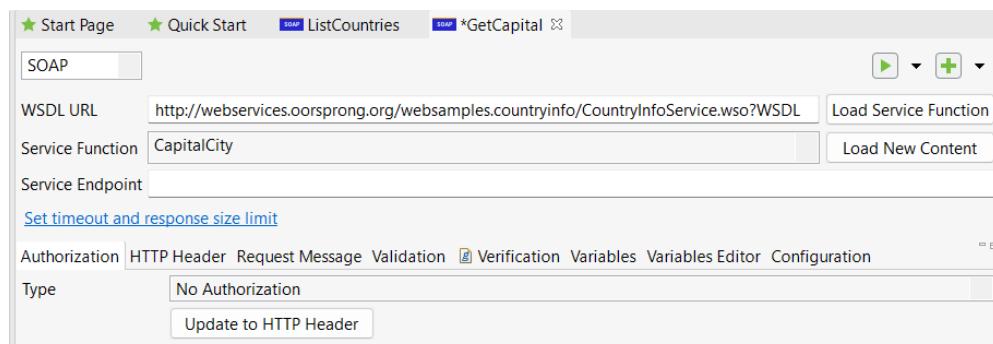
Gambar 6.53 List Nama dari Berbagai Negara

18. Buat web service request lainnya.



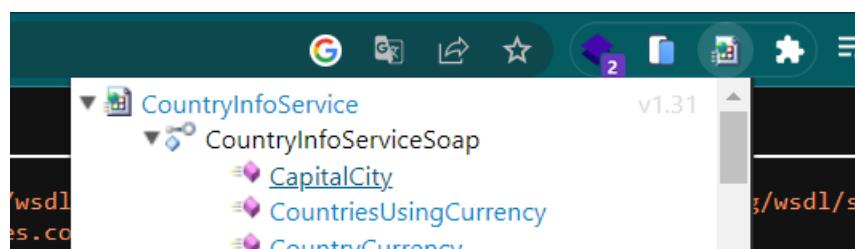
Gambar 6.54 Web Service Request SOAP

19. Pilih server function nya CapitalCity.



Gambar 6.55 Server Function

20. Pada extensi wizdler, pilih CapitalCity.



Gambar 6.56 Capital City

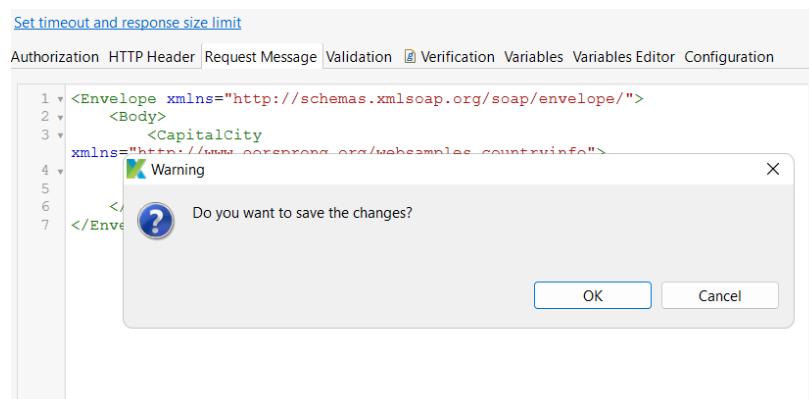
21. Berikut script dari CapitalCity.



```
POST http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <CapitalCity xmlns="http://www.oorsprong.org/websamples.countryinfo">
      <sCountryISOCode>[string]</sCountryISOCode>
    </CapitalCity>
  </Body>
</Envelope>
```

Gambar 6.57 Script CapitalCity

22. Save script.



Gambar 6.58 Save Script

23. Berikut ini hasil dari request yang diberikan, dengan menampilkan nama-nama ibukota dari tiap negara.



```
607 <m:sISOCode>IS
608 </m:sISOCode>
609 <m:sName>Iceland
610 </m:sName>
611 </m:tCountryCodeAndName>
612 <m:tCountryCodeAndName>
613 <m:sISOCode>IN
614 </m:sISOCode>
615 <m:sName>India
616 </m:sName>
617 </m:tCountryCodeAndName>
618 <m:tCountryCodeAndName>
619 <m:sISOCode>ID
620 </m:sISOCode>
621 <m:sName>Indonesia
622 </m:sName>
623 </m:tCountryCodeAndName>
624 <m:tCountryCodeAndName>
625 <m:sISOCode>IR
626 </m:sISOCode>
627 <m:sName>Iran
```

Gambar 6.59 Hasil Request

24. Kita juga dapat mengambil request berdasarkan variabel tertentu, seperti yang terlihat pada gambar di bawah ini terjadi perubahan script pada syntaks pemanggilan ISO code.

[Set timeout and response size limit](#)

Authorization HTTP Header Request Message Validation  Verification Variables Variables Editor Configuration

```
1 <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
2   <Body>
3     <CapitalCity
4       xmlns="http://www.oorsprong.org/websamples.countryinfo">
5         <sCountryISOCode>${CountryISOCode}</sCountryISOCode>
6       </CapitalCity>
7     </Body>
8   </Envelope>
```

Gambar 6.60 Mengambil Request Berdasarkan Variabel Tertentu

25. Kemudian pilih menu variabel dan klik add.

Gambar 6.61 Menu Variabel

26. Selanjutnya, pilih name yaitu CountryISOCODE, pilih type sebagai Global Variable.

No.	Name	Type	Default value	Description
1	CountryISO...	Global Vari...	GlobalVariable.null	

Gambar 6.62 Isi Tabel Variabel

27. Buka profiles, kemudian pilih default, isi value dengan ISO CODE yang ingin anda tampilkan, untuk kasus ini saya isi value dengan inisial ID.

Name	Value Type	Value	Description
CountryISO...	String	"IN"	

Name	Value	Description
CountryISOCode	'IN'	

Gambar 6.63 Isi Value

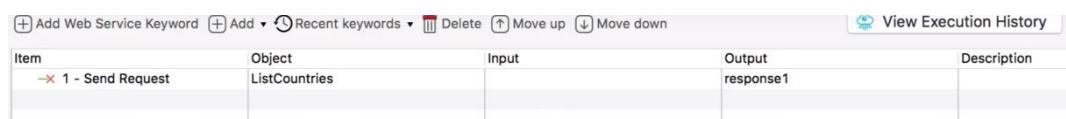
28. Berikut hasil request yang diberikan dari modifikasi sesuai dengan variabel ISO code dengan inisial ID.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <soap:Envelope
3   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
4     <soap:Body>
5       <m:CapitalCityResponse
6         xmlns:m="http://www.oorsprong.org/websamples.countryinfo">
7           <m:CapitalCityResult>Jakarta
8           </m:CapitalCityResult>
9         </m:CapitalCityResponse>
10      </soap:Body>
11    </soap:Envelope>
  
```

Gambar 6.64 Hasil Request yang Sudah Dimodifikasi

29. Selanjutnya kita akan membuat pengujian test case/kasus uji, terhadap kedua web service request. Anda dapat klik icon tambah, kemudian pilih add to new test case, kemudian beri dengan nama TestOne, lalu klik ok.



Gambar 6.65 Pengujian Test Case dari Kedua Web Service Request

30. Selanjutnya kita akan menyimpan output dari API ListCountries pada variabel yang disebut response1.



Gambar 6.66 Variabel Response1

31. Kemudian pergi ke script, dan anda dapat melihat script dari variabel response1 yang telah dibuat.

```

import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
import static com.kms.katalon.core.testcase.TestCaseFactory.findTestCase
import static com.kms.katalon.coretestdata.TestDataFactory.findTestData
import static com.kms.katalon.core.testobject.ObjectRepository.findTestObject
import com.kms.katalon.core.checkpoint.Checkpoint as Checkpoint
import com.kms.katalon.core.cucumber.keyword.CucumberBuiltinKeywords as CucumberKW
import com.kms.katalon.core.mobile.keyword.MobileBuiltInKeywords as Mobile
import com.kms.katalon.core.model.FailureHandling as FailureHandling
import com.kms.katalon.core.testcase.TestCase as TestCase
import com.kms.katalon.coretestdata.TestData as TestData
import com.kms.katalon.core.testobject.TestObject as TestObject
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
import internal.GlobalVariable as GlobalVariable

response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountries'))

```

Gambar 6.67 Script dari Variabel Response1

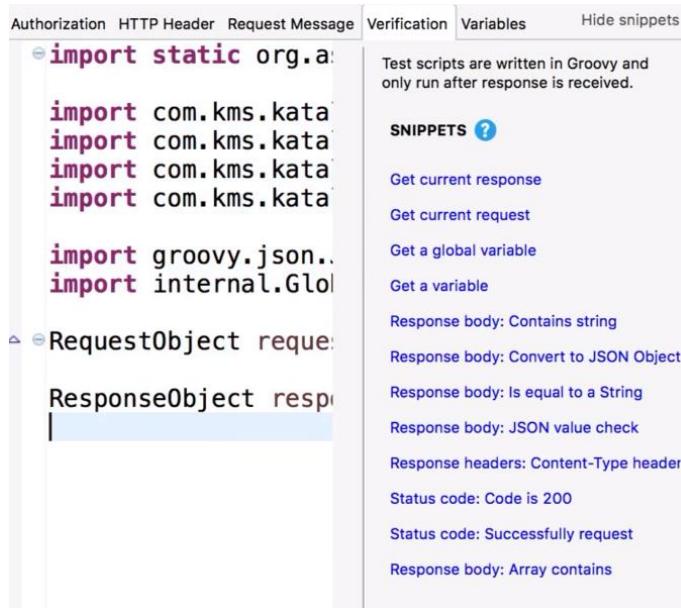
32. Sekarang kita akan melakukan fetch pada respon, dan mengambil nilai tertentu dari respon tersebut.

The screenshot shows the Katalon Studio interface with the following details:

- Tests Explorer:** Shows a tree structure with profiles, test cases, and object repositories.
- Request History:** Shows a recent request for "ListCountryNamesByName".
- SOAP:** Selected, with URL "http://webservices.oorsprong.org/websamples.countryinfo/".
- Service Function:** "ListOfCountryNamesByName".
- Response:**
 - Status: 200 OK
 - Elapsed: 464 ms
 - Size: 35 KB
- Body:** Displays the XML response body.
- Verification Log:** Shows a snippet of Groovy code for extracting data from the response.

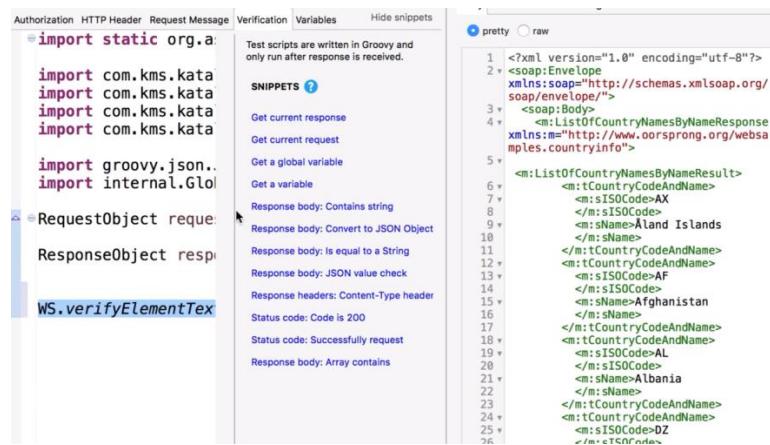
Gambar 6.68 Mengambil Nilai Tertentu dari Respon

33. Pada jendela verifikasi kita dapat melihat list cuplikan yang tersedia, dan dapat digunakan untuk cuplikan verifikasi.



Gambar 6.69 List Cuplikan

34. Untuk melihat pemeriksaan atau verifikasi nilai XML kita dapat melakukannya dengan mengarahkan kursor ke syntaks Iso Code, kemudian tekan **ctrl + case** pada keyboard, sehingga kita dapat melihat pada jendela show snipped , telah ada verifikasi khusus yang sudah dibuat.



Gambar 6.70 Verifikasi Nilai XML

35. Selanjutnya kita akan menuliskan kode untuk menyimpan nilai respon yang ada pada variabel response1 tadi, dimana nilainya kita tampung pada variabel xml1.

```
+ import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
  response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountries'))
  String xml1 = response1.responseBodyContent
```

Gambar 6.71 Variabel xml1

36. Kemudian parsing nilai respon tersebut ke dalam bentuk xml, dengan kode berikut, dimana hasil akan disimpan pada variabel `dataValue`.

```
+ import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
  response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountries'))
  String xml1 = response1.responseBodyContent
  def dataValue= new XmlSlurper().parseText(xml1)
```

Gambar 6.72 Parsing Nilai Response

37. Untuk mendapatkan nilai atau simpul tertentu, hasil parsing yang ada pada variabel `dataValue` tadi, kita simpulkan berdasarkan list country name by name, lalu simpan hasil pada variabel `countryCode`, seperti yang terlihat pada kode berikut.

```
+ import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
  response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountries'))
  String xml1 = response1.responseBodyContent
  def dataValue= new XmlSlurper().parseText(xml1)
  def countryCode = dataValue.ListOfCountryNamesByNameResult.tCountryCodeAndName[2]
```

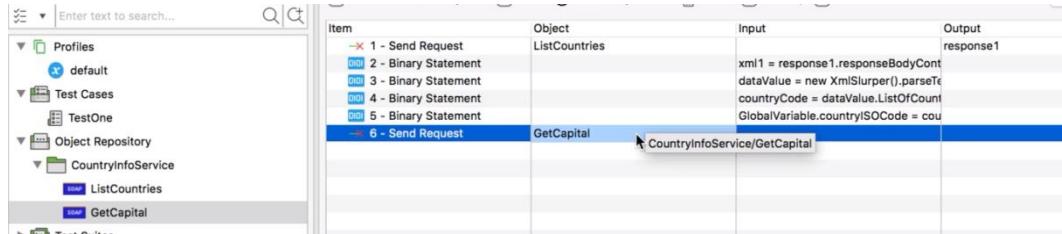
Gambar 6.73 Code dari Variabel Countrycode

38. Selanjutnya hasil pada variabel `countryCode` tadi kita simpan pada global variabel, dengan perintah seperti berikut.

```
+ import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
  response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountries'))
  String xml1 = response1.responseBodyContent
  def dataValue= new XmlSlurper().parseText(xml1)
  def countryCode = dataValue.ListOfCountryNamesByNameResult.tCountryCodeAndName[2]
  GlobalVariable.countryISOCode = countryCode
```

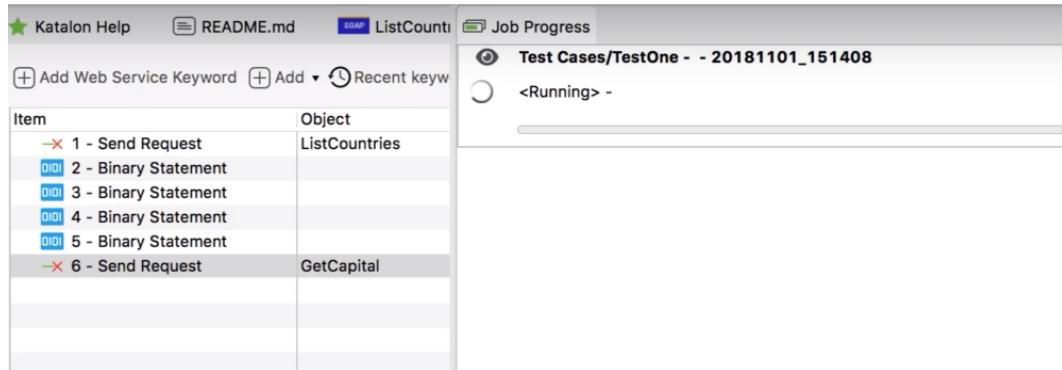
Gambar 6.74 Variabel Countrycode disimpan pada Variabel GlobalVariable

39. Setelah itu pergi ke jendela test case, dan panggil API GetCapital yang telah dibuat sebelumnya.



Gambar 6.75 Panggil API GetCapital

40. Lalu save dan jalankan, dan berikut hasil yang diberikan.



Gambar 6.76 Simpan dan Jalankan API GetCapital

41. Buka jendela console, dan memperlihatkan hasil verifikasi yang telah kita lakukan.

```

Problems Console Log Viewer
<terminated> TempTestCase1541065448450 [Katalon] /Applications/Katalon Studio 6.app/Contents/Eclipse/jre/Contents/Home/jre/bin/java (01-Nov-2018, 3:14:09 PM)
11-01-2018 03:14:13 PM - [START] - Start Test Case : Test Cases/TestOne
11-01-2018 03:14:13 PM - [INFO] - Evaluating variables for test case
11-01-2018 03:14:14 PM - [START] - Start action : Statement - response1 = com.kms.kal
11-01-2018 03:14:14 PM - [INFO] - Finding Test Object with id 'Object Repository/(
11-01-2018 03:14:14 PM - [INFO] - Checking request object
11-01-2018 03:14:16 PM - [PASSED] - Send request successfully
11-01-2018 03:14:16 PM - [END] - End action : Statement - response1 = com.kms.kal
11-01-2018 03:14:16 PM - [START] - Start action : Statement - xml1 = responseBodyContent
11-01-2018 03:14:17 PM - [END] - End action : Statement - xml1 = responseBodyContent
11-01-2018 03:14:17 PM - [START] - Start action : Statement - dataValue = new groov
11-01-2018 03:14:17 PM - [END] - End action : Statement - dataValue = new groovy.
11-01-2018 03:14:17 PM - [START] - Start action : Statement - countryCode = sISOCode
11-01-2018 03:14:17 PM - [END] - End action : Statement - countryCode = sISOCode.
11-01-2018 03:14:17 PM - [START] - Start action : Statement - countryISOCode = cou
11-01-2018 03:14:17 PM - [END] - End action : Statement - countryISOCode = counti
11-01-2018 03:14:17 PM - [START] - Start action : sendRequest
11-01-2018 03:14:17 PM - [INFO] - Finding Test Object with id 'Object Repository/(
11-01-2018 03:14:17 PM - [INFO] - Checking request object
11-01-2018 03:14:18 PM - [PASSED] - Send request successfully
11-01-2018 03:14:18 PM - [END] - End action : sendRequest
11-01-2018 03:14:18 PM - [PASSED] - Test Cases/TestOne
11-01-2018 03:14:18 PM - [END] - End Test Case : Test Cases/TestOne

```

Gambar 6.77 Hasil Verifikasi

42. Untuk melihat country code yang di ekstrak, dapat tambahkan code berikut.

```

import com.kms.katalon.core.testobject.TestObject as TestObject
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WebUI
import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WU
import internal.GlobalVariable as GlobalVariable

response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountryNamesByName'))

String xml1 = response1.responseBodyContent

def dataValue = new XmlSlurper().parseText(xml1)

def countryCode = dataValue.ListOfCountryNamesByNameResult.tCountryName

println("The extracted country code is : "+countryCode)

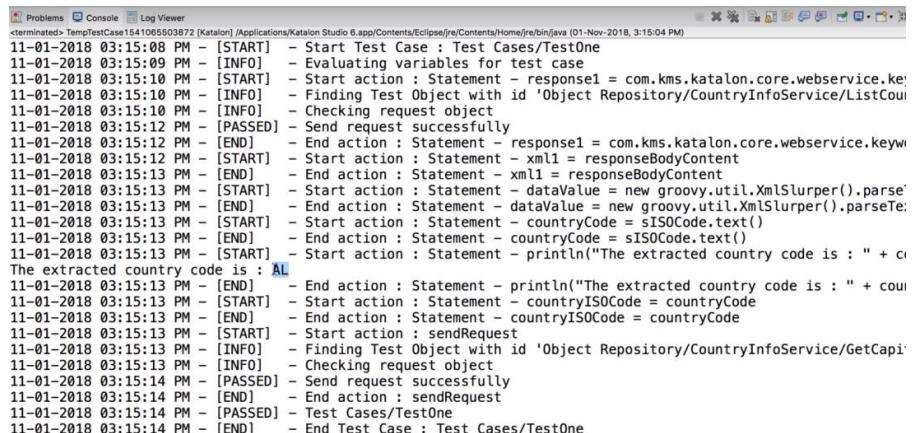
GlobalVariable.countryISOCode = countryCode

WS.sendRequest(findTestObject('CountryInfoService/GetCapital'))

```

Gambar 6.78 Ekstrak Country Code

43. Dapat dilihat berikut hasil dari country code yang di ekstrak, yaitu country code dengan kode negara AL.



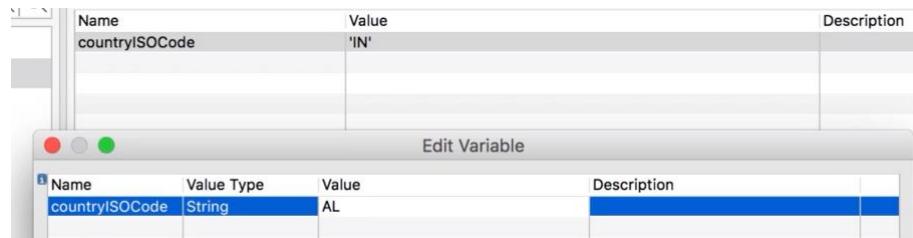
```

11-01-2018 03:15:08 PM - [START] - Start Test Case : Test Cases/TestOne
11-01-2018 03:15:09 PM - [INFO] - Evaluating variables for test case
11-01-2018 03:15:10 PM - [START] - Start action : Statement - response1 = com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WebUI
11-01-2018 03:15:10 PM - [INFO] - Finding Test Object with id 'Object Repository/CountryInfoService/ListCountryNamesByName'
11-01-2018 03:15:10 PM - [INFO] - Checking request object
11-01-2018 03:15:12 PM - [PASSED] - Send request successfully
11-01-2018 03:15:12 PM - [END] - End action : Statement - response1 = com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WebUI
11-01-2018 03:15:12 PM - [START] - Start action : Statement - xml1 = responseBodyContent
11-01-2018 03:15:13 PM - [END] - End action : Statement - xml1 = responseBodyContent
11-01-2018 03:15:13 PM - [START] - Start action : Statement - dataValue = new groovy.util.XmlSlurper().parseText(xml1)
11-01-2018 03:15:13 PM - [END] - End action : Statement - dataValue = new groovy.util.XmlSlurper().parseText(xml1)
11-01-2018 03:15:13 PM - [START] - Start action : Statement - countryCode = sISOCode.text()
11-01-2018 03:15:13 PM - [END] - End action : Statement - countryCode = sISOCode.text()
11-01-2018 03:15:13 PM - [START] - Start action : Statement - println("The extracted country code is : "+countryCode)
11-01-2018 03:15:13 PM - [END] - End action : Statement - println("The extracted country code is : "+countryCode)
11-01-2018 03:15:13 PM - [START] - Start action : Statement - countryISOCode = countryCode
11-01-2018 03:15:13 PM - [END] - End action : Statement - countryISOCode = countryCode
11-01-2018 03:15:13 PM - [START] - Start action : sendRequest
11-01-2018 03:15:13 PM - [INFO] - Finding Test Object with id 'Object Repository/CountryInfoService/GetCapital'
11-01-2018 03:15:13 PM - [INFO] - Checking request object
11-01-2018 03:15:14 PM - [PASSED] - Send request successfully
11-01-2018 03:15:14 PM - [END] - End action : sendRequest
11-01-2018 03:15:14 PM - [PASSED] - Test Cases/TestOne
11-01-2018 03:15:14 PM - [END] - End Test Case : Test Cases/TestOne

```

Gambar 6.79 Hasil Country Code yang Diekstrak

44. Untuk memvalidasi country code tersebut, pergi ke halaman default dan ubah value dari country code menjadi AL.



Name	Value	Description	
countryISOCode	'IN'		
Name	Type	Value	Description
countryISOCode	String	AL	

Gambar 6.80 Validasi Country Code

45. Lalu save dan jalankan, hasil memperlihatkan country code dengan inisial AL adalah negara Tirana.

```

1 <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  2   <Body>
  3     <CapitalCity xmlns="http://www.oorsprong.org/websamples.countryinfo">
  4       <sCountryISOCode>${countryISOCode}</sCountryISOCode>
  5     </CapitalCity>
  6   </Body>
  7 </Envelope>
```

```

1 <?xml version="1.0" encoding="utf-8"?>
  2 <soap:Envelope
  3   xmlns:soap="http://schemas.xmlsoap.org/soap/
  4   envelope/">
  5   <soap:Body>
  6     <CapitalCityResponse
  7       xmlns:m="http://www.oorsprong.org/websamples.
  8       countryinfo">
  9       <m:CapitalCityResult>Tirana
  10      </m:CapitalCityResult>
  11    </CapitalCityResponse>
  12  </soap:Body>
  13 </soap:Envelope>
```

Gambar 6.81 Memperlihatkan Hasil Country Code

46. Kembali ke halaman verifikasi, arahkan kursor ke “Tirana” lalu tekan ctrl + case pada keyboard, dan dapat dilihat code verifikasi telah terbentuk.

```

@import static org.assertj.core.api.Assertions
import com.kms.katalon.core.testobject.RequestObject
import com.kms.katalon.core.testobject.ResponseObject
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
import groovy.json.JsonSlurper
import internal.GlobalVariable as GlobalVariable
```

```

RequestObject request = WSResponseManager.get('GetCapital')
ResponseObject response = WSResponseManager.get('GetCapital')
```

```

WS.verifyElementText(response, 'CapitalCityResult', 'Tirana')
```

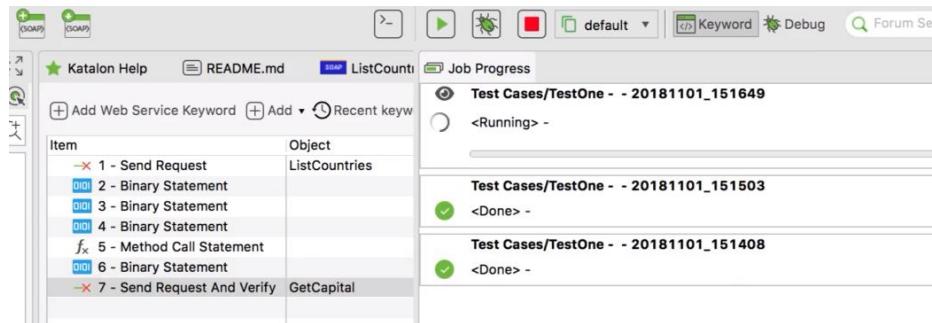
Gambar 6.82 Code Verifikasi

47. Selanjutnya masuk ke halaman test case, lalu update item dengan “send request and verify”. Hal ini dilakukan untuk memverifikasi permintaan/request dari API yang dilakukan.

Item	Object	Input	Output
1 - Send Request	ListCountries		response1
2 - Binary Statement		xml1 = response1.responseBodyContent	
3 - Binary Statement		dataValue = new XmlSlurper().parseText(xml1)	
4 - Binary Statement		countryCode = dataValue.ListOfCountry[0].countryCode	
5 - Method Call Statement		println("The extracted country code is: " + countryCode)	
6 - Binary Statement		GlobalVariable.countryISOCode = countryCode	
Send Request And Verify	GetCapital		

Gambar 6.83 Memverifikasi Request

48. Kemudian save script, dan jalankan test case.



Gambar 6.84 Simpan Script dan Jalankan Test Case

49. Setelah proses running selesai dan berhasil, pergi ke halaman log viewer dan kita dapat melihat verifikasi yang sudah berhasil dan berfungsi dengan baik.

Test Cases/TestOne (6.095s)		Name:	verifyElementText
		Start:	11-01-2018 03:16:59 PM
		Elapsed time:	0.080s
		Message:	Verify element text successfully
<pre>Evaluating variables for test case 1 - Statement - response1 = com.kms.katalon.core.webservice.keyword.WSBuiltinKeywords.sendRequestAndVerify 2 - Statement - xml1 = responseBodyContent (0.704s) 3 - Statement - dataValue = new groovy.util.XmlSlurper().parseText(xml1) (0.082s) 4 - Statement - countryCode = sISOCode.text() (0.048s) 5 - Statement - println("The extracted country code is: " + countryCode) (0.011s) 6 - Statement - countryISOCode = countryCode (0.024s) 7 - sendRequestAndVerify (1.524s) Finding Test Object with id 'Object Repository/CountryInfoService/GetCapital' Checking request object Start verifying response Verification (0.632s) 1 - Statement - request = com.kms.katalon.core.webservice.verification.WSResponseManager.getInfo() 2 - Statement - response = com.kms.katalon.core.webservice.verification.WSResponseManager.getInfo() 3 - verifyElementText (0.080s)</pre>			

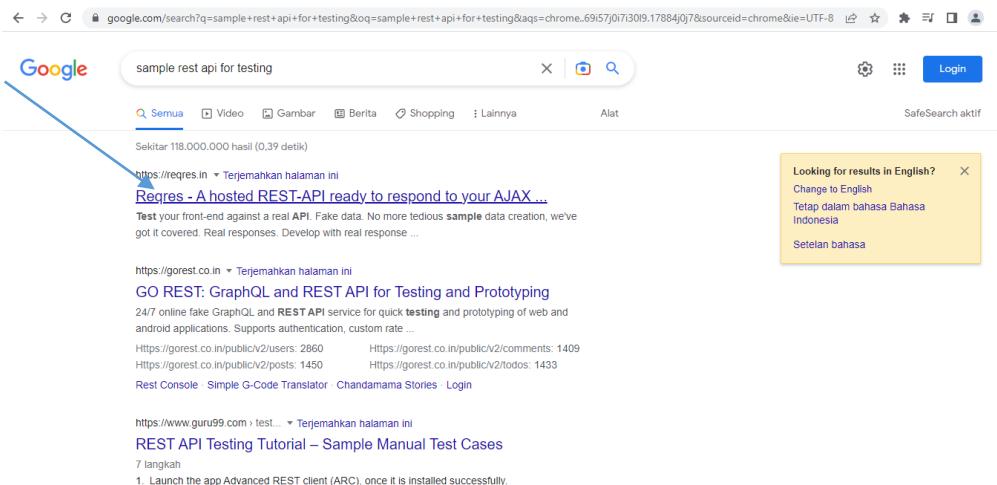
Gambar 6.85 Verifikasi yang Sudah Berhasil

c. API CHAINING | REST - JSON | HOW TO SEND VALUE FROM ONE API TO OTHER

Setelah rangkaian pengujian atau kumpulan rangkaian pengujian, sesi terakhir kita telah melihat cara mengek nilai kita dari respon XML kita dan kemudian memberikan nilai itu dalam permintaan API, selanjutnya kita akan

menggunakan API Rest dan melihat bagaimana melakukannya dengan respons Json. Berikut tahapan-tahapannya.

1. Setelah membuka katalon studio. Selanjutnya kita akan mencari sampel sisa untuk pengujian dan disini kita memiliki sampel eqres di web



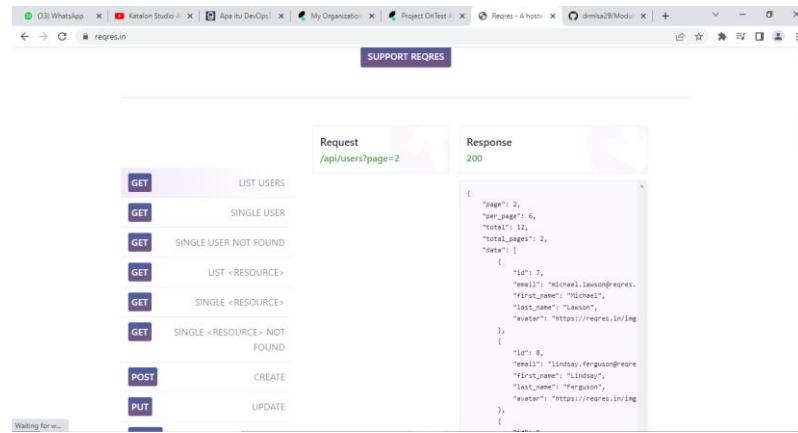
Gambar 6.86 Sample Rest API

2. Situs web ini memiliki api istirahat yang baik untuk pengujian dan disini kita dapat melihat ini memiliki api palsu sehingga kita dapat menggunakan untuk pengujian dan yang paling pasti berlaku kita akan membuat url ini tersedia sehingga kita dapat merujuk mereka menjadi



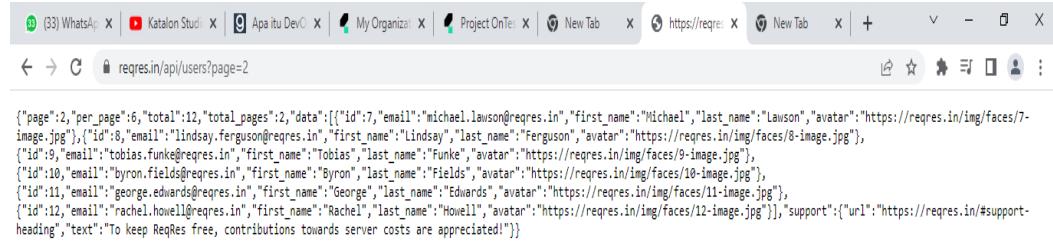
Gambar 6.87 URL Request

3. Disini bisa kita lihat jika kita turun kita akan memiliki banyak aplikasi dan kita telah mendapatkan posting yang dihapus batch dan seterusnya kita akan menggunakan daftar ini, pengguna mendapatkan API, jadi kita akan menyalin url ini dan meletakkan nya di tab baru.



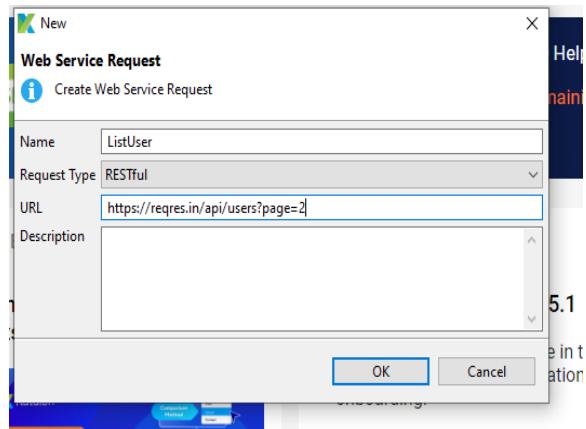
Gambar 6.88 Salin URL

- Menyalin teks url reqres.in dan menyalin sisa titik akhir dari sini dan menambahkan nya lagi, dan ini adalah titik terakhir API jika kita menjalankannya kita dapat melihat respon. Jadi saya akan menyalin url ini



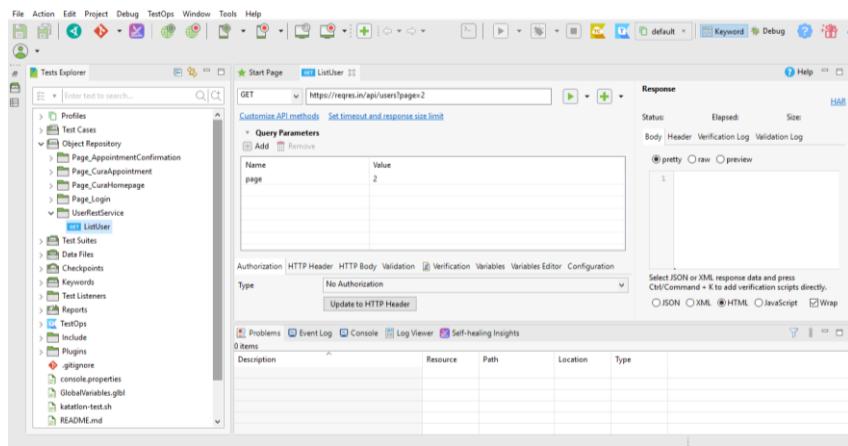
Gambar 6.89 Respon dari Teks URL reqres.in

- Setelah menyalin url, kemudian buka katalon studio dan buat folder baru pada object repository dengan nama UserRestService dan memasukkan link url ke web service request



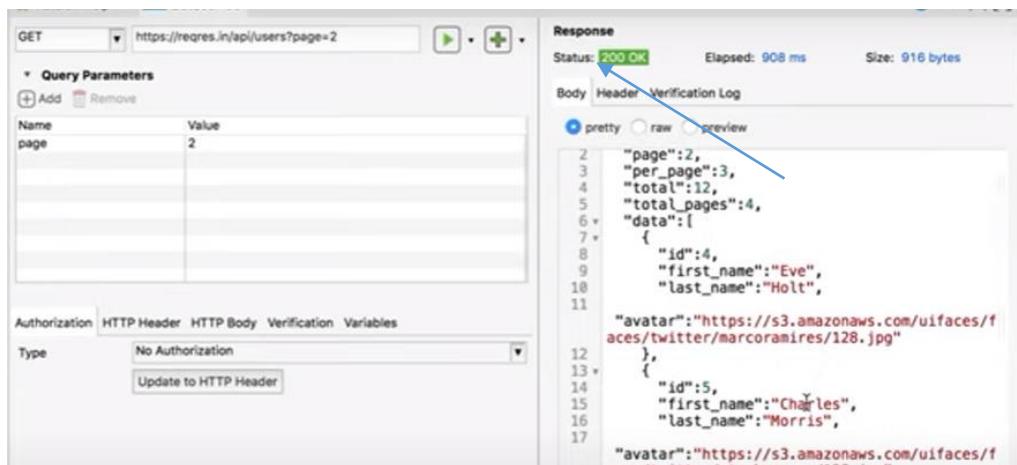
Gambar 6.90 Web Service Request pada reqres.in

6. Dan ini hasil setelah kita masukkan link url dan tidak ada otorsasi yang diperlukan untuk contoh API khusus ini, jika kita memerlukan otorisasi kita dapat mengedit di verifikasi dan dalam verifikasi kita dapat menambahkan beberapa cuplikan dan variable



Gambar 6.91 Contoh API

7. Dan kita jalankan apakah sesuai dan kita akan melihat apa yang harus dilakukan kemudian kita memeriksa apakah kita mendapatkan respons yang valid. Dan hasil nya seperti pada gambar dibawah dengan status 200 ok. Dan kita juga mendapatkan daftar pengguna dengan API ini dan pengujian ini berfungsi dengan baik.



Gambar 6.92 Hasil Pengujian API-CHAINING

DAFTAR PUSTAKA

There are no sources in the current document.