

**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
<b>KATA PENGANTAR .....</b>	<b>ii</b>
<b>DAFTAR ISI .....</b>	<b>iv</b>
<b>DAFTAR GAMBAR.....</b>	<b>vii</b>
<b>BAB 1 PENDAHULUAN .....</b>	<b>1</b>
1.1. Tujuan .....	1
1.2. Dasar Teori.....	1
1.3. Tahap-tahap Instalasi Katalon .....	2
<b>BAB 2 MULAI MEMBUAT PROJEK.....</b>	<b>13</b>
2.1. Tujuan .....	13
2.2. Membuat Projek Katalon.....	13
2.2.1. <i>Record and Replay</i> .....	13
2.2.2. <i>Manual Mode</i> .....	30
2.2.3. <i>Script Mode</i> .....	38
<b>BAB 3 TEST SUITE .....</b>	<b>43</b>
3.1. Tujuan .....	43
3.2. Dasar Teori.....	43
3.3. Percobaan.....	44
<b>BAB 4 MENGAMBIL DATA CSV.....</b>	<b>53</b>
4.1. Tujuan .....	53
4.2. Dasar Teori.....	53
4.3. Percobaan.....	53
<b>BAB 5 RESULT AND REPORTING .....</b>	<b>59</b>
5.1. Tujuan .....	59
5.2. Dasar Teori.....	59
5.3. Percobaan.....	59
5.3.1. Test Execution Reports .....	59
5.3.2. Basic Reports Plugins.....	64
5.3.3. How To Email Results .....	68
5.3.4. How To Use Katalon Analytics Step by Step .....	74
<b>BAB 6 MENJALANKAN TEST MENGGUNAKAN COMMANDLINE DAN PLUGIN JENKINS PADA KATALON STUDIO .....</b>	<b>78</b>
6.1. Tujuan .....	78

6.2. Dasar Teori.....	78
6.3. Percobaan.....	79
<b>BAB 7 INTEGRATION WITH GIT AND JENKINS .....</b>	<b>84</b>
7.1. Tujuan .....	84
7.2. Dasar Teori.....	84
7.3. Percobaan Integrasi Dengan GIT .....	84
7.4. Percobaan Integrasi Dengan Jenkins .....	94
<b>BAB 8 API TESTING .....</b>	<b>99</b>
8.1. Tujuan .....	99
8.2. Dasar Teori.....	99
8.3. Percobaan API Testing .....	99
<b>BAB 9 MOBILE TEST MENGGUNAKAN KATALON STUDIO .....</b>	<b>132</b>
9.1. Tujuan .....	132
9.2. Dasar Teori.....	132
9.3. Percobaan.....	132
<b>BAB 10 MEMBUAT ENVIRONMENT .....</b>	<b>143</b>
10.1. Tujuan .....	143
10.2. Dasar Teori.....	143
10.3. Percobaan.....	143
<b>BAB 11 CUSTOM KEYWORD.....</b>	<b>150</b>
11.1. Tujuan .....	150
11.2. Dasar Teori.....	150
11.3. Percobaan.....	150
<b>BAB 12 WINDOWS DEKSTOP APPLICATION TESTING .....</b>	<b>157</b>
12.1. Tujuan .....	157
12.2. Dasar Teori.....	157
12.3. Percobaan.....	157
<b>BAB 13 CARA MENGGUNAKAN PLUGIN XPATH DAN PLUGIN VISUAL VALIDATION .....</b>	<b>168</b>
13.1. Tujuan .....	168
13.2. Dasar Teori.....	168
13.3. Langkah-langkah dalam Menggunakan Plugin .....	171
<b>BAB 14 BDD CUCUMBER TEST.....</b>	<b>199</b>
14.1. Tujuan .....	199
14.2. Landasan Teori.....	199

14.2.1. <i>Example</i> .....	199
14.2.2. Bagaimana cara ini dapat bekerja? .....	200
14.2.3. Keuntungan Cucumber dibandingkan dengan <i>tools</i> lainnya? .....	201
14.3. Pembahasan.....	202
14.3.1. Membuat BDD Cucumber tests di Katalon Studio .....	202
14.3.2. How To Create Cucumber Runner .....	246
14.4. Penutupan.....	249
14.4.1. Simpulan .....	249
<b>DAFTAR PUSTAKA .....</b>	<b>250</b>

## DAFTAR GAMBAR

Gambar 1.1 Jendela <i>Run</i> .....	2
Gambar 1.2 Detail spesifikasi laptop.....	2
Gambar 1.3 Tampilan <i>landing page</i> www.katalon.com .....	3
Gambar 1.4 <i>Free Plan</i> Katalon .....	3
Gambar 1.5 Tampilan halaman <i>register</i> .....	3
Gambar 1.6 Halaman <i>Verify Email</i> .....	4
Gambar 1.7 Email aktivasi akun Katalon .....	4
Gambar 1.8 Halaman jika akun telah teraktivasi .....	4
Gambar 1.9 Halaman pengisian data profesi .....	5
Gambar 1.10 <i>Pop up toturial</i> pengenalan <i>dashboard</i> Katalon .....	5
Gambar 1.11 <i>Pop up download</i> Katalon Studio .....	6
Gambar 1.12 <i>Download</i> Katalon Studio.....	6
Gambar 1.13 <i>Extract</i> file katalon .....	7
Gambar 1.14 <i>Directory</i> Katalon Studio.....	7
Gambar 1.15 Jendela <i>Windows Defender Firewall</i> .....	8
Gambar 1.16 Jendela <i>cloning repository</i> git.....	8
Gambar 1.17 <i>Activate</i> Katalon Studio dengan akun .....	9
Gambar 1.18 Tampilan awal Katalon Studio .....	9
Gambar 1.19 Tampilan jika berhasil <i>Login</i> .....	9
Gambar 2.1 <i>Create a New Project</i> .....	14
Gambar 2.2 <i>New Katalon Project</i> .....	14
Gambar 2.3 Proses pembuatan projek baru .....	14
Gambar 2.4 Integrasi project TestOps.....	15
Gambar 2.5 Hasil setelah membuat projek.....	15
Gambar 2.6 Membuat <i>Test Case</i> .....	15
Gambar 2.7 Memberi nama <i>Test Case</i> .....	16
Gambar 2.8 Tampilan <i>Test Case</i> .....	16
Gambar 2.9 <i>Tool Record Web</i> .....	16
Gambar 2.10 Jendela <i>Web Recorder</i> .....	17
Gambar 2.11 <i>Install Extension</i> .....	17
Gambar 2.12 <i>Extension Katalon Recorder</i> .....	17
Gambar 2.13 Tampilan Website OpenSource-Demo .....	18
Gambar 2.14 Percobaan pada tombol <i>Search</i> .....	18
Gambar 2.15 Menampilkan <i>alert</i> .....	19
Gambar 2.16 Menambahkan item objek.....	19
Gambar 2.17 Mengatur objek dari item.....	20
Gambar 2.18 <i>Stop Recording</i> .....	20
Gambar 2.19 <i>Captured Objects</i> .....	20
Gambar 2.20 Detail item .....	21
Gambar 2.21 <i>Save Script Recording</i> .....	21
Gambar 2.22 Menyimpan Elemen ke <i>Object Repository</i> .....	21
Gambar 2.23 Membuat Folder Test2.....	22

Gambar 2.24 Menyimpan ke folder Test2 .....	22
Gambar 2.25 Hasil folder Test2 .....	23
Gambar 2.26 <i>Run in Chrome Browser</i> .....	23
Gambar 2.27 Proses <i>Running</i> .....	23
Gambar 2.28 <i>Failed WebDriver</i> .....	24
Gambar 2.29 <i>Update WebDriver</i> .....	24
Gambar 2.30 Hasil <i>Running Login</i> .....	25
Gambar 2.31 <i>Running Dashboard</i> .....	25
Gambar 2.32 <i>End Running</i> .....	26
Gambar 2.33 <i>Log Viewer Record Test</i> .....	26
Gambar 2.34 <i>Object Repository Test2</i> .....	27
Gambar 2.35 <i>Settings element</i> .....	27
Gambar 2.36 Fitur <i>Spy Web</i> .....	27
Gambar 2.37 Jendela <i>Spy Web</i> .....	28
Gambar 2.38 <i>Capture Login Button</i> .....	28
Gambar 2.39 <i>Object hasil capture</i> .....	29
Gambar 2.40 <i>Captured Objects</i> .....	29
Gambar 2.41 Jendela <i>Add Element to Object Repository</i> .....	30
Gambar 2.42 <i>Katalon Toolbar</i> .....	31
Gambar 2.43 Jendela <i>Spy Web</i> .....	31
Gambar 2.44 <i>Spy Web Running</i> .....	31
Gambar 2.45 <i>Capture elemen yang diperlukan</i> .....	32
Gambar 2.46 Hasil <i>Capture Login Page</i> .....	32
Gambar 2.47 <i>Login as Admin</i> .....	32
Gambar 2.48 <i>Capture Picture</i> .....	33
Gambar 2.49 <i>Capture Menu Time</i> .....	33
Gambar 2.50 Hasil <i>Capture Website</i> .....	33
Gambar 2.51 <i>Save Captured Object to Test3</i> .....	33
Gambar 2.52 Isi folder Test3.....	34
Gambar 2.53 <i>Create a New Test Case</i> .....	34
Gambar 2.54 SampleTestCase3 .....	34
Gambar 2.55 <i>Add Web UI Keyword</i> .....	35
Gambar 2.56 <i>Create Open Browser Item</i> .....	35
Gambar 2.57 Mengisi nilai/ <i>value</i> .....	35
Gambar 2.58 <i>Create Set Text Item</i> .....	35
Gambar 2.59 <i>Add Object to Set Text</i> .....	35
Gambar 2.60 Mengisi nilai <i>username</i> .....	36
Gambar 2.61 <i>Create a New Set Text Item</i> .....	36
Gambar 2.62 <i>Add Value as Password</i> .....	36
Gambar 2.63 <i>Add Click Item</i> .....	36
Gambar 2.64 <i>Add Close Browser Item</i> .....	36
Gambar 2.65 <i>Log Viewer Result for Manual Mode</i> .....	37
Gambar 2.66 <i>Toolbar Project</i> .....	37
Gambar 2.67 <i>Setting Test Case Failure</i> .....	37
Gambar 2.68 Tampilan <i>Script Mode</i> .....	38
Gambar 2.69 <i>Captured Objects from rzq-perpus.epizy.com</i> .....	39
Gambar 2.70 Simpan seluruh elemen ke folder Test4 .....	39

Gambar 2.71 <i>Create a New Test Case</i> .....	40
Gambar 2.72 <i>Open Script Tool</i> .....	40
Gambar 2.73 <i>Planning Script Path</i> .....	40
Gambar 2.74 <i>Open Browser Script</i> .....	41
Gambar 2.75 <i>Set Text Script</i> .....	41
Gambar 2.76 <i>Click to Login Button</i> .....	41
Gambar 2.77 <i>Click to Image</i> .....	41
Gambar 2.78 <i>Close Browser Script</i> .....	41
Gambar 2.79 <i>Test Script Mode on Running</i> .....	41
Gambar 2.80 <i>Test Success Running</i> .....	42
Gambar 2.81 <i>Notation of Disable Script</i> .....	42
Gambar 3.1 test suite.....	43
Gambar 3.2 test suite paraller.....	44
Gambar 3.3 Tampilan pemilihan langkah pertama.....	44
Gambar 3.4 Tampilan atur nama test suite .....	45
Gambar 3.5 Tampilan hasil setelah membuat test suite.....	45
Gambar 3.6 Tampilan test case browser.....	45
Gambar 3.7 Tampilan hasil setelah masuk test case ke test suite .....	46
Gambar 3.8 Tampilan browser.....	46
Gambar 3.9 Tampilan proses berjalan.....	46
Gambar 3.10 Tampilan <i>website orangehrmlive</i> .....	46
Gambar 3.11 Tampilan hasil test suite .....	47
Gambar 3.12 Tampilan detail test suite .....	47
Gambar 3.13 Tampilan hasil test suite di folder.....	47
Gambar 3.14 Tampilan hasil laporan test suite .....	47
Gambar 3.15 Tampilan daftar laporan.....	48
Gambar 3.16 Tampilan menambahkan test case pada test suite .....	48
Gambar 3.17 Tampilan file sample test ke dalam folder login .....	48
Gambar 3.18 Tampilan test suite collection .....	49
Gambar 3.19 Tampilan create new test suite .....	49
Gambar 3.20 Tampilan hasil test suite collection.....	49
Gambar 3.21 Tampilan test suite collection browser.....	50
Gambar 3.22 Tampilan hasil setelah test suite collection dirun.....	50
Gambar 3.23 Tampilan proses berjalan.....	50
Gambar 3.24 Tampilan hasil browser .....	51
Gambar 3.25 Tampilan isi dari Log Viewer.....	51
Gambar 3.26 Tampilan pengujian menggunakan Test Suite Collection .....	51
Gambar 3.27 Tampilan isi dari file laporan test suite collection .....	52
Gambar 3.28 Tampilan show detail dibrowser chrome .....	52
Gambar 3.29 Tampilan show detail dibrowser firebox .....	52
Gambar 4.1 File Excel .....	53
Gambar 4.2 Add Variabels.....	53
Gambar 4.3 Hasil Variables .....	54
Gambar 4.4 Mengubah Value .....	54
Gambar 4.5 Object Password .....	54
Gambar 4.6 Save.....	54
Gambar 4.7 Create New Suite .....	55

Gambar 4.8 Add Test 2 .....	55
Gambar 4.9 Show Data Binding.....	56
Gambar 4.10 Variabel Binding .....	56
Gambar 4.11 <i>New Test Data</i> .....	56
Gambar 4.12 Test Explore .....	56
Gambar 4.13 Tahap Browser .....	57
Gambar 4.14 File xlsx .....	57
Gambar 4.15 Test Data .....	57
Gambar 4.16 Proses Variabel Binding .....	58
Gambar 4.17 Proses menjalankan .....	58
Gambar 4.18 Hasil Pengujian.....	58
Gambar 5.1 Menu log view pada katalon.....	59
Gambar 5.2 Tes yang dijalankan.....	60
Gambar 5.3 Menu filter log.....	60
Gambar 5.4 Tampilan hirarki log .....	60
Gambar 5.5 Pengaturan format report .....	61
Gambar 5.6 Pilih menu report .....	61
Gambar 5.7 Report yang sudah dipilih.....	61
Gambar 5.8 Tampilan jendela report.....	62
Gambar 5.9 Tampilan test case table.....	62
Gambar 5.10 Tampilan tab summary .....	62
Gambar 5.11 Tampilan tab execution setting .....	63
Gambar 5.12 Tampilan tab execution environtment.....	63
Gambar 5.13 Menu test case detail .....	63
Gambar 5.14 Test case dialog .....	64
Gambar 5.15 Tampilan test case suite .....	64
Gambar 5.16 Detail test case suite .....	64
Gambar 5.17 Tampilan menu project .....	65
Gambar 5.18 Menu pada project setting.....	65
Gambar 5.19 List pengujian yang telah dibuat .....	65
Gambar 5.20 Open containing folder .....	66
Gambar 5.21 Lokasi folder disimpan .....	66
Gambar 5.22 Hasil pengujian.....	66
Gambar 5.23 Laporan pengujian dalam format HTML .....	67
Gambar 5.24 Laporan pengujian dalam format CSV .....	67
Gambar 5.25 Laporan di ekspor dalam format PDF.....	68
Gambar 5.26 Pilih lokasi penyimpanan file .....	68
Gambar 5.27 Pilih menu project setting pada toolbar.....	69
Gambar 5.28 Pilih template email .....	69
Gambar 5.29 Tampilan mail server setting .....	69
Gambar 5.30 Pengisian Email pada mail server setting .....	70
Gambar 5.31 Pilih apply .....	70
Gambar 5.32 Tampilan saat proses berjalan.....	70
Gambar 5.33 Tampilan proses gagal (error).....	71
Gambar 5.34 Tampilan menu security .....	71
Gambar 5.35 Tampilan jendela app password.....	71
Gambar 5.36 Pilih generate .....	72

Gambar 5.37 Tampilan halaman App password.....	72
Gambar 5.38 Proses berhasil .....	72
Gambar 5.39 Tampilan report format.....	73
Gambar 5.40 Mengubah template test Suite.....	73
Gambar 5.41 Halaman Execution Infromation.....	73
Gambar 5.42 Notifikasi file PDF masuk pada email .....	74
Gambar 5.43 Pilihan untuk memulai project baru.....	74
Gambar 5.44 Open project .....	75
Gambar 5.45 Halaman test activities.....	75
Gambar 5.46 Pilihan test Suite.....	76
Gambar 5.47 Test Suite yang sudah ditambahkan.....	76
Gambar 5.48 Test Suite dijalankan menggunakan Browser .....	76
Gambar 5.49 Test Suite sedang di eksekusi .....	77
Gambar 5.50 Tampilan hasil detail report .....	77
Gambar 6.1 Melihat Versi Java Pada CMD .....	79
Gambar 6.2 Mengenerate Command Pada Aplikasi Katalon .....	80
Gambar 6.3 Copy to Clipboard Generated Command.....	81
Gambar 6.4 Path Instalasi Folder Pada CMD.....	81
Gambar 6.5 Menempel Perintah Generated Command .....	81
Gambar 6.6 Mengganti Path Ke File Runtime Engine .....	82
Gambar 6.7 Testing Website Otomatis Pada Browser Chrome .....	82
Gambar 6.8 Output Testing Pada CMD .....	83
Gambar 7.1 Membuat <i>repository</i> baru .....	85
Gambar 7.2 Pengaturan <i>repository</i> .....	85
Gambar 7.3 Membuat projek di <i>Dashboard</i> Katalon .....	86
Gambar 7.4 Memilih projek tim.....	86
Gambar 7.5 Memberi nama projek.....	86
Gambar 7.6 Hasil pembuatan projek .....	86
Gambar 7.7 Koneksi dengan <i>repository</i> GIT .....	87
Gambar 7.8 Meng-copy link untuk <i>clone</i> repo .....	87
Gambar 7.9 Copy API token .....	88
Gambar 7.10 Membuka projek di Katalon Studio.....	88
Gambar 7.11 Memilih projek dengan API token.....	88
Gambar 7.12 Melakukan <i>commit</i> projek .....	88
Gambar 7.13 <i>Commit</i> projek Katalon Studio .....	89
Gambar 7.14 Terdapat <i>Pull Request</i> di Repo GitHub.....	89
Gambar 7.15 Membuat <i>test case</i> .....	90
Gambar 7.16 <i>Commit test case</i> .....	90
Gambar 7.17 Hasil pada repo GitHub .....	90
Gambar 7.18 Isi folder Test Case .....	91
Gambar 7.19 Koneksi Katalon dengan GitHub .....	91
Gambar 7.20 <i>Open Project</i> .....	92
Gambar 7.21 Hasil <i>Open Project</i> .....	92
Gambar 7.22 Membuat <i>test case</i> baru .....	92
Gambar 7.23 <i>Commit</i> perubahan projek.....	93
Gambar 7.24 Hasil perubahan repo GitHub .....	93
Gambar 7.25 Isi folder <i>Test Case</i> .....	93

Gambar 7.26 Isi <i>test case 1</i> by <i>tom</i> .....	94
Gambar 7.27 isi <i>test case 2</i> by <i>ranran</i> .....	94
Gambar 7.28 Membuat <i>Test Suite</i> .....	95
Gambar 7.29 Memberi nama <i>Test Suite</i> .....	95
Gambar 7.30 Memasukkan <i>test case</i> ke dalam <i>test suite</i> .....	95
Gambar 7.31 Hasil setelah menambahkan <i>test case</i> .....	96
Gambar 7.32 Jendela <i>Generate Command for Console Mode</i> .....	96
Gambar 7.33 <i>Copy</i> teks ke <i>Clipboard</i> .....	96
Gambar 7.34 <i>Download GIT</i> .....	97
Gambar 7.35 <i>Installer GIT</i> .....	97
Gambar 7.36 Menyalin link <i>clone repository</i> .....	97
Gambar 7.37 Melakukan <i>pull repository</i> .....	98
Gambar 7.38 Melakukan <i>push repository</i> .....	98
Gambar 8.1 Membuat Folder Baru.....	99
Gambar 8.2 Pilih Web Service Request .....	100
Gambar 8.3 Membuat Web Service Request.....	100
Gambar 8.4 URL API Pada Web .....	100
Gambar 8.5 Tampilan File .....	101
Gambar 8.6 Tampilan Response .....	101
Gambar 8.7 Membuat Folder Baru.....	102
Gambar 8.8 Pilih Web Service Request .....	102
Gambar 8.9 Membuat Web Service Request.....	102
Gambar 8.10 Searching Website di Google .....	103
Gambar 8.11 Tampilan Website.....	103
Gambar 8.12 Tampilan File .....	103
Gambar 8.13 Tampilan Service Function .....	104
Gambar 8.14 Membuat Folder User .....	104
Gambar 8.15 Folder User .....	104
Gambar 8.16 Form Test Case.....	105
Gambar 8.17 Web Service Keyword.....	105
Gambar 8.18 Send Request .....	105
Gambar 8.19 Get User Details .....	106
Gambar 8.20 Response Pada Kolom Output .....	106
Gambar 8.21 Verify Response Status Code .....	106
Gambar 8.22 Verify Response Status Code .....	106
Gambar 8.23 Property Value.....	107
Gambar 8.24 Verify Element Property Value .....	107
Gambar 8.25 Klik Icon Play.....	107
Gambar 8.26 Melihat Hasil di Log Viewer .....	107
Gambar 8.27 Mengganti isi dari Verify Element Property Value.....	108
Gambar 8.28 Hasil Error .....	108
Gambar 8.29 Penyebab Error .....	108
Gambar 8.30 Membuat File UserValidation .....	109
Gambar 8.31 Tampilan dari File UserValidation .....	109
Gambar 8.32 Centang Validate User Details.....	109
Gambar 8.33 Klik Icon Play.....	109
Gambar 8.34 Hasil dari Log Viewer .....	110

Gambar 8.35 Folder Reports Melihat Hasil Pengujian .....	110
Gambar 8.36 Hasil Pengujian File Json .....	110
Gambar 8.37 Contoh SOAP API.....	111
Gambar 8.38 Script SOAP API.....	111
Gambar 8.39 Membuat Folder .....	112
Gambar 8.40 Folder Countryinfoservices .....	112
Gambar 8.41 Membuat Web Service Request.....	112
Gambar 8.42 Tambahkan Ekstensi Wizdler pada Chrome .....	113
Gambar 8.43 Ikon dari Ekstensi Wizdler .....	113
Gambar 8.44 List Request.....	113
Gambar 8.45 Request dari ListOfCountryNamesByName.....	113
Gambar 8.46 Web Service Request dengan nama ListCountries.....	114
Gambar 8.47 List Countries Sudah Dibuat.....	114
Gambar 8.48 Load Service Function.....	115
Gambar 8.49 Service Function.....	115
Gambar 8.50 Copy Script.....	115
Gambar 8.51 Save Script.....	116
Gambar 8.52 Request Script.....	116
Gambar 8.53 List Nama dari Berbagai Negara .....	116
Gambar 8.54 Web Service Request SOAP.....	117
Gambar 8.55 Server Function .....	117
Gambar 8.56 Capital City .....	117
Gambar 8.57 Script CapitalCity .....	118
Gambar 8.58 Save Script.....	118
Gambar 8.59 Hasil Request.....	118
Gambar 8.60 Mengambil Request Berdasarkan Variabel Tertentu .....	119
Gambar 8.61 Menu Variabel.....	119
Gambar 8.62 Isi Tabel Variabel .....	120
Gambar 8.63 Isi Value .....	120
Gambar 8.64 Hasil Request yang Sudah Dimodifikasi .....	121
Gambar 8.65 Pengujian Test Case dari Kedua Web Service Request .....	121
Gambar 8.66 Variabel Response1 .....	121
Gambar 8.67 Script dari Variabel Response1 .....	122
Gambar 8.68 Mengambil Nilai Tertentu dari Respon .....	122
Gambar 8.69 List Cuplikan .....	123
Gambar 8.70 Verifikasi Nilai XML .....	123
Gambar 8.71 Variabel xml1 .....	124
Gambar 8.72 Parsing Nilai Response .....	124
Gambar 8.73 Code dari Variabel Countrycode .....	124
Gambar 8.74 Variabel Countrycode disimpan pada Variabel GlobalVariable .....	124
Gambar 8.75 Panggil API GetCapital .....	125
Gambar 8.76 Simpan dan Jalankan API GetCapital .....	125
Gambar 8.77 Hasil Verifikasi.....	125
Gambar 8.78 Ekstrak Country Code .....	126
Gambar 8.79 Hasil Country Code yang Diekstrak .....	126
Gambar 8.80 Validasi Country Code .....	126
Gambar 8.81 Memperlihatkan Hasil Country Code .....	127

Gambar 8.82 Code Verifikasi.....	127
Gambar 8.83 Memverifikasi Request.....	128
Gambar 8.84 Simpan Script dan Jalankan Test Case .....	128
Gambar 8.85 Verifikasi yang Sudah Berhasil .....	128
Gambar 8.86 Sample Rest API .....	129
Gambar 8.87 URL Request .....	129
Gambar 8.88 Salin URL.....	130
Gambar 8.89 Respon dari Teks URL reqres.in.....	130
Gambar 8.90 Web Service Request pada reqres.in.....	130
Gambar 8.91 Contoh API.....	131
Gambar 8.92 Hasil Pengujian API-CHAINING .....	131
Gambar 9.1 Node.js Setup .....	132
Gambar 9.2 Lokasi Instalasi Node.js.....	133
Gambar 9.3 Setting Default.....	133
Gambar 9.4 Tool Automatically Install.....	133
Gambar 9.5 Update License Tems .....	134
Gambar 9.6 Tahap Instalasi.....	134
Gambar 9.7 Mengecek Instalasi Java .....	134
Gambar 9.8 Install Appium.....	135
Gambar 9.9 Tahap Instalasi Selesai.....	135
Gambar 9.10 Membuat New Project .....	135
Gambar 9.11 Memilih Destination Folder .....	135
Gambar 9.12 Membuat New Folder Test Case .....	136
Gambar 9.13 Create Folder .....	136
Gambar 9.14 Create Test Case .....	136
Gambar 9.15 Mengecek About Phone.....	137
Gambar 9.16 Setting Opsi Pengembang.....	137
Gambar 9.17 Menjalankan Record Mobile .....	138
Gambar 9.18 Configuration pada Mobile Recorder .....	138
Gambar 9.19 Open APIDemos.....	138
Gambar 9.20 Start application APIDemos .....	138
Gambar 9.21 Menginstall Appium dan API.....	139
Gambar 9.22 Test pada APIDemos .....	139
Gambar 9.23 Save Script APIDemos .....	140
Gambar 9.24 Create Folder APIDemos.....	140
Gambar 9.25 Export Option Mobile Automation Testing .....	141
Gambar 9.26 Run Mobile Automation Testing .....	141
Gambar 9.27 Android Device .....	141
Gambar 9.28 Proses Running Mobile Automation Testing .....	142
Gambar 9.29 Tampilan Log Viewer Test.....	142
Gambar 10.1 Membuat Test Case .....	143
Gambar 10.2 Memberi Nama Test .....	144
Gambar 10.3 Record Web.....	144
Gambar 10.4 Link Website yang akan Dilakukan Pengujian .....	144
Gambar 10.5 Login .....	144
Gambar 10.6 Test Tercatat di Tabel .....	145
Gambar 10.7 Membuat Test Suite.....	145

Gambar 10.8 Mengisi Default Profiles.....	145
Gambar 10.9 Isi dari Nilai dari Profiles .....	146
Gambar 10.10 Script Mode .....	146
Gambar 10.11 Rename Sesuai Dengan Environment.....	146
Gambar 10.12 Membuat Test Suite Baru .....	147
Gambar 10.13 Menambahkan Test Case Dalam Test Suite.....	147
Gambar 10.14 Tambahkan Test Case.....	147
Gambar 10.15 Eksekusi Test Case Profiles .....	148
Gambar 10.16 Membuat Test Suite Collection .....	148
Gambar 10.17 Tambahkan Test Suite pada Test Suite Collection.....	149
Gambar 10.18 Pilih Environment yang akan dieksekusi .....	149
Gambar 11.1 Buka Aplikasi Katalon.....	150
Gambar 11.2 Pilih Custome Keywords .....	151
Gambar 11.3 Fitur-fitur Pada Custom Keyword .....	151
Gambar 11.4 Membuat Keyword .....	152
Gambar 11.5 Masukkan Nama Package dan Nama Class .....	152
Gambar 11.6 Tampilan dari Keyword.....	153
Gambar 11.7 Masukkan Kode.....	153
Gambar 11.8 Membuat Test Case Baru di dalam Common Tes Case .....	153
Gambar 11.9 Membuat Test Case Baru “Test1” .....	154
Gambar 11.10 Tampilan dari Test Case Baru .....	154
Gambar 11.11 Tampilan dari Test Case Setelah ditambahkan .....	154
Gambar 11.12 Tampilan dari Console.....	155
Gambar 11.13 Membuat Kata Kunci CustomFungtion.groovy .....	155
Gambar 11.14 Masukkan File .....	155
Gambar 11.15 Tampilan Script dari Test1 .....	156
Gambar 11.16 Memasukkan Kata Kunci .....	156
Gambar 11.17 Jalankan Test Case Test1 .....	156
Gambar 11.18 Tampilan Console Setelah dijalankan .....	156
Gambar 12.1 Tampilan Halaman Utama Katalon .....	158
Gambar 12.2 Tampilan Hasil Katalon Setelah Didownload.....	158
Gambar 12.3 Tampilan Menu test Case .....	159
Gambar 12.4 Tampilan Jendela Rekaman .....	159
Gambar 12.5 Tampilan application file Yang Akan Diuji.....	160
Gambar 12.6 Tampilan Proses Recording .....	160
Gambar 12.7 Tampilan instalasi wizard .....	161
Gambar 12.8 Tampilan License agreement .....	161
Gambar 12.9 Tampilan Lokasi Penyimpanan File Install.....	161
Gambar 12.10 Tampilan Hasil Install .....	162
Gambar 12.11 Tampilan Hasil Install Berhasil .....	162
Gambar 12.12 Tampilan WinappDriver .....	162
Gambar 12.13 Tampilan Start WinappDriver .....	163
Gambar 12.14 Tampilan Mengaktifkan Developer Mode .....	163
Gambar 12.15 Tampilan Developer Mode Berhasil Di Aktifkan .....	164
Gambar 12.16 Tampilan notepad.exe.....	164
Gambar 12.17Tampilan tombol Start .....	164
Gambar 12.18 Tampilan Halaman CMD Pada Notapad.....	165

Gambar 12.19 Tampilan Pengujian Terhadap Desktop .....	165
Gambar 12.20 Tampilan Objek Repository Baru .....	166
Gambar 12.21 Tampilan Folder Notepad Objects berhasil.....	166
Gambar 12.22 Membuat Satu Test Case Baru.....	166
Gambar 12.23 Tampilan Test Case Pada Kasus Uji .....	167
Gambar 12.24 Tampilan Jendela Script .....	167
Gambar 12.25 Tampilan Running Test Case.....	167
Gambar 13.1 Website Katalon Store .....	171
Gambar 13.2 Utilities Katalon Store Websites .....	171
Gambar 13.3 Menginstall Auto-healing Smart Xpath .....	172
Gambar 13.4 Dokumentasi Plugin Xpath.....	172
Gambar 13.5 Halaman Login Katalon Store .....	173
Gambar 13.6 Pop up Accept and Install Xpath .....	173
Gambar 13.7 Notifikasi Plugin Xpath Successfully Instaled.....	173
Gambar 13.8 Case test 1 Katalon Studio Project.....	174
Gambar 13.9 Record Web Tools Katalon Studio .....	174
Gambar 13.10 Memulai Recording Action Katalon Studio.....	174
Gambar 13.11 Pengujian Website CURA Healthcare Service .....	175
Gambar 13.12 Memasukkan Username dan Password akun Demo.....	175
Gambar 13.13 Jendela Web Recorder Katalon Studio .....	176
Gambar 13.14 Menjalankan Kembali Test Recording Web .....	176
Gambar 13.15 Menyimpan Object Yang Sudah Dilakukan Test.....	177
Gambar 13.16 Informasi Object Beserta Properties Web Recorder test1 .....	177
Gambar 13.17 Informasi Object Beserta Properties Web Recorder Katalon Studio .....	178
Gambar 13.18 Mengubah wait element for timeout menjadi 5 detik .....	178
Gambar 13.19 Menjalankan kembali web recording katalon studio.....	179
Gambar 13.20 Menjalankan Web Recorder Dengan Website Yang Lainnya .....	179
Gambar 13.21 Jendela Informasi Web Recorder .....	179
Gambar 13.22 Informasi Object Beserta Properties Web Recorder test2 .....	180
Gambar 13.23 Inspect dari web test2 .....	180
Gambar 13.24 Value Identifikasi Obejct Pada Web Test2 .....	181
Gambar 13.25 Reload Plugins Pada Icon Profile .....	181
Gambar 13.26 Status Active Plugin Applitools Intergration .....	182
Gambar 13.27 Tampilan Utama Applitools .....	182
Gambar 13.28 My API key pada icon profile.....	183
Gambar 13.29 Menyalin API Key.....	183
Gambar 13.30 Membuka Setting pada tab Project .....	183
Gambar 13.31 Menyesuaikan Isian Applitools Intergration .....	184
Gambar 13.32Membuat test case baru pada aplikasi katalon studio.....	185
Gambar 13.33 Memberi label/nama test casenya .....	185
Gambar 13.34 Mengklik Link Walkthrough pada website applitools .....	186
Gambar 13.35 Mengklik link The Hello World Web Page .....	186
Gambar 13.36 Mengklik link diff1.....	187
Gambar 13.37 Mengklik Tombol Click me! .....	187
Gambar 13.38 Perubahan angka.....	187
Gambar 13.39 Membuka Web Record Tools Aplikasi Katalon Studo .....	188
Gambar 13.40 Masukkan Link The Hello World Page ke Web Recorder .....	188

Gambar 13.41 Proses Pengujian The Hello World Page Website .....	188
Gambar 13.42 Menambahkan Custom Keyword Pada Web Recorder .....	189
Gambar 13.43 Menuliskan check1 pada Input Web Recorder .....	189
Gambar 13.44 Memilih Keywords.checkWindows pada IsKeywords .....	190
Gambar 13.45 Mengklik link ?diff1 .....	190
Gambar 13.46 Melihat Informasi hasil record.....	190
Gambar 13.47 Menambahkan Custom Keyword untuk kedua kalinya.....	191
Gambar 13.48 Menuliskan check2 pada Input Web Recorder .....	191
Gambar 13.49 Mengklik tombol stop recording.....	191
Gambar 13.50 Klik OK pada web recorder .....	192
Gambar 13.51 Klik OK pada jendela Add Element to Obejct Repository .....	192
Gambar 13.52 Test Check2 .....	192
Gambar 13.53 Opsi Chrome Pada Web Service.....	193
Gambar 13.54 Menghapus Variabel a dan b pada script .....	193
Gambar 13.55 Aplikasi The Hello World Page Berjalan.....	194
Gambar 13.56 Merefresh Applitools.....	194
Gambar 13.57 Status berjalannya Check1 dan Check2 pada Applitools .....	194
Gambar 13.58 Merefresh kembali Applitools .....	194
Gambar 13.59 Hasil Caputure The Hello World Page Website.....	195
Gambar 13.60 Menjalankan kembali aplikasi sederhana dari katalon .....	195
Gambar 13.61 Pesan Error Test Case pada Log Viewer.....	196
Gambar 13.62 Check 1 yang telah Passed .....	196
Gambar 13.63 Meluluskan Test Case The Hello World Page .....	196
Gambar 13.64 Tombol Like Untuk Menerima Gambar ini .....	197
Gambar 13.65 Tools untuk mencrop area .....	197
Gambar 13.66 Mengcrop area yang akan diabaikan.....	197
Gambar 13.67 Memilih Region Ignore .....	197
Gambar 13.68 Pop Up Ketika Di Klik tombol like .....	198
Gambar 13.69 Menyimpan Preset .....	198
Gambar 13.70 Check2 telah berstatus Passed .....	198
Gambar 14.1 BDD Cucumber Test proses .....	200
Gambar 14.2 open project .....	202
Gambar 14.3 pilih folder include .....	202
Gambar 14.4 buat fitur baru .....	203
Gambar 14.5 tampilan file fitur login.....	203
Gambar 14.6 pilih recalculate steps.....	204
Gambar 14.7 membuat new step definition .....	204
Gambar 14.8 isi nama class name .....	205
Gambar 14.9 hapus dan buat script baru .....	205
Gambar 14.10 buat fitur navigasi .....	206
Gambar 14.11 cek login fitur .....	206
Gambar 14.12 hasil recalculate .....	207
Gambar 14.13 cek script dengan cara recalculate.....	207
Gambar 14.14 menuju ke script .....	208
Gambar 14.15 buat script untuk membuat fitur.....	208
Gambar 14.16 cek statement script .....	209
Gambar 14.17 hasil script yang sudah memiliki statement .....	209

Gambar 14.18 pilih pretty format untuk merapikan script.....	210
Gambar 14.19 hasil script yang sudah rapi.....	210
Gambar 14.20 klik button play untuk menjalankan script .....	210
Gambar 14.21 script berhasil dijalankan .....	211
Gambar 14.22 klik log viewer untuk melihat detail script.....	211
Gambar 14.23 lihat detail script pada bagian console .....	212
Gambar 14.24 membuat fitur enter .....	212
Gambar 14.25 tambahkan script.....	213
Gambar 14.26 recalculate script.....	213
Gambar 14.27 hasil recalculate tidak ada warning .....	214
Gambar 14.28 script yang dibuat berhasil .....	214
Gambar 14.29 melihat detail script dengan log viewer .....	215
Gambar 14.30 pilih test case .....	215
Gambar 14.31 isi nama test case yang ingin dibuat.....	216
Gambar 14.32 klik icon record web .....	216
Gambar 14.33 isi URL yang akan diuji.....	217
Gambar 14.34 klik action pada setiap pengujinya .....	217
Gambar 14.35 save script.....	218
Gambar 14.36 klik stop record .....	218
Gambar 14.37 klik OK untuk membuat folder baru penyimpanan file.....	219
Gambar 14.38 klik ubah menjadi verify element .....	219
Gambar 14.39 buka tab script .....	220
Gambar 14.40 pengujian yang berhasil dijalankan.....	220
Gambar 14.41 salin script file LoginTest .....	221
Gambar 14.42 pastekan script pada bagian navigate to login page.....	221
Gambar 14.43 salin script .....	221
Gambar 14.44 enter username dan password .....	222
Gambar 14.45 ubah username dan password .....	222
Gambar 14.46 mengubah text password biasa menjadi password enkripsi .....	222
Gambar 14.47 masukkan password yang sesuai dengan web pengujian .....	223
Gambar 14.48 paste pada bagian password .....	223
Gambar 14.49 salin script dari file login test.....	224
Gambar 14.50 paste kan script pada bagian login button .....	224
Gambar 14.51 salin script dari file LoginTest .....	225
Gambar 14.52 pastekan script pada bagian home page .....	225
Gambar 14.53 jalankan file login fitur dengan chrome .....	226
Gambar 14.54 hasil setelah dijalankan .....	226
Gambar 14.55 lihat detail error pada log viewer .....	227
Gambar 14.56 error ditemukan .....	227
Gambar 14.57 kembali ke file LoginTest .....	227
Gambar 14.58 pastekan script yang kurang pada file LoginTest.....	228
Gambar 14.59 jalankan lagi file tersebut.....	228
Gambar 14.60 hasil saat dijalankan.....	229
Gambar 14.61 Terjadi error pada skenario kedua.....	229
Gambar 14.62 lihat skenario error pada log viewer.....	230
Gambar 14.63 buat test case baru pada folder Test Case.....	230
Gambar 14.64 beri nama file test case baru.....	231

Gambar 14.65 klik keyword browser .....	231
Gambar 14.66 klik RunFeaturFile.....	232
Gambar 14.67 tambahkan item dengan klik cucumber keyword .....	232
Gambar 14.68 beri nama item yang ditambahkan .....	233
Gambar 14.69 salin ID item .....	233
Gambar 14.70 masukkan ID pada bagian value .....	234
Gambar 14.71 id yang telah ditambahkan .....	234
Gambar 14.72 disable kan debug file .....	234
Gambar 14.73 jalankan file dengan chrome .....	235
Gambar 14.74 skenario pertama berhasil dijalankan .....	235
Gambar 14.75 skenario kedua error .....	236
Gambar 14.76 kembali ke Login Feature .....	236
Gambar 14.77 pada item kedua enable-kan debug .....	237
Gambar 14.78 disable kan item nomor 1.....	237
Gambar 14.79 salin ID .....	238
Gambar 14.80 salin pada kolom paste.....	238
Gambar 14.81 jalankan file dengan chrome .....	239
Gambar 14.82 hasil skenario pertama .....	239
Gambar 14.83 keterangan berhasil dijalankan pada job progress .....	240
Gambar 14.84 buat Test Suite baru .....	240
Gambar 14.85 berikan nama file Test Suite .....	241
Gambar 14.86 tampilan setelah file Test Suite berhasil dibuat.....	241
Gambar 14.87 pilih RunFeatureFile .....	242
Gambar 14.88 jalankan file Test Suite .....	242
Gambar 14.89 berhasil dijalankan.....	243
Gambar 14.90 klik button build CMD .....	243
Gambar 14.91 pilih lokasi file Test Suite .....	244
Gambar 14.92 buka folder reports untuk menampilkan report file Test Suite .....	244
Gambar 14.93 cari lokasi penyimpanan .....	245
Gambar 14.94 pada lokasi penyimpanan pilih file HTML .....	245
Gambar 14.95 log dari pengujian yang dilakukan .....	246
Gambar 14.96 membuat class baru dalam folder groovy .....	246
Gambar 14.97 tambahkan code di dalam class.....	247
Gambar 14.98 add cucumber didalam test .....	247
Gambar 14.99 masuk ke script Runfeaturefile dan jalakan .....	247
Gambar 14.100 proses running file .....	247
Gambar 14.101 pilih reports.....	248
Gambar 14.102 pilih reports folder .....	248
Gambar 14.103 klik 2 kali pada index.html .....	248
Gambar 14.104 hasil reports folder .....	249

## **BAB 1** **PENDAHULUAN**

### **1.1. Tujuan**

1. Mahasiswa dapat memahami dari Aplikasi Katalon
2. Mahasiswa dapat membuat akun Katalon
3. Mahasiswa dapat melakukan instalasi Katalon

### **1.2. Dasar Teori**

Katalon adalah sebuah aplikasi yang dapat melakukan tes otomatis terhadap program web dan *mobile* (termasuk kapabilitas dari Selenium dan Appium). Katalon tersedia secara gratis untuk Windows maupun Mac OS, website resmi dari katalon dapat dikunjungi pada link [www.katalon.com](http://www.katalon.com). Katalon dilengkapi dengan tampilan *Graphical User Interface*(GUI) yang *friendly*. Katalon juga dapat melakukan pengujian secara otomatis dengan mudah.

Adapun beberapa hal yang dilakukan dengan menggunakan aplikasi katalon ini yaitu:

1. Otomatis untuk tes web dan tes aplikasi *mobile*
2. Dapat melakukan tes terhadap *web service* (API)
3. Membuat secara cepat tes otomatis
4. Dapat membuat *record* dan *playback*
5. Tes dapat berjalan antar browser yang berbeda

Katalon juga telah terintegrasi dengan : JIRA, qTest, Git, dan Jenkins. Katalon dapat membantu untuk melakukan tes yang otomatis, yaitu sebagai berikut:

1. Membuat pengujian secara otomatis dengan cepat
2. Memiliki pendukung untuk perangkat Web, *Mobile* dan API
3. Penguji manual dapat memulai dengan menggunakan caranya sendiri
4. Kurva belajar yang sangat pendek
5. Fitur yang sangat berguna, seperti membuat tes, eksekusi, dan laporan
6. Telah terintegrasi dengan JIRA, Jenkins, Git, dan lain-lain

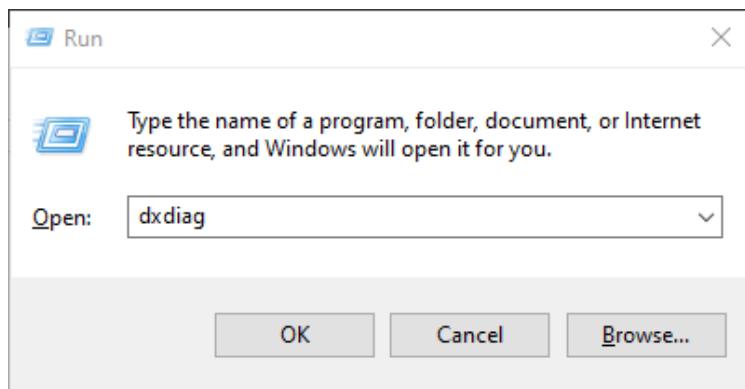
### 1.3. Tahap-tahap Instalasi Katalon

Sebelum memulai proses instalasi aplikasi katalon, langkah pertama yang harus dilakukan adalah cek spesifikasi laptop masing-masing. Adapun spesifikasi yang disarankan untuk menggunakan katalon adalah sebagai berikut :

1. **Operasi sistem** : Windows 7 atau yang terbaru, MacOS 10.11+
2. **CPU** : 1 GHz atau cepat dari 32-bit (x86) atau 64-bit (x64) processor
3. **Memory** : 1 GB RAM (32-bit) atau 2 GB RAM (64-bit)
4. **Hard Drive** : setidaknya tersisa 1 GB dari ruang disk

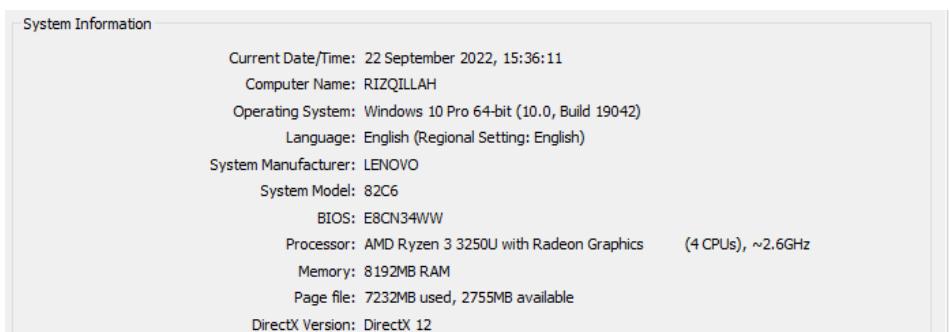
Untuk melakukan pengecekan spesifikasi laptop adalah dengan mengikuti tahap berikut :

1. Tekan Win+R
2. Kemudian ketik dxdiag dan OK



Gambar 1.1 Jendela Run

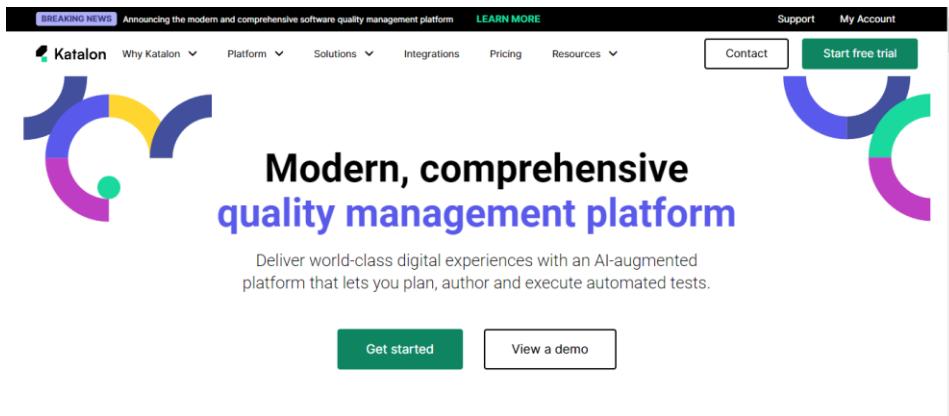
3. Perhatikan jendela yang muncul, maka pada jendela tersebut akan berisi informasi mengenai spesifikasi laptop



Gambar 1.2 Detail spesifikasi laptop

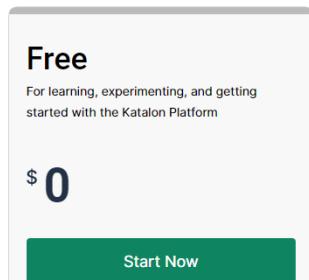
Langkah selanjutnya adalah melakukan registrasi akun pada *website* [www.katalon.com](http://www.katalon.com). Langkah-langkahnya sebagai berikut :

1. Buka website [www.katalon.com](http://www.katalon.com), pilih menu *pricing*



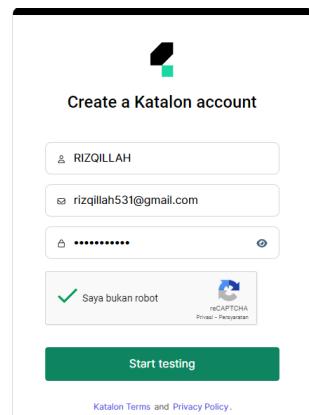
Gambar 1.3 Tampilan *landing page* [www.katalon.com](http://www.katalon.com)

2. Pilih bagian *Free* dan tekan *Start Now*



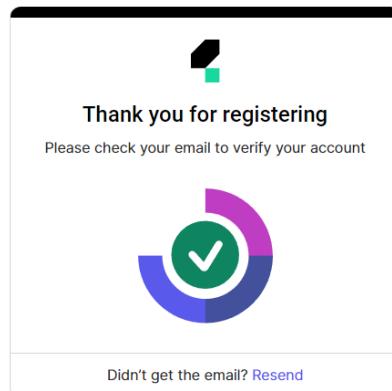
Gambar 1.4 *Free Plan* Katalon

3. Buat akun anda dengan menginputkan Nama, Email, dan Password. Adapun password yang harus dimasukkan adalah password dengan huruf besar, huruf kecil, angka, dan karakter



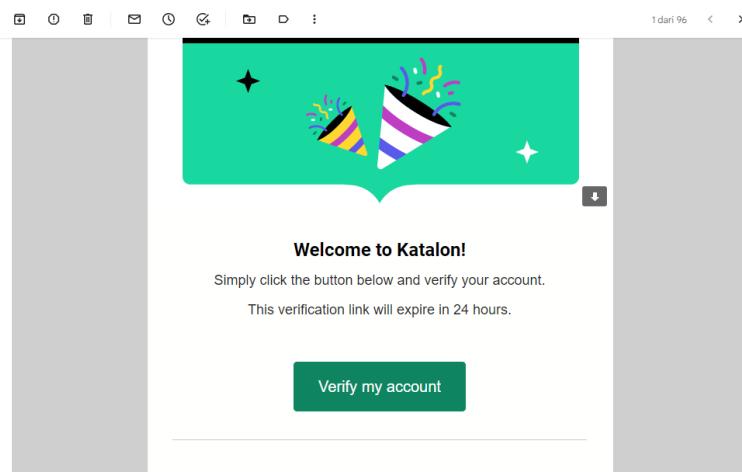
Gambar 1.5 Tampilan halaman *register*

4. Buka email aktivasi di email yang telah didaftarkan, jika email tidak ada, cek bagian kotak Spam



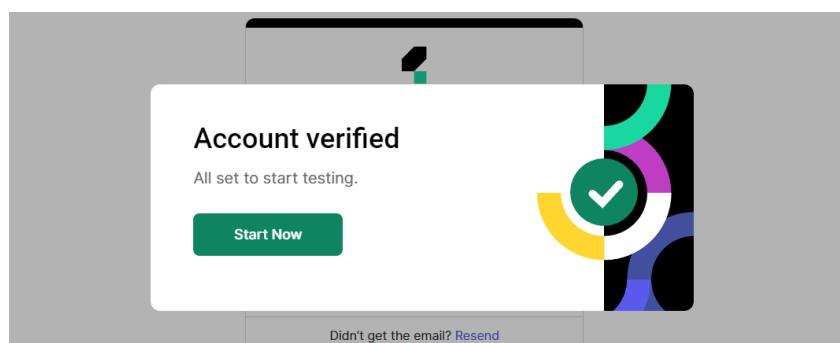
Gambar 1.6 Halaman *Verify Email*

5. Klik *Verify My Account* pada email yang diterima



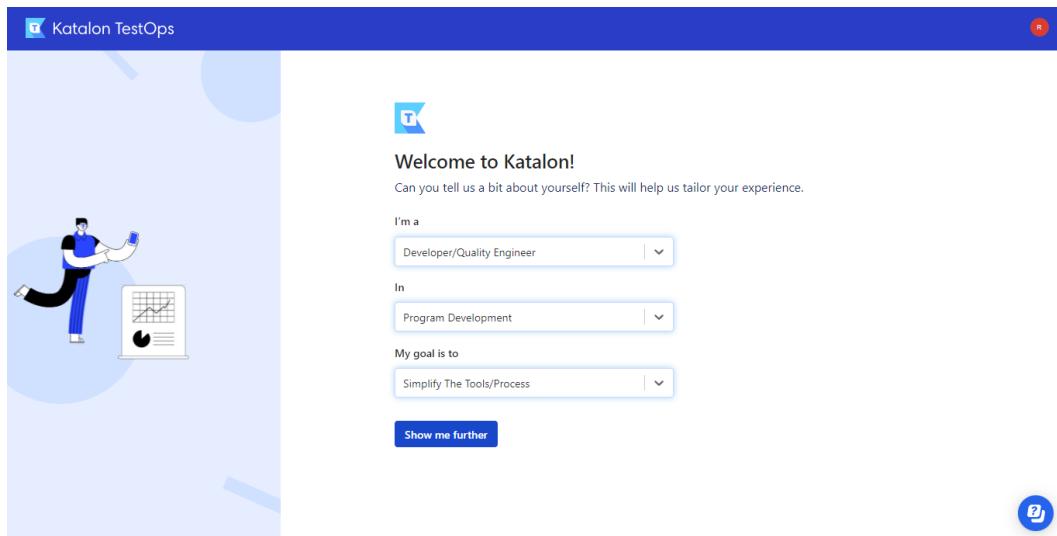
Gambar 1.7 Email aktivasi akun Katalon

6. Berikut adalah tampilan jika akun berhasil di aktivasi, langkah selanjutnya klik tombol *Start Now*



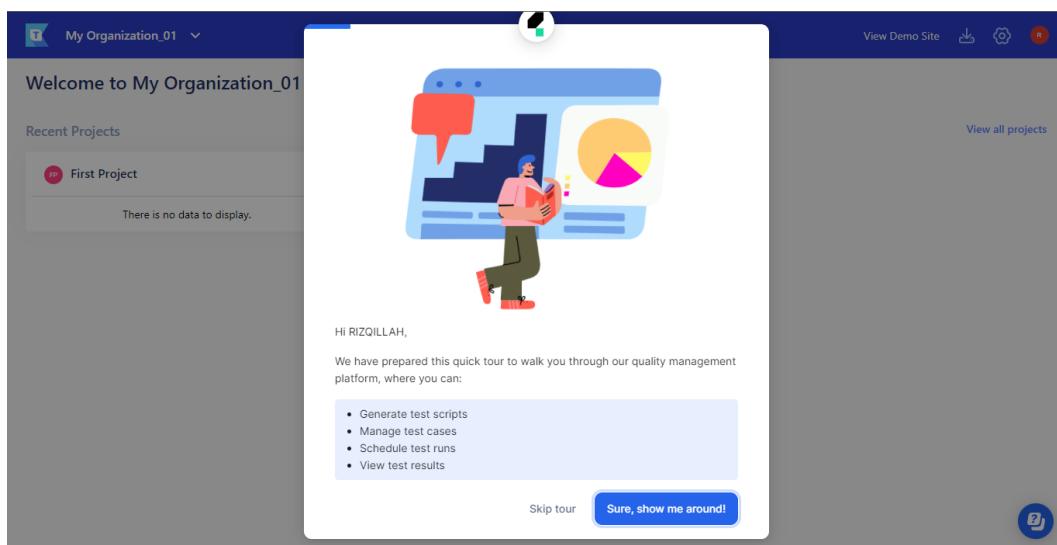
Gambar 1.8 Halaman jika akun telah teraktivasi

7. Pada halaman *Welcome to Katalon!* Isikan data anda berupa data profesi, ketertarikan, dan tujuan dari menggunakan Katalon. Kemudian klik *Show me Futher*



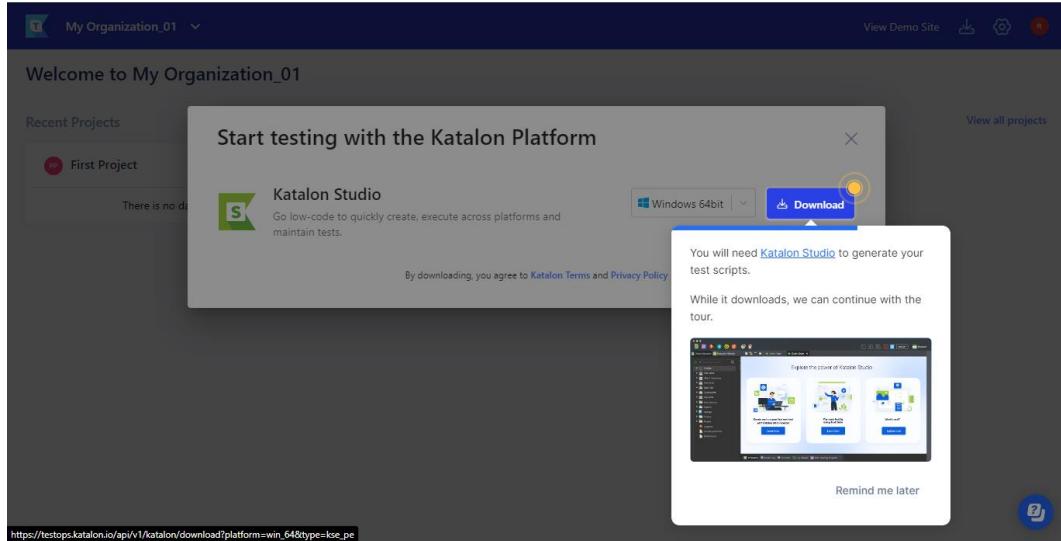
Gambar 1.9 Halaman pengisian data profesi

8. Jika sudah berhasil, maka akan mendapatkan *pop up* untuk *tutorial* dari *dashboard* Katalon, ikuti *tutorial* sampai selesai



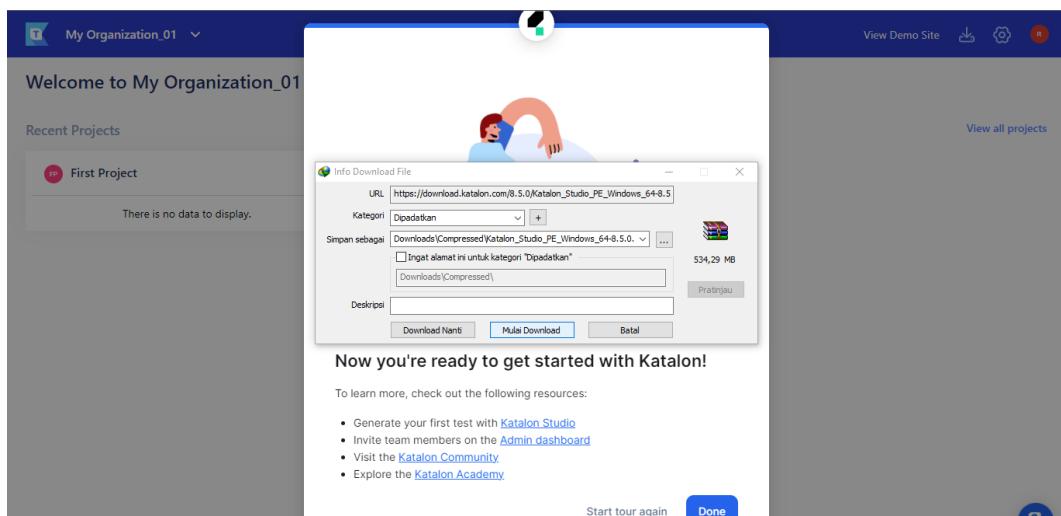
Gambar 1.10 Pop up toturial pengenalan dashboard Katalon

9. Kemudian anda akan diminta *download* aplikasi Katalon Studio sesuai dengan *platform OS* yang digunakan, klik *Download*



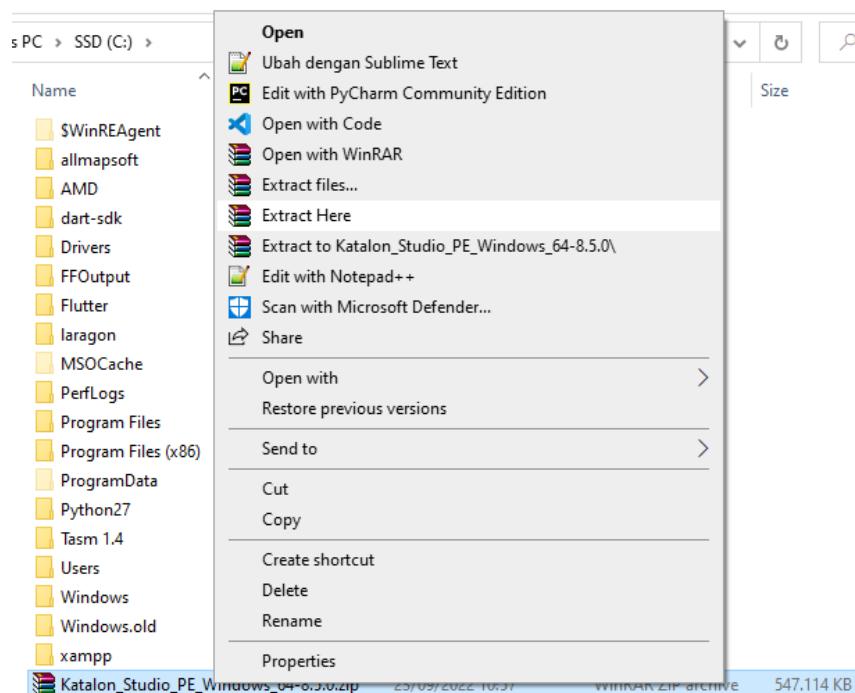
Gambar 1.11 Pop up download Katalon Studio

10. *Download* aplikasi Katalon Studio sampai selesai, jika sudah selesai, langkah selanjutnya adalah menginstal aplikasi Katalon Studio pada perangkat PC anda



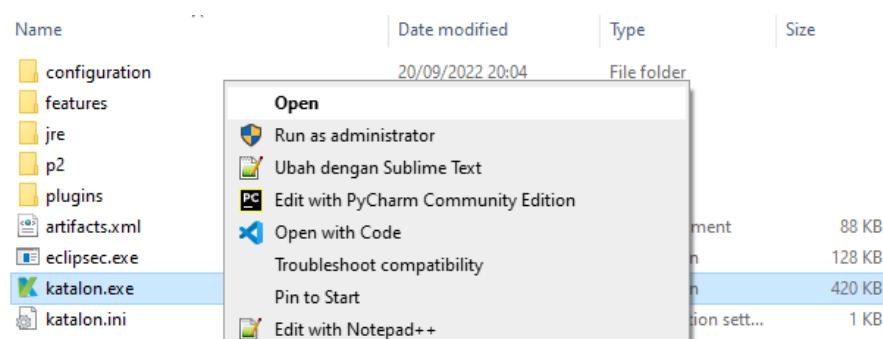
Gambar 1.12 Download Katalon Studio

11. Extract hasil download pada partisi C, dikarenakan aplikasi Katalon tidak perlu menggunakan instalasi, hal yang diperlukan hanyalah membuka aplikasi setelah download



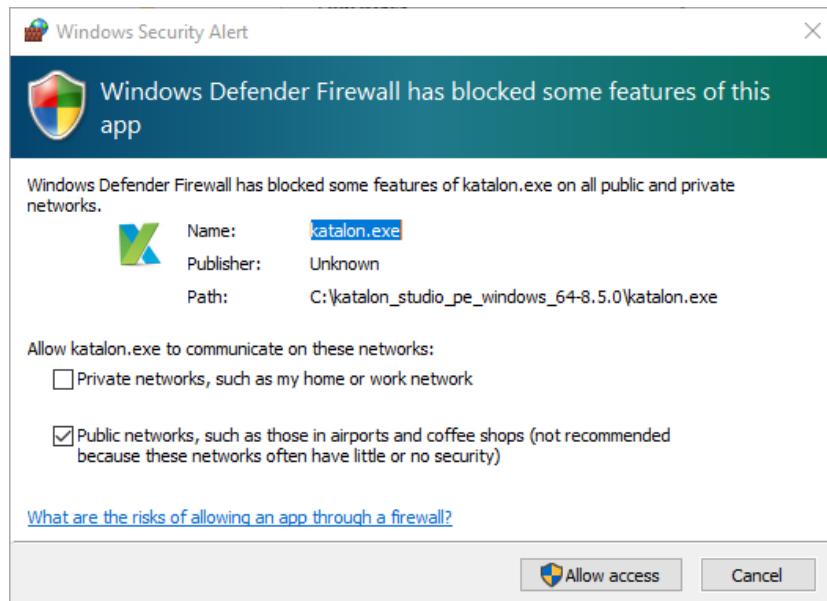
Gambar 1.13 Extract file katalon

12. Buka folder hasil ekstraksi dan buka file katalon.exe



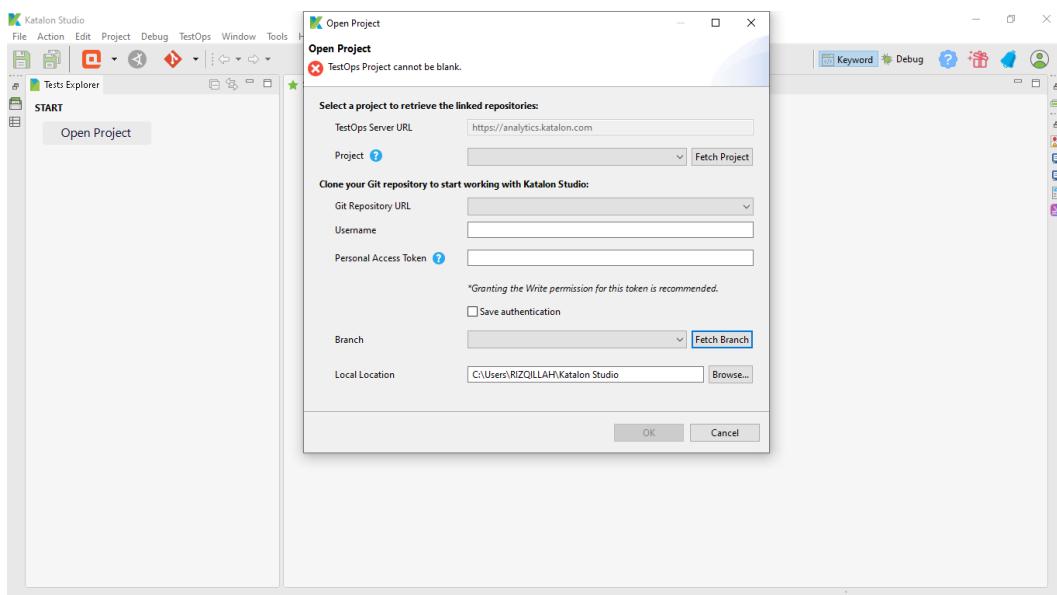
Gambar 1.14 Directory Katalon Studio

13. Jika muncul jendela *Windows Defender Firewall*, pilih *Allow access*



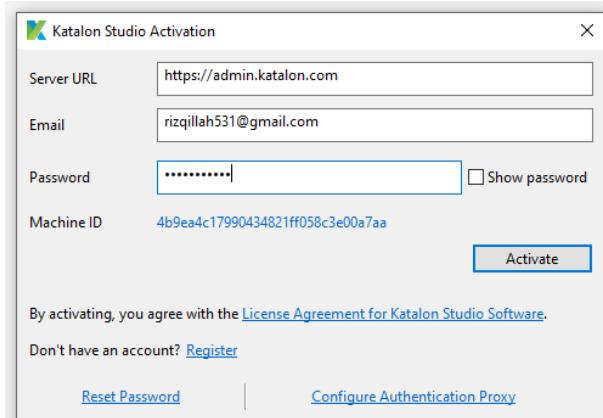
Gambar 1.15 Jendela *Windows Defender Firewall*

14. Pada jendela ini, anda akan diminta untuk melakukan *clone* atau *linked* dengan *repository* dari git, jika tidak ingin melakukan *clone* pilih *Cancel*

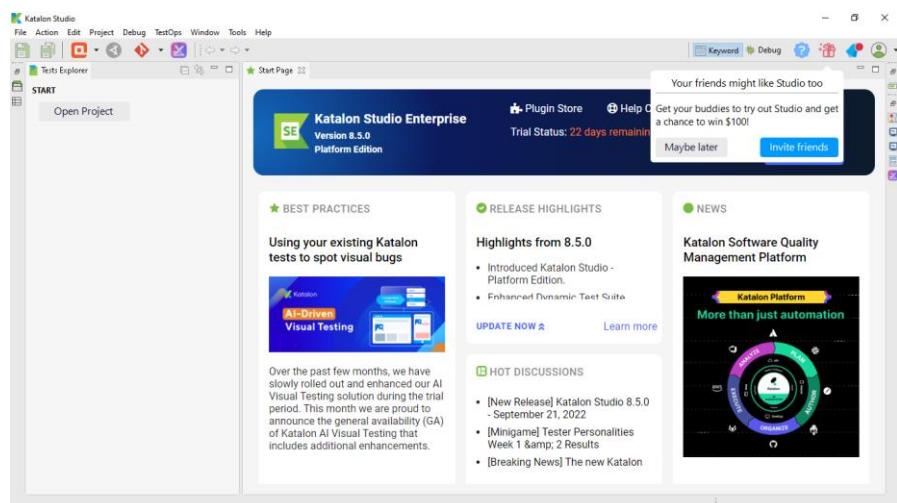


Gambar 1.16 Jendela *cloning repository* git

15. Tampilan awal dari Katalon Studio, jika sebelumnya anda diminta untuk masuk akun, masukkan data email dan password dari akun yang terdaftar pada *website* Katalon sebelumnya

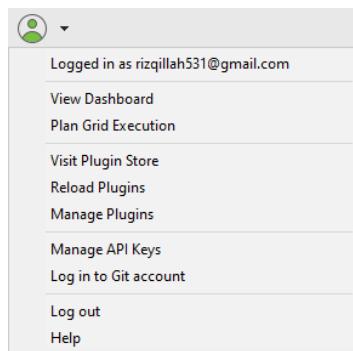


Gambar 1.17 Activate Katalon Studio dengan akun



Gambar 1.18 Tampilan awal Katalon Studio

16. Klik pada *icon people* maka akan muncul jika telah *login* atau belum

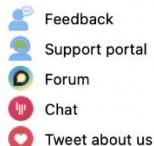


Gambar 1.19 Tampilan jika berhasil *Login*

17. Adapun fungsi-fungsi dari tombol *toolbar* adalah sebagai berikut

### Fitur Bagian Sistem

Icon	Deskripsi
	Simpan project dari testing
	Simpan semua pengujian yang dibuka.
	Buka Spy Web untuk menangkap elemen di situs web.
	Buka Spy Mobile untuk menangkap elemen pada aplikasi seluler.
	Buka Spy Windows Object untuk menangkap elemen di Windows.
	Buka Rekam Web untuk merekam kasus uji WebUI.
	Buka Rekam Seluler untuk merekam kasus uji seluler.
	Buka Rekam Tindakan Windows untuk merekam kasus uji Windows.
	Buat kasus uji baru.
	Jalankan kasus uji yang saat ini terbuka. Anda dapat memilih aplikasi untuk menjalankan pengujian mulai dari: <ul style="list-style-type: none"><li>▪ Chrome</li><li>▪ Firefox</li><li>▪ IE</li><li>▪ Safari</li></ul>

	<ul style="list-style-type: none"> <li>▪ Edge</li> <li>▪ Remote</li> <li>▪ Headless</li> <li>▪ Android</li> <li>▪ iOS (on macOS)</li> <li>▪ Custom</li> </ul>
	<p>Jalankan dan debug kasus uji yang saat ini terbuka. Anda dapat memilih aplikasi untuk menjalankan pengujian mulai dari:</p> <ul style="list-style-type: none"> <li>▪ Chrome</li> <li>▪ Firefox</li> <li>▪ IE</li> <li>▪ Safari</li> <li>▪ Edge</li> <li>▪ Remote</li> <li>▪ Headless</li> <li>▪ Android</li> <li>▪ iOS (on macOS)</li> <li>▪ Custom</li> </ul>
 Debug	Debug kasus uji yang saat ini terbuka.
	Hentikan sesi eksekusi pengujian saat ini.
	Buka Command Builder untuk menghasilkan perintah untuk eksekusi konsol.
 default	Profil eksekusi (lingkungan pengujian) yang akan diterapkan saat menjalankan pengujian.
	Akses Pusat Bantuan dan Forum Katalon.
 <ul style="list-style-type: none"> <li> Feedback</li> <li> Support portal</li> <li> Forum</li> <li> Chat</li> <li> Tweet about us</li> </ul>	<p>Anda juga dapat mengirimkan umpan balik atau pertanyaan lebih lanjut ke saluran kami. Pakar dan pengguna Katalon akan mencoba membantu Anda sesegera mungkin.</p>

## Fitur Plugin

Icon	Deskripsi
	Katalon TestOps adalah platform khusus kami untuk orkestrasi QA, analisis pengujian, dan laporan lanjutan.
	Katalon TestCloud adalah lingkungan eksekusi pengujian berbasis cloud tempat Anda dapat mengotomatiskan skrip pengujian di seluruh browser dan/atau sistem operasi (OS) yang paling umum dan diperbarui dan/atau kombinasi keduanya.
	Self-Healing secara otomatis mencoba locator lain ketika locator default gagal.
	Perintah untuk aktivitas Git. Anda dapat memilih opsi ini dengan memilih opsi yang muncul di daftar dropdown (setelah mengaktifkan Git): <ul style="list-style-type: none"> <li>▪ Clone Project</li> <li>▪ Share Project</li> <li>▪ Show History</li> <li>▪ Manage Branches</li> <li>▪ Commit</li> <li>▪ Push</li> <li>▪ Pull</li> <li>▪ Fetch</li> </ul>
	Applitools adalah alat pengujian visual bawaan untuk mode Perekaman dan Pembuatan Skrip.
	Impor test case dari akun terintegrasi JIRA.

## **BAB 2**

### **MULAI MEMBUAT PROJEK**

#### **2.1. Tujuan**

Setelah mempelajari cara membuat projek pada katalon, diharapkan:

1. Mahasiswa dapat membuat projek katalon
2. Mahasiswa dapat membuat tes dengan *Record* dan *Replay*
3. Mahasiswa dapat membuat tes dengan *Manual*
4. Mahasiswa dapat membuat tes dengan *Script*

#### **2.2. Membuat Projek Katalon**

Untuk membuat projek tes baru di Katalon Studio terbagi atas 3 cara, yaitu:

1. *Record and Replay*
2. *Manual Mode*
3. *Script Mode*

Adapun ketiga cara tersebut akan dijelaskan pada bab ini dengan selengkap mungkin.

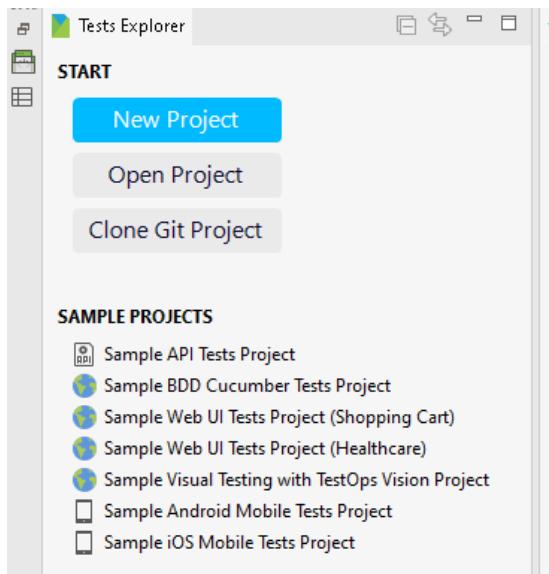
##### **2.2.1. Record and Replay**

Pada metode menggunakan *record and replay* akan memberi beberapa peran yang penting, diantaranya yaitu:

1. Dapat dengan cepat merekam tes yang dilakukan
2. Menangkap aksi beserta objek dari tes
3. Dapat mengupdate/edit langkap dari perekaman
4. Dapat melakukan pengulangan pada semua jenis browser

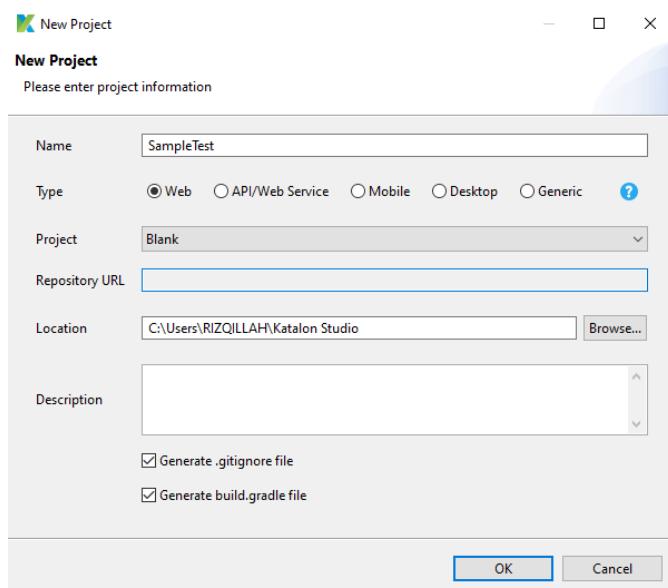
Adapun cara melakukan tes secara *Record and Replay* adalah sebagai berikut :

1. Klik New Project



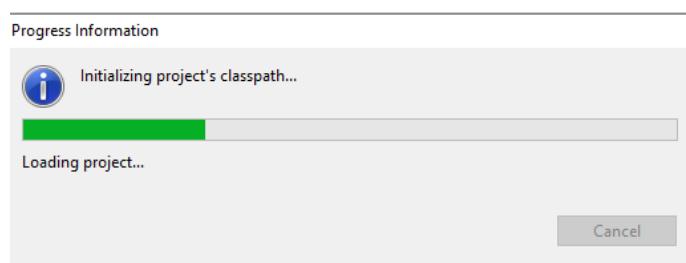
Gambar 2.1 *Create a New Project*

2. Isikan nama project, dan klik OK



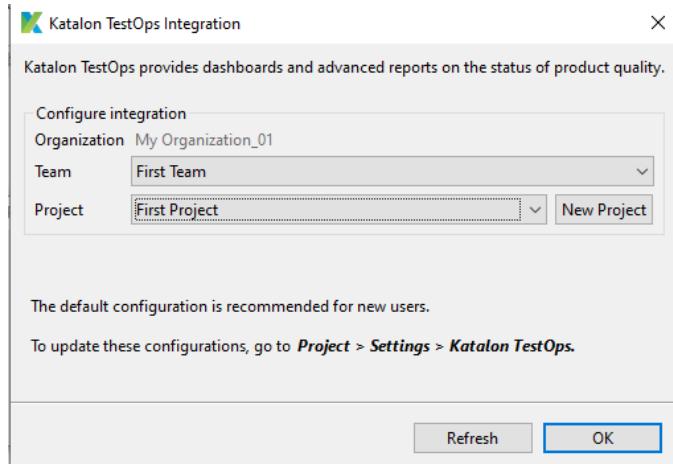
Gambar 2.2 *New Katalon Project*

3. Tunggu hingga proses selesai



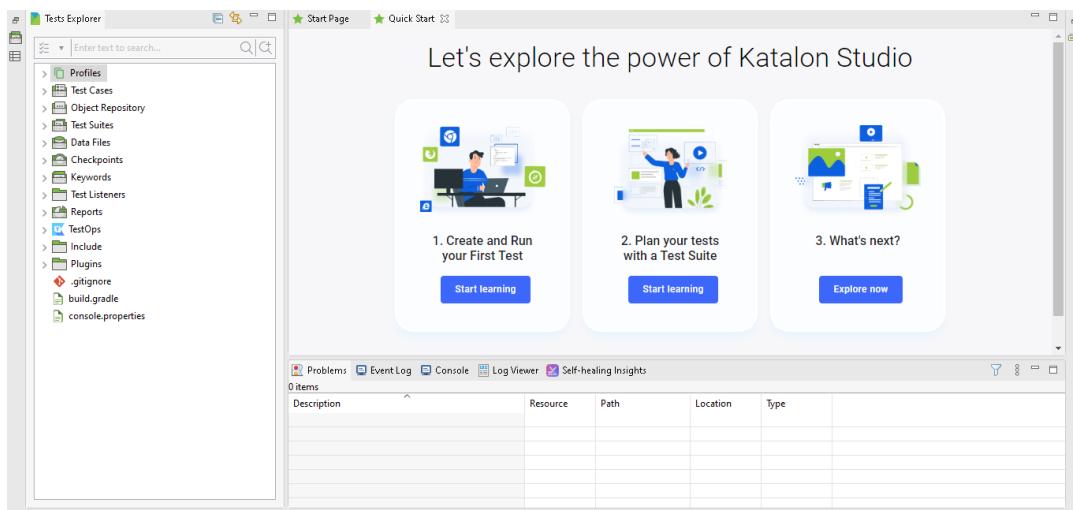
Gambar 2.3 *Proses pembuatan projek baru*

4. Jika keluar integrasi klik OK



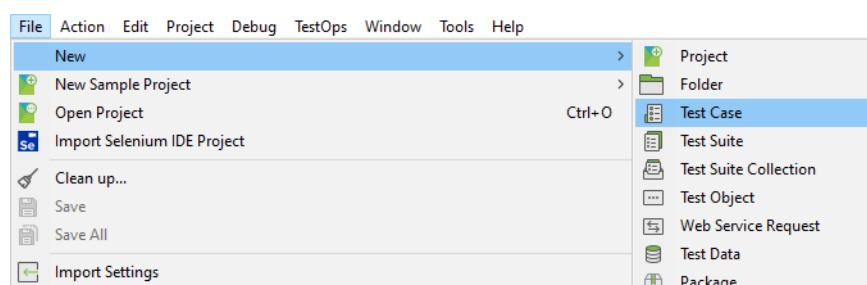
Gambar 2.4 Integrasi project TestOps

5. Hasil projek baru



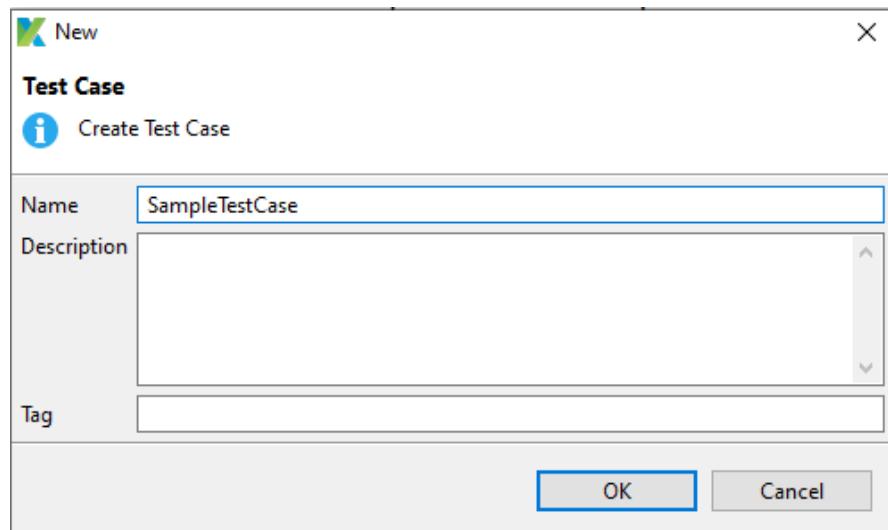
Gambar 2.5 Hasil setelah membuat projek

6. Kemudian pilih *File>New>Test Case*



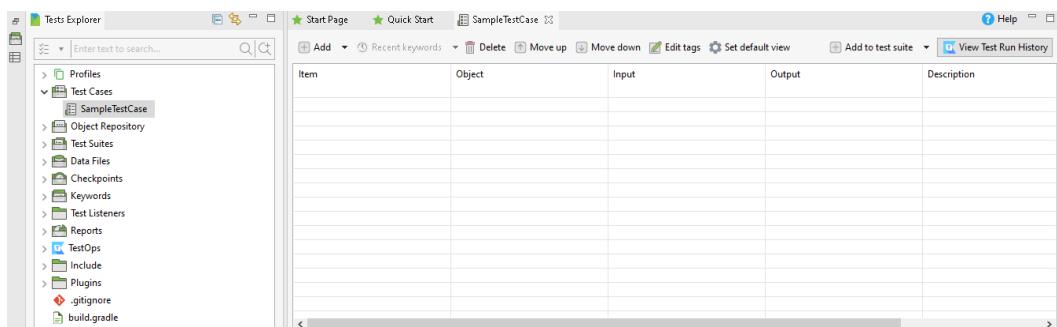
Gambar 2.6 Membuat Test Case

7. Tulis nama *test case* dan OK



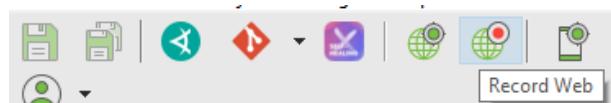
Gambar 2.7 Memberi nama *Test Case*

8. Hasil setelah membuat *Test Case*



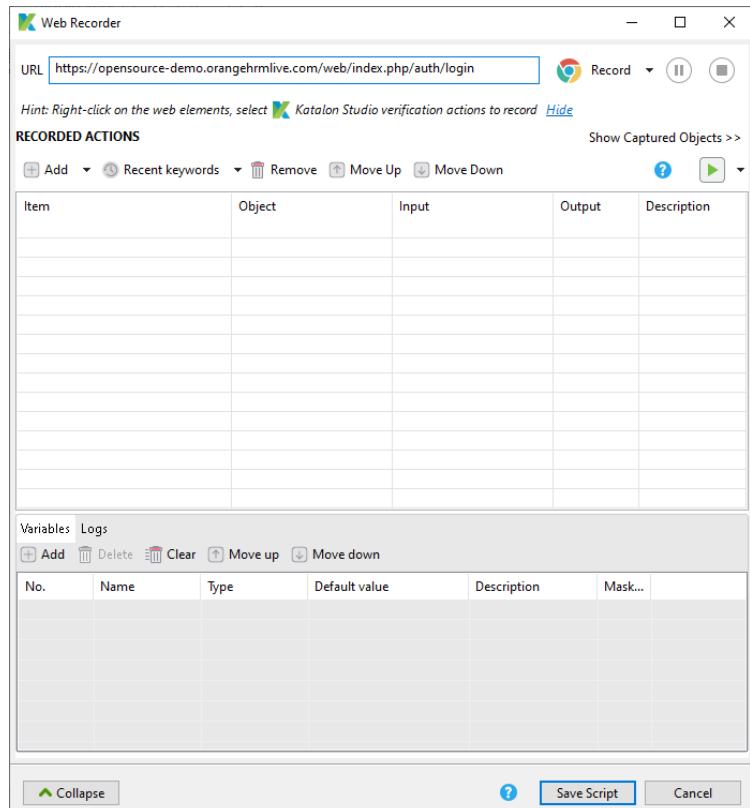
Gambar 2.8 Tampilan *Test Case*

9. Untuk memulai proses perkemanan pada website, klik *Record Web*



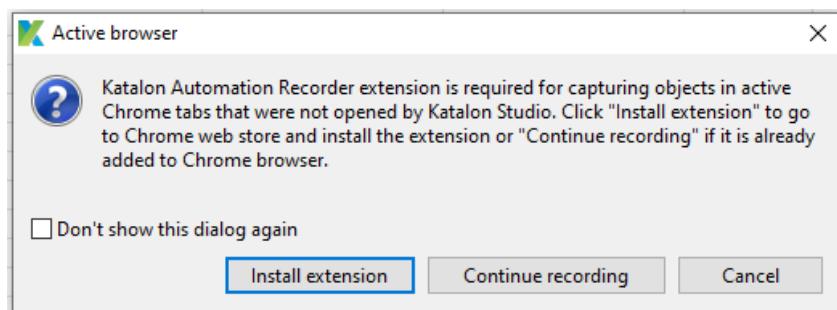
Gambar 2.9 Tool *Record Web*

10. Untuk memulai merekam website, isikan pada bagian URL dengan link dari website yang akan di rekam, contohnya saya akan merekam link <https://opensource-demo.orangehrmlive.com/web/index.php/auth/login>. Kemudian pilih web browser yang hendak digunakan



Gambar 2.10 Jendela Web Recorder

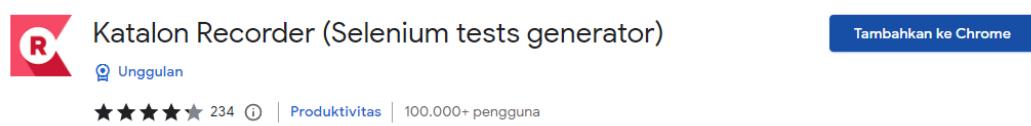
11. Jika muncul pemberitahuan untuk menginstall ekstensi, klik Install Extension



Gambar 2.11 Install Extension

12. Pilih Tambahkan ke Chrome

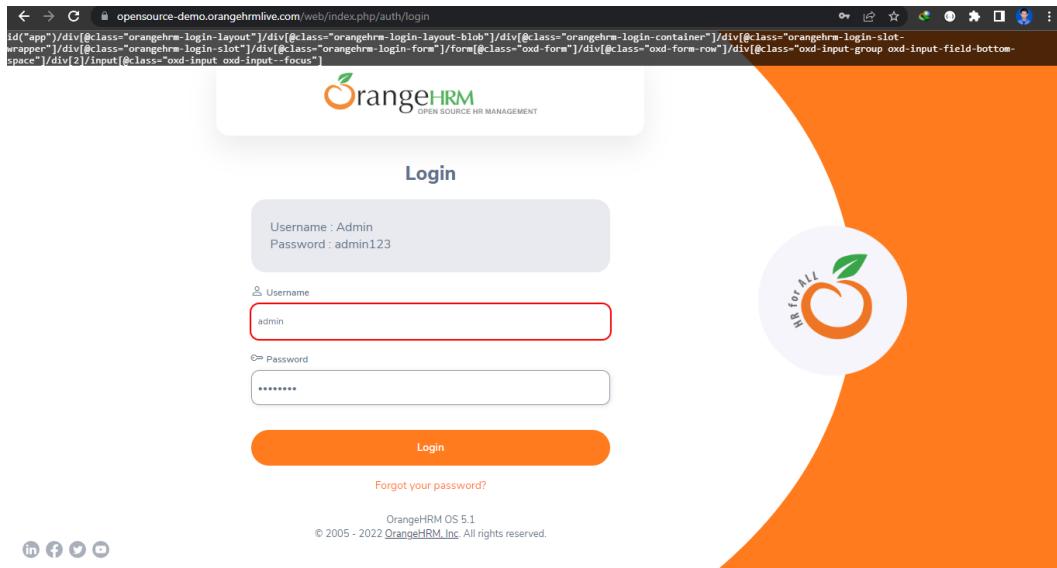
Beranda > Ekstensi > Katalon Recorder (Selenium tests generator)



Gambar 2.12 Extension Katalon Recorder

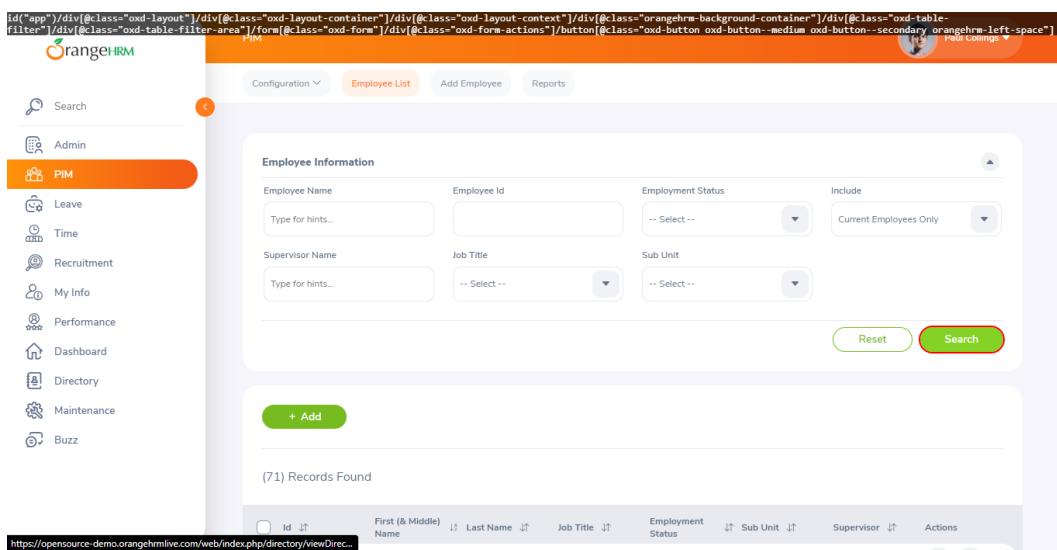
13. Jika sudah menginstal ekstensi, selanjutnya Start dan buka website

<https://opensource-demo.orangehrmlive.com/web/index.php/auth/login>



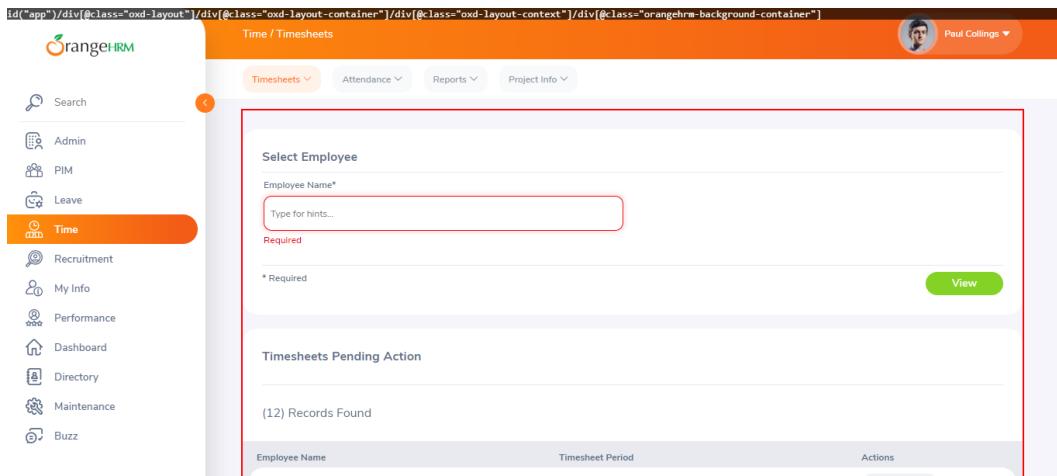
Gambar 2.13 Tampilan Website OpenSource-Demo

14. Kemudian lakukan hal seperti input atau menekan tombol Search



Gambar 2.14 Percobaan pada tombol Search

15. Kemudian buka bagian Time, dan klik View. Maka akan memunculkan *error* seperti di bawah. Klik pada tulisan Required agar objek dari teks masuk ke *recorder*



Gambar 2.15 Menampilkan *alert*

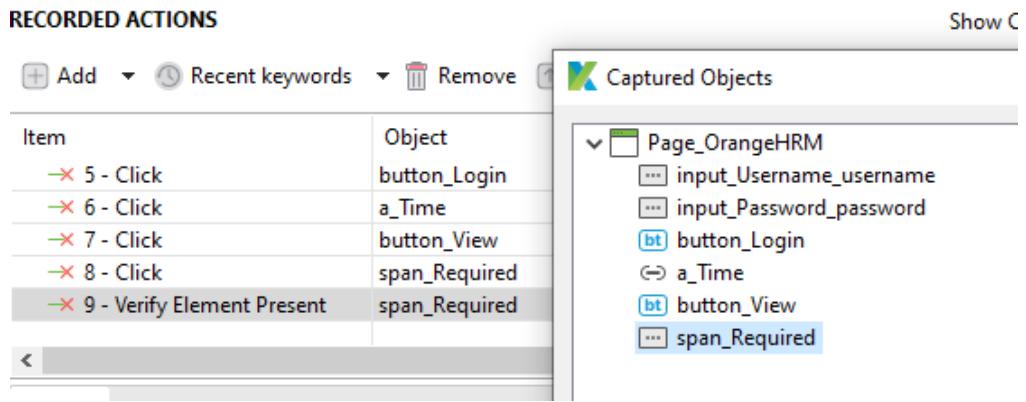
16. Klik Add pada jendela Web Recorder Katalon. Pada item pilih tanda panah, kemudian cari item dengan nama Verify Element Present.

```

@CompileStatic
@com.kms.katalon.core.annotation.Keyword(keywordObject =
StringConstants.KW_CATEGORIZE_ELEMENT)
static boolean verifyElementPresent(com.kms.katalon.core.testobject.TestObject to, int timeOut,
com.kms.katalon.core.model.FailureHandling flowControl)
Verify if the given web element presents on the DOM
throws:
StepFailedException
Parameters:
to - represent a web element
timeOut - system will wait at most timeout (seconds) to return result
flowControl
Returns:
true if element presents; otherwise, false
  
```

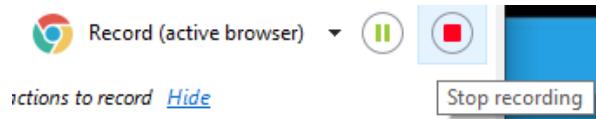
Gambar 2.16 Menambahkan item objek

17. Kemudian klik 2 kali pada bagian kolom element dari item Verify Element Present. Pilih span\_Required dan klik OK



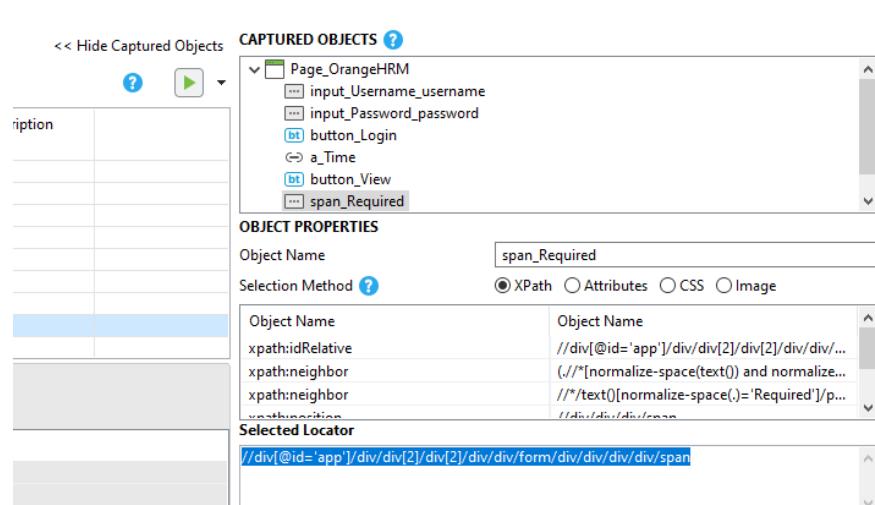
Gambar 2.17 Mengatur objek dari item

#### 18. Kemudian *Stop Recording*



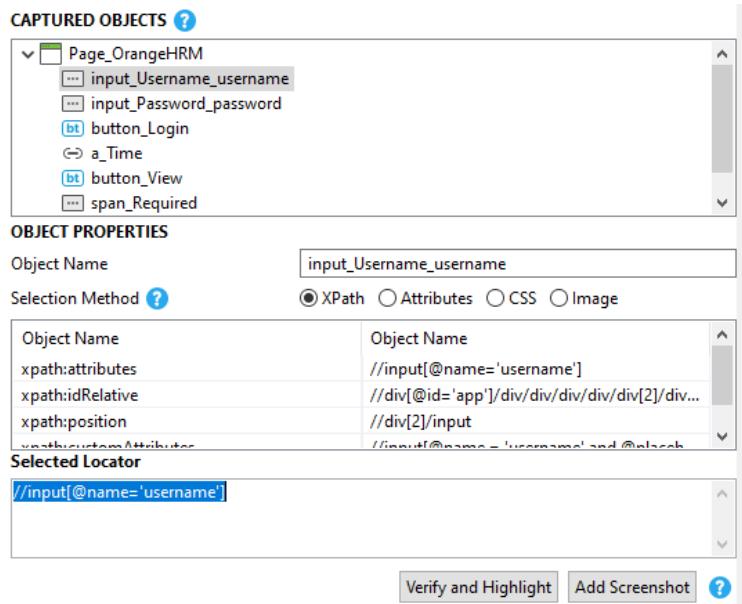
Gambar 2.18 *Stop Recording*

#### 19. Setelah melakukan *Capture* kita dapat melihat telah meng-*capture* satu section dengan nama Page\_OrangeHRM



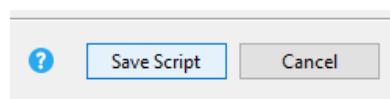
Gambar 2.19 *Captured Objects*

#### 20. Untuk melihat detail/*properties* dari objek, pilih pada objek yang hendak dilihat detail, maka akan muncul detail dibagian Object Properties



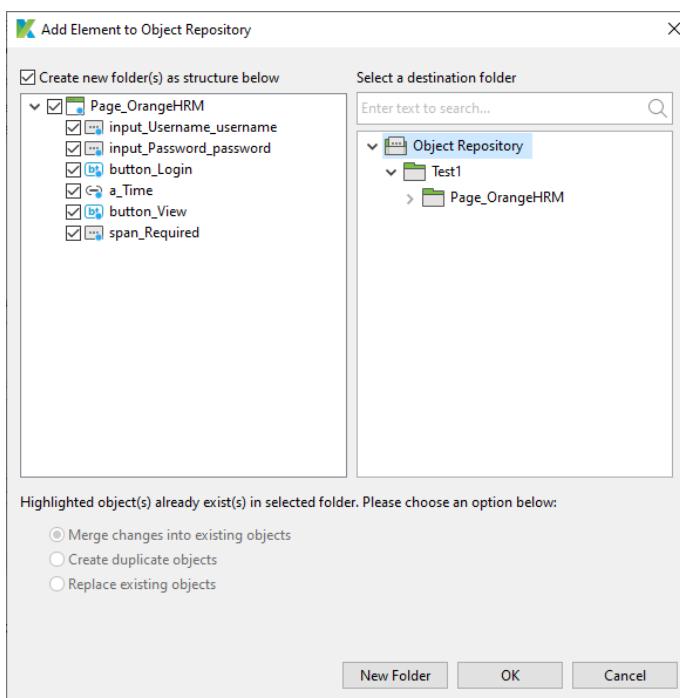
Gambar 2.20 Detail item

21. Untuk menyimpan hasil tes ke test case, tekan tombol Save Script



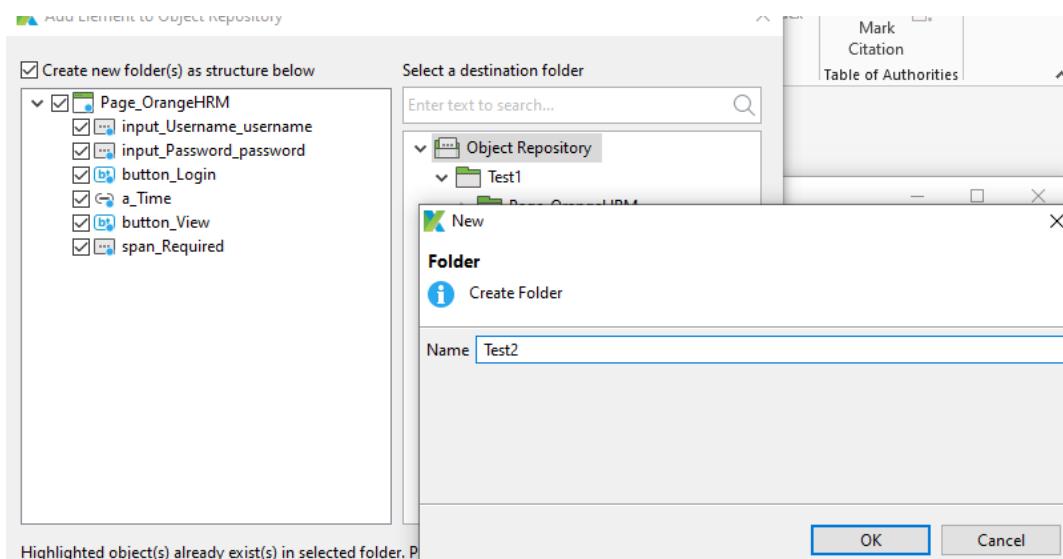
Gambar 2.21 Save Script Recording

22. Pilih elemen apa saja yang akan disimpan, kemudian klik New Folder



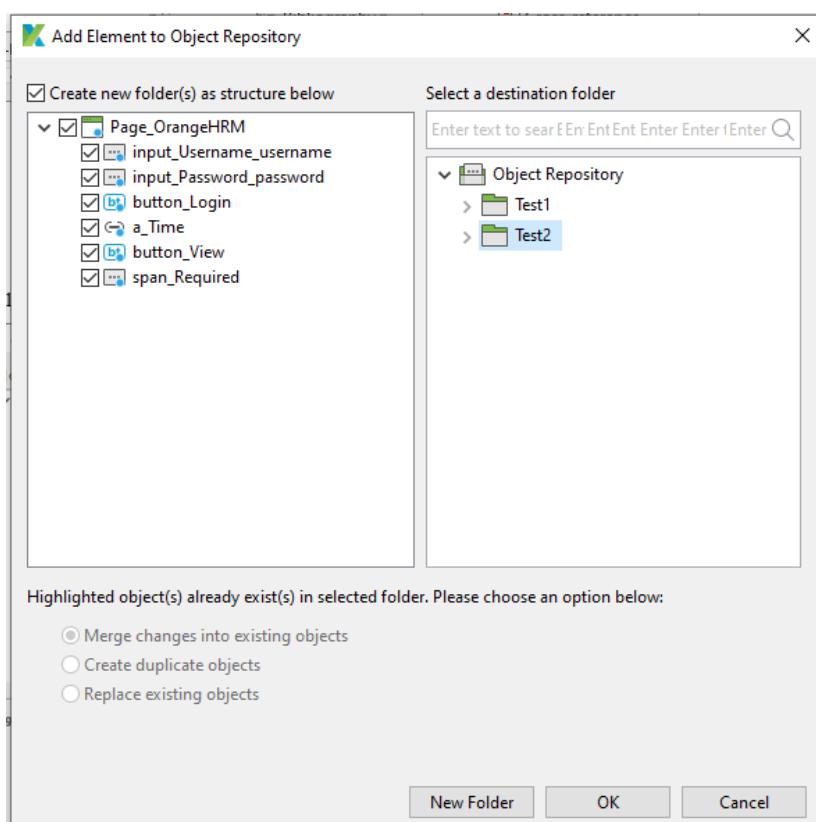
Gambar 2.22 Menyimpan Elemen ke *Object Repository*

23. Buat nama folder dengan nama Test2



Gambar 2.23 Membuat Folder Test2

24. Pilih folder yang telah dibuat (Test2) dan kemudian klik OK



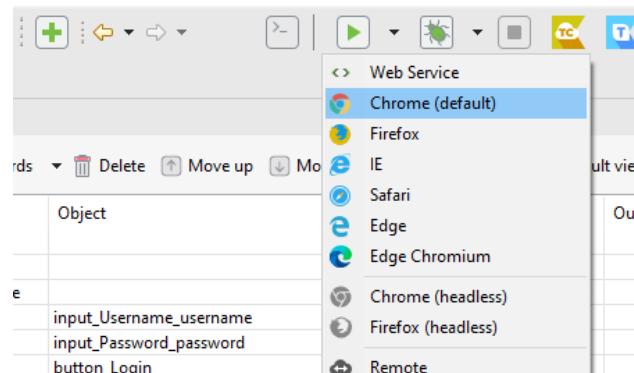
Gambar 2.24 Menyimpan ke folder Test2

25. Maka akan menghasilkan data tes *recorder* seperti berikut, jika ingin mengubah data objek, maka dapat menggunakan metode manual

Item	Object	Input
1 - Open Browser		""
2 - Navigate To Url		"https://opensource-demo.orangehrm.com"
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	button_View	
8 - Click	span_Required	
9 - Verify Element Present	span_Required	0

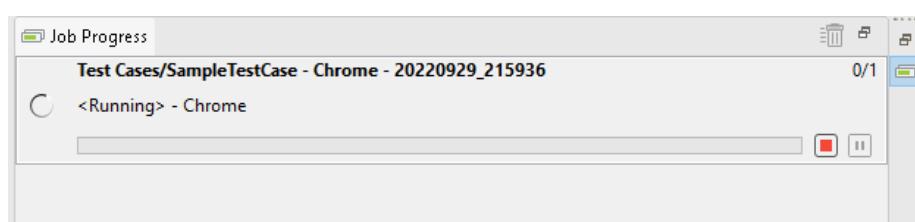
Gambar 2.25 Hasil folder Test2

26. Adapun untuk melakukan *replay* terhadap tes yang telah dilakukan yaitu hal pertama harus save data dari tes terlebih dahulu. Kemudian klik pada icon play dan pilih Chrome (default)



Gambar 2.26 Run in Chrome Browser

27. Proses *replay* akan berjalan secara otomatis



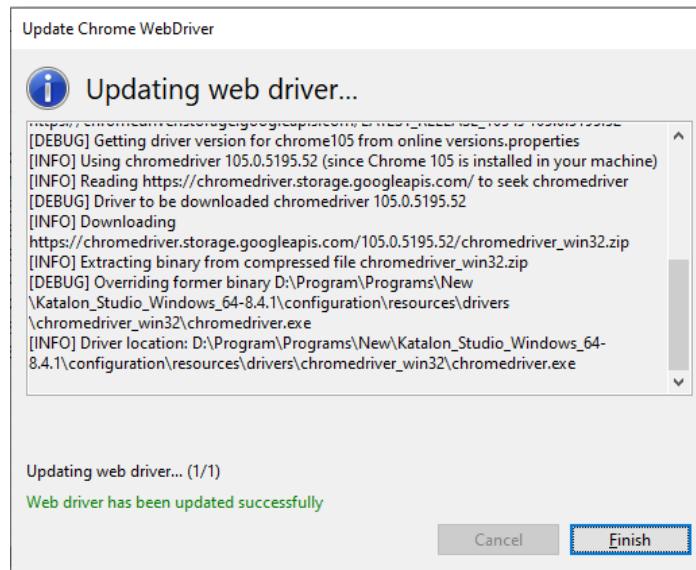
Gambar 2.27 Proses Running

28. Jika muncul error dikarenakan oleh web driver, terlebih dahulu update web driver dengan cara pilih Tools>Update WebDriver>Update Chrome WebDriver



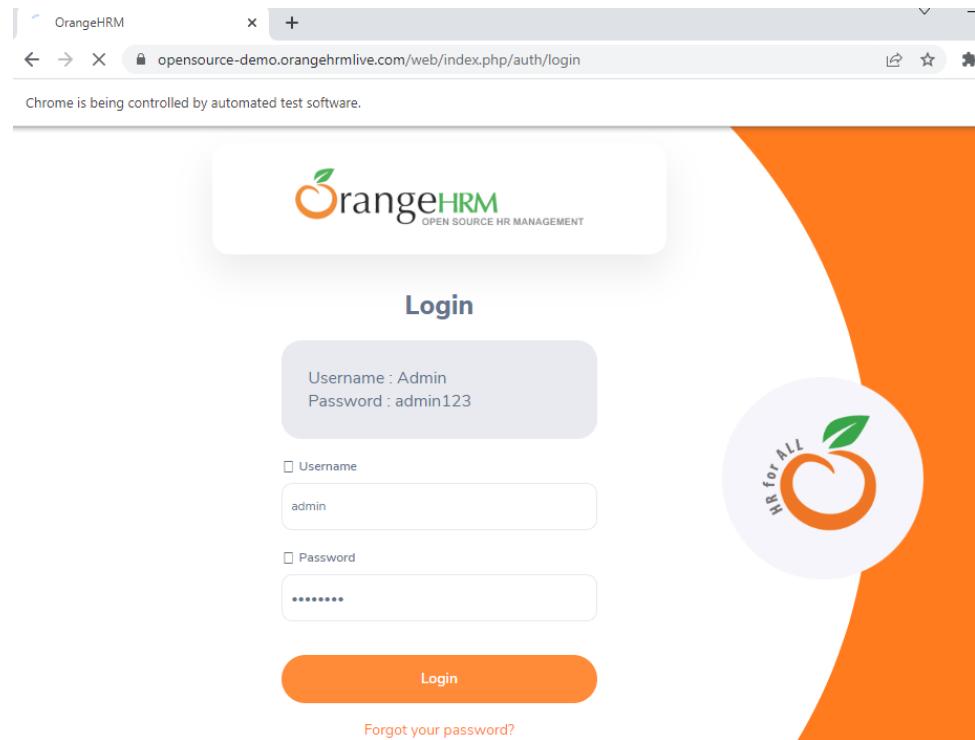
Gambar 2.28 Failed WebDriver

29. Tunggu hingga proses update selesai dan klik Finish

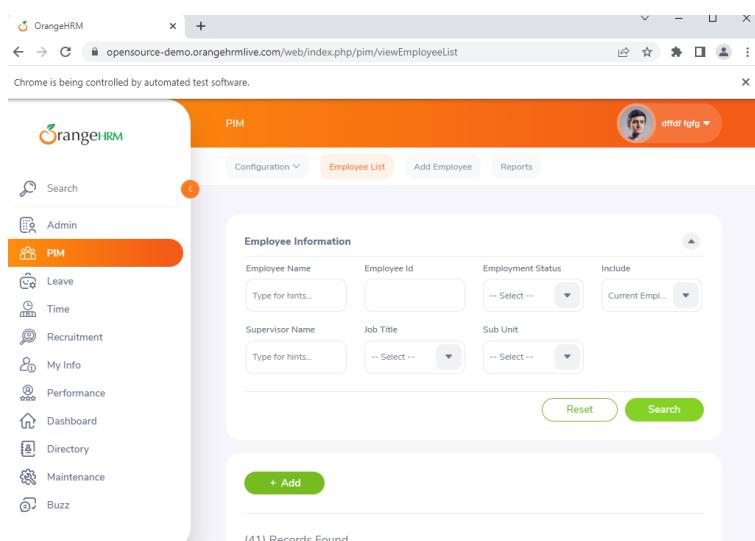


Gambar 2.29 Update WebDriver

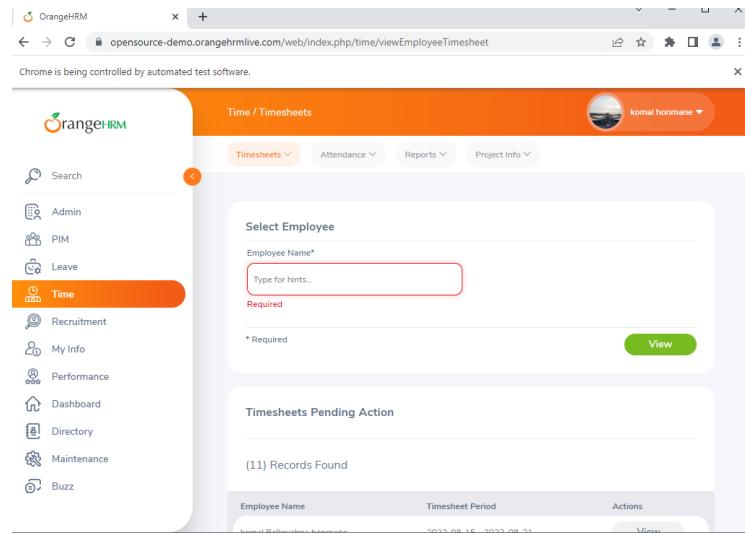
30. Kemudian lakukan *replay* seperti sebelumnya dan tunggu hingga proses pengujian selesai dilakukan secara otomatis



Gambar 2.30 Hasil *Running Login*

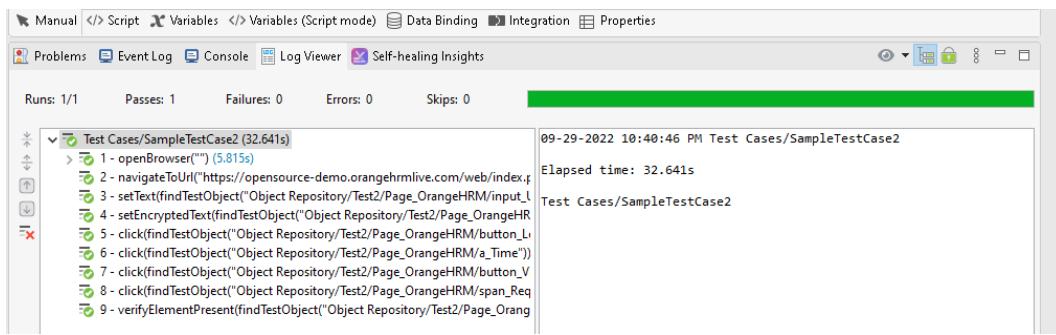


Gambar 2.31 *Running Dashboard*



Gambar 2.32 End Running

31. Maka akan menghasilkan status proses yang berhasil dan memakan waktu selama 32,641 Detik dan dengan 9 tahapan seperti berikut



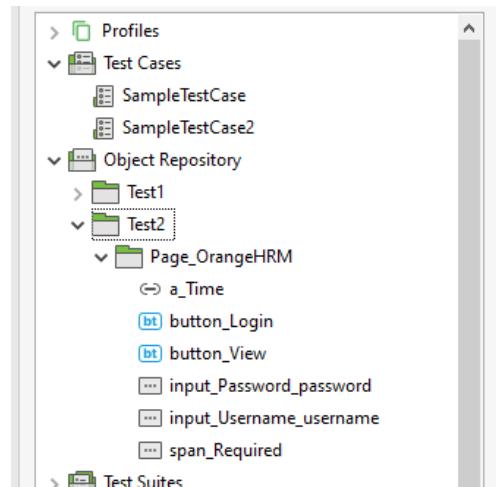
Gambar 2.33 Log Viewer Record Test

Jika suatu ketika anda membutuhkan mengubah atau memperbarui identifikasi dari suatu objek atau kehilangan suatu objek selama proses perekaman, maka ada beberapa tips yang dapat dilakukan untuk mengatasi hal tersebut.

### Tips Pertama

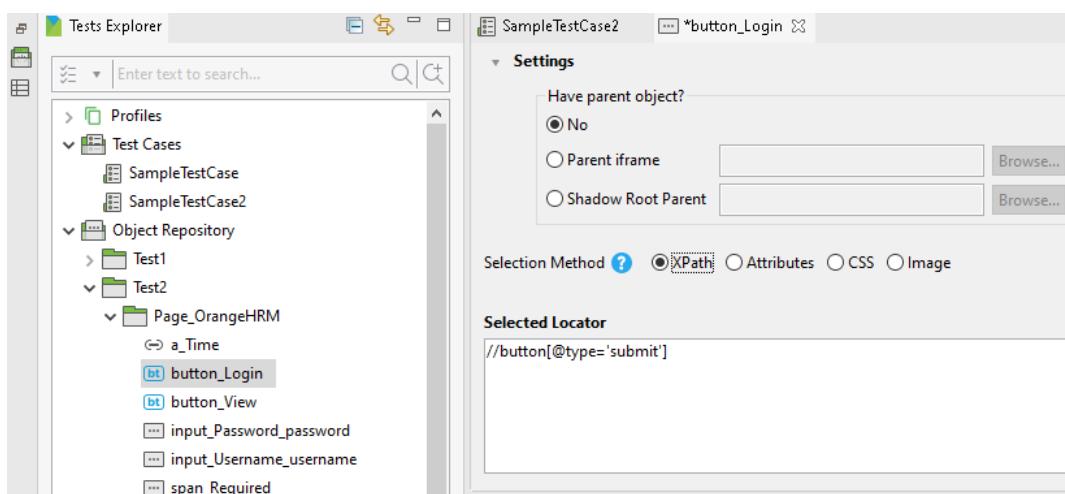
Tips pertama adalah dengan cara mengupdate objek identifikasi menggunakan *object repository*.

1. Adapun langkah jika ingin melakukan perubahan terhadap propertis dari objek adalah dengan memilih objek yang akan diubah dari folder Test2



Gambar 2.34 *Object Repository Test2*

2. Maka akan muncul settings dari objek yang dipilih, pada bagian ini anda dapat mengubah segala sesuatu mengenai objek yang telah dipilih

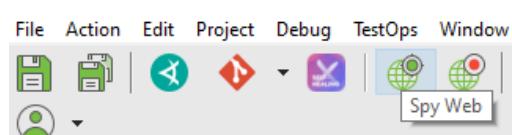


Gambar 2.35 *Settings element*

### Tips Kedua

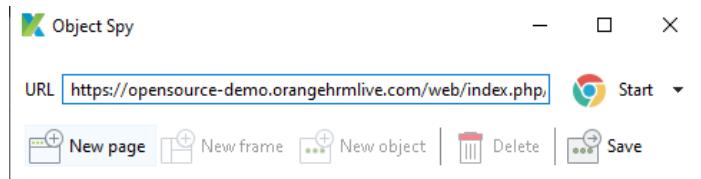
Tips kedua adalah dengan menggunakan fitur dari *Spy Web*, fitur ini berguna untuk meng-*capture* objek baru dari suatu web.

1. Pilih icon *Spy Web*



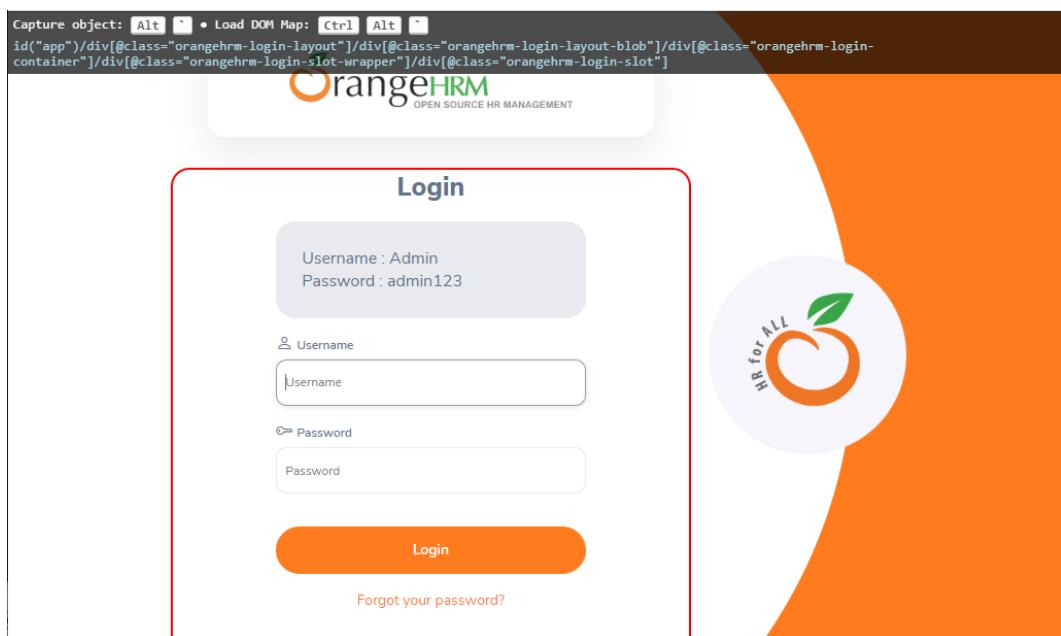
Gambar 2.36 Fitur *Spy Web*

2. Pada kolom URL masukkan link <https://opensource-demo.orangehrmlive.com/web/index.php/auth/login> kemudian klik *Start*



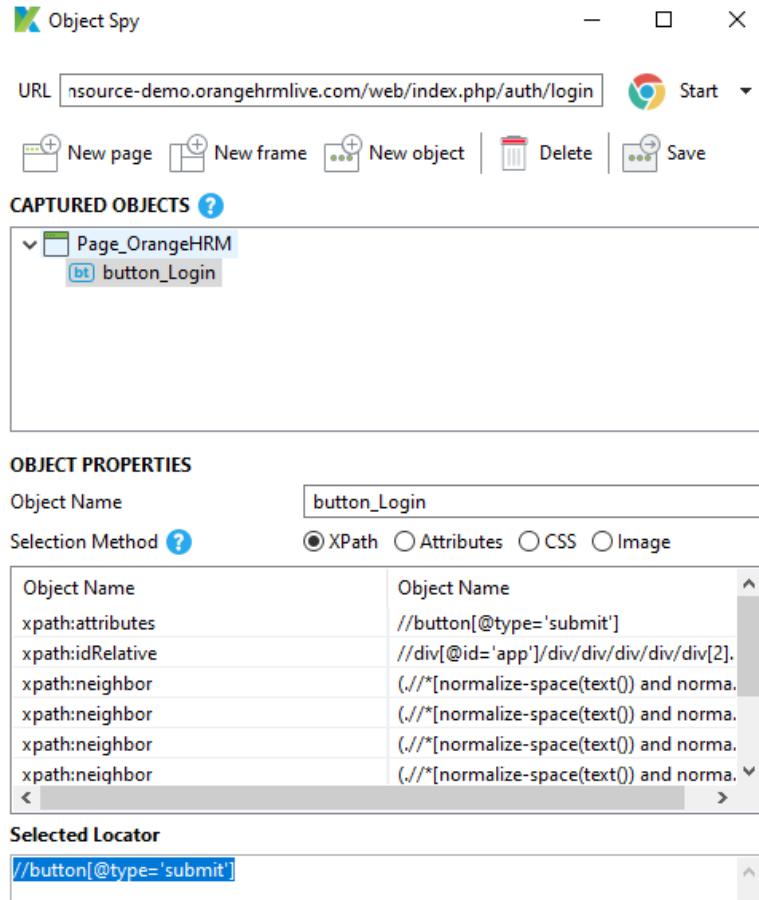
Gambar 2.37 Jendela *Spy Web*

3. Seperti yang terlihat pada gambar di bawah, jika ingin meng-*capture* suatu elemen, maka sebelum click, tekan Alt+~ terlebih dahulu. Pada tahap ini yang akan di *capture* adalah tombol login



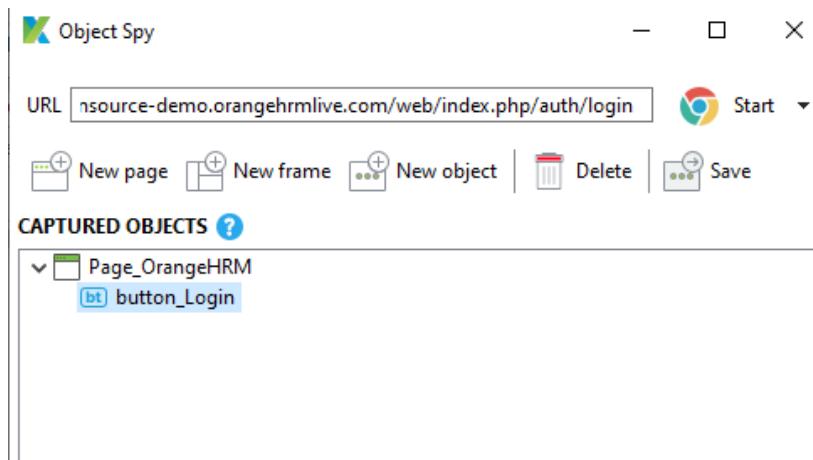
Gambar 2.38 *Capture Login Button*

4. Secara otomatis elemen yang di *capture* akan masuk ke bagian *Captured Objects* dengan menggunakan fitur *Spy Web*



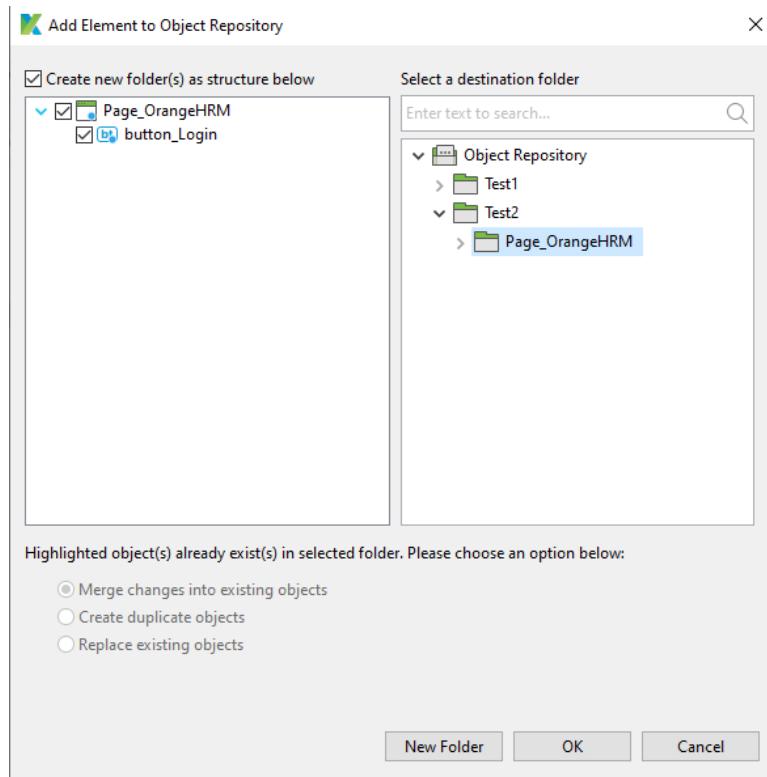
Gambar 2.39 *Object hasil capture*

- Untuk menambahkan elemen yang telah di *capture* ke projek tes adalah dengan cara klik Save



Gambar 2.40 *Captured Objects*

- Dan pilih elemen yang akan di save, kemudian pilih folder dan klik OK



Gambar 2.41 Jendela Add Element to Object Repository

7. Sekian dari tips untuk *capture* elemen menggunakan *Spy Web*

### 2.2.2. Manual Mode

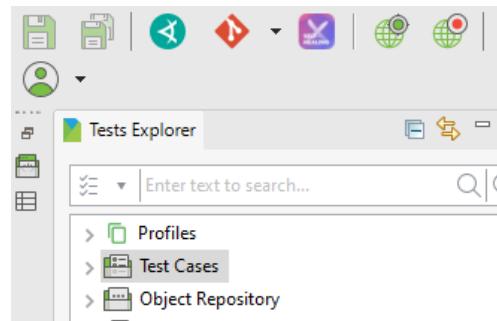
Metode Manual dapat memberikan beberapa peran penting, diantaranya yaitu:

1. Dapat menambahkan kata kunci dan aksi secara bertahap
2. Dapat merujuk ke objek yang diambil
3. Dapat melakukan *drag and drop* secara tepat
4. Dapat mengulang pada segala jenis browser yang tersedia

Pada Manual Mode, ada 3 hal yang akan dilakukan yaitu : Menangkap seluruh elemen yang akan digunakan pada tes, Membuat tes secara manual, dan menjalankan tes.

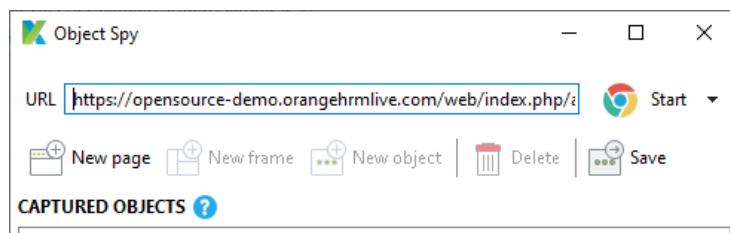
Pada tes secara manual, kita akan mencoba membuat perintah untuk membuka suatu web, kemudian login ke web tersebut berdasarkan dari perintah yang diberikan. Adapun tahap-tahap untuk melakukan metode tersebut sebagai berikut :

1. Buka projek sebelumnya, kemudian pilih *icon Spy Web* untuk menangkap seluruh elemen yang akan digunakan pada tes secara manual



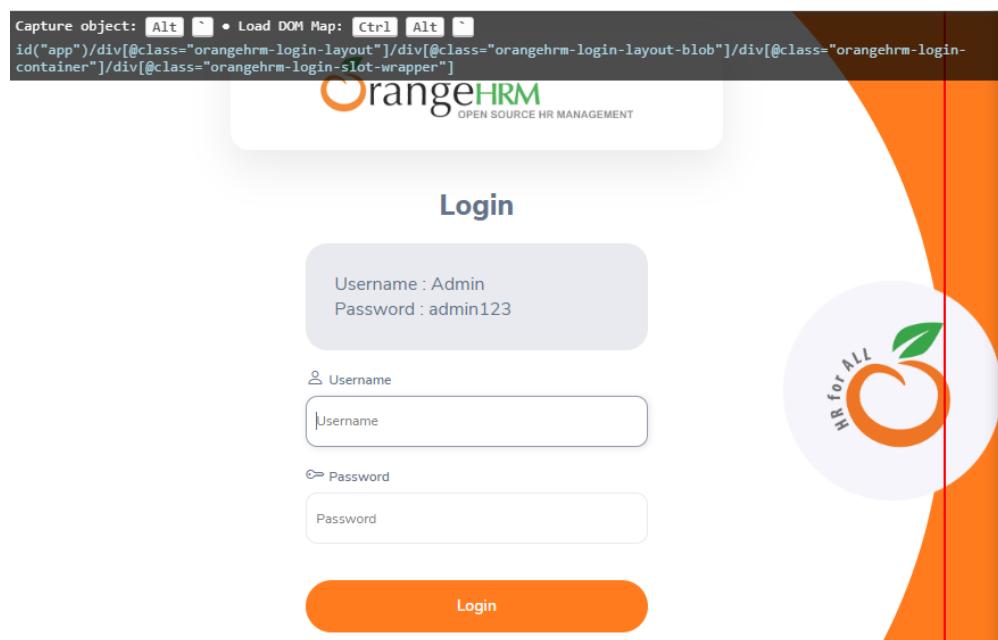
Gambar 2.42 Katalon Toolbar

2. Masukkan link yang sebelumnya, yaitu <https://opensource-demo.orangehrmlive.com/web/index.php/auth/login> pada kolom URL



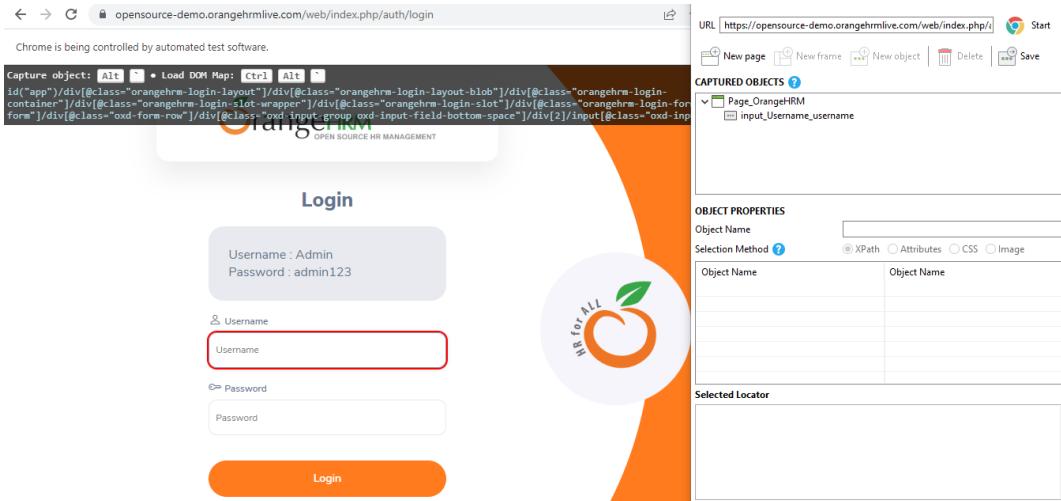
Gambar 2.43 Jendela Spy Web

3. Kemudian lakukan Start pada *Spy Web*, berikut adalah hasil dari *Spy Web*



Gambar 2.44 Spy Web Running

4. Lakukan *capture* terhadap elemen *textfield username* dan *password*, beserta tombol Login juga



Gambar 2.45 *Capture* elemen yang diperlukan

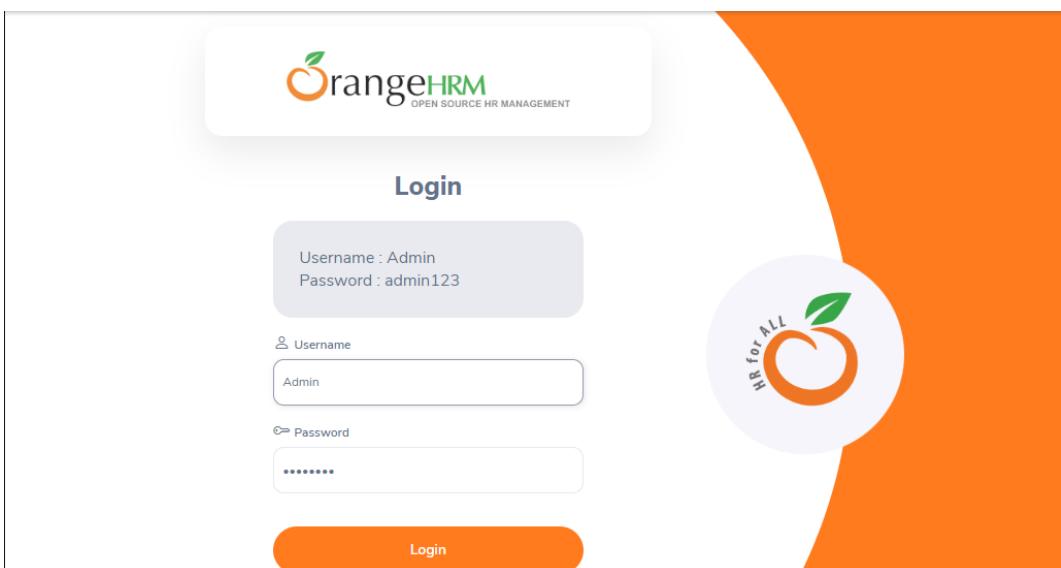
5. Hasil setelah *capture objects*

#### CAPTURED OBJECTS

- ✓ Page\_OrangeHRM
  - ... input\_Username\_username
  - ... input\_Password\_password
  - bt button\_Login

Gambar 2.46 Hasil *Capture Login Page*

6. Kemudian login ke Web dengan akun Admin

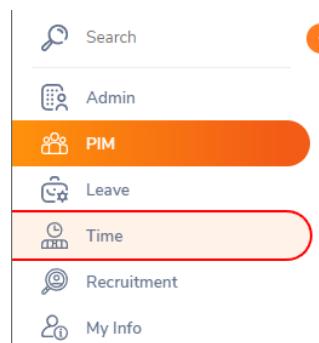


Gambar 2.47 *Login as Admin*

7. Pada halaman *dashboard*, yang akan di *capture* hanyalah bagian foto akun dan tombol menu *Time*

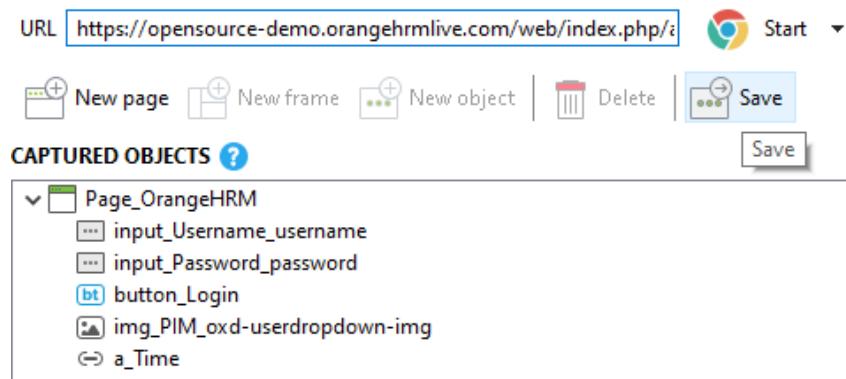


Gambar 2.48 *Capture Picture*



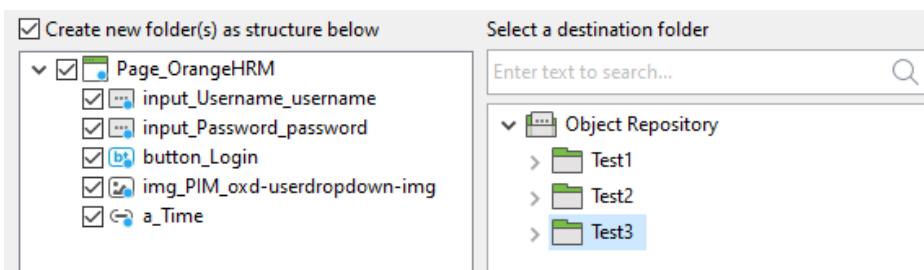
Gambar 2.49 *Capture Menu Time*

8. *Close browser*, inilah hasil *capture* yang telah dilakukan. Lakukan save dengan klik Save



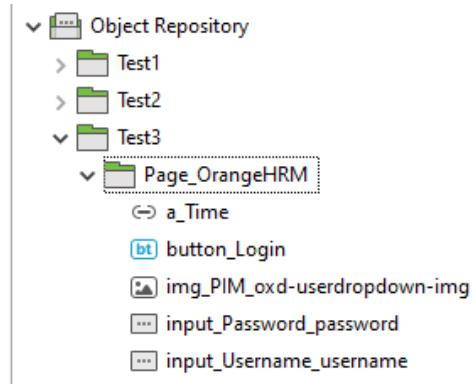
Gambar 2.50 Hasil *Capture Website*

9. Centang semua elemen yang telah di *capture*. Buat folder baru dengan nama Test3 dan klik OK



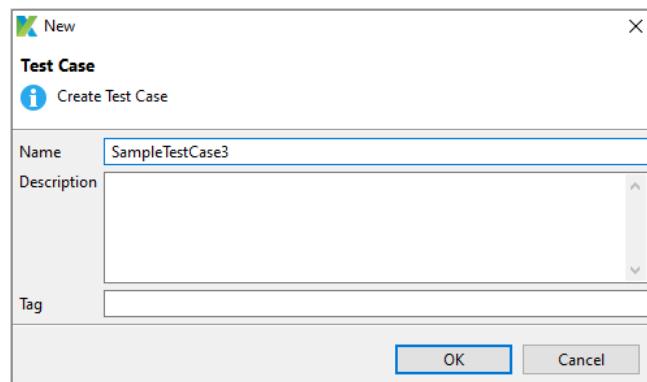
Gambar 2.51 *Save Captured Object to Test3*

10. Anda bisa *close* jendela *Spy Web* sebelumnya. Dan ini adalah hasil dari percobaan *capture* dari *Spy Web* yang dimasukkan ke folder Test3



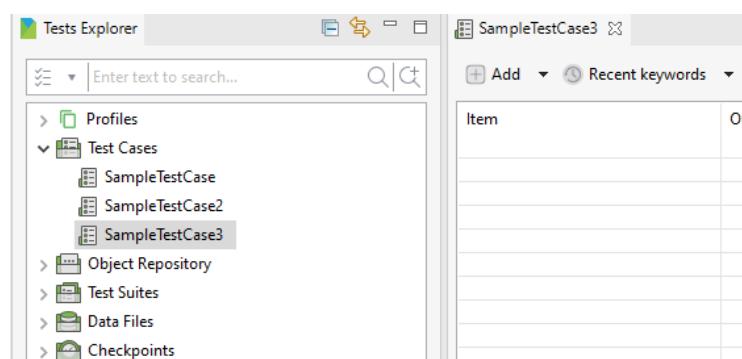
Gambar 2.52 Isi folder Test3

11. Langkah selanjutnya adalah melakukan *testing* secara manual. Untuk melakukan hal tersebut, langkah pertama adalah membuat *Test Case* baru dengan nama SampleTestCase3 dengan cara klik menu File>New>Test Case. Dan beri nama dengan SampleTestCase3 dan OK



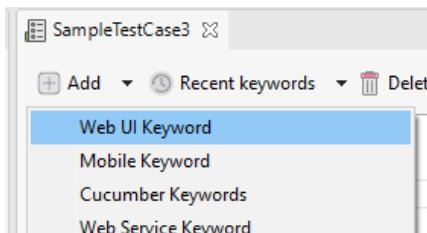
Gambar 2.53 Create a New Test Case

12. Hasil dari *Test Case* dengan nama SampleTestCase3



Gambar 2.54 SampleTestCase3

13. Dan langkah selanjutnya adalah menjalankan *Test case* yang telah dibuat. Sebelum menjalankan hasil *capture* elemen, lakukanlah beberapa konfigurasi agar *test case* dapat menjalankan browser dan membuka link yang dituju dengan benar. Pilih Add>Web UI Keyword



Gambar 2.55 Add Web UI Keyword

14. Pada item yang telah dibuat, pilih Open Browser

Item	Object	Input
1 - Open Browser		""

Gambar 2.56 Create Open Browser Item

15. Pada Input, isikan dengan link dari website yang sebelumnya telah di *capture*

Add	Recent keywords	Delete	Move up	Move down	Edit tags	Set default view

Item	Object	Input	Out
1 - Open Browser		"https://opensource-demo.orangehrm.com"	

Gambar 2.57 Mengisi nilai/value

16. Kemudian Add item baru, dan pilih dengan Set Text. Item ini akan digunakan sebagai nilai/value dari elemen *username* dan *password*

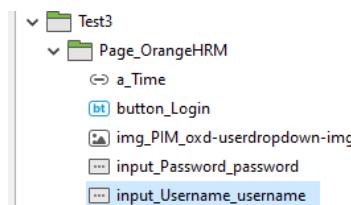
Add	Recent keywords	Delete	Move up	Move down

Item	Object
1 - Open Browser	
2 - Set Text	null

Gambar 2.58 Create Set Text Item

17. Pilih elemen *input\_Username\_username* pada folder Test3 yang sebelumnya telah di *capture* dan OK



Gambar 2.59 Add Object to Set Text

18. Atur nilai kolom input dengan Admin

Item	Object	Input
1 - Open Browser		"https://opensource-demo.orangehrm.com"
2 - Set Text	input_Username_username	"Admin"

Gambar 2.60 Mengisi nilai *username*

19. Buat item untuk *password* juga, dan anda dapat melakukan *drag and drop* dari Object Repository ke kolom Object dari item yang dipilih

Item	Object	Input
1 - Open Browser	input_Username_username	"https://opensource-demo.orangehrm.com"
2 - Set Text	null	"Admin"
3 - Set Text	null	""

Gambar 2.61 Create a New Set Text Item

20. Maka hasilnya seperti berikut

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

Gambar 2.62 Add Value as Password

21. Tambahkan item *Click* untuk tombol login, klik gambar, dan klik menu *Time*

Item	Object	Input
1 - Open Browser	input_Username_username	"https://opensource-demo.orangehrm.com"
2 - Set Text	input_Password_password	"admin123"
3 - Set Text	button_Login	null
4 - Click	img_PIM_oxd-userdropdown-img	null
5 - Click	a_Time	null
6 - Click		

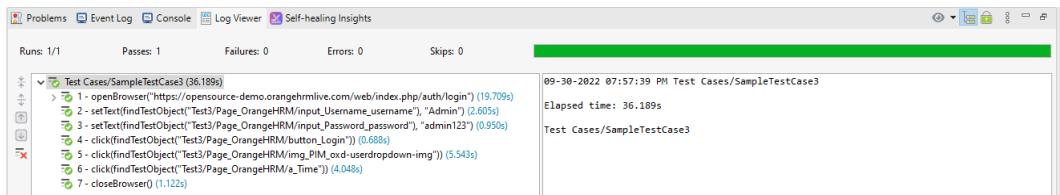
Gambar 2.63 Add Click Item

22. Kemudian tambahkan item untuk *Close Browser*



Gambar 2.64 Add Close Browser Item

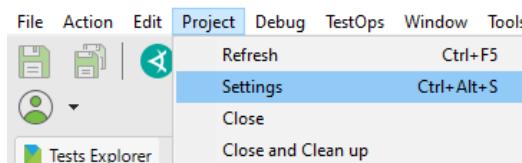
23. Jalankan tes dengan cara klik icon *Run* dan pilih web browser yang akan dijalankan, tunggu hingga tes selesai dilakukan secara otomatis. Setelah selesai browser akan tertutup secara otomatis. Ini adalah hasil dari log *testing*



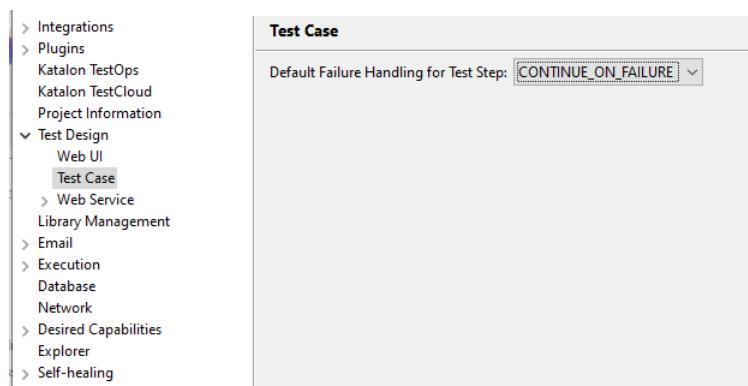
Gambar 2.65 Log Viewer Result for Manual Mode

24. Itulah semua tahap-tahap untuk melakukan tes secara manual.

Adapun tips agar mengatasi jika terjadi *Failure* atau gagal *Test Case* tidak berhenti adalah dengan cara mengatur pada bagian menu Project>Settings>Test Design>Test Case>CONTINUE\_ON\_FAILURE. Dengan begitu, walau terjadi *error* saat menjalankan *Test Case*, program tidak akan berhenti dan akan berjalan dengan melanjutkan proses selanjutnya. Adapun pilihan lainnya yaitu OPTIONAL, pilihan ini akan mengubah proses FAILED menjadi WARNING. Lebih lengkap mengenai *Test Case* ada di link <https://docs.katalon.com/docs/maintain/configure-failure-handling-settings-in-katalon-studio>.



Gambar 2.66 Toolbar Project



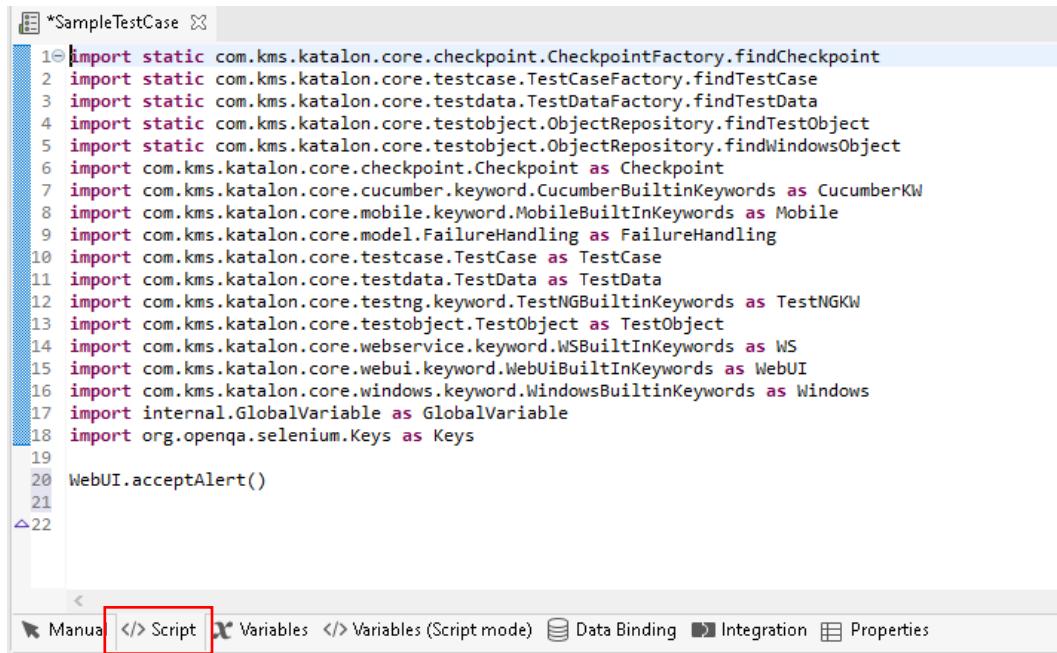
Gambar 2.67 Setting Test Case Failure

### 2.2.3. Script Mode

Pada bagian *Script Mode*, user dapat melakukan tes berupa bahasa Java ataupun Groovy. Adapun beberapa hal yang dapat dilakukan dengan tes menggunakan script yaitu sebagai berikut :

1. Membuat tes melalui *scripting*
2. Dapat menggunakan bahasa Java maupun Groovy
3. Dapat melakukan *drag and drop* objek secara langsung
4. Dapat mengulang pada segala jenis web browser

Adapun cara melakukan *Script Mode* yaitu dengan cara menekan pada bagian Script.



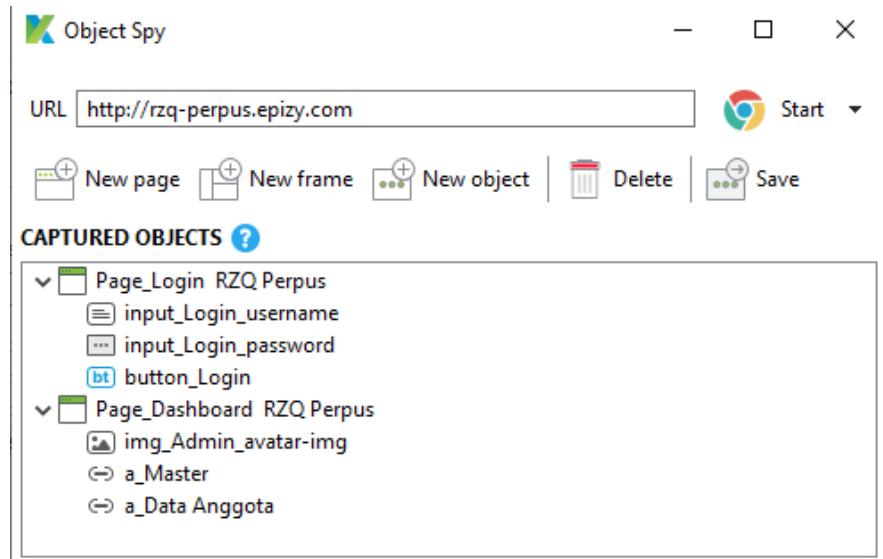
The screenshot shows the Katalon Studio interface with the 'Script' mode tab selected. The code editor displays a Groovy script named 'SampleTestCase'. The script imports various Katalon core classes and methods, such as CheckpointFactory, TestcaseFactory, TestDataFactory, ObjectRepository, Checkpoint, CucumberKW, Mobile, FailureHandling, TestCase, TestData, TestNGBuiltInKeywords, TestObject, WSBuiltInKeywords, WebUIBuiltInKeywords, WindowsBuiltInKeywords, GlobalVariable, and Keys. It concludes with a call to 'WebUI.acceptAlert()'. Below the code editor, the bottom navigation bar is visible, with the 'Script' tab highlighted by a red box.

```
1① import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
2 import static com.kms.katalon.core.testcase.TestCaseFactory.findTestCase
3 import static com.kms.katalon.coretestdata.TestDataFactory.findTestData
4 import static com.kms.katalon.core.testobject.ObjectRepository.findTestObject
5 import static com.kms.katalon.core.testobject.ObjectRepository.findWindowsObject
6 import com.kms.katalon.core.checkpoint.Checkpoint as Checkpoint
7 import com.kms.katalon.core.cucumber.keyword.CucumberBuiltInKeywords as CucumberKW
8 import com.kms.katalon.core.mobile.keyword.MobileBuiltInKeywords as Mobile
9 import com.kms.katalon.core.model.FailureHandling as FailureHandling
10 import com.kms.katalon.core.testcase.TestCase as TestCase
11 import com.kms.katalon.coretestdata.TestData as TestData
12 import com.kms.katalon.core.testng.keyword.TestNGBuiltInKeywords as TestNGKW
13 import com.kms.katalon.core.testobject.TestObject as TestObject
14 import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
15 import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
16 import com.kms.katalon.core.windows.keyword.WindowsBuiltInKeywords as Windows
17 import internal.GlobalVariable as GlobalVariable
18 import org.openqa.selenium.Keys as Keys
19
20 WebUI.acceptAlert()
21
22
```

Gambar 2.68 Tampilan *Script Mode*

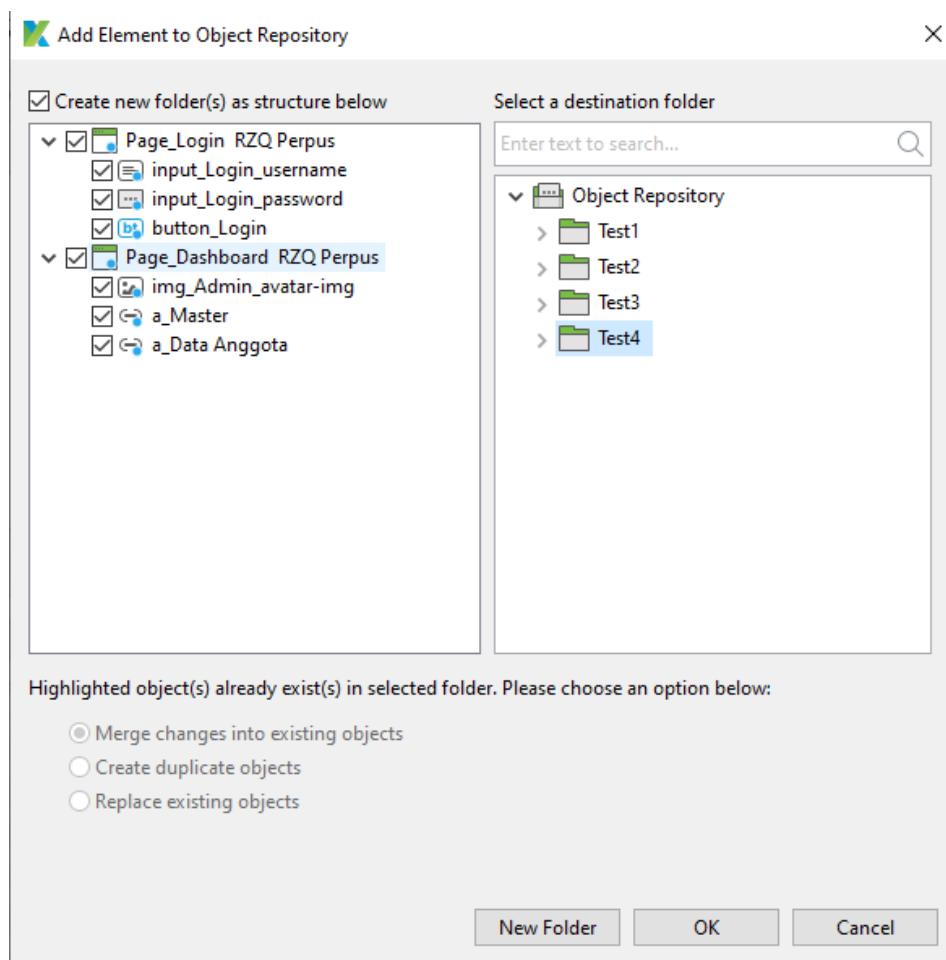
Pada *Script Mode* kita akan mencoba melakukan logi dan klik menu pada website saya sendiri yaitu <http://rzq-perpus.epizy.com>. Tahap-tahap untuk melakukannya yaitu sebagai berikut :

1. Lakukan *capture* elemen seperti yang telah dilakukan sebelumnya menggunakan fitur *Spy Web*. Adapun objek hasil *capture* sebagai berikut



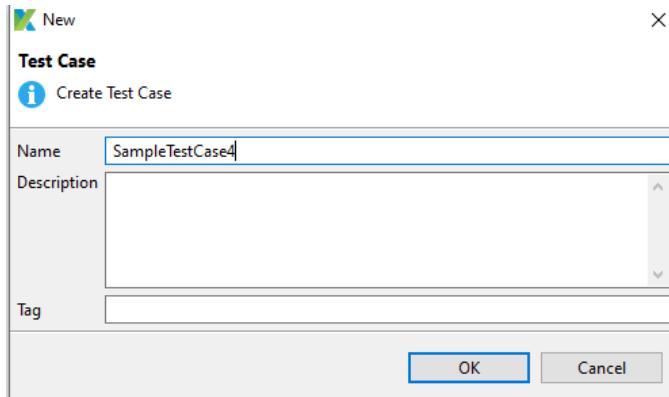
Gambar 2.69 *Captured Objects from rzq-perpus.epizy.com*

2. Simpan seluruh elemen ke dalam folder baru dengan nama Test4 dan OK



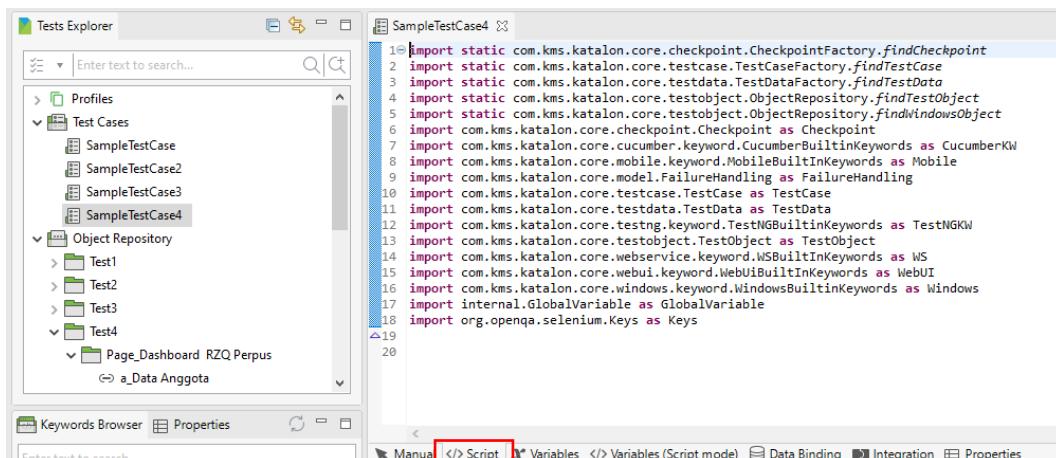
Gambar 2.70 Simpan seluruh elemen ke folder Test4

3. Buat *Test Case* baru dengan nama SampleTestCase4



Gambar 2.71 Create a New Test Case

4. Buka file SampleTestCase4 dan masuk ke bagian *Script*. Abaikan *code import* yang ada, secara *default* Katalon telah mengimport seluruh *package* yang diperlukan



Gambar 2.72 Open Script Tool

5. Berikut adalah hal yang akan dilakukan melalui *Script Mode*. *Script Mode* mendukung penggunaan Bahasa Java

```

1⑩ import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
19
20
21 // Buka Browser
22
23
24 // Input username dan password
25
26
27 // Click Login
28
29
30 // Click gambar
31

```

Gambar 2.73 Planning Script Path

6. Berikut adalah *script* untuk membuka browser, yaitu dengan cara memanggil fungsi OpenBrowser() dengan mengisi parameter berupa string dari URL

```

21 // Buka Browser
22 WebUI.openBrowser("http://rzq-perpus.epizy.com")
23

```

Gambar 2.74 *Open Browser Script*

7. Berikut *script* untuk input *username* dan *password*. setText() berguna untuk menginput teks, dan parameter dari setText() ada 2, yaitu objek dan nilai/*value* dari teks. Untuk menambahkan objek pada parameter setText(), anda bisa melakukan *drag and drop* objek ke dalam parameter

```

24 // Input username dan password
25 WebUI.setText(findTestObject('Object Repository/Test4/Page_Login_RZQ_Perpus/input_Login_username'), "admin")
26 WebUI.setText(findTestObject('Object Repository/Test4/Page_Login_RZQ_Perpus/input_Login_password'), "admin")
27

```

Gambar 2.75 *Set Text Script*

8. Berikut *script* untuk *click* tombol login dengan menggunakan fungsi click(). Parameter yang diterima adalah objek yang akan diklik

```

28 // Click Login
29 WebUI.click(findTestObject('Object Repository/Test4/Page_Login_RZQ_Perpus/button_Login'))
30

```

Gambar 2.76 *Click to Login Button*

9. Berikut *script* untuk *click* pada gambar

```

31 // Click gambar
32 WebUI.click(findTestObject('Object Repository/Test4/Page_Dashboard_RZQ_Perpus/img_Admin_avatar-img'))
33

```

Gambar 2.77 *Click to Image*

10. Berikut adalah proses terakhir, yaitu *Close Browser*

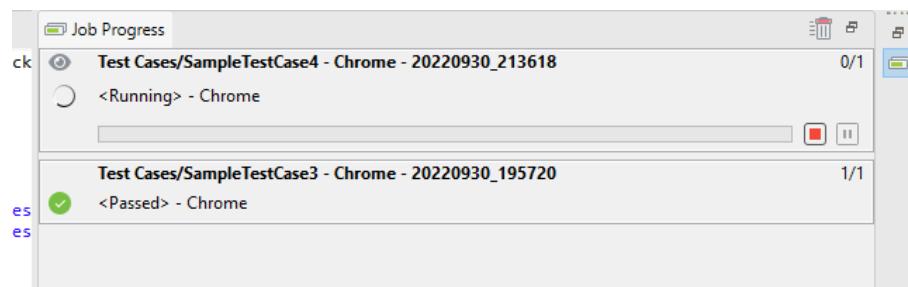
```

34
35 // Close Browser
36 WebUI.closeBrowser()
37

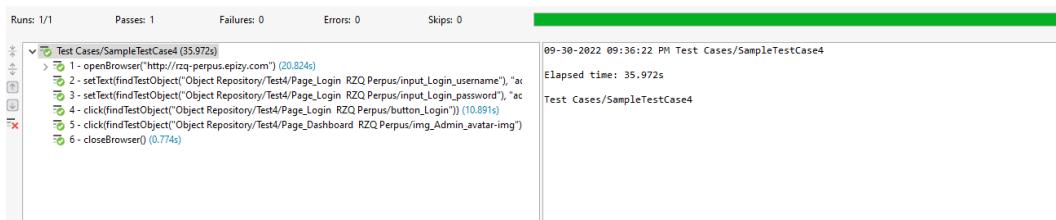
```

Gambar 2.78 *Close Browser Script*

11. Jalankan *Test* dengan menekan Run atau Ctrl+Shift+A



Gambar 2.79 *Test Script Mode on Running*



Gambar 2.80 *Test Success Running*

12. Dengan melakukan tes dan berhasil terjalankan hingga selesai, maka tes secara *script* telah berhasil dilakukan.

Jika hendak membuat salah satu dari script tidak berjalan pada *script mode*, cukup dengan hanya menambahkan notasi `not_run:` sebelum tulisan *script*. Contohnya seperti berikut

```
35 // Close Browser
36 not_run:WebUI.closeBrowser()
37
```

Gambar 2.81 *Notation of Disable Script*

*Script* tersebut akan membuat browser tidak menutup secara otomatis.

Adapun *Quick Tips* dari ketiga mode yang telah dibahas yaitu :

- Untuk memulai tes, pertama mulailah dengan melakukan *Record* atau pengambilan data elemen dari web terhadap tes yang akan dilakukan
- Kemudian pergi ke bagian manual mode dan *update/edit* elemen yang dibutuhkan
- *Script Mode* mungkin akan dibutuhkan sebagai pembuatan aksi yang bebas
- Ulangi tes yang telah dibuat

## BAB 3

### TEST SUITE

#### 3.1. Tujuan

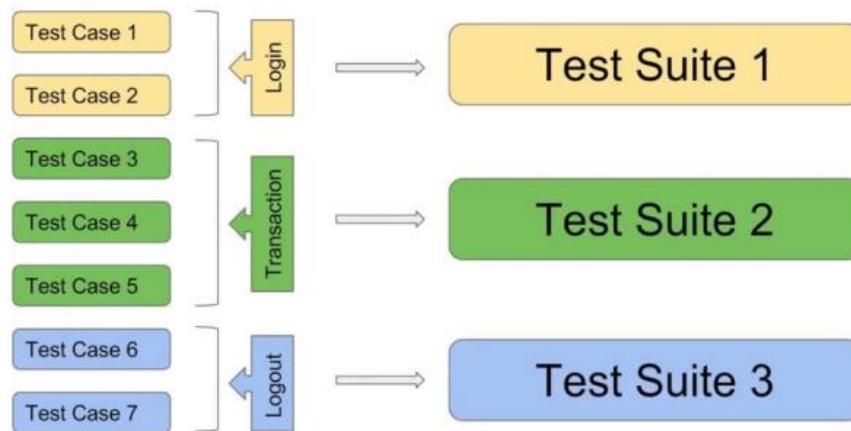
Pada bagian ini akan membahas tentang Test Suite (TS) dan Test Suite Parallel pada katalon studio, diharapkan :

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

#### 3.2. 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.



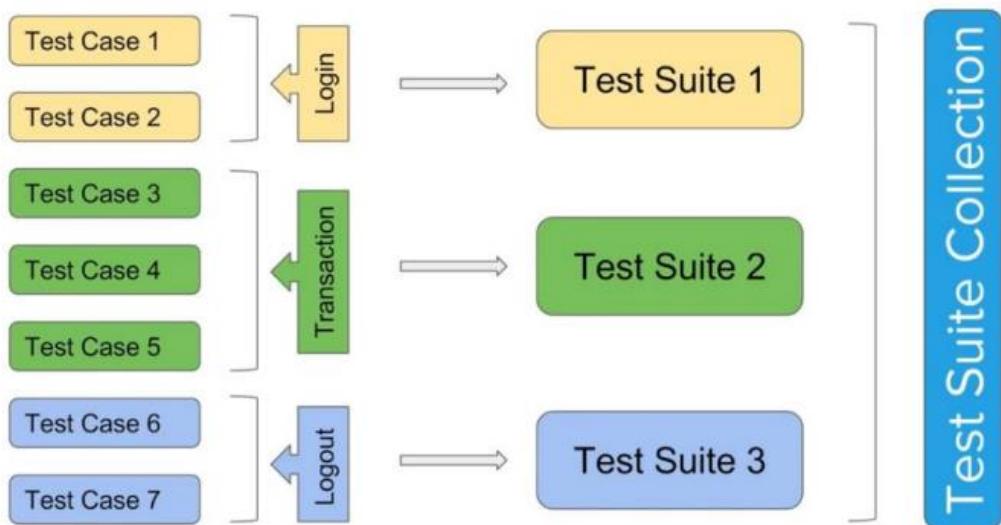
Gambar 3.1 test suite

##### 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.

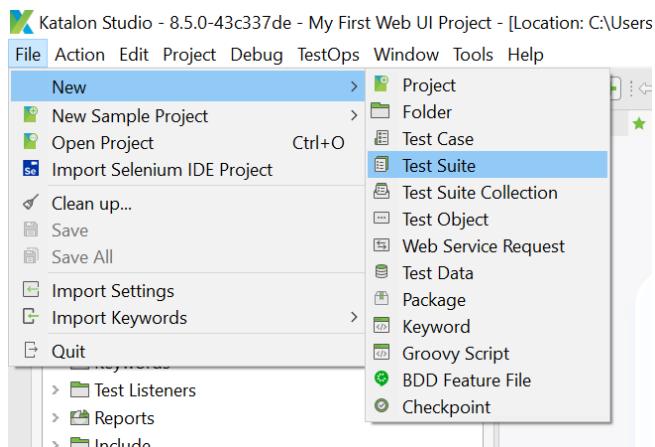


Gambar 3.2 test suite paraller

### 3.3. Percobaan

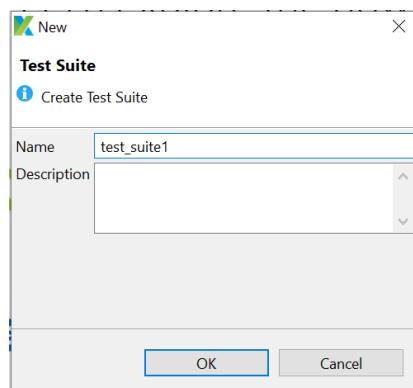
A. Berikut ini adalah Langkah-langkah untuk membuat Test Suite :

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



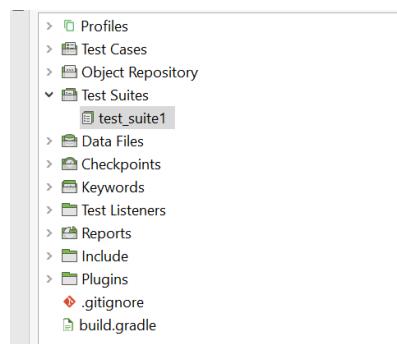
Gambar 3.3 Tampilan pemilihan langkah pertama

- Atur nama Test Suite dengan test\_suite1



Gambar 3.4 Tampilan atur nama test suite

- Hasil setelah membuat Test Suite



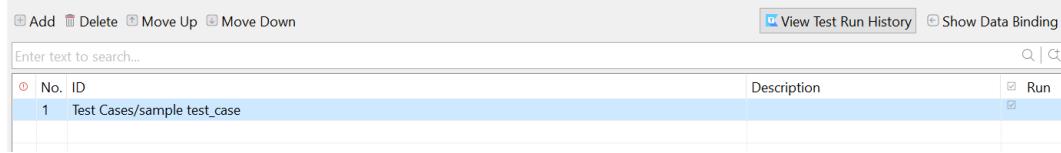
Gambar 3.5 Tampilan hasil setelah membuat test suite

- 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 3.6 Tampilan test case browser

- Hasil setelah memasukkan Test Case ke dalam Test Suite, kemudian tambahkan centang di sample\_test\_case pada bagian run

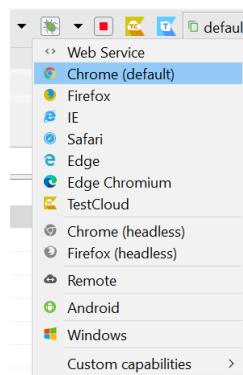


The screenshot shows a table in the TestRail interface. The columns are 'No. ID', 'Description', and 'Run'. There is one row with the ID '1' and the description 'Test Cases/sample test\_case'. A checkbox in the 'Run' column is checked.

No. ID	Description	Run
1	Test Cases/sample test_case	<input checked="" type="checkbox"/>

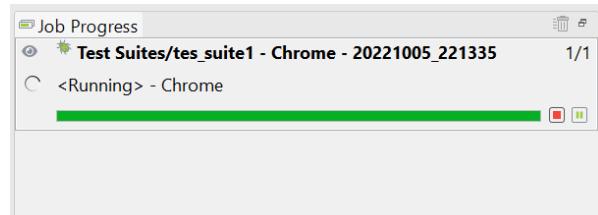
Gambar 3.7 Tampilan hasil setelah masuk test case ke test suite

- Kemudian jalankan pada browser, dan lihat hasilnya



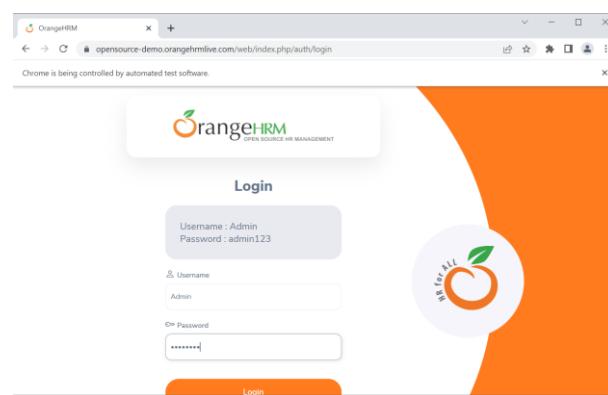
Gambar 3.8 Tampilan browser

- Tunggu hingga proses selesai berjalan

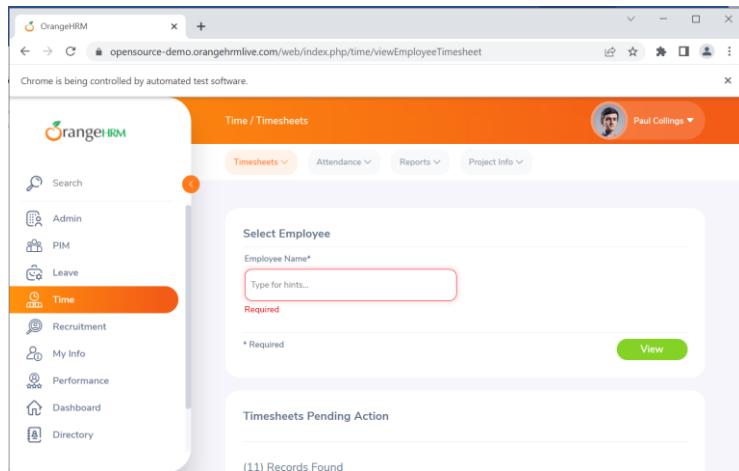


Gambar 3.9 Tampilan proses berjalan

- Tampilan Test Suite setelah dijalankan



Gambar 3.10 Tampilan website orangehrmlive



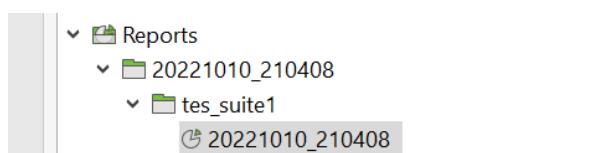
Gambar 3.11 Tampilan hasil test suite

- 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 3.12 Tampilan detail test suite

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



Gambar 3.13 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	

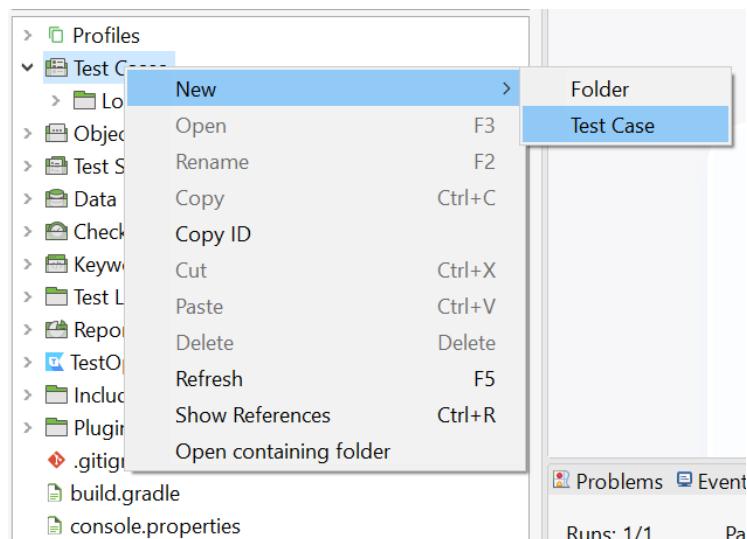
Gambar 3.14 Tampilan hasil laporan test suite

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

Summary		Execution Settings	Execution Environment
Test Suite ID	<a href="#">Test Suites/tes suite1</a>		
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		

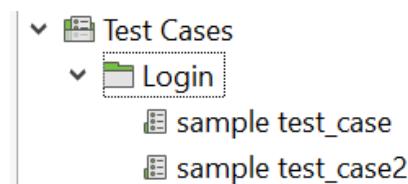
Gambar 3.15 Tampilan daftar laporan

- 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 3.16 Tampilan menambahkan test case pada test suite

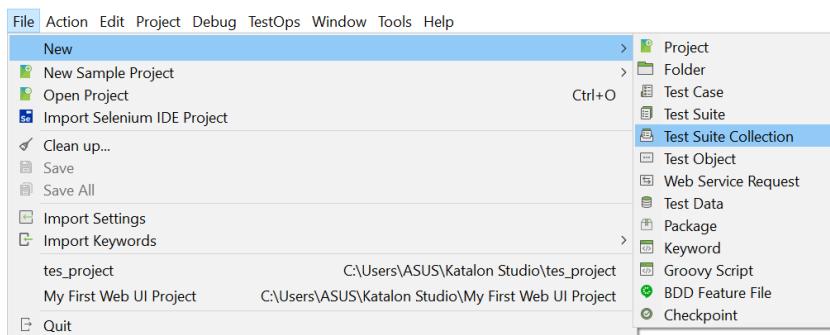
- Kemudian pindahkan file sample test ke dalam folder login dan inilah hasil setelah membuat folder Login.



Gambar 3.17 Tampilan file sample test ke dalam folder login

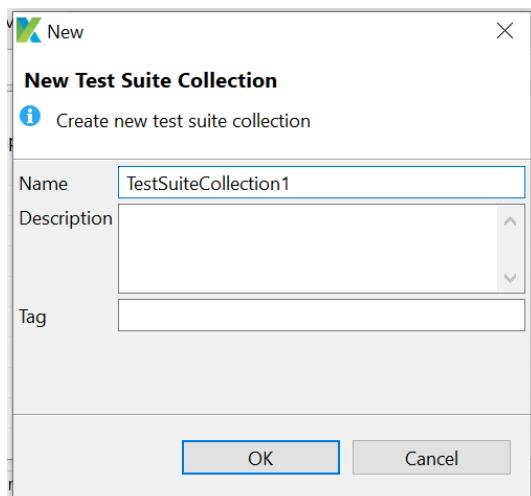
B. Berikut ini adalah langkah-langkah untuk menjalankan Test Suite secara parallel dengan menggunakan Test Suite Collection :

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



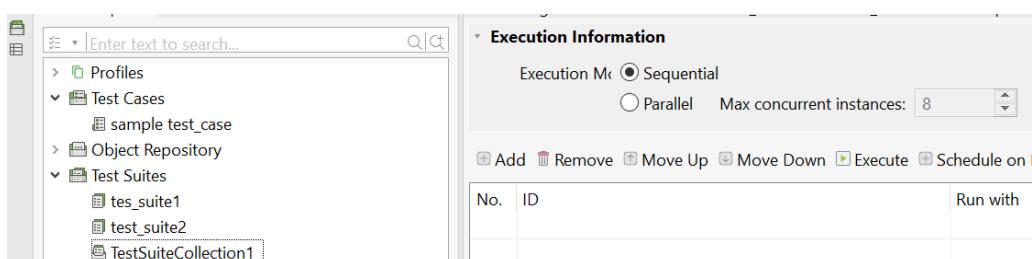
Gambar 3.18 Tampilan test suite collection

- Atur nama dengan Test Suite Collection1 dan OK



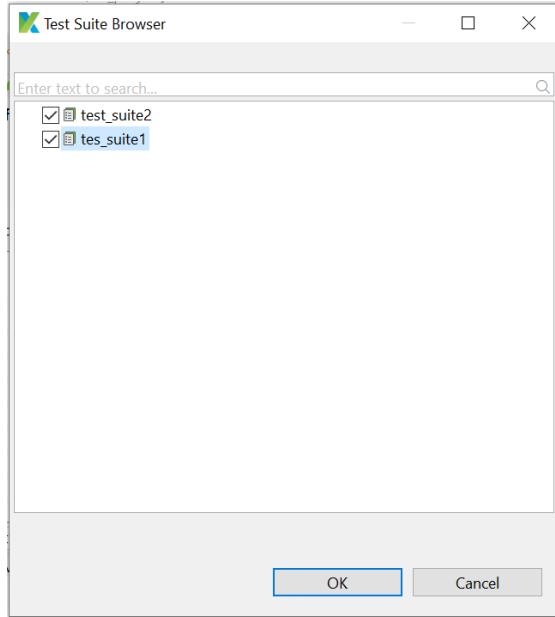
Gambar 3.19 Tampilan create new test suite

- Berikut adalah tampilan dari Test Suite Collection



Gambar 3.20 Tampilan hasil test suite collection

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



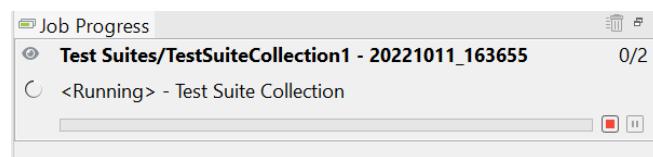
Gambar 3.21 Tampilan test suite collection browser

- 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					
Execution Mode					
<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	
1	Test Suites/test_suite2	Firefox		default	<input type="checkbox"/> Run
2	Test Suites/tes_suite1	Chrome		default	<input type="checkbox"/>

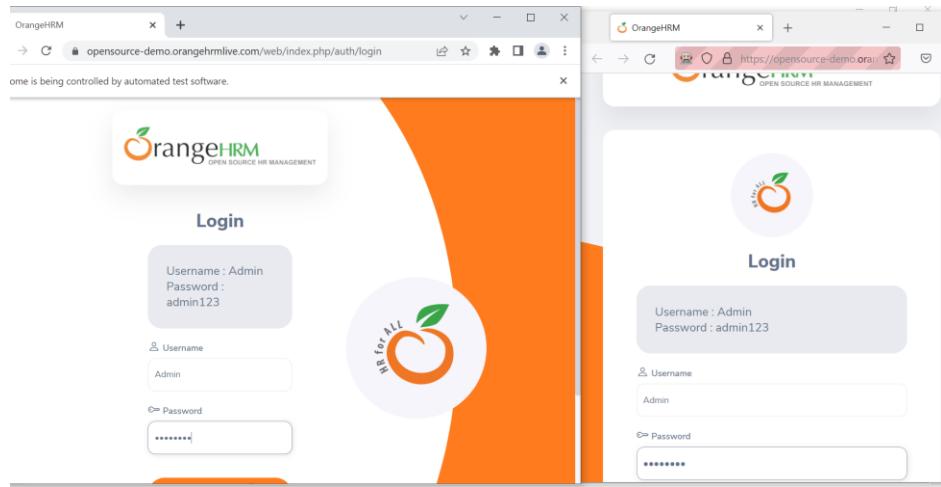
Gambar 3.22 Tampilan hasil setelah test suite collection dirun

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



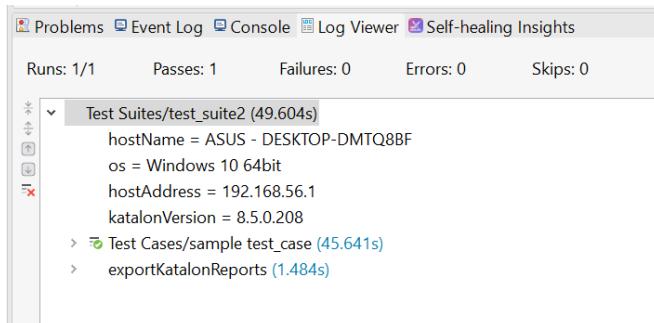
Gambar 3.23 Tampilan proses berjalan

- 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).



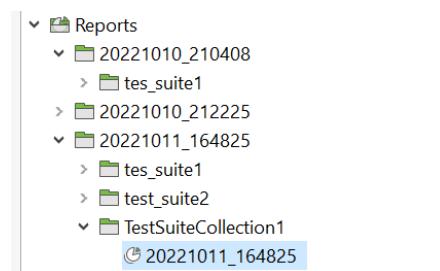
Gambar 3.24 Tampilan hasil browser

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



Gambar 3.25 Tampilan isi dari Log Viewer

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



Gambar 3.26 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 3.27 Tampilan isi dari file laporan test suite collection

- Hasil dari menekan show detail dengan browser chrome.

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	<a href="#">Export report</a> <a href="#">Katalon TestOps</a> <a href="#">Show Test Case Details</a>
Search here...	
No.	Name
> 1	<a href="#">sample test_case (23.589s)</a>
<a href="#">Resolution</a> <a href="#">Video</a>	
<a href="#">Summary</a> <a href="#">Execution Settings</a> <a href="#">Execution Environment</a>	
<b>Test Suite ID:</b> <a href="#">Test Suites/tes_suite1</a>	
Host name	ASUS - DESKTOP-DMTQ8BF
Katalon version	8.5.0.208
Start	2022-10-11 16:55:07
Elapsed	1m - 4.447s
Total TC	1
Passed	1
Error	0
Incomplete	0
Failed	0
Skip	0

Gambar 3.28 Tampilan show detail dibrowser chrome

- Hasil dari menekan show detail dengan browser firefox.

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	<a href="#">Export report</a> <a href="#">Katalon TestOps</a> <a href="#">Show Test Case Details</a>
Search here...	
No.	Name
> 1	<a href="#">sample test_case (45.641s)</a>
<a href="#">Resolution</a> <a href="#">Video</a>	
<a href="#">Summary</a> <a href="#">Execution Settings</a> <a href="#">Execution Environment</a>	
<b>Test Suite ID:</b> <a href="#">Test Suites/test_suite2</a>	
Host name	ASUS - DESKTOP-DMTQ8BF
Katalon version	8.5.0.208
Start	2022-10-11 16:54:54
Elapsed	46.344s
Total TC	1
Passed	1
Error	0
Incomplete	0
Failed	0
Skip	0

Gambar 3.29 Tampilan show detail dibrowser firebox

## BAB 4

### MENGAMBIL DATA CSV

#### 4.1. 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*.

#### 4.2. 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.

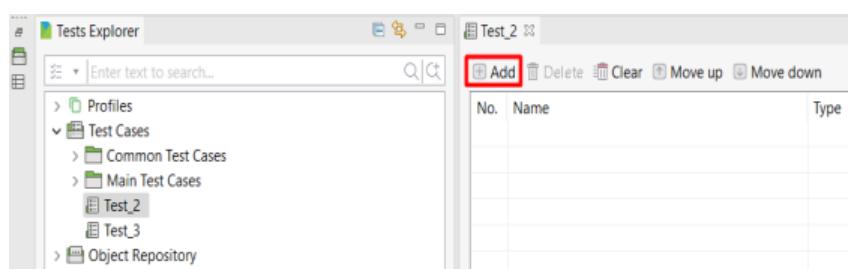
#### 4.3. Percobaan

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

	A	B
1	Username	Password
2	Admin	admin123
3	Admin2	admin1234

Gambar 4.1 File Excel

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



Gambar 4.2 Add Variabels

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

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

Gambar 4.3 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 4.4 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

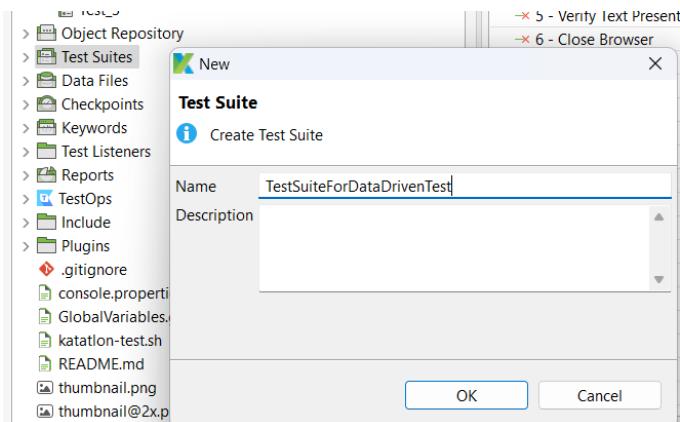
Gambar 4.5 Object Password

6. Kemudian save.



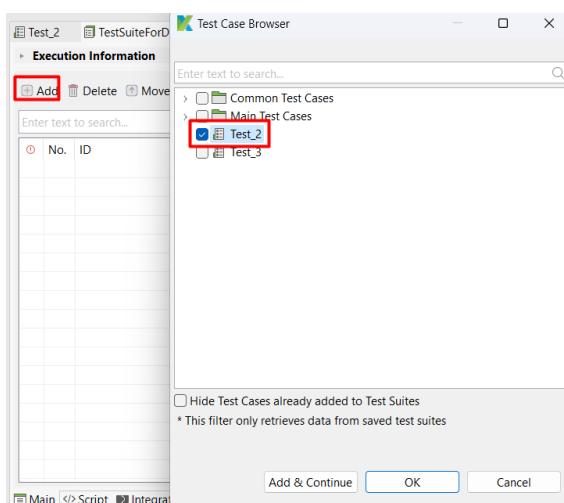
Gambar 4.6 Save

7. Setelah itu Create Test Suite.



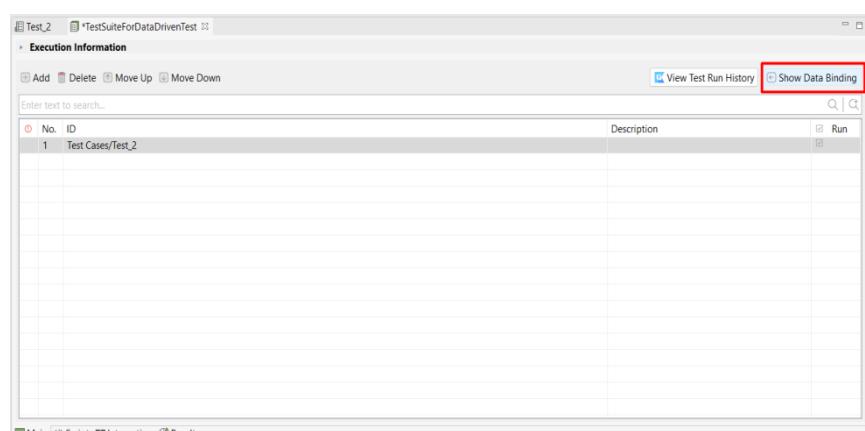
Gambar 4.7 Create New Suite

8. Tambahkan test case yang telah ada sebelumnya.



Gambar 4.8 Add Test 2

9. Kemudian klik *Show Data Binding*.



Gambar 4.9 Show Data Binding

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

Select Data Binding option:

Use Variables and Binding at [Test Case](#)

Use Variables and Binding at Suite Test Case

Test Data

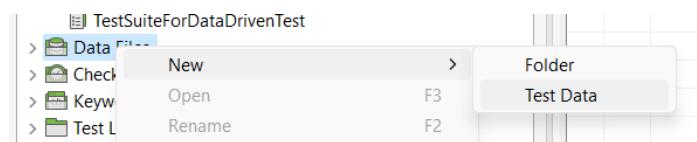
No.	ID	Data Iteration	Type

Variable Binding

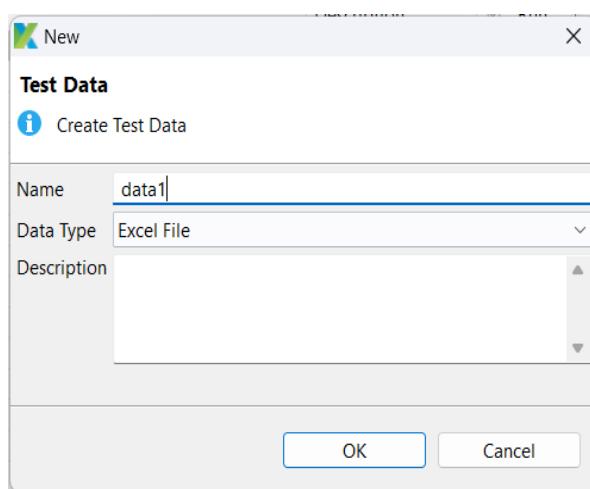
No.	Name	Default va...	Type	Test Data	Value
1	Username	"	Default		
2	Password	"	Default		

Gambar 4.10 Variabel Binding

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

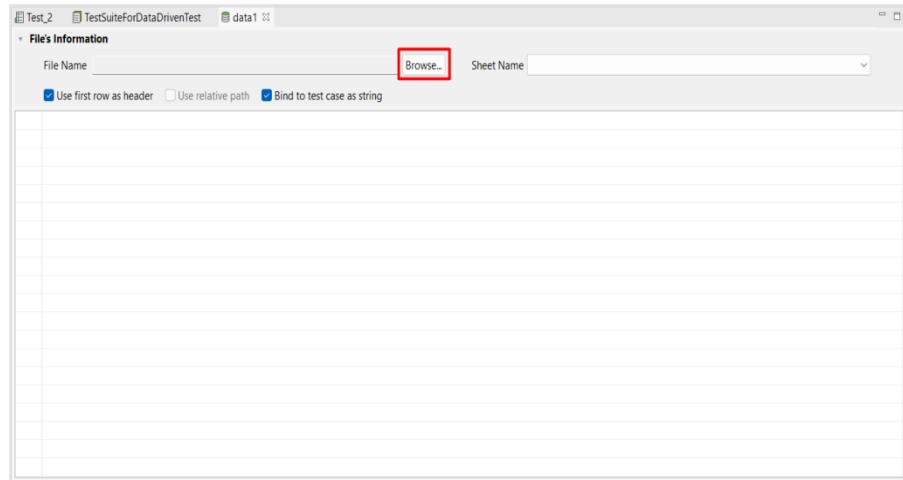


Gambar 4.11 New Test Data



Gambar 4.12 Test Explore

12. Pada datafiles klik *Browse*.



Gambar 4.13 Tahap Browser

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

A screenshot of the 'Test Data Browser' interface. The 'File's Information' tab is active. The 'File Name' field contains the path 'D:\File Kuliah\Semester 7\Kualitas Perangkat Lunak\Praktikum\katalon\data1.xlsx'. The 'Sheet Name' is set to 'Sheet1'. Below the tabs, there is a table with two rows of data:

No.	Username	Password
1	Admin	admin123
2	Admin2	admin1234

Gambar 4.14 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 'Execution Information' tab is active. In the 'Test Data' section, there is a table with one row:

No.	ID
1	Test Cases/Test_2

Below the table, a sub-dialog titled 'Test Data Browser' shows the contents of the 'data1' sheet from the previous screenshot.

Gambar 4.15 Test Data

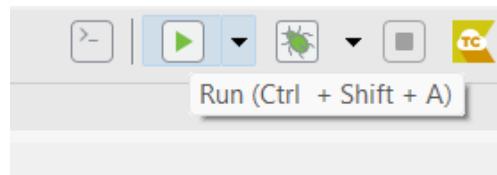
15. Pada Variable Binding ubah kedua tipe variabel menjadi *Data Column* kemudian pilih test datanya-nya, dan ubah valuenya sesuai dengan variable masing-masing

### 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 4.16 Proses Variabel Binding

16. Kemudian coba jalankan.



Gambar 4.17 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 4.18 Hasil Pengujian

## BAB 5

### RESULT AND REPORTING

#### 5.1. 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.

#### 5.2. Dasar Teori

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

#### 5.3. Percobaan

##### 5.3.1. 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 5.1 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.

The screenshot shows the Katalon Studio interface with the 'Log Viewer' tab selected. The table displays the following data:

Level	Time	Message
PASSED	2022-10-13 07:58:34.082	Browser is opened with url: "
PASSED	2022-10-13 07:58:43.383	Navigate to 'https://opensource-demo.orangehrmlive.com/web/index.php/auth/login'
PASSED	2022-10-13 07:58:48.807	Text 'Admin' is set on object 'Object Repository/test2/Page_OrangeHRM/input_Username_username'
PASSED	2022-10-13 07:58:49.629	Text ***** has been set on object 'Object Repository/test2/Page_OrangeHRM/input_Password_password'
PASSED	2022-10-13 07:58:50.208	Object: 'Object Repository/test2/Page_OrangeHRM/button_Login'
PASSED	2022-10-13 07:58:51.004	Page was closed

Gambar 5.2 Tes yang dijalankan

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

The screenshot shows the Katalon Studio interface with the 'Log Viewer' tab selected. The table displays the following data, with the 'Passed' filter applied:

Level	Time	Message
START	2022-10-13 07:58:54.106	
PASSED	2022-10-13 07:59:00.118	
END	2022-10-13 07:59:00.120	
START	2022-10-13 07:59:00.120	
PASSED	2022-10-13 07:59:03.628	
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 5.3 Menu filter log

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

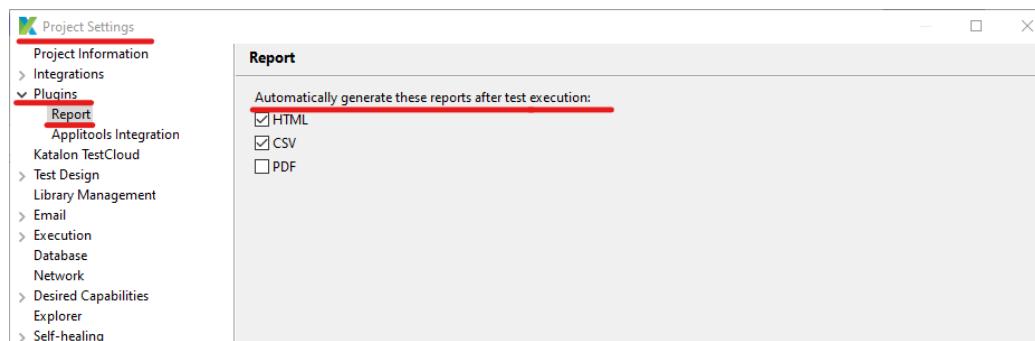
The screenshot shows the Katalon Studio interface with the 'Log Viewer' tab selected. The left sidebar shows a tree view of test cases and steps. The right panel shows the detailed log for the 'Test Suites/TestSuiteOne (45.301s)' step, which includes the following details:

- 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"), "nUKwTbfqfG") (0.581s)
  - 5 - click(findTestObject("test2/Page\_OrangeHRM/button\_Login")) (0.870s)
  - 6 - closeBrowser() (0.870s)
- Test Cases/Third Test Case (14.455s)
  - exportKatalonReports (1.666s)

Gambar 5.4 Tampilan hirarki log

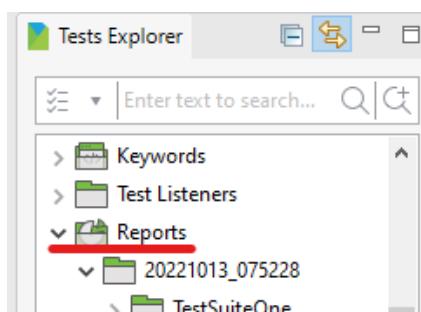
## B. Report

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.



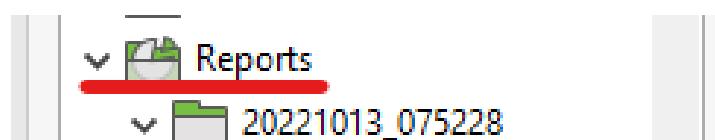
Gambar 5.5 Pengaturan format report

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



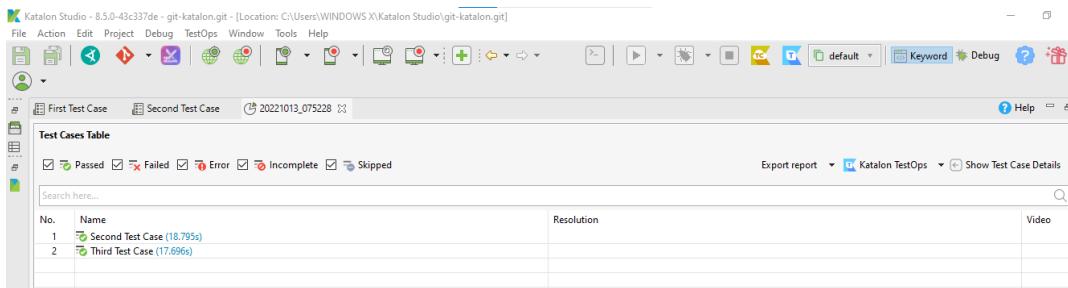
Gambar 5.6 Pilih menu report

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 5.7 Report yang sudah dipilih

4. Berikut ini adalah tampilan jendela report secara keseluruhan.



Gambar 5.8 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 type="checkbox"/> Failed <input type="checkbox"/> Error <input type="checkbox"/> Incomplete <input type="checkbox"/> Skipped	
Search here...	
No.	Name
1	Second Test Case (18.795s)
2	Third Test Case (17.696s)

Gambar 5.9 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	<a href="#">Test Suites/TestSuiteOne</a>		
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 5.10 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": ""}, prox...
defaultFailureHandling	CONTINUE_ON_FAILURE
terminateDriverAfterTestCase	false
defaultPageLoadTimeout	30.0
report	{videoRecorderOption=[enable=false, useBrowserRecorder=true, videoFormat=AVI, videoQuality=LOW, recordAllTes...
enablePageLoadTimeout	false
terminateDriverAfterTestSuite	true
useActionDelayInSecond	SECONDS
testDataInfo	{}
selfHealingEnabled	true
WebUI	{}

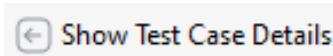
Gambar 5.11 Tampilan tab execution setting

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	75ec0030e4ef4c6017f904d89cbdb8
seleniumVersion	3.141.59
proxyInformation	ProxyInformation { proxyOption=NO_PROXY, proxyServerType=HTTP, username=, password=*****}, proxyServerAd...
platform	Windows 10

Gambar 5.12 Tampilan tab execution environtment

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



Gambar 5.13 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 5.14 Test case dialog

### 5.3.2. 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.

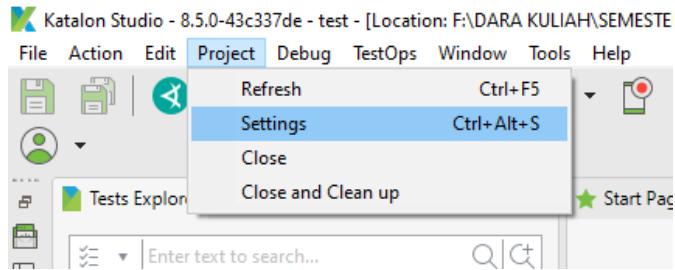
TestCase_1			
<span>Add</span> <span>Recent keywords</span> <span>Delete</span> <span>Move up</span> <span>Move down</span> <span>Edit tags</span> <span>Set default view</span>			
Item	Object	Input	Output
→ 1 - Open Browser		""	
→ 2 - Navigate To Url		"https://opensource-demo.orangehr...	
→ 3 - Set Text	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 5.15 Tampilan test case suite

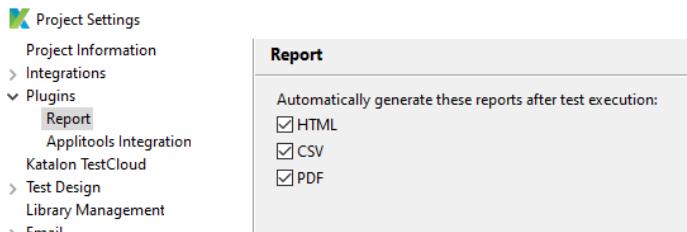
<input checked="" type="checkbox"/> Passed <input checked="" type="checkbox"/> Failed <input checked="" type="checkbox"/> Error <input checked="" type="checkbox"/> Incomplete <input checked="" type="checkbox"/> Skipped		Export
Search here...		
No.	Name	Resolution
1	Test1 (34.378s)	
2	TestCase_2 (24.477s)	

Gambar 5.16 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.

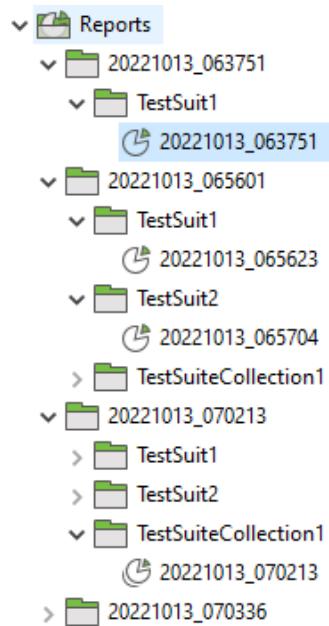


Gambar 5.17 Tampilan menu project



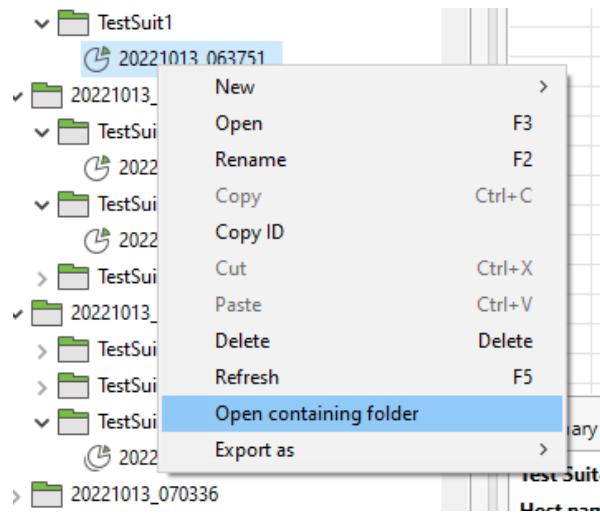
Gambar 5.18 Menu pada project setting

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 5.19 List pengujian yang telah dibuat

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



Gambar 5.20 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 5.21 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 5.22 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**

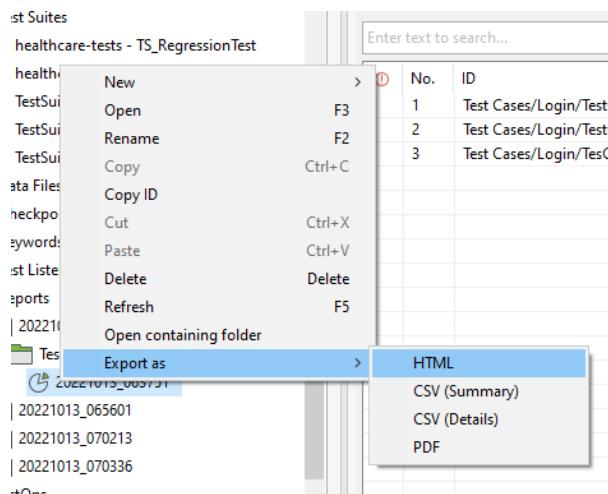
- **TEST SUITE:** TestSuit1
  - Full Name: TestSuit1
  - Start / End / Elapsed: 2022-10-13 06:38:26.583 / 2022-10-13 06:39:26.409 / 00:00:59.826
  - Status: 2 test total, 2 passed, 0 failed, 0 error, 0 incomplete, 0 skipped
- **TEST CASE:** Test Cases/Test1
  - Full Name: TestSuit1/Test Cases/Test1
  - 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"), "hUKwJTb0fgPU9eVlw/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 5.23 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(find	Chrome 106.0.0.0			#####	#####	0.730s	PASSED			
10 click(find	Chrome 106.0.0.0			#####	#####	5.336s	PASSED			
11 click(find	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(find	Chrome 106.0.0.0			#####	#####	0.517s	PASSED			
21 click(find	Chrome 106.0.0.0			#####	#####	5.388s	PASSED			
22 click(find	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 5.24 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 5.25 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

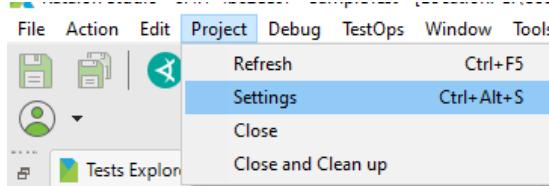
Gambar 5.26 Pilih lokasi penyimpanan file

### 5.3.3. How To Email Results

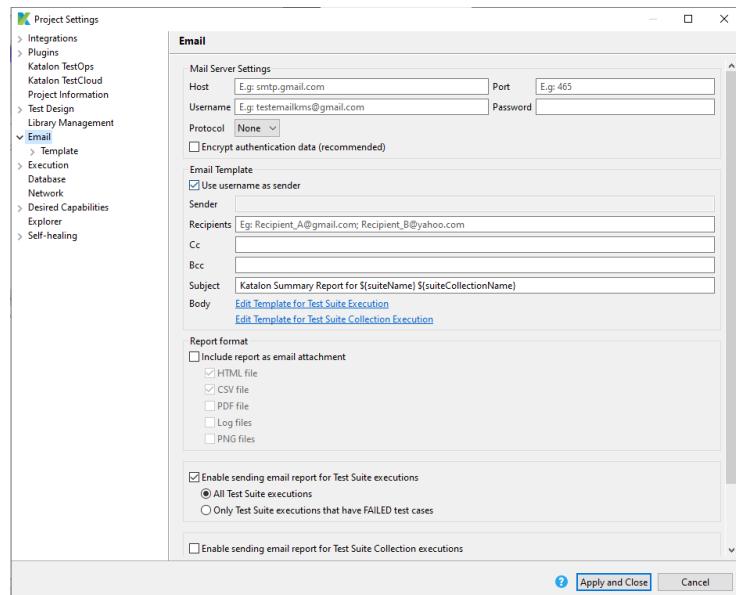
Setelah melakukan pengujian perangkat lunak, terkadang seorang dari tim ingin hasil pengujian tersebut dikirim ke email secara otomatis agar menghemat

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 5.27 Pilih menu project setting pada toolbar



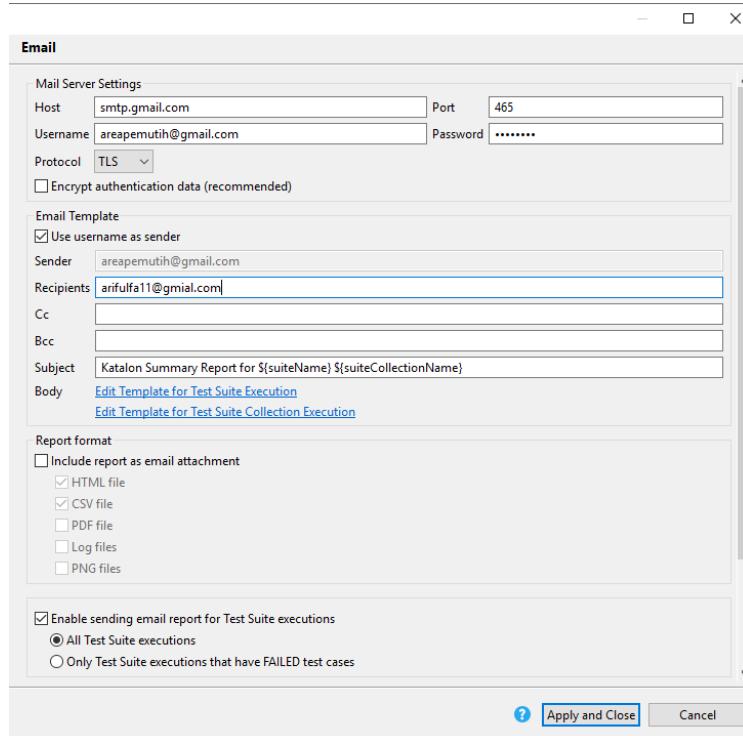
Gambar 5.28 Pilih template email

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

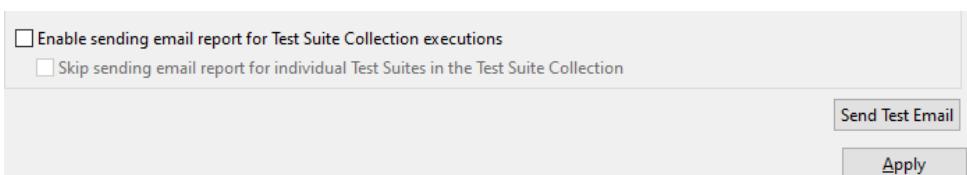
Email sever	Host	Port	Reference
Gmail	smtp.gmail.com	465 or 587	<a href="#">Check Gmail through other email platforms</a>
Outlook	smtp.office365.com	587 or 25	<a href="#">How to set up a multifunction device or application to send email using Microsoft 365 or Office 365</a>
Yahoo! Mail	smtp.mail.yahoo.com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

Gambar 5.29 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*.

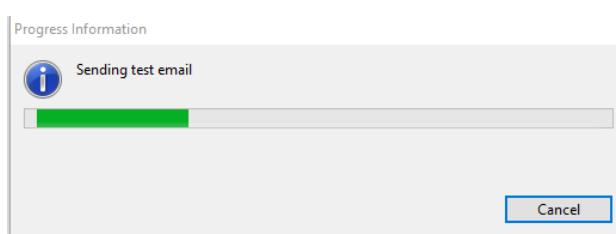


Gambar 5.30 Pengisian Email pada mail server setting

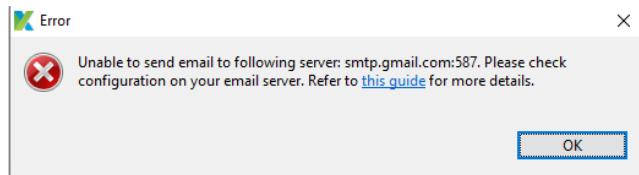


Gambar 5.31 Pilih apply

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

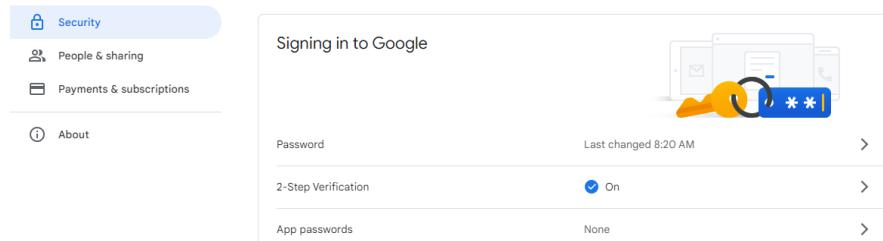


Gambar 5.32 Tampilan saat proses berjalan



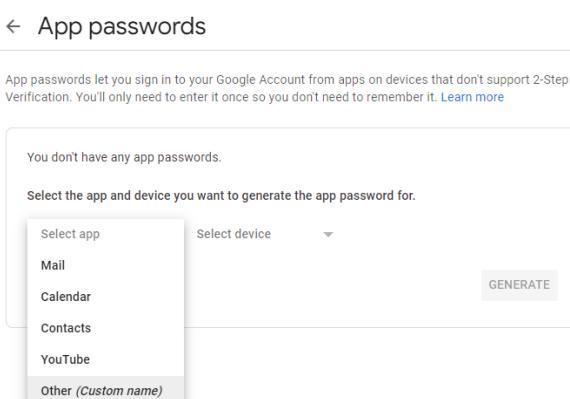
Gambar 5.33 Tampilan proses gagal (error)

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



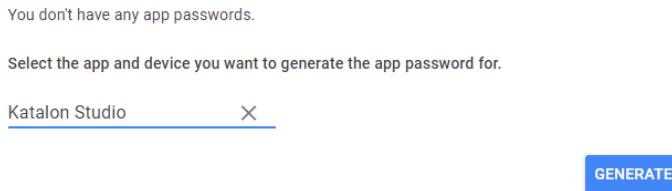
Gambar 5.34 Tampilan menu security

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



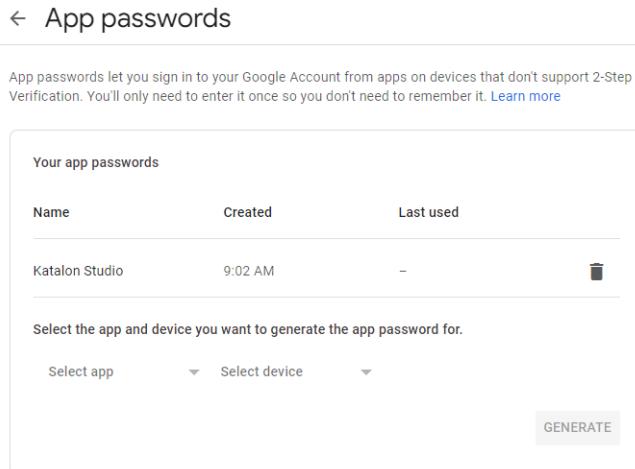
Gambar 5.35 Tampilan jendela app password

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



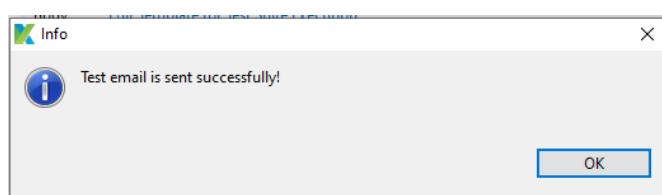
Gambar 5.36 Pilih generate

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



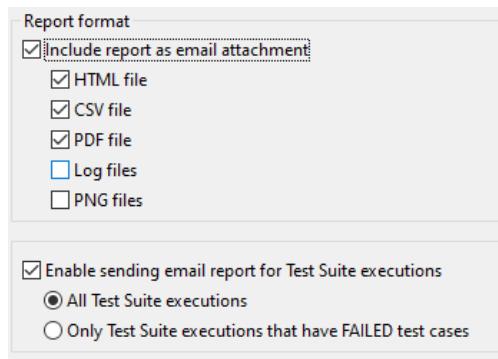
Gambar 5.37 Tampilan halaman App password

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



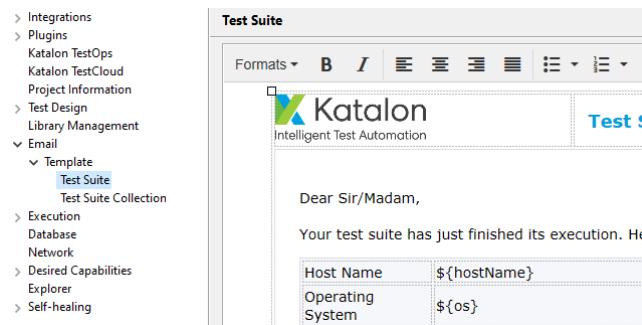
Gambar 5.38 Proses berhasil

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



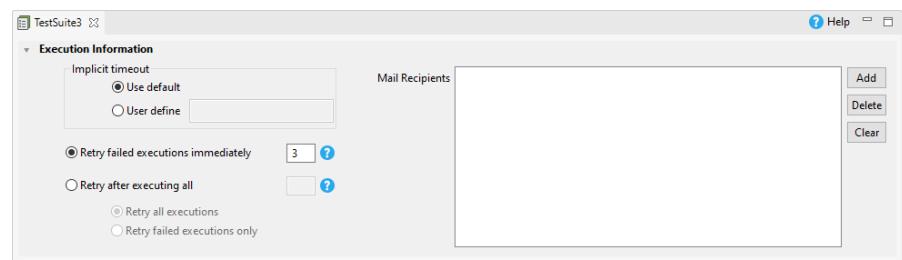
Gambar 5.39 Tampilan report format

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



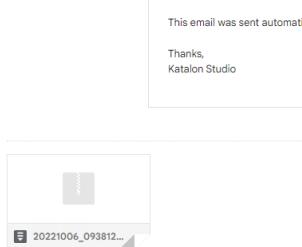
Gambar 5.40 Mengubah template test Suite

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 5.41 Halaman Execution Infomration

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.



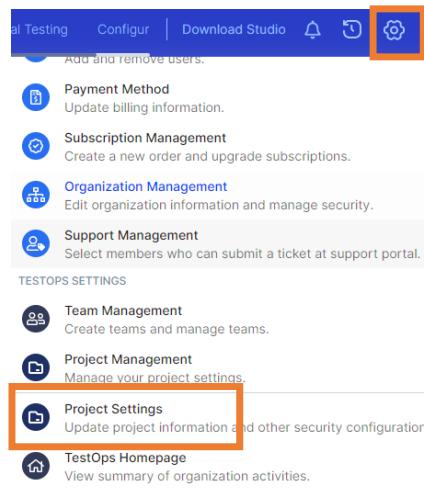
Gambar 5.42 Notifikasi file PDF masuk pada email

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

#### 5.3.4. How To Use Katalon Analytics Step by Step

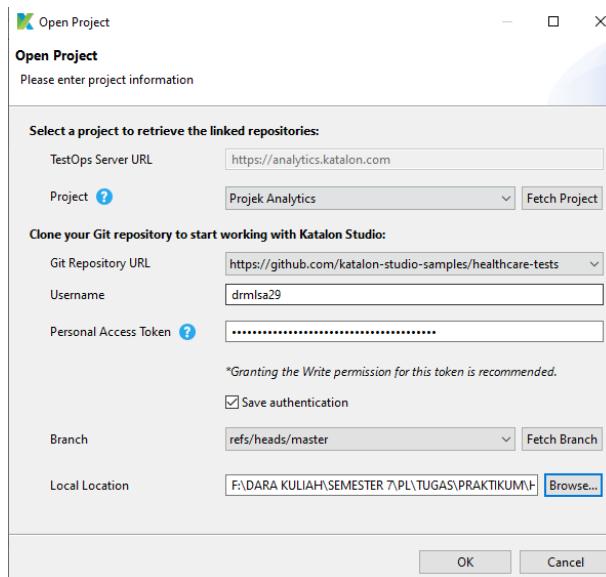
Pada modul ini kita akan mempelajari bagaimana cara menggunakan katalon analytics. Katalon analytic merupakan suatu proses yang berfungsi untuk melihat, menyimpan, dan menganalisis hasil atau laporan dari pengujian yang dilakukan pada katalon. Berikut ini merupakan tahapan menggunakan katalon analytics.

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 5.43 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.

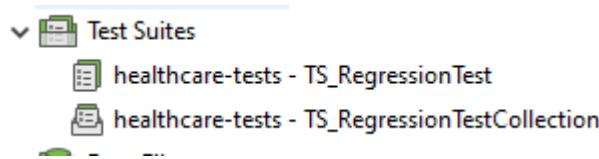


Gambar 5.44 Open project

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 5.45 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 5.46 Pilihan test Suite

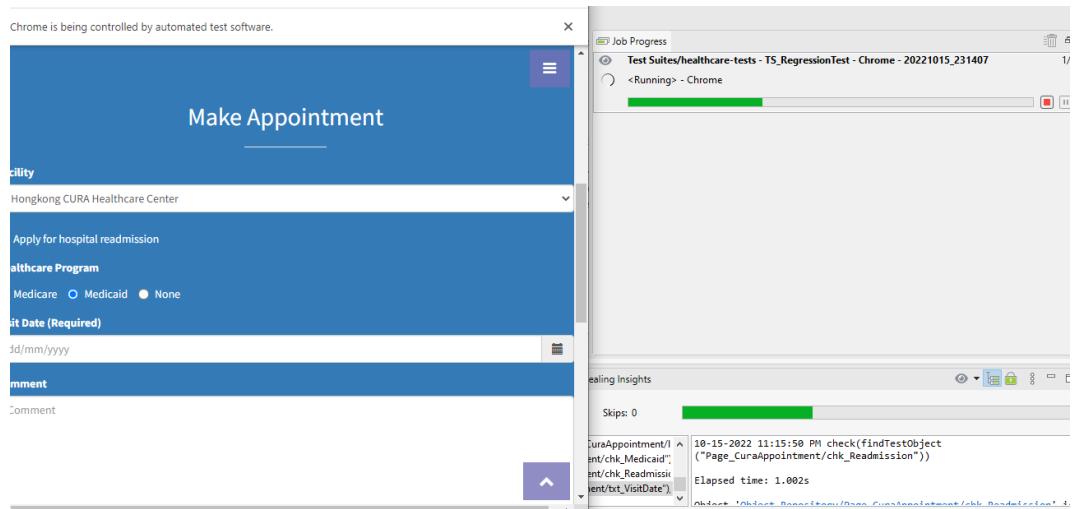
Seperti yang dapat dilihat pada gambar di bawah ini, pada test suit tersebut sudah ditambahkan beberapa test case.

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

Gambar 5.47 Test Suite yang sudah ditambahkan

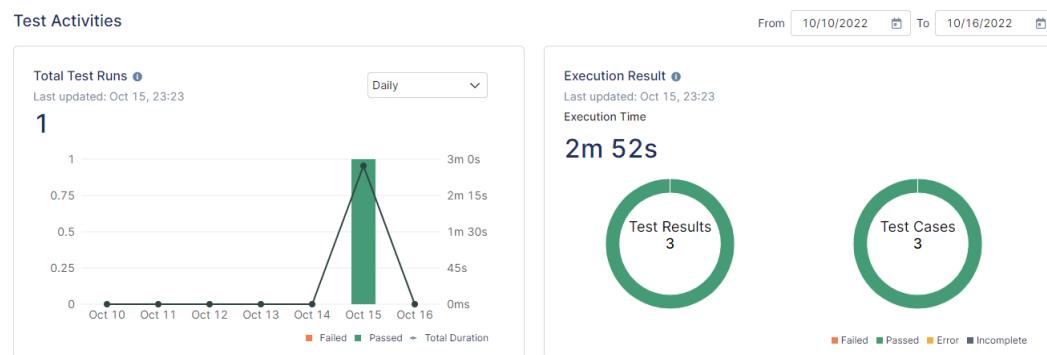
5. Jalankan test suit dengan browser pilihan anda, dan tunggu sampai proses eksekusi selesai. Berikut ini gambar tampilan test suit yang sedang dieksekusi.

Gambar 5.48 Test Suite dijalankan menggunakan Browser



Gambar 5.49 Test Suite sedang di eksekusi

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



Gambar 5.50 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 6**

### **MENJALANKAN TEST MENGGUNAKAN COMMANDLINE DAN PLUGIN JENKINS PADA KATALON STUDIO**

#### **6.1. 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.

#### **6.2. 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:

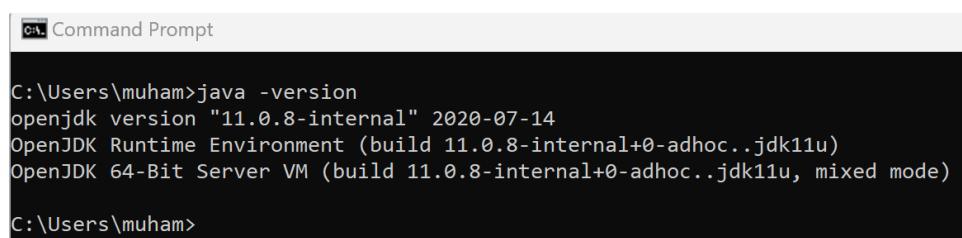
1. Build software menggunakan sistem build seperti Gradle, Maven, dan lainnya.
2. Pengujian otomasi menggunakan kerangka kerja pengujian seperti Nose2, PyTest, Robot, Selenium, dan banyak lagi.
3. Jalankan skrip pengujian (menggunakan terminal Windows, shell Linux, dll).

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.

### 6.3. 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>
```

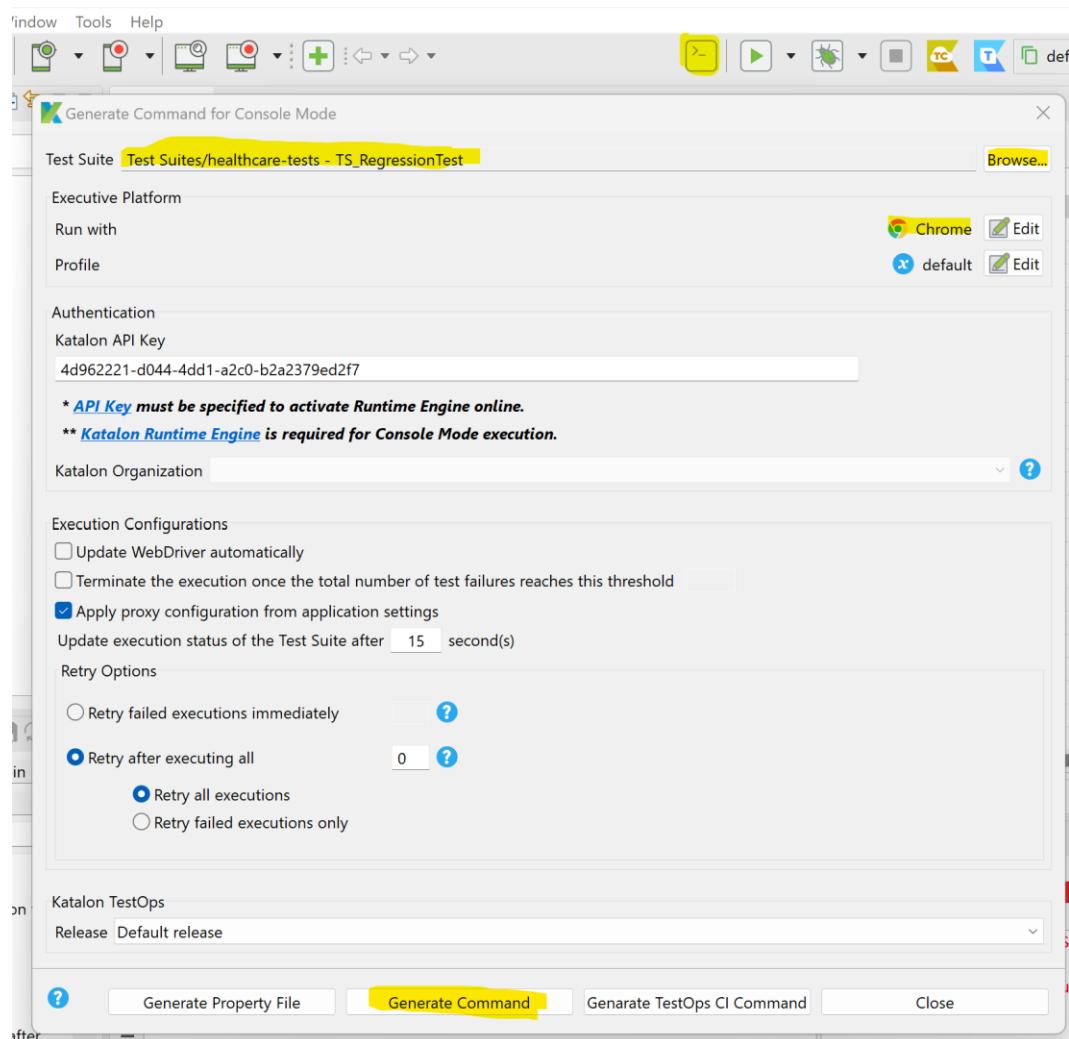
Gambar 6.1 Melihat Versi Java Pada CMD

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>

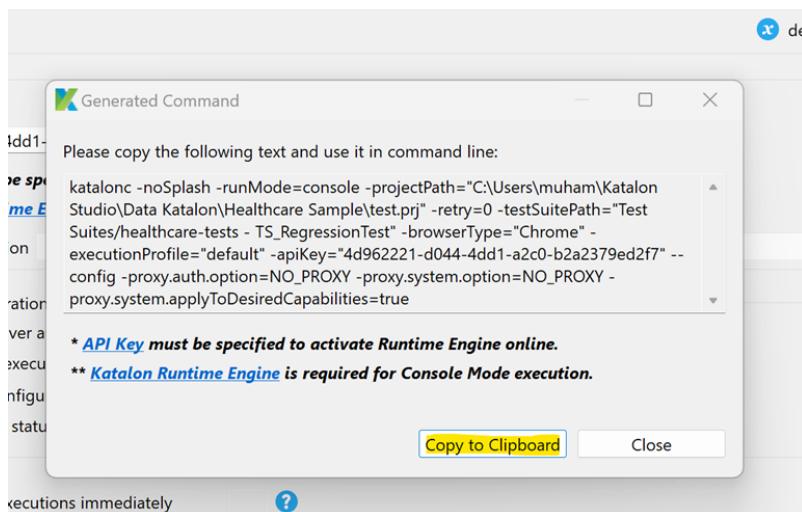
Langkah selanjutnya adalah membuka aplikasi testing catalon studio untuk mempersiapkan command dengan cara mengenerate command seperti gambar di bawah ini:



Gambar 6.2 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 6.3 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 6.4 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 6.5 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](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:

```

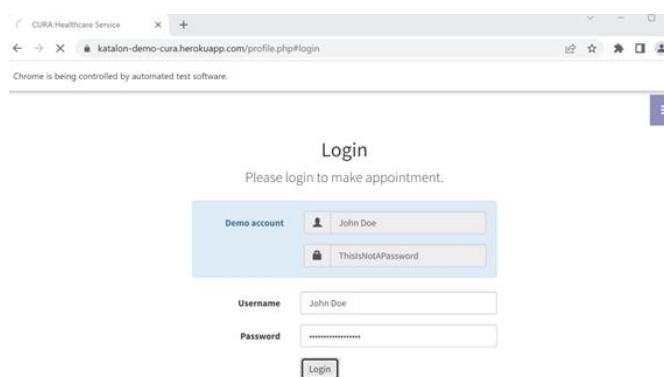
INFO: Katalon Version: 8.5.1
INFO: Katalon Edition: Platform
INFO: Command-line arguments: -runMode=console -projectPath=C:\Users\muham\Katalon Studio\Data Katalon\Healthcare Sample\test.prj -retry=0 -testSuitePath=Test Suites/healthcare-tests - TS_RegressionTest -browserType=Chrome -executionProfile=default -apikeys=***** --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 6.6 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 6.7 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 6.8 Output Testing Pada CMD

## **BAB 7** **INTEGRATION WITH GIT AND JENKINS**

### **7.1. Tujuan**

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

### **7.2. 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**

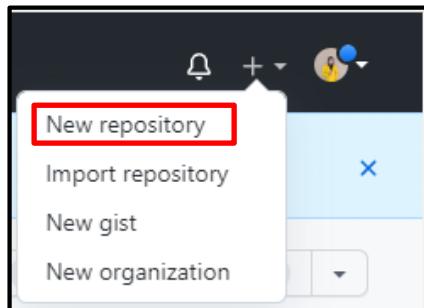
Integration with Git berarti dapat mengintegrasikan TestComplete dengan Git dan bekerja dengan repositori langsung dari antarmuka pengguna TestComplete. Tanpa meninggalkan IDE TestComplete, Anda dapat menambahkan file proyek ke repositori, melakukan perubahan, membatalkan perubahan dan memasukkan ke 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.

### **7.3. 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 7.1 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

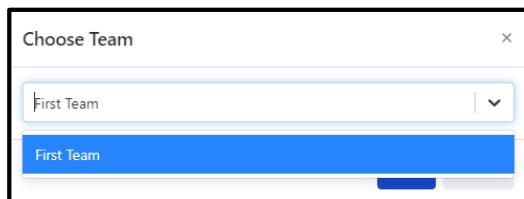
The screenshot shows the GitHub 'Create New Repository' form. It includes fields for 'Owner' (set to 'khairaul'), 'Repository name' (set to 'KatalonGitTes'), and 'Description (optional)'. Under 'Visibility', the 'Public' radio button is selected, with a note: 'Anyone on the internet can see this repository. You choose who can commit.' Below that, the 'Private' radio button is available with its own note: 'You choose who can see and commit to this repository.' There is also a section for initializing the repository with a README file, which has a checked checkbox labeled 'Add a README file'. At the bottom, there are sections for '.gitignore' and 'Choose a license'.

Gambar 7.2 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 7.3 Membuat projek di *Dashboard* Katalon



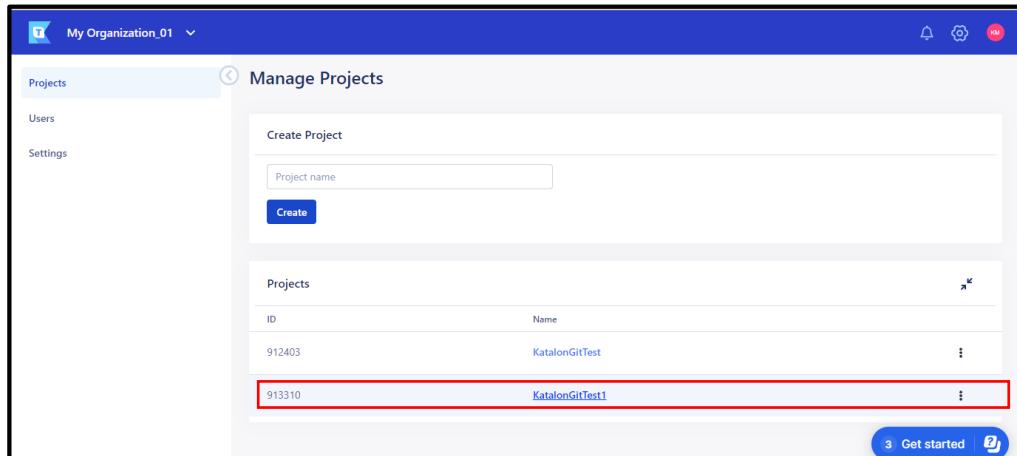
Gambar 7.4 Memilih projek tim

4. Masukkan nama project yang diinginkan



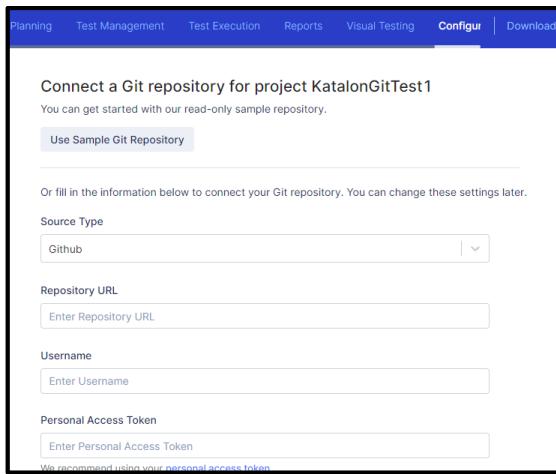
Gambar 7.5 Memberi nama projek

Maka projek yang telah dibuat akan tampil seperti pada gambar berikut



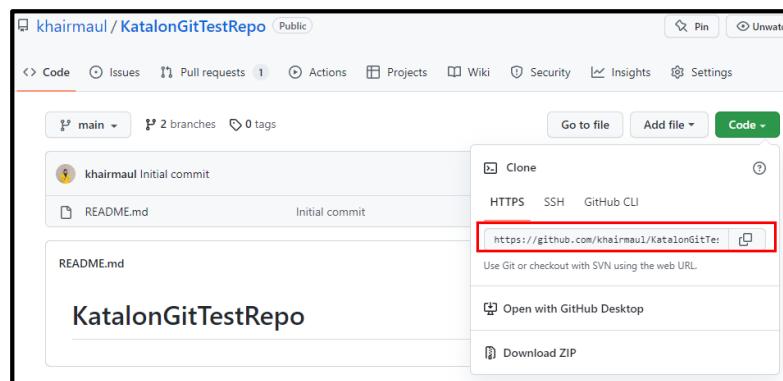
Gambar 7.6 Hasil pembuatan projek

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



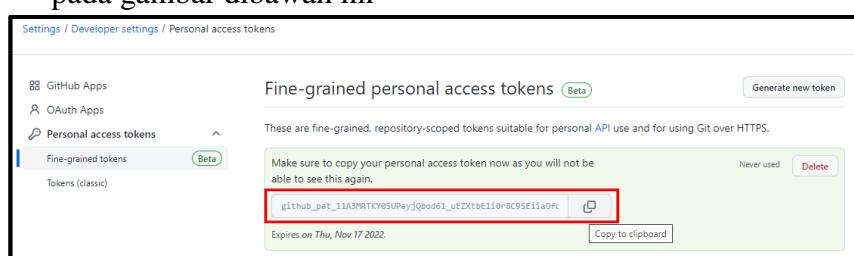
Gambar 7.7 Koneksi dengan *repository* GIT

- 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



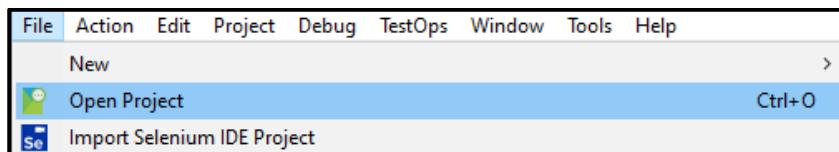
Gambar 7.8 Meng-copy link untuk *clone* repo

- 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



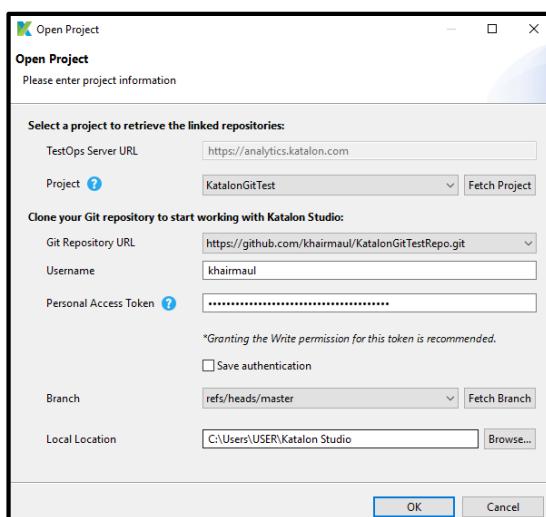
Gambar 7.9 *Copy API token*

4. Open dan clone project



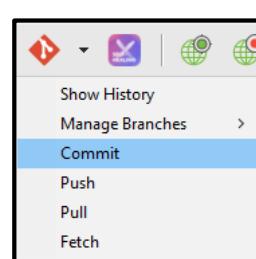
Gambar 7.10 Membuka projek di Katalon Studio

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



Gambar 7.11 Memilih projek dengan API token

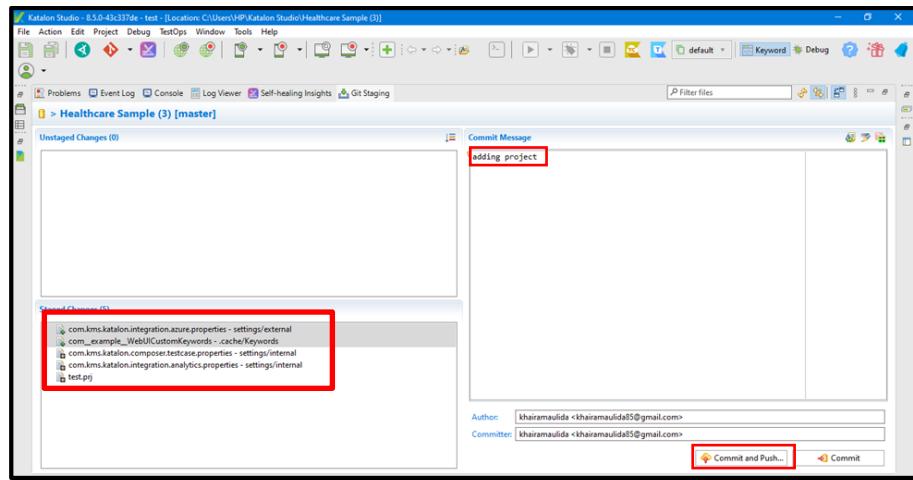
5. Commit project



Gambar 7.12 Melakukan *commit* projek

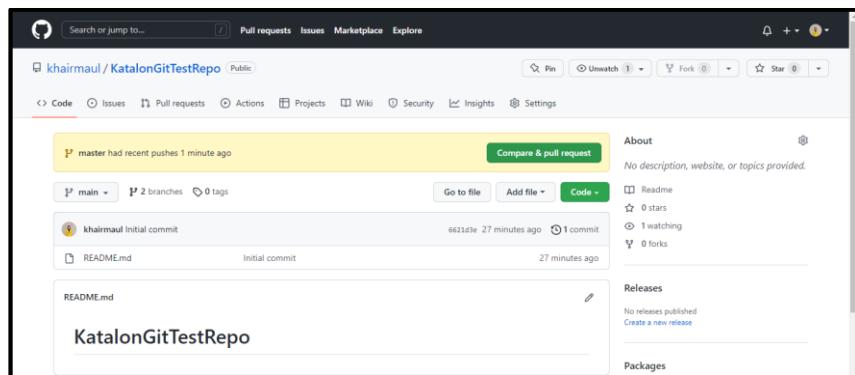
Kemudian lakukan langkah-langkah dibawah ini:

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 7.13 *Commit* projek Katalon Studio

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

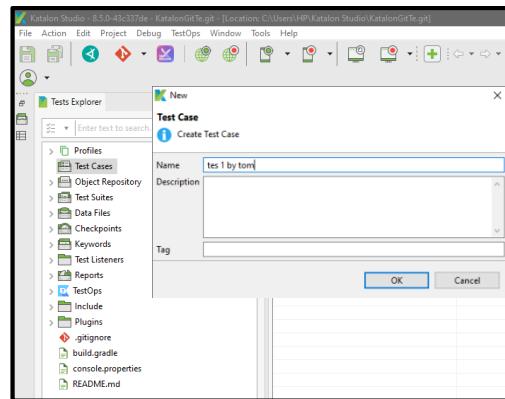


Gambar 7.14 Terdapat *Pull Request* di Repo GitHub

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.

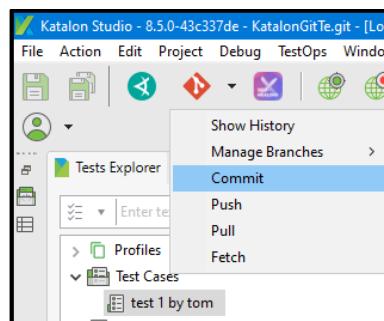
### Contoh 1:

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



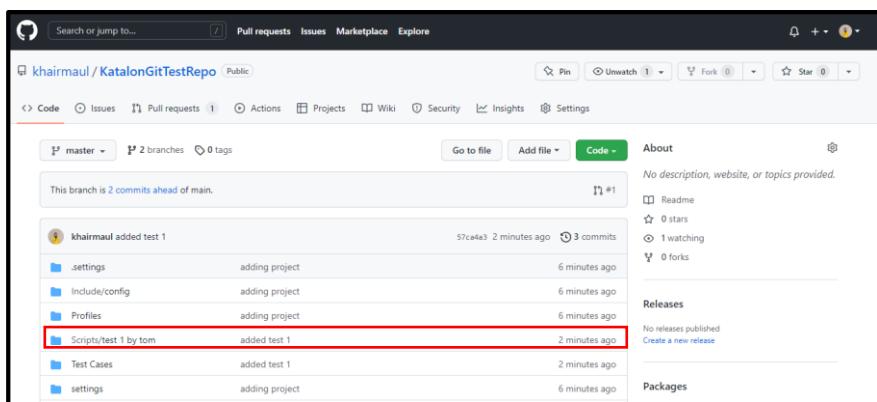
Gambar 7.15 Membuat *test case*

Kemudian klik icon github dan pilih commit



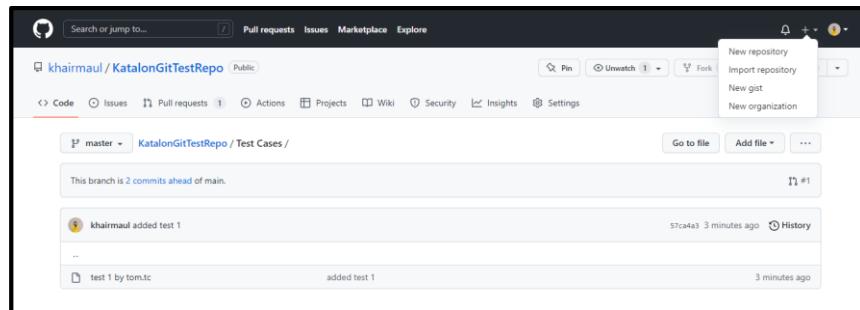
Gambar 7.16 Commit *test case*

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 7.17 Hasil pada repo GitHub

Ketika folder dibuka maka akan ditampilkan file testcase tersebut.



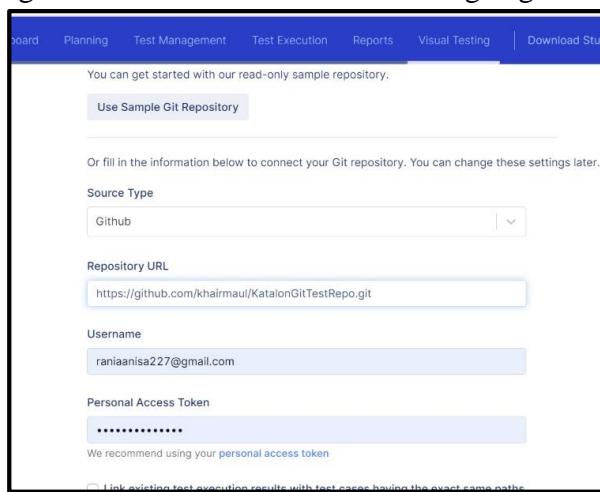
Gambar 7.18 Isi folder Test Case

### Contoh 2:

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

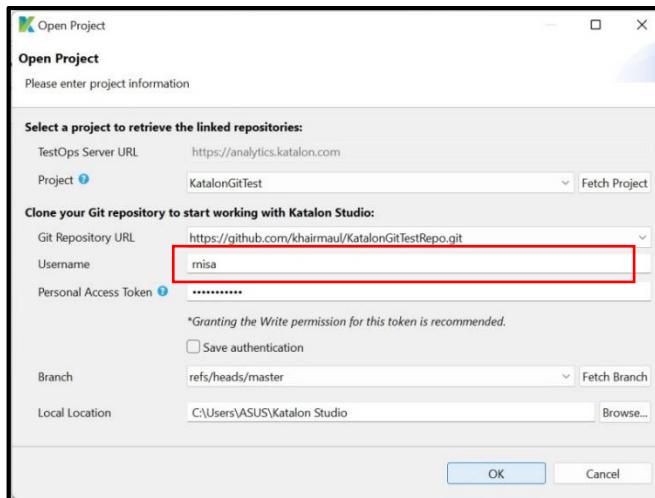
Berikut langkah-langkahnya:

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



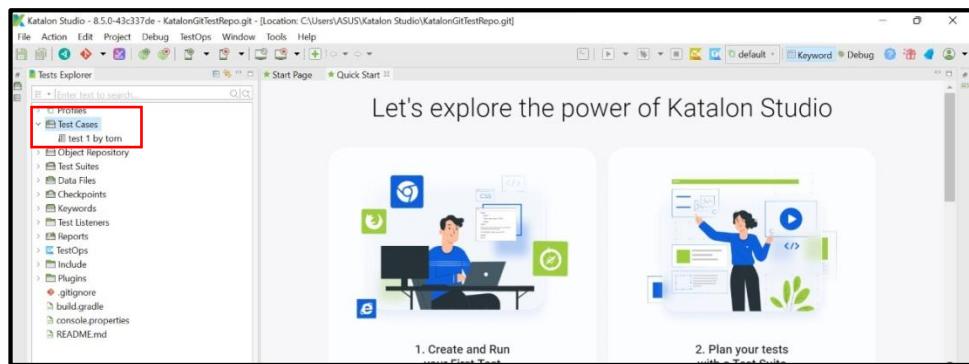
Gambar 7.19 Koneksi Katalon dengan GitHub

2. Open project dan clone repository tom



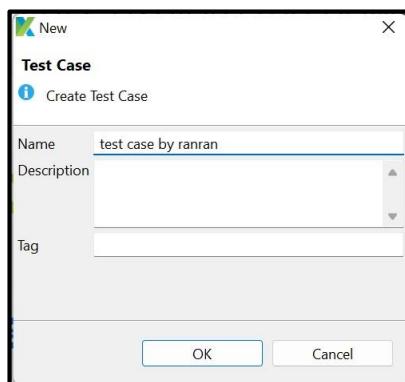
Gambar 7.20 Open Project

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



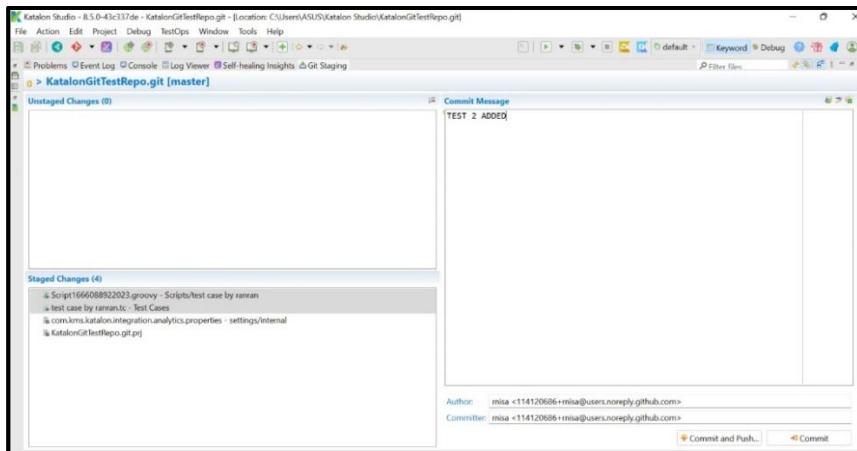
Gambar 7.21 Hasil Open Project

3. Kemudian ran akan membuat perubahan pada repository tom yaitu membuat testcase baru



Gambar 7.22 Membuat test case baru

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



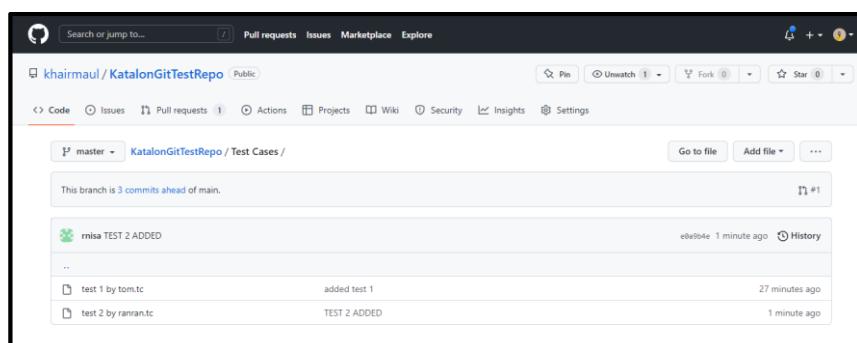
Gambar 7.23 Commit perubahan projek

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

The screenshot shows a GitHub commit history for a repository. The commit 'misa TEST 2 ADDED' (commit e0a9b4e) was made 24 seconds ago. It contains 4 commits. The commit details show changes to '.settings', 'Include/config', 'Profiles', 'Scripts' (with 'TEST 2 ADDED'), and 'Test Cases' (with 'TEST 2 ADDED'). The 'Test Cases' commit is highlighted with a red box. Other files listed include 'settings', '.gitignore', 'KatalonGitTestRepo.git.prj', 'README.md', 'build.gradle', and 'console.properties'.

Gambar 7.24 Hasil perubahan repo GitHub

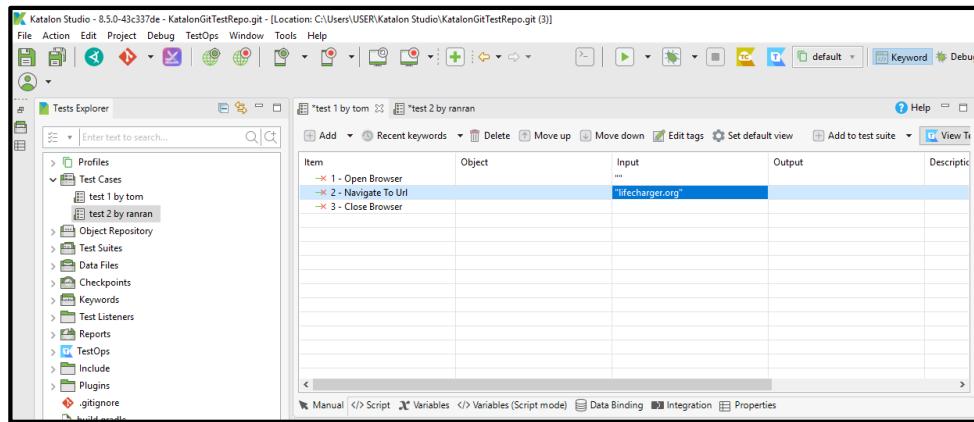
Ketika folder tersebut di klik maka akan menampilkan test case yang telah dibuat oleh ran



Gambar 7.25 Isi folder Test Case

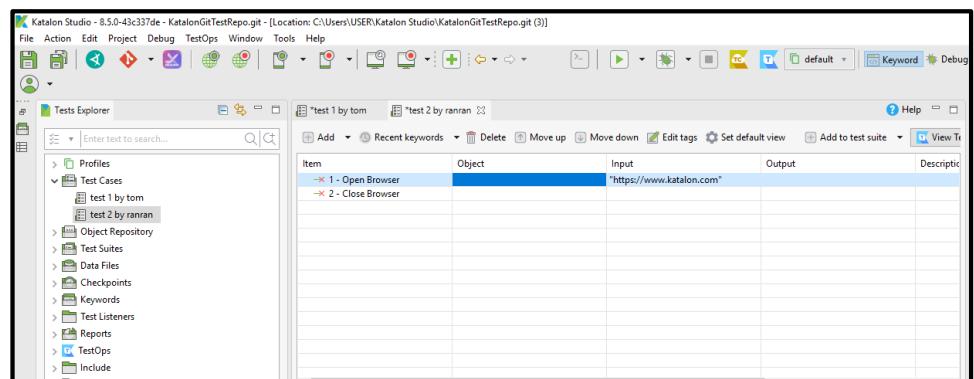
## 7.4. Percobaan Integrasi Dengan Jenkins

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



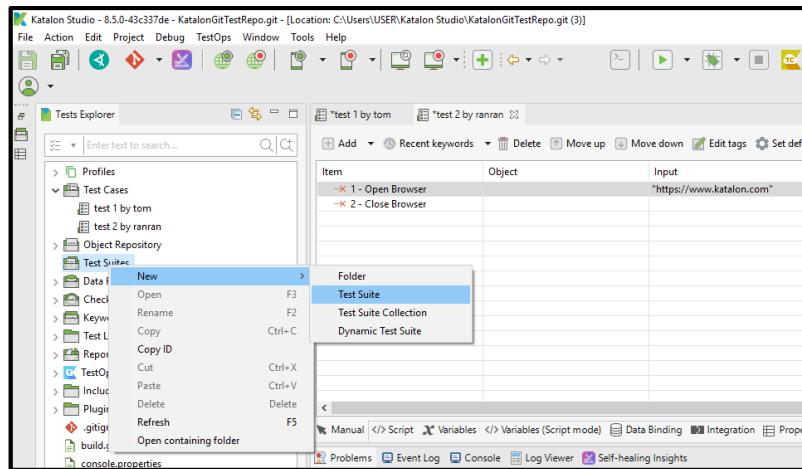
Gambar 7.26 Isi *test case 1 by tom*

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



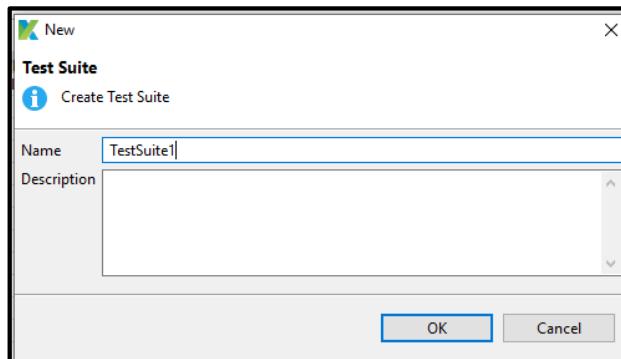
Gambar 7.27 isi *test case 2 by ranran*

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



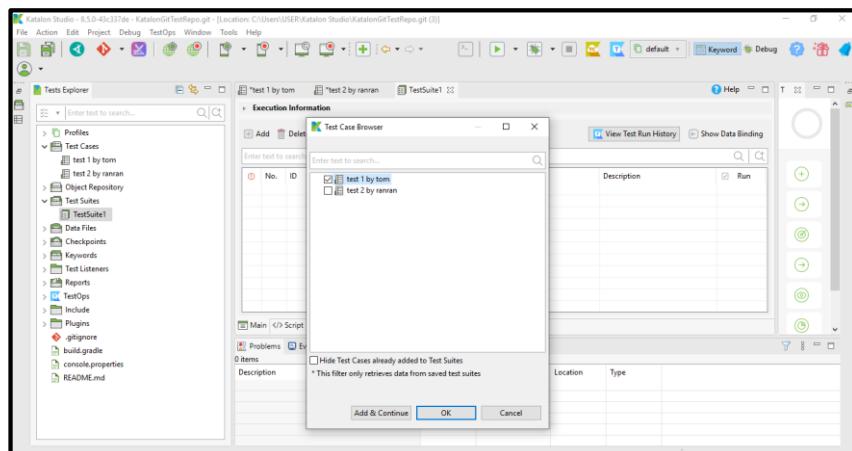
Gambar 7.28 Membuat *Test Suite*

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



Gambar 7.29 Memberi nama *Test Suite*

5. Kemudian klik add dan pilih test 1 by tom



Gambar 7.30 Memasukkan *test case* ke dalam *test suite*

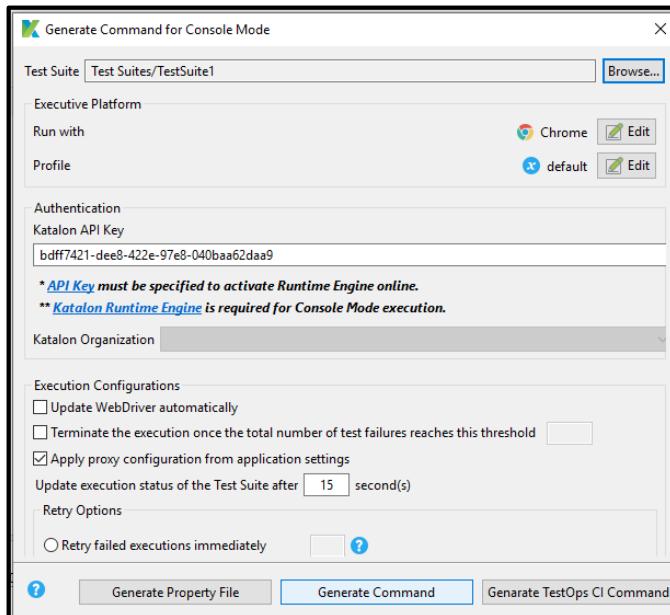
6. Maka hasilnya seperti berikut

The screenshot shows the 'Execution Information' window in Katalon Studio. It displays a table with one row, where the first column is 'No.', the second is 'ID', the third is 'Description', and the fourth is 'Run'. The row contains the value '1' in 'No.', 'Test Cases/test 1 by tom' in 'ID', and a checked checkbox in 'Run'.

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

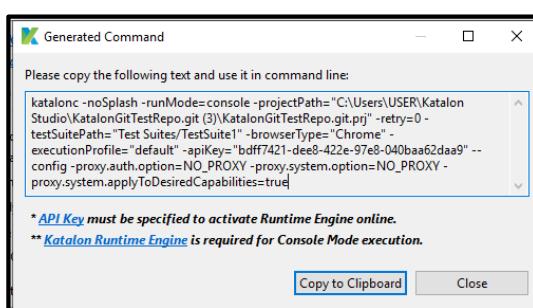
Gambar 7.31 Hasil setelah menambahkan *test case*

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



Gambar 7.32 Jendela *Generate Command for Console Mode*

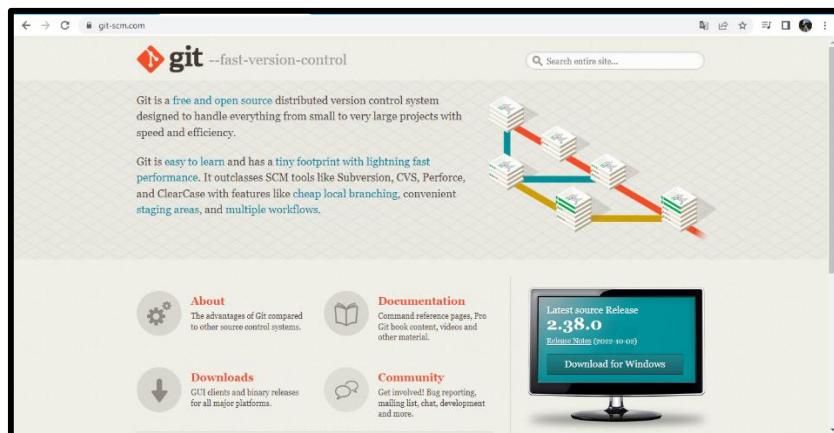
- Lalu klik pada tombol Copy to Clipboard



Gambar 7.33 *Copy* teks ke *Clipboard*

## Tahap 2 Install Git

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



Gambar 7.34 Download GIT

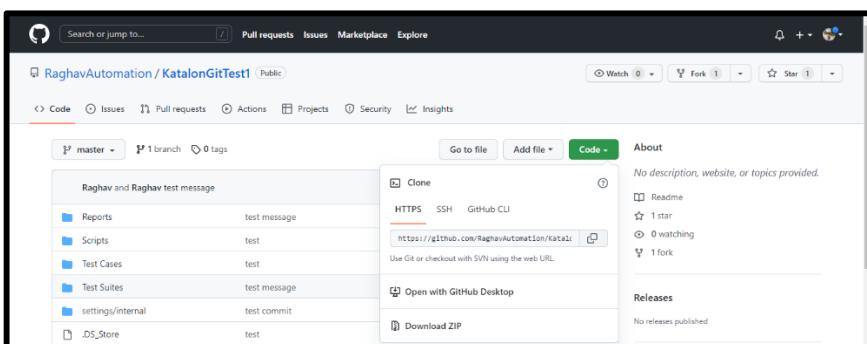
3. Kemudian install

📁	Katalon_Studio_PE_Windows_64-8.5.0	9/22/2022 3:15 PM	File folder
💻	[gigapurlalingga.net]_AIMpSftUnvMD10...	9/20/2022 10:08 AM	WinRAR archive
💿	bodhi-6.0.0-64.iso	9/21/2022 9:14 AM	Disc Image File
🌐	GitHubDesktopSetup-x64.exe	10/13/2022 3:37 PM	Application
gMaps	googleearthprowin-7.3.4.exe	9/5/2022 3:40 PM	Application
📁	Katalon_Studio_PE_Windows_64-8.5.0.zip	9/23/2022 4:44 AM	WinRAR ZIP archive
🐍	pycharm-community-2022.2.1.exe	9/7/2022 8:51 PM	Application
💻	SAS.Planet.Release.211230[geojamal.com]...	9/5/2022 3:25 PM	WinRAR ZIP archive
_VM_	VirtualBox-6.0.14-133895-Win.exe	10/24/2019 8:36 AM	Application
Git	Git-2.38.0-64-bit.exe	10/10/2022 10:39 AM	Application

Gambar 7.35 Installer GIT

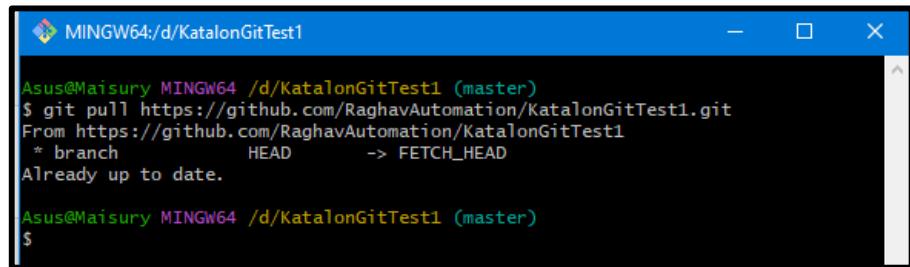
## Tahap 3 Uji perintah Git

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



Gambar 7.36 Menyalin link clone repository

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

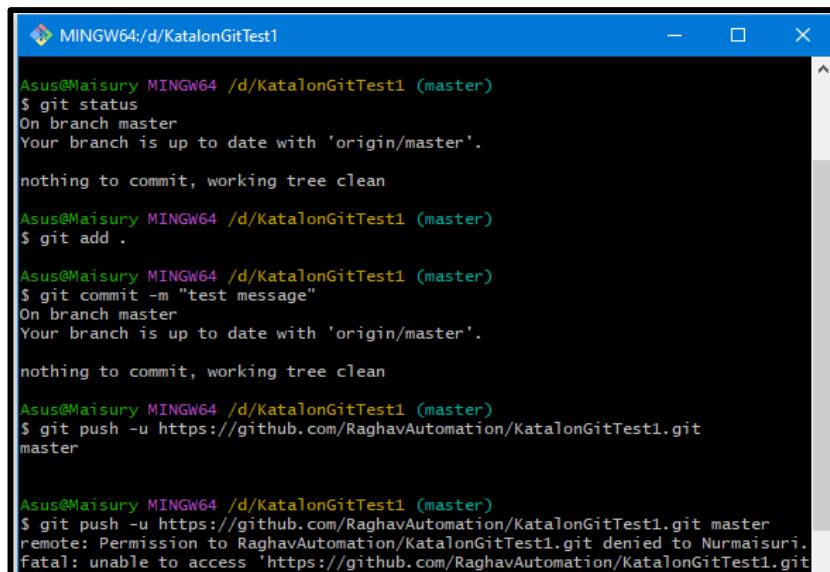


```
Asus@Maisury MINGW64 /d/KatalonGitTest1 (master)
$ git pull https://github.com/RaghavAutomation/KatalonGitTest1.git
From https://github.com/RaghavAutomation/KatalonGitTest1
 * branch            HEAD      -> FETCH_HEAD
Already up to date.

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

Gambar 7.37 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 7.38 Melakukan *push repository*

## BAB 8

### API TESTING

#### 8.1. 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.

#### 8.2. Dasar Teori

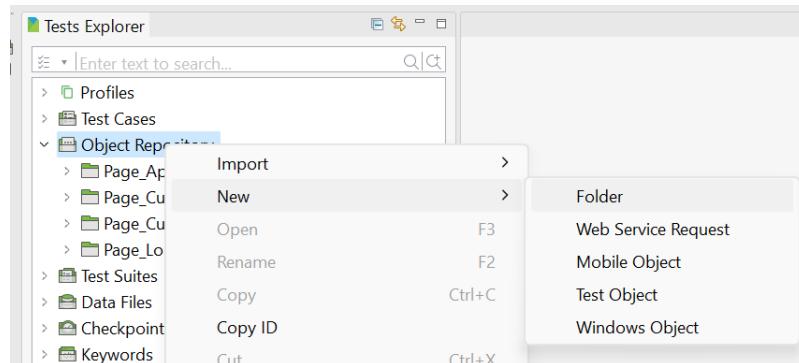
Api merupakan aplikasi yang memungkinkan para developer untuk mengintegrasikan bagian aplikasi dengan bagian aplikasi lainnya.

#### 8.3. Percobaan API Testing

##### a. How To Do Api Testing With Katalon

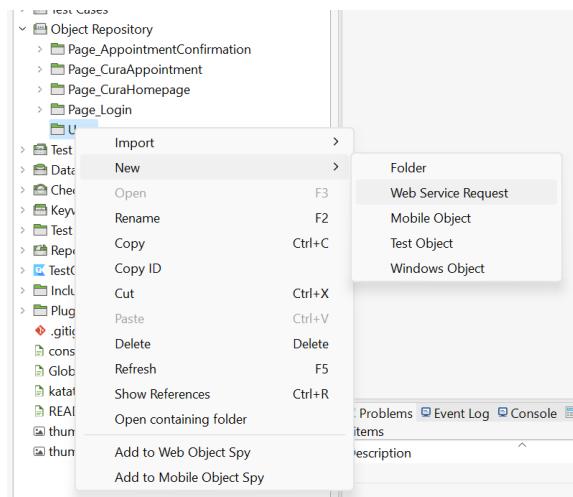
Katalon juga dapat digunakan sebagai aplikasi untuk melakukan pengujian terhadap API. Berikut langkah-langkah yang dapat dilakukan dalam menguji API pada katalon.

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



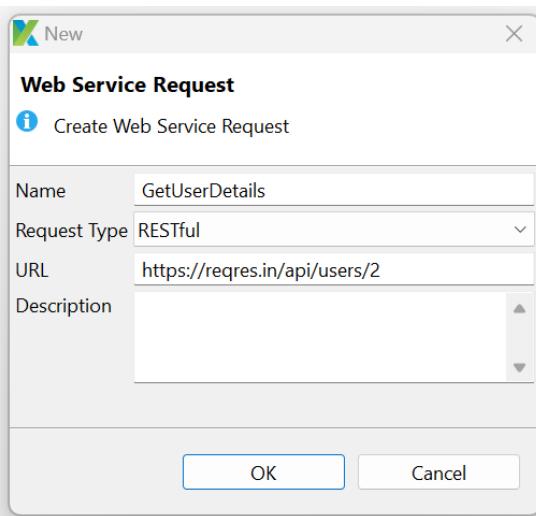
Gambar 8.1 Membuat Folder Baru

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



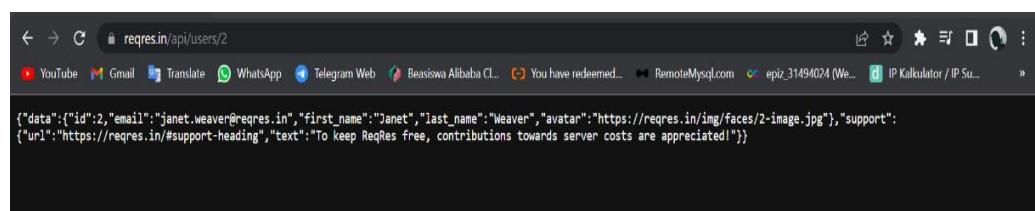
Gambar 8.2 Pilih Web Service Request

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



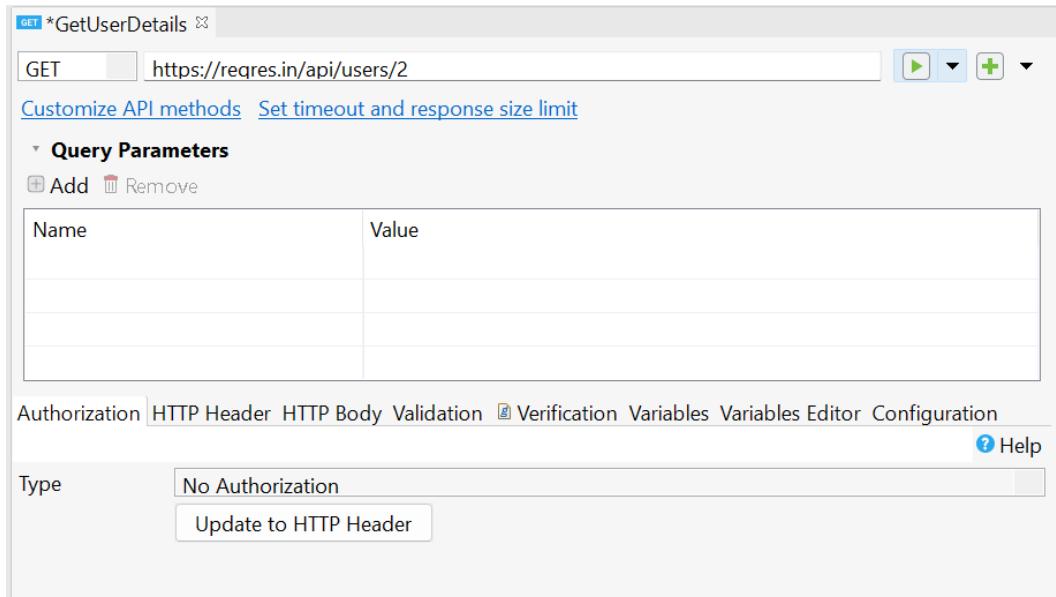
Gambar 8.3 Membuat Web Service Request

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



Gambar 8.4 URL API Pada Web

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



Gambar 8.5 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 'Response' tab in Postman. The status is 200 OK, elapsed time is 1131 ms, and size is 644 bytes. The response body is a JSON object representing a user:

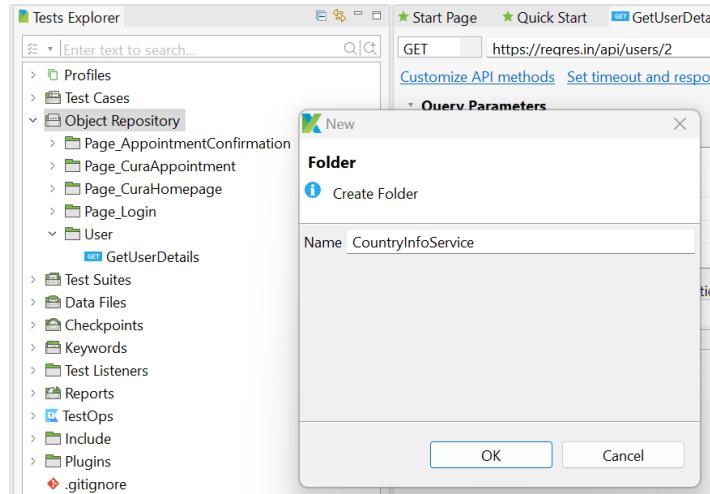
```
1 {  
2   "data": {  
3     "id": 2,  
4     "email": "janet.weaver@reqres.in",  
5     "first_name": "Janet",  
6     "last_name": "Weaver",  
7     "avatar": "https://reqres.in/img/faces/2-image.jpg"  
8   },  
9   "support": {  
10     "url": "https://reqres.in/#support-link"  
11   }  
12 }
```

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

JSON XML HTML JavaScript Wrap Line

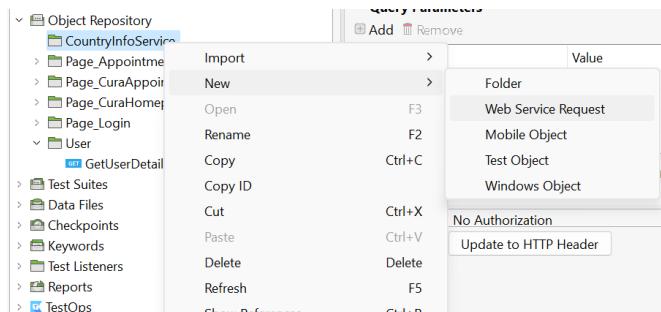
Gambar 8.6 Tampilan Response

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



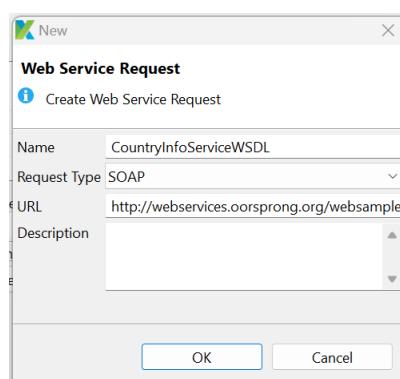
Gambar 8.7 Membuat Folder Baru

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



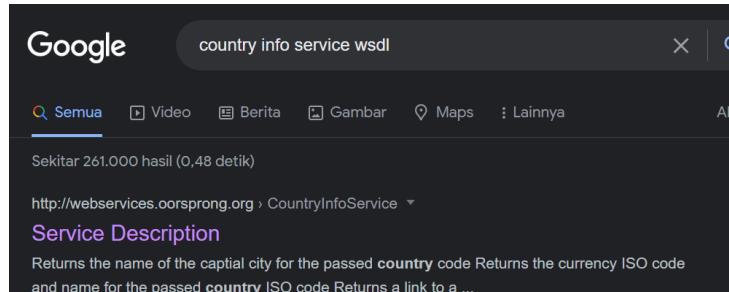
Gambar 8.8 Pilih Web Service Request

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



Gambar 8.9 Membuat Web Service Request

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



Gambar 8.10 Searching Website di Google

11. Berikut tampilan website, lalu copy link addressnya.

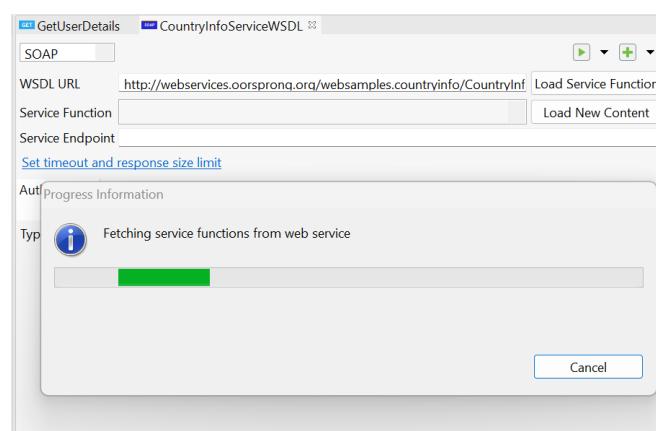
```

<?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/" targetNamespace="http://www.oorsprong.org/wsdl12/" xmlns:tns="http://www.oorsprong.org/websamples.countryinfo" name="CountryInfoService">
  <types>
    <xsd:schema elementFormDefault="qualified" targetNamespace="http://www.oorsprong.org/websamples.countryinfo">
      <xsd:complexType name="tContinent">
        <xsd:sequence>
          <xsd:element name="tCode" type="xs:string"/>
          <xsd:element name="tName" type="xs:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="tCurrency">
        <xsd:sequence>
          <xsd:element name="sISOCode" type="xs:string"/>
          <xsd:element name="sName" type="xs:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="tCountryCodeAndName">
        <xsd:sequence>
          <xsd:element name="sISOCode" type="xs:string"/>
          <xsd:element name="sName" type="xs:string"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="tCountryCodeAndNameGroupedByContinent">
        <xsd:sequence>
          <xsd:element name="Continent" type="tns:tContinent"/>
          <xsd:element name="CountryCodeAndNames" type="tns:ArrayOfCountryCodeAndName"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="tCountryInfo">
        <xsd:sequence>
          <xsd:element name="sISOCode" type="xs:string"/>
          <xsd:element name="sName" type="xs:string"/>
          <xsd:element name="sCapitalCity" type="xs:string"/>
          <xsd:element name="sPhoneCode" type="xs:string"/>
          <xsd:element name="sContinentCode" type="xs:string"/>
          <xsd:element name="sCurrencyISOCode" type="xs:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
  </types>
  <message name="GetUserDetails" type="tns:UserDetailsType">
    <part name="parameters" type="tns:UserDetailsType"/>
    <part name="request" type="tns:UserDetailsType"/>
    <part name="fault" type="tns:UserDetailsFaultType"/>
  </message>
  <message name="GetCountryInfo" type="tns:CountryInfoType">
    <part name="parameters" type="tns:CountryInfoType"/>
    <part name="request" type="tns:CountryInfoType"/>
    <part name="fault" type="tns:CountryInfoFaultType"/>
  </message>
  <message name="GetContinent" type="tns:ContinentType">
    <part name="parameters" type="tns:ContinentType"/>
    <part name="request" type="tns:ContinentType"/>
    <part name="fault" type="tns:ContinentFaultType"/>
  </message>
  <message name="GetCurrency" type="tns:CurrencyType">
    <part name="parameters" type="tns:CurrencyType"/>
    <part name="request" type="tns:CurrencyType"/>
    <part name="fault" type="tns:CurrencyFaultType"/>
  </message>
  <message name="GetCountryCodeAndName" type="tns:CountryCodeAndNameType">
    <part name="parameters" type="tns:CountryCodeAndNameType"/>
    <part name="request" type="tns:CountryCodeAndNameType"/>
    <part name="fault" type="tns:CountryCodeAndNameFaultType"/>
  </message>
  <message name="GetCountryCodeAndNameGroupedByContinent" type="tns:CountryCodeAndNameGroupedByContinentType">
    <part name="parameters" type="tns:CountryCodeAndNameGroupedByContinentType"/>
    <part name="request" type="tns:CountryCodeAndNameGroupedByContinentType"/>
    <part name="fault" type="tns:CountryCodeAndNameGroupedByContinentFaultType"/>
  </message>
  <message name="GetCountryInfoGroupedByContinent" type="tns:CountryInfoGroupedByContinentType">
    <part name="parameters" type="tns:CountryInfoGroupedByContinentType"/>
    <part name="request" type="tns:CountryInfoGroupedByContinentType"/>
    <part name="fault" type="tns:CountryInfoGroupedByContinentFaultType"/>
  </message>
  <port name="CountryInfoServicePort" type="tns:CountryInfoServicePortType">
    <soap:operation soapAction="http://www.oorsprong.org/websamples.countryinfo/CountryInfoService/GetUserDetails"/>
    <binding name="CountryInfoServicePortBinding" type="tns:CountryInfoServicePortTypeBinding">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="GetUserDetails">
        <soap:operation soapAction="http://www.oorsprong.org/websamples.countryinfo/CountryInfoService/GetUserDetails"/>
        <input message="tns:GetUserDetails"/>
        <output message="tns:UserDetailsType"/>
        <fault name="UserDetailsFault" message="tns:UserDetailsFaultType"/>
      </operation>
      <operation name="GetCountryInfo">
        <soap:operation soapAction="http://www.oorsprong.org/websamples.countryinfo/CountryInfoService/GetCountryInfo"/>
        <input message="tns:GetCountryInfo"/>
        <output message="tns:CountryInfoType"/>
        <fault name="CountryInfoFault" message="tns:CountryInfoFaultType"/>
      </operation>
      <operation name="GetContinent">
        <soap:operation soapAction="http://www.oorsprong.org/websamples.countryinfo/CountryInfoService/GetContinent"/>
        <input message="tns:GetContinent"/>
        <output message="tns:ContinentType"/>
        <fault name="ContinentFault" message="tns:ContinentFaultType"/>
      </operation>
      <operation name="GetCurrency">
        <soap:operation soapAction="http://www.oorsprong.org/websamples.countryinfo/CountryInfoService/GetCurrency"/>
        <input message="tns:GetCurrency"/>
        <output message="tns:CurrencyType"/>
        <fault name="CurrencyFault" message="tns:CurrencyFaultType"/>
      </operation>
      <operation name="GetCountryCodeAndName">
        <soap:operation soapAction="http://www.oorsprong.org/websamples.countryinfo/CountryInfoService/GetCountryCodeAndName"/>
        <input message="tns:GetCountryCodeAndName"/>
        <output message="tns:CountryCodeAndNameType"/>
        <fault name="CountryCodeAndNameFault" message="tns:CountryCodeAndNameFaultType"/>
      </operation>
      <operation name="GetCountryCodeAndNameGroupedByContinent">
        <soap:operation soapAction="http://www.oorsprong.org/websamples.countryinfo/CountryInfoService/GetCountryCodeAndNameGroupedByContinent"/>
        <input message="tns:GetCountryCodeAndNameGroupedByContinent"/>
        <output message="tns:CountryCodeAndNameGroupedByContinentType"/>
        <fault name="CountryCodeAndNameGroupedByContinentFault" message="tns:CountryCodeAndNameGroupedByContinentFaultType"/>
      </operation>
      <operation name="GetCountryInfoGroupedByContinent">
        <soap:operation soapAction="http://www.oorsprong.org/websamples.countryinfo/CountryInfoService/GetCountryInfoGroupedByContinent"/>
        <input message="tns:GetCountryInfoGroupedByContinent"/>
        <output message="tns:CountryInfoGroupedByContinentType"/>
        <fault name="CountryInfoGroupedByContinentFault" message="tns:CountryInfoGroupedByContinentFaultType"/>
      </operation>
    </binding>
  </port>

```

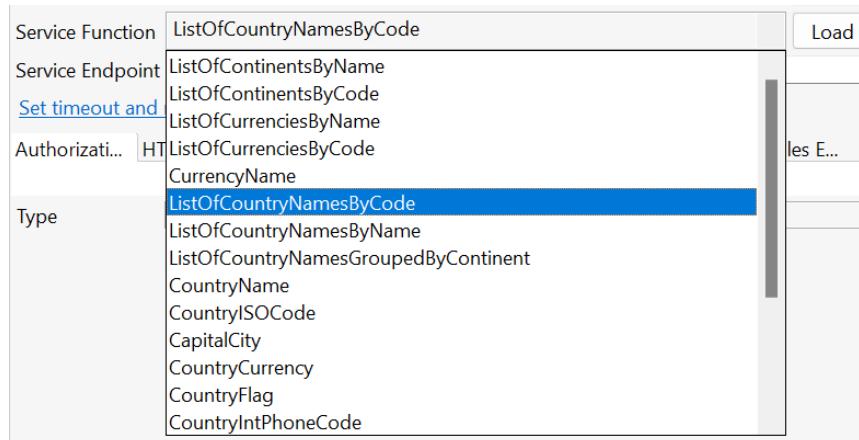
Gambar 8.11 Tampilan Website

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



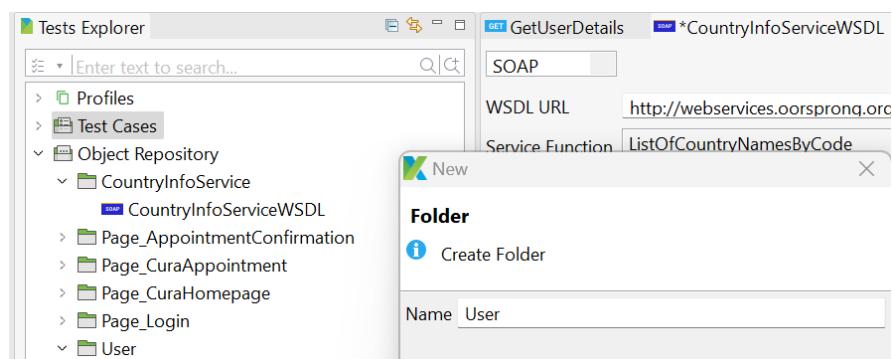
Gambar 8.12 Tampilan File

13. Pada Service Function pilih “ListOfCountryNamesByCode”



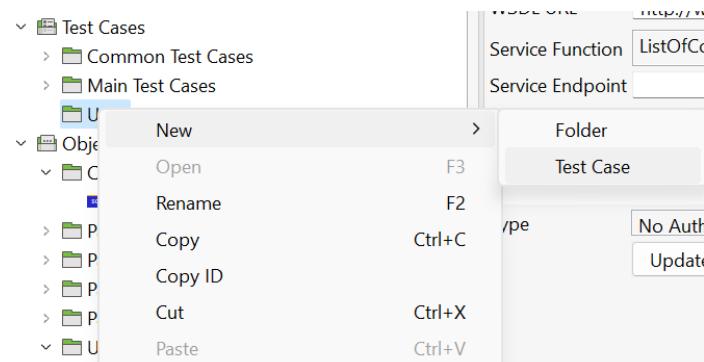
Gambar 8.13 Tampilan Service Function

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



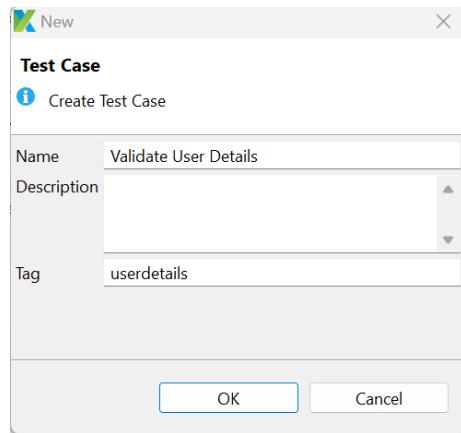
Gambar 8.14 Membuat Folder User

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



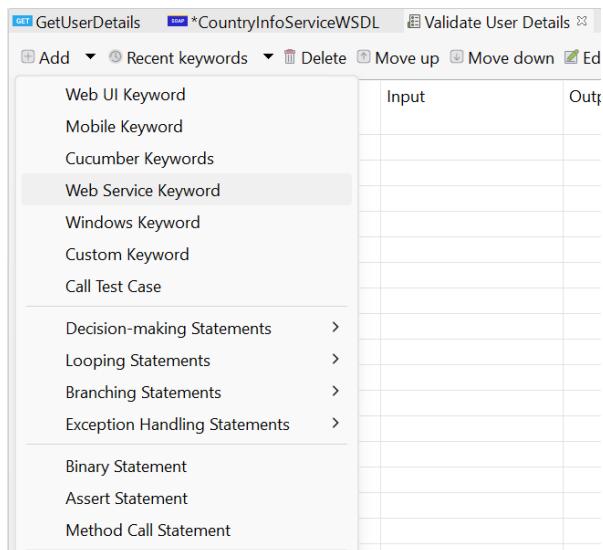
Gambar 8.15 Folder User

16. Isikan form seperti berikut, klik Ok



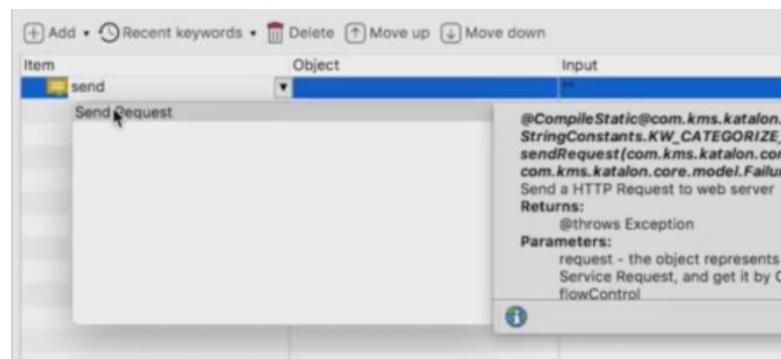
Gambar 8.16 Form Test Case

17. Selanjutnya klik dropdown Add pilih Web Service Keyword



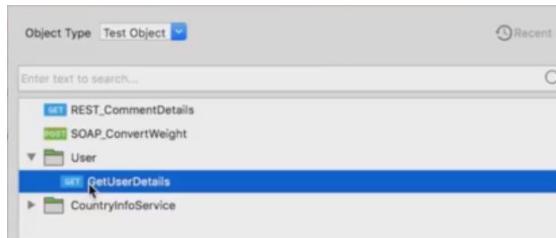
Gambar 8.17 Web Service Keyword

18. Gantikan Namanya menjadi “Send Request”



Gambar 8.18 Send Request

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



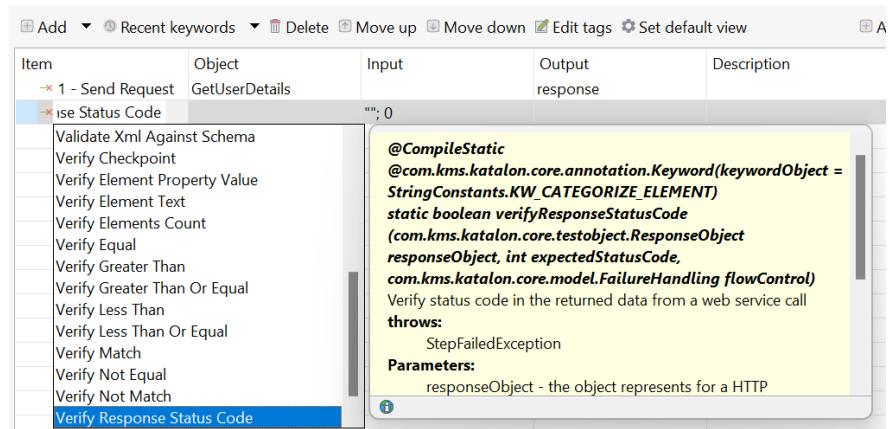
Gambar 8.19 Get User Details

20. Pada Output isikan “response”

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

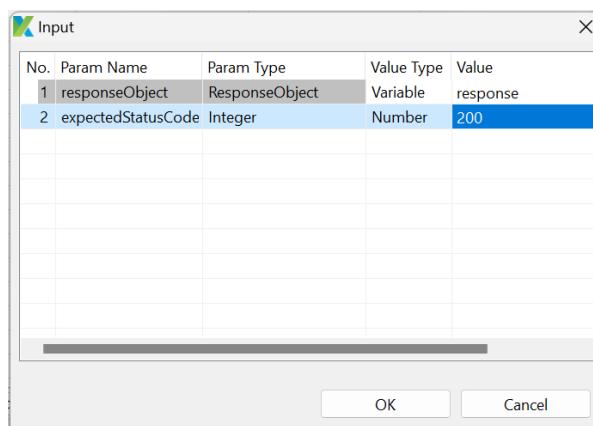
Gambar 8.20 Response Pada Kolom Output

21. Tambahan “Verify Response Status Code”



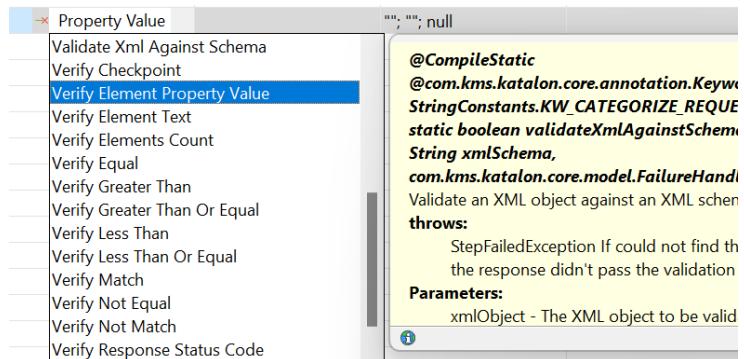
Gambar 8.21 Verify Response Status Code

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



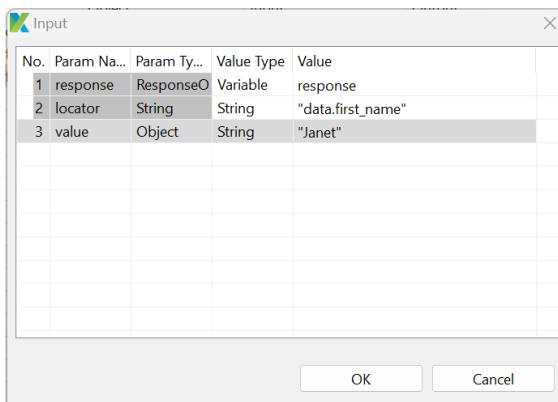
Gambar 8.22 Verify Response Status Code

23. Kemudian tambahkan “Verify Element Property Value”



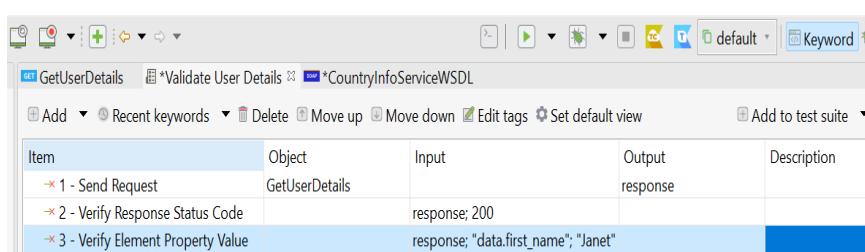
Gambar 8.23 Property Value

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



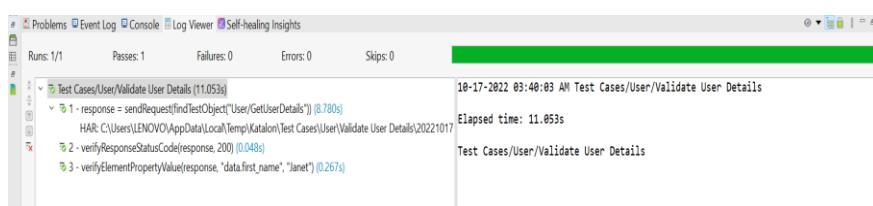
Gambar 8.24 Verify Element Property Value

25. Sehingga menjadi seperti berikut, selanjutnya klik icon play



Gambar 8.25 Klik Icon Play

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



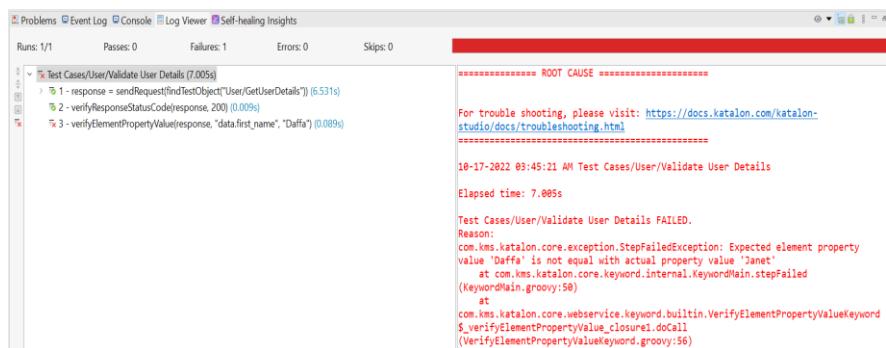
Gambar 8.26 Melihat Hasil di Log Viewer

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 8.27 Mengganti isi dari Verify Element Property Value

28. Maka hasilnya akan error, seperti berikut



Gambar 8.28 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     "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    supporturi: https://reqres.in/#su
}

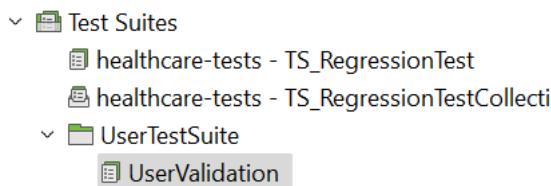
```

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

JSON  XML  HTML  JavaScript  Wrap Lin

Gambar 8.29 Penyebab Error

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



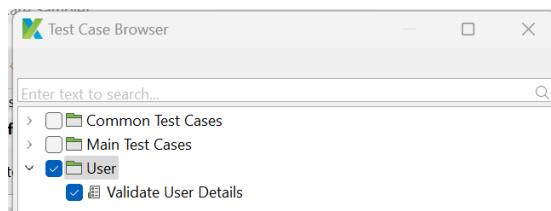
Gambar 8.30 Membuat File UserValidation

31. Berikut tampilannya, klik add

No.	ID	Description	Run
1			
2			
3			
4			

Gambar 8.31 Tampilan dari File UserValidation

32. Centang Validate User Details



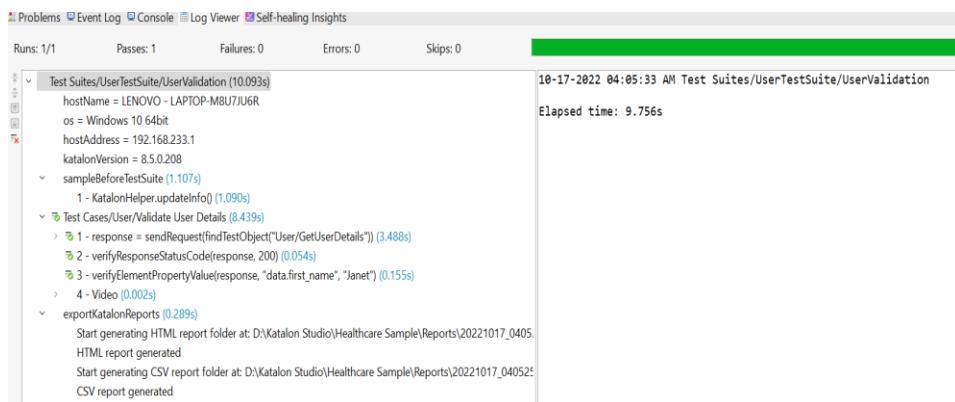
Gambar 8.32 Centang Validate User Details

33. Berikut setelah di add, klik icon play

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

Gambar 8.33 Klik Icon Play

### 34. Berikut hasil dari Log Viewer

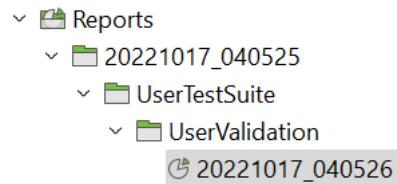


```
Runs: 1/1 Passes: 1 Failures: 0 Errors: 0 Skips: 0
10-17-2022 04:05:33 AM Test Suites/UserTestSuite/UserValidation
Elapsed time: 9.756s

Test Suites/UserTestSuite/UserValidation (10.093s)
  hostName = LENOVO - LAPTOP-M8U7JU6R
  os = Windows 10 64bit
  hostAddress = 192.168.233.1
  katalonVersion = 8.5.0.208
  sampleBeforeTestSuite (1.107s)
    1 - KatalonHelper.updateInfo() (1.090s)
  ✓ Test Cases/User/Validate User Details (8.439s)
    > 1 - response = sendRequest(findTestObject("User/GetUserDetails")) (3.488s)
      2 - verifyResponseStatusCode(response, 200) (0.054s)
      3 - verifyElementPropertyValue(response, "data.first_name", "Janet") (0.155s)
    > 4 - Video (0.002s)
  ✓ exportKatalonReports (0.289s)
    Start generating HTML report folder at: D:\Katalon Studio\Healthcare Sample\Reports\20221017_0405...
    HTML report generated
    Start generating CSV report folder at: D:\Katalon Studio\Healthcare Sample\Reports\20221017_0405...
    CSV report generated
```

Gambar 8.34 Hasil dari Log Viewer

### 35. Selanjutnya kita buka Reports untuk melihat hasil pengujian



Gambar 8.35 Folder Reports Melihat Hasil Pengujian

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

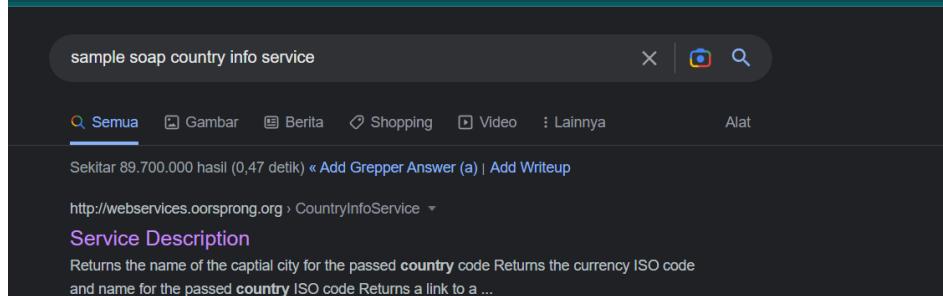
Test Cases Table		Resolution
<input checked="" type="checkbox"/> Passed	<input checked="" type="checkbox"/> Failed	<input checked="" type="checkbox"/> Error
<input checked="" type="checkbox"/> Incomplete		
<input checked="" type="checkbox"/> Skipped		
Export report <a href="#">Katalon TestOps</a>		
Search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

Gambar 8.36 Hasil Pengujian File Json

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

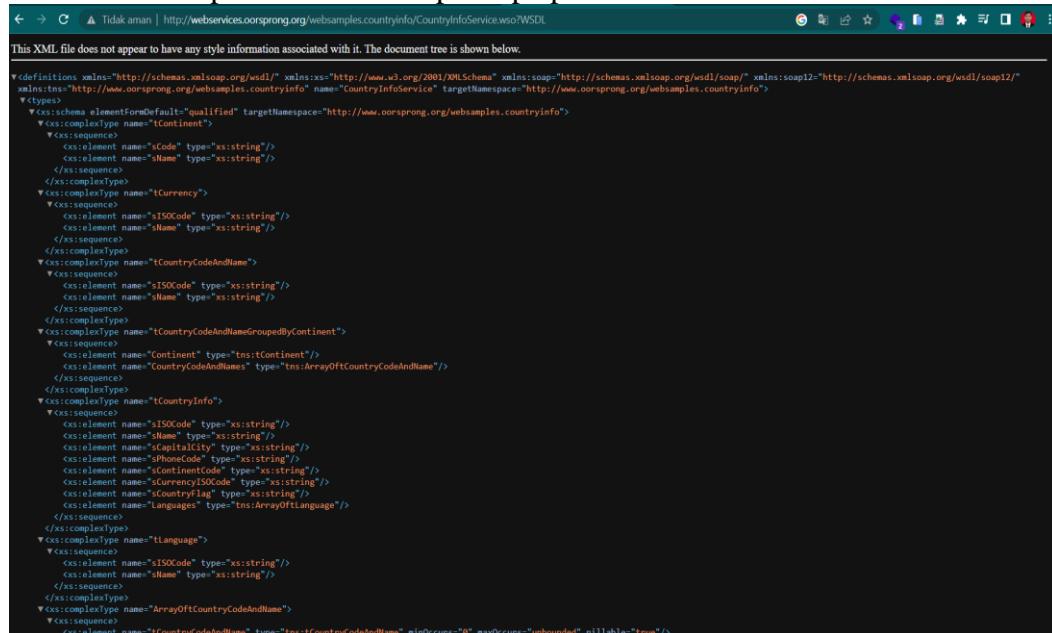
Katalon juga dapat melakukan chaining API. Chaining API adalah suatu proses untuk mengekstrak nilai dari respon suatu API dan memberikan nilai ke permintaan API berikutnya. Pada tahapan ini kita akan melakukan chaining API pada API SOAP yang memiliki respon XML. Berikut tahapannya.

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



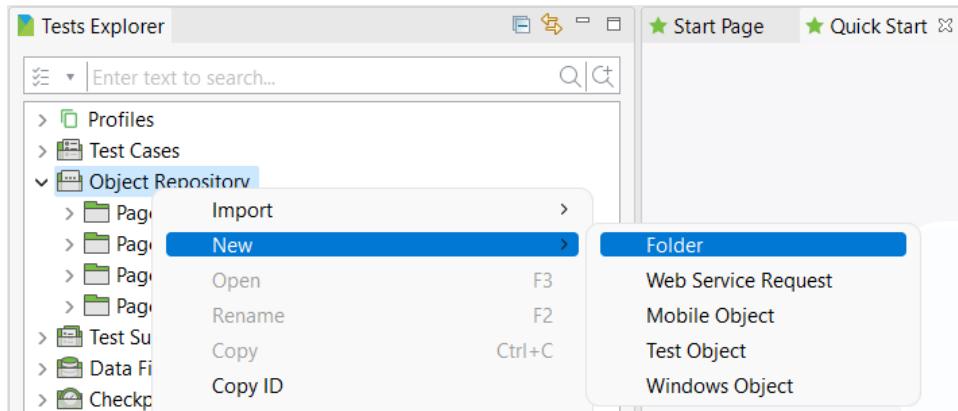
Gambar 8.37 Contoh SOAP API

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



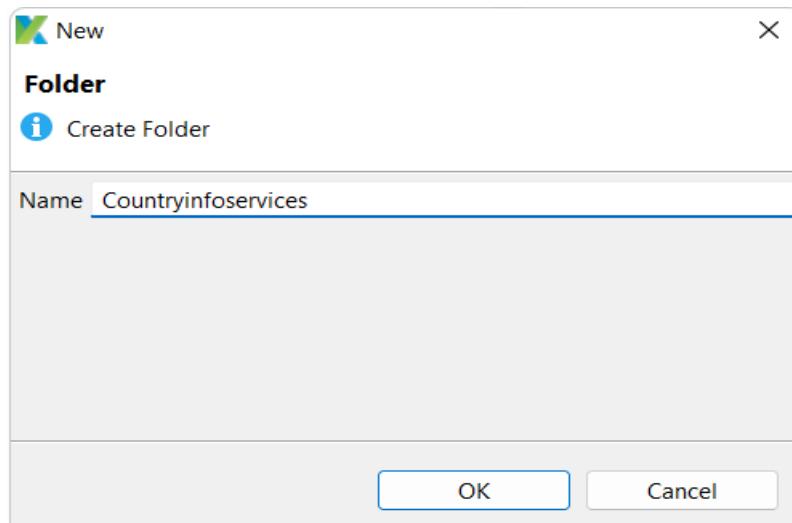
Gambar 8.38 Script SOAP API

3. Selanjutnya buat 1 folder baru.



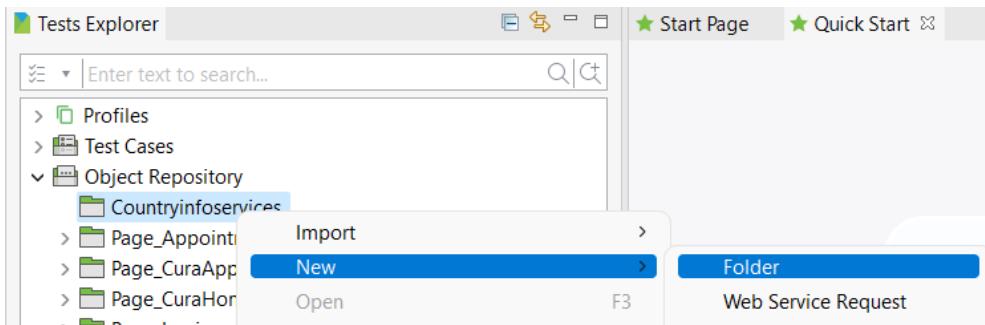
Gambar 8.39 Membuat Folder

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



Gambar 8.40 Folder Countryinfoservices

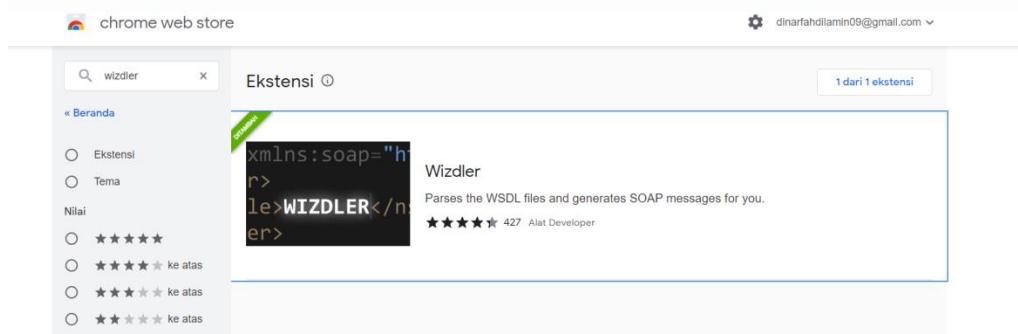
5. Pada folder Countrinfoservice, buat 1 web service request.



Gambar 8.41 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 8.42 Tambahkan Ekstansi Wizdler pada Chrome

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



Gambar 8.43 Ikon dari Ekstansi Wizdler

8. Berikut ini merupakan list request yang tersedia.



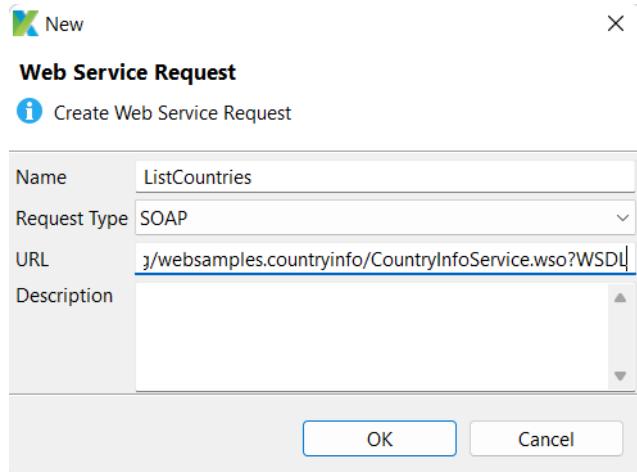
Gambar 8.44 List Request

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



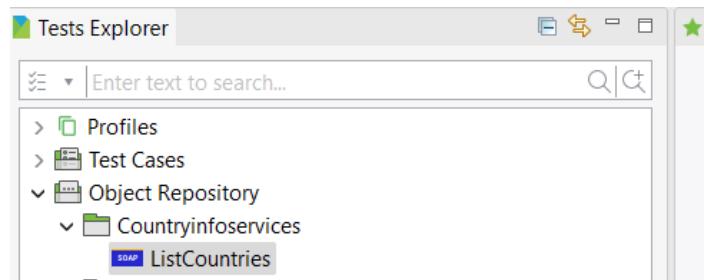
Gambar 8.45 Request dari `ListOfCountryNamesByName`

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



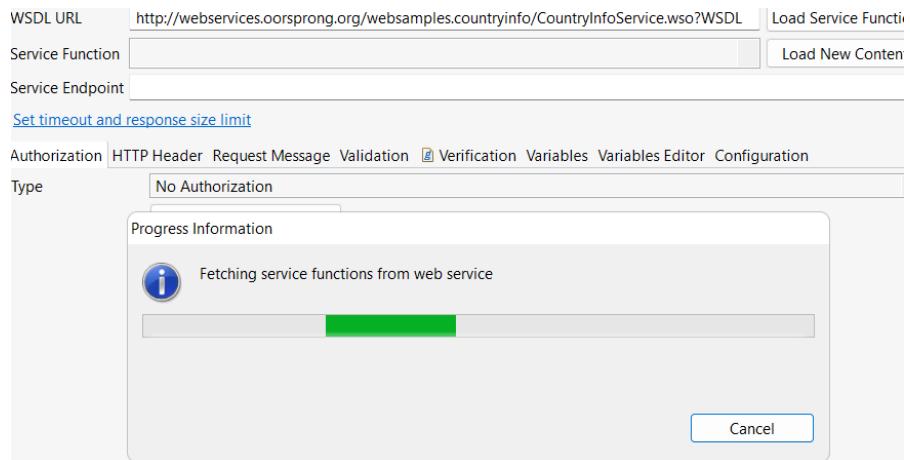
Gambar 8.46 Web Service Request dengan nama ListCountries

11. Dapat kita lihat web service request sudah terbentuk.



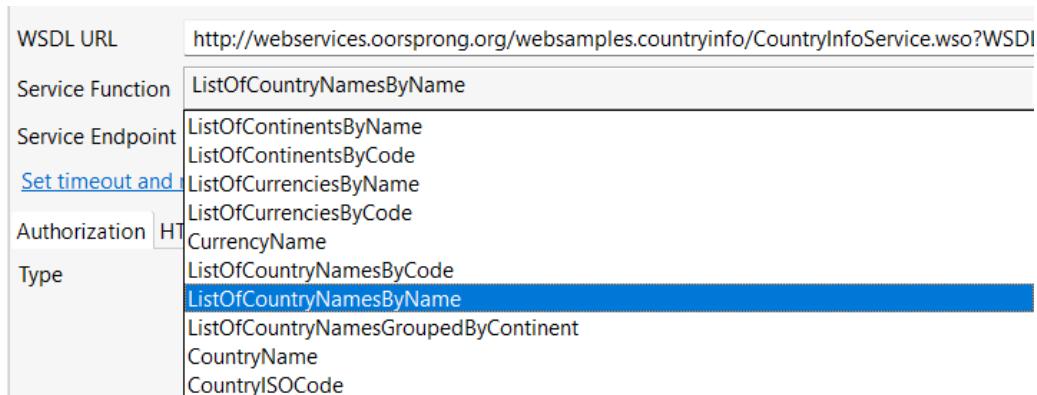
Gambar 8.47 List Countries Sudah Dibuat

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



Gambar 8.48 Load Service Function

13. Pilih ListOfCountryNamesByName untuk service functionnya.



Gambar 8.49 Service Function

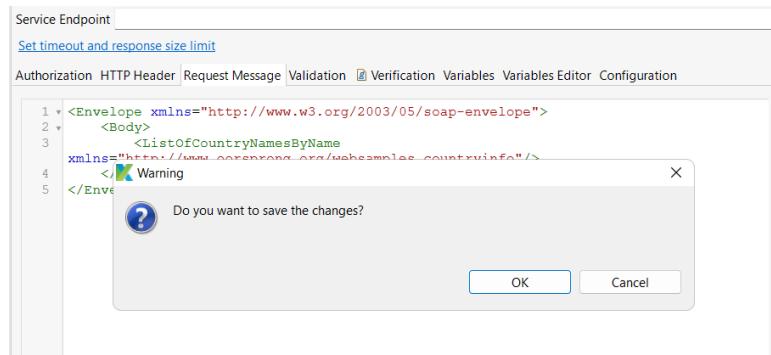
14. Kemudian pada request message copi script yang ada pada url ListOfCountryNamesByName tadi.

A screenshot of a 'Request Message' editor. The tab bar includes 'Authorization', 'HTTP Header', 'Request Message' (selected), 'Validation', 'Verification', 'Variables', 'Variables Editor', and 'Configuration'. The main area contains the following XML code:

```
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>
```

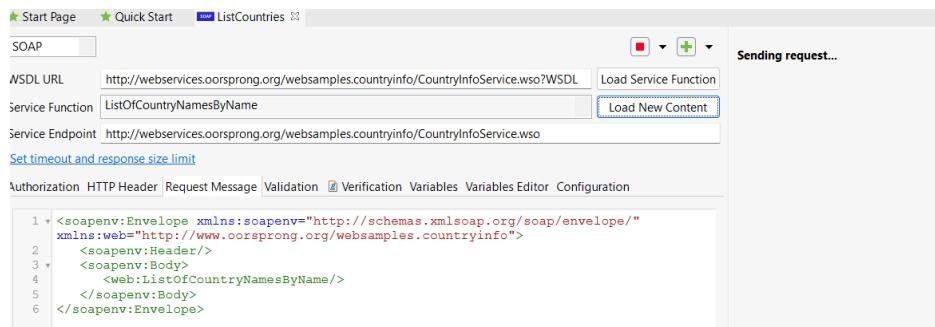
Gambar 8.50 Copy Script

## 15. Kemudian save script.



Gambar 8.51 Save Script

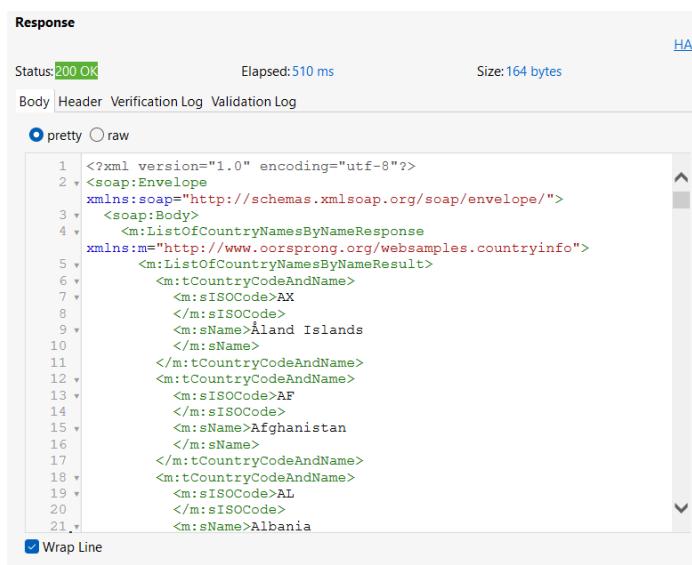
## 16. Selanjutnya jalankan script untuk melakukan request/permintaan.



```
1 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
2   xmlns:web="http://www.oorsprong.org/websamples.countryinfo"  
3   <soapenv:Header>  
4     <soapenv:Body>  
5       <web:ListOfCountryNamesByName/>  
6     </soapenv:Body>  
7   </soapenv:Envelope>
```

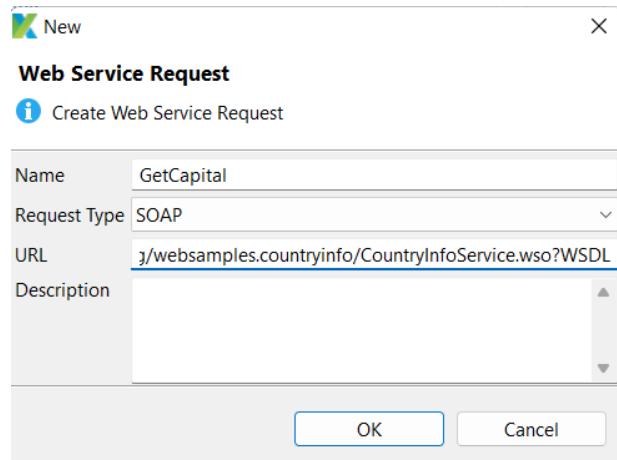
Gambar 8.52 Request Script

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



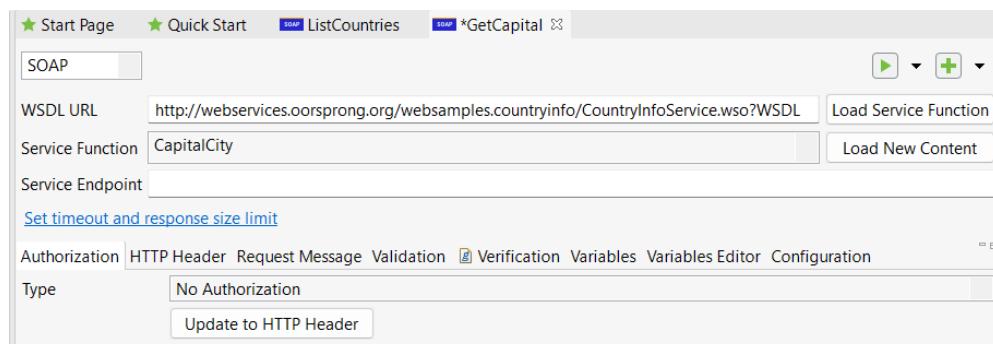
Gambar 8.53 List Nama dari Berbagai Negara

18. Buat web service request lainnya.



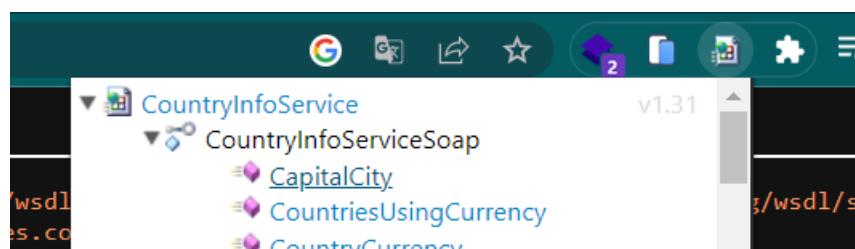
Gambar 8.54 Web Service Request SOAP

19. Pilih server function nya CapitalCity.



Gambar 8.55 Server Function

20. Pada extensi wizdler, pilih CapitalCity.



Gambar 8.56 Capital City

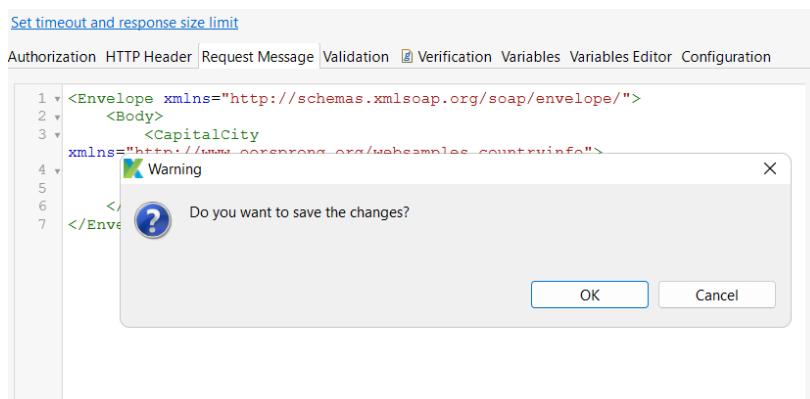
21. Berikut script dari CapitalCity.



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

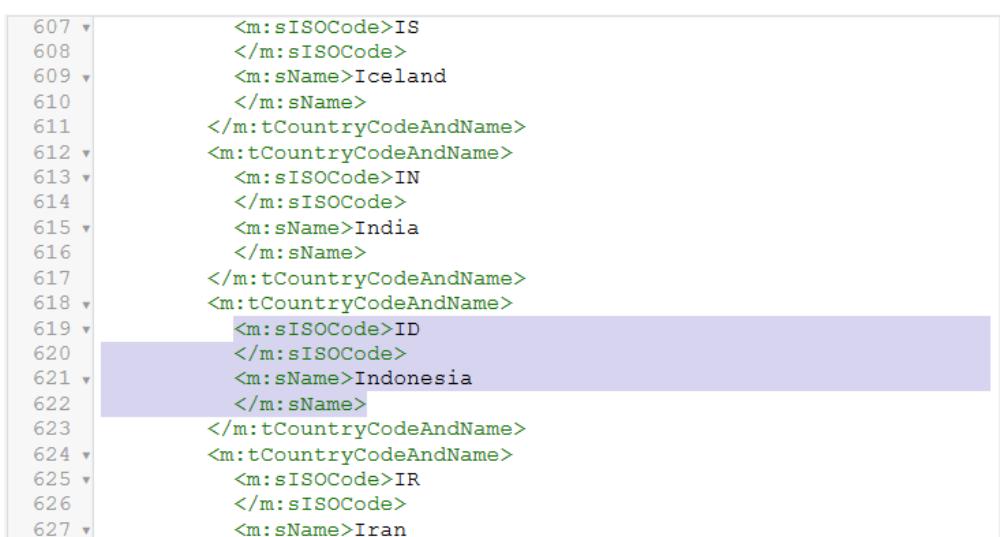
Gambar 8.57 Script CapitalCity

22. Save script.



Gambar 8.58 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 8.59 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 8.60 Mengambil Request Berdasarkan Variabel Tertentu

25. Kemudian pilih menu variabel dan klik add.

Gambar 8.61 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 8.62 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 8.63 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 8.64 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.

Test Case Manager				
Actions		Test Case Details		
Item	Object	Input	Output	Description
-x 1 - Send Request	ListCountries		response1	

Gambar 8.65 Pengujian Test Case dari Kedua Web Service Request

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

Output
response1

Gambar 8.66 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 8.67 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. One test case named "TestOne" is selected.
- Request History:** Shows a recent request for "ListCountryNamesByName".
- Katalon Help:** A help center.
- README.md:** A file viewer.
- ListCountryNamesByName:** The current service function selected.
- GetCapital:** Another service function listed.
- default:** The active profile.
- Response:** Tab showing the response details. Status: 200 OK, Elapsed: 464 ms, Size: 35 KB.
- Body:** Tab showing the raw XML response. The XML content is as follows:

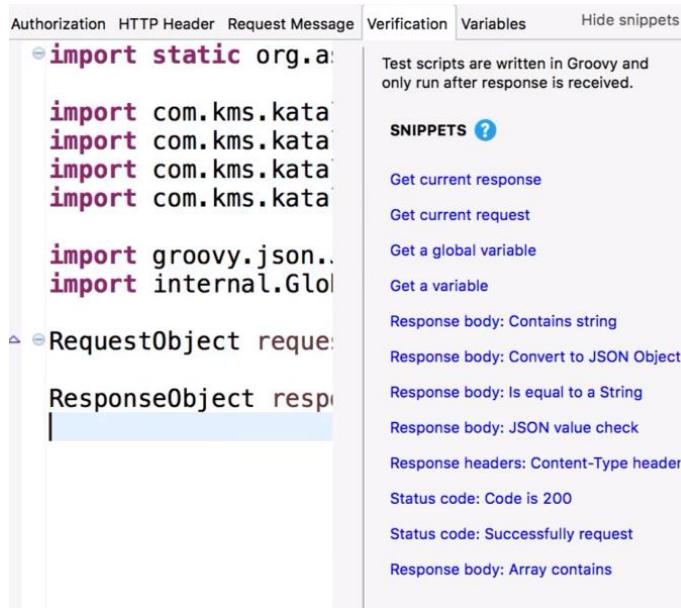
```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <m:ListofCountryNamesByNamesResponse xmlns:m="http://www.oorsprong.org/websamples/countryinfo">
      <m:ListofCountryNamesByNamesResult>
        <m:tCountryCodeAndName>
          <m:sISOCode>AX</m:sISOCode>
          <m:sName>Aland Islands</m:sName>
        </m:tCountryCodeAndName>
        <m:tCountryCodeAndName>
          <m:sISOCode>AF</m:sISOCode>
          <m:sName>Afghanistan</m:sName>
        </m:tCountryCodeAndName>
        <m:tCountryCodeAndName>
          <m:sISOCode>AL</m:sISOCode>
          <m:sName>Albania</m:sName>
        </m:tCountryCodeAndName>
        <m:tCountryCodeAndName>
          <m:sISOCode>DZ</m:sISOCode>
          <m:sName>Algeria</m:sName>
        </m:tCountryCodeAndName>
      </m:ListofCountryNamesByNamesResult>
    </m:ListofCountryNamesByNamesResponse>
  </soap:Body>
</soap:Envelope>

```

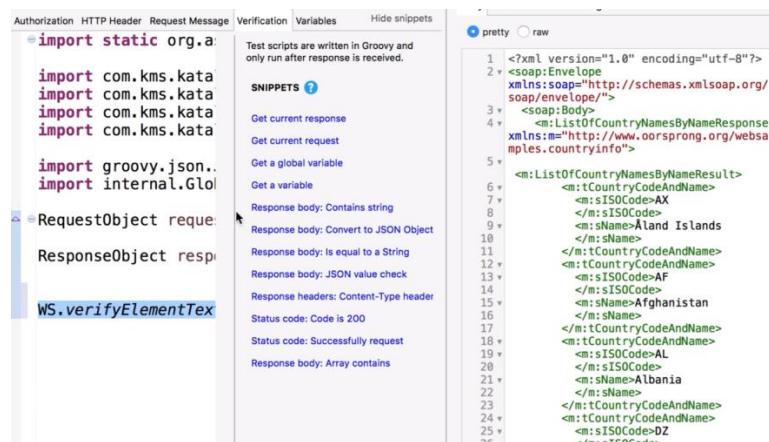
Gambar 8.68 Mengambil Nilai Tertentu dari Respon

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



Gambar 8.69 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 8.70 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 8.71 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 8.72 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]
```

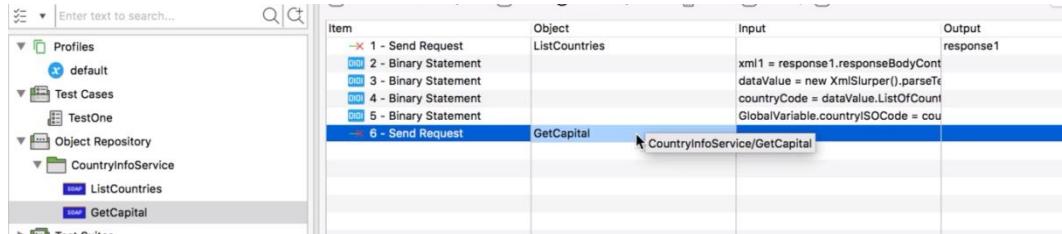
Gambar 8.73 Code dari Variabel Countrycode

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
```

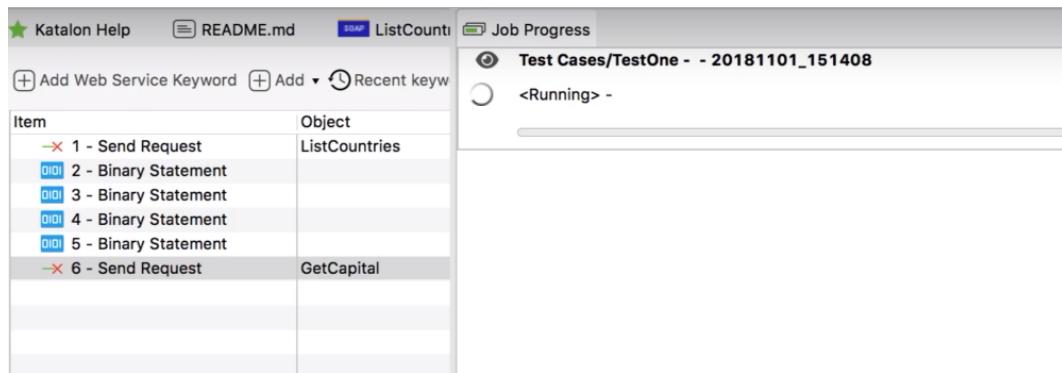
Gambar 8.74 Variabel Countrycode disimpan pada Variabel GlobalVariable

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



Gambar 8.75 Panggil API GetCapital

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



Gambar 8.76 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 = responseBodyCont
11-01-2018 03:14:17 PM - [END] - End action : Statement - xml1 = responseBodyCont
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 = sISOCo
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 8.77 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 WS
import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
import internal.GlobalVariable as GlobalVariable

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

String xml1 = response1.responseBodyContent

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

def countryCode = dataValue.ListOfCountryNamesByNameResult.tCountry[0].countryCode

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

GlobalVariable.countryISOCode = countryCode

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

Gambar 8.78 Ekstrak Country Code

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

```
[Problem] [Console] [Log Viewer]
*terminated* TempCase1541065830872 [Katalon] /Applications/Katalon Studio 6.app/Contents/Eclipse/jre/Contents/Home/jre/bin/java (01-Nov-2018, 3:15:04 PM)

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 - response$ = com.kms.katalon.core.webservice.keyword.katalon.HttpResponse
11-01-2018 03:15:10 PM - [INFO] - Finding Test Object with id 'Object Repository/CountryInfoService/ListCountry'
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 - response$ = com.kms.katalon.core.webservice.keyword.katalon.HttpResponse
11-01-2018 03:15:12 PM - [START] - Start action : Statement - $xml = responseBodyContent
11-01-2018 03:15:13 PM - [END] - End action : Statement - $xml = responseBodyContent
11-01-2018 03:15:13 PM - [START] - Start action : Statement - dataValue = new groovy.util.XmlSlurper().parse($xml)
11-01-2018 03:15:13 PM - [END] - End action : Statement - dataValue = new groovy.util.XmlSlurper().parse($xml)
11-01-2018 03:15:13 PM - [START] - Start action : Statement - countryCode = $isoCode.text()
11-01-2018 03:15:13 PM - [END] - End action : Statement - countryCode = $isoCode.text()
11-01-2018 03:15:13 PM - [START] - Start action : Statement - println("The extracted country code is : " + countryCode)
The extracted country code is : AL
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 8.79 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'	

Gambar 8.80 Validasi Country Code

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

```

<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope">
<Body>
<CapitalCity xmlns="http://www.oorsprong.org/websamples.countryinfo">
<sCountryISOCode>${countryISOCode}</sCountryISOCode>
</CapitalCity>
</Body>
</Envelope>

```

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<CapitalCityResponse xmlns:m="http://www.oorsprong.org/websamples.countryinfo">
<m:CapitalCityResult>Tirana</m:CapitalCityResult>
</CapitalCityResponse>
</soap:Body>
</soap:Envelope>

```

Gambar 8.81 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.getRequest()
ResponseObject response = WSResponseManager.getRespon

```

```

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

```

Test scripts are written in Groovy and only run after response is received.

**SNIPPETS**

- Get current response
- Get current request
- Get a global variable
- Get a variable
- Response body: Contains string
- Response body: Convert to JSON Object
- Response body: Is equal to a String
- Response body: JSON value check
- Response headers: Content-Type header
- Status code: Code is 200
- Status code: Successfully request
- Response body: Array contains

Gambar 8.82 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].countryISOCode	
5 - Method Call Statement		println("The extracted country code is: " + countryCode)	
6 - Binary Statement		GlobalVariable.countryISOCode = countryCode	
Send Request And Verify	GetCapital		

Gambar 8.83 Memverifikasi Request

48. Kemudian save script, dan jalankan test case.

Job Progress
Test Cases/TestOne - 20181101_151649
<Running> -
Test Cases/TestOne - 20181101_151503
<Done> -
Test Cases/TestOne - 20181101_151408
<Done> -

Gambar 8.84 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.

Name:	verifyElementText
Start:	11-01-2018 03:16:59 PM
Elapsed time:	0.080s
Message:	Verify element text successfully

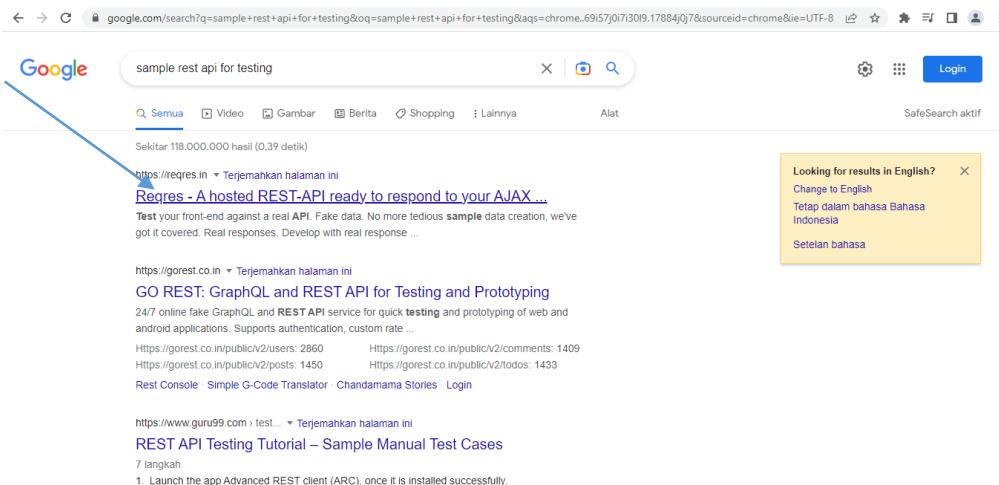
Gambar 8.85 Verifikasi yang Sudah Berhasil

### 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



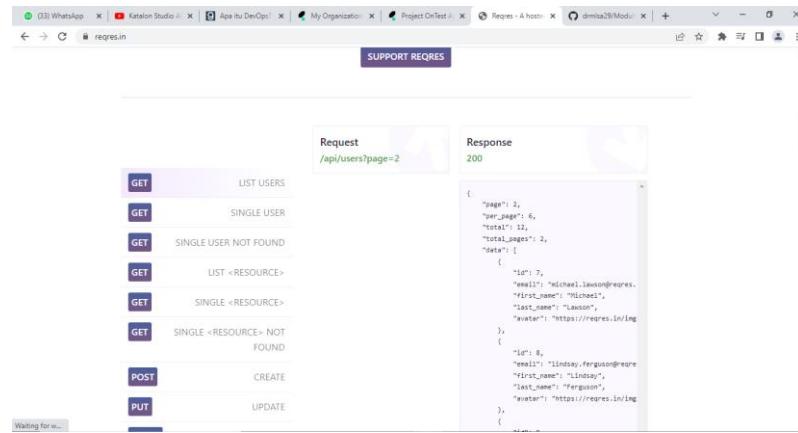
Gambar 8.86 Sample Rest API

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



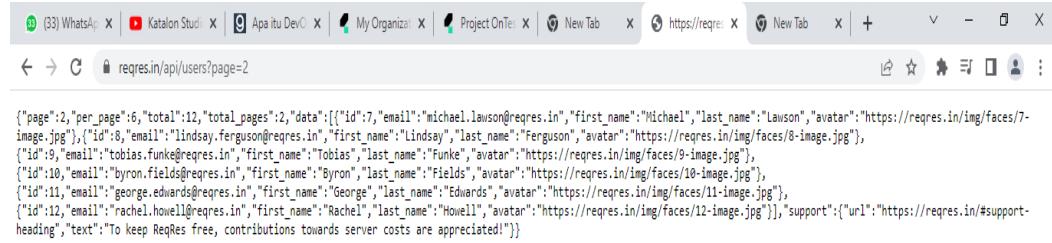
Gambar 8.87 URL Request

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.



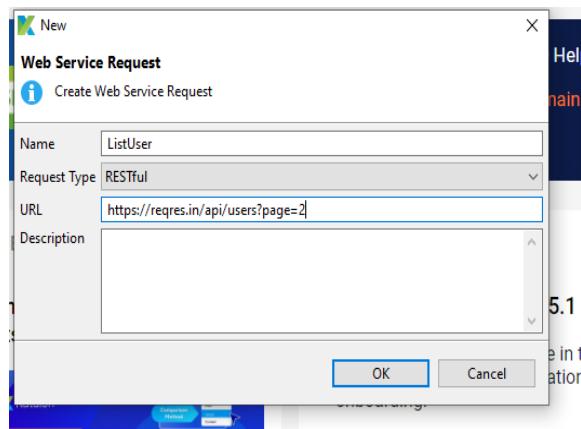
Gambar 8.88 Salin URL

- 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



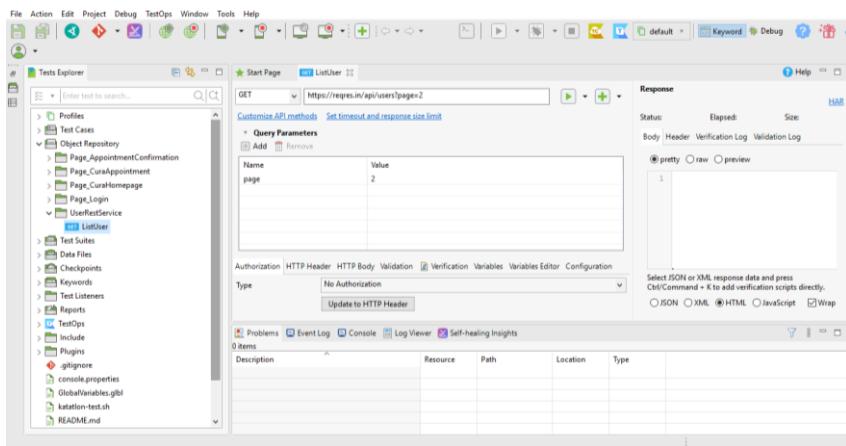
Gambar 8.89 Respon dari Teks URL reqres.in

- 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



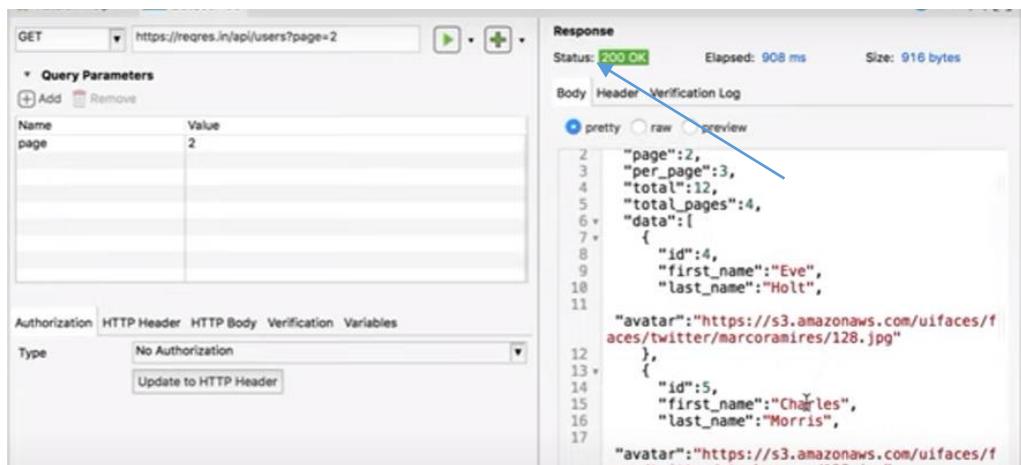
Gambar 8.90 Web Service Request pada reqres.in

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 8.91 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 8.92 Hasil Pengujian API-CHAINING

## BAB 9

### MOBILE TEST MENGGUNAKAN KATALON STUDIO

#### 9.1. Tujuan

Setelah menyelesaikan bab ini, praktikan diharapkan dapat :

- Mampu menginstal Node.js, Java dan Appium.
- Dapat melakukan *mobile (android) testing* menggunakan Katalon Studio.

#### 9.2. Dasar Teori

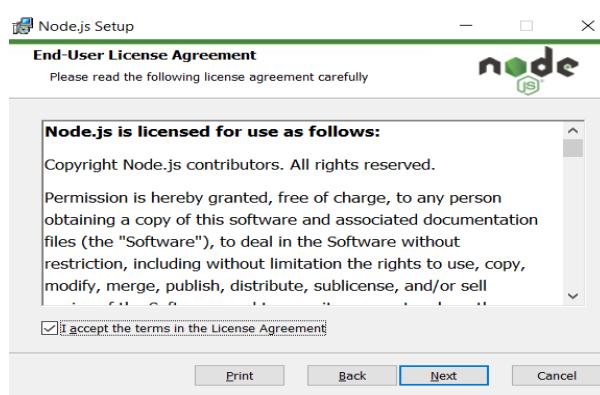
Katalon Studio adalah salah satu *software* yang digunakan untuk *automation testing* aplikasi berbasis web dan mobile. Pada *Mobile testing* khususnya android, memiliki beberapa tipe *test* tergantung oleh subjeknya sebagai berikut:

- a. **Functional testing** berfungsi untuk mengetahui apakah aplikasi berjalan dengan sesuai yang diinginkan.
- b. **Perfomance testing** berfungsi untuk mengetahui apakah aplikasi berjalan dengan tanggap dan efisien.
- c. **Accessibility testing** berfungsi untuk mengetahui apakah aplikasi berjalan dengan servis aksebilitas.
- d. **Complality testing** berfungsi untuk mengetahui apakah aplikasi berjalan pada setiap device dan level API.

#### 9.3. Percobaan

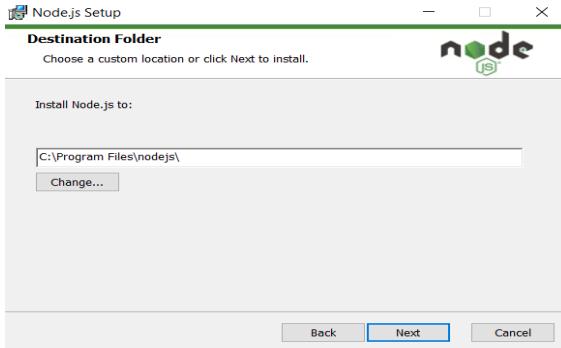
Mobile (Android) Testing di Window Menggunakan Katalon

1. Download node.js pada <https://nodejs.org/en/download/>
2. Buka aplikasi instalasi node.js, lalu centang *license agreement* dan klik next.



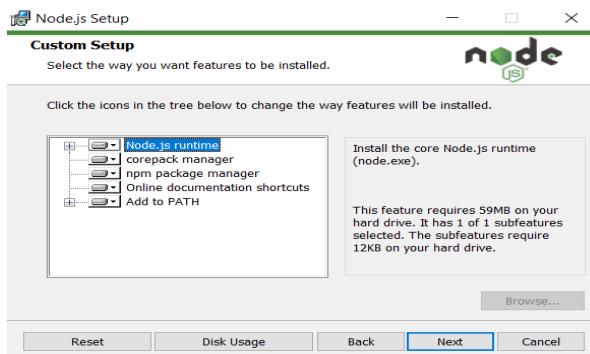
Gambar 9.1 Node.js Setup

3. Tentukan lokasi instalasi node.js, lalu klik next.



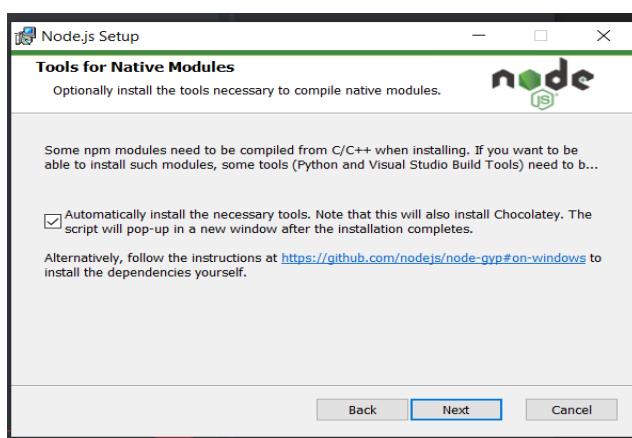
Gambar 9.2 Lokasi Instalasi Node.js

4. Pilih fitur yang ingin diinstal, pada bagian ini bisa menggunakan *setting default*, klik next.



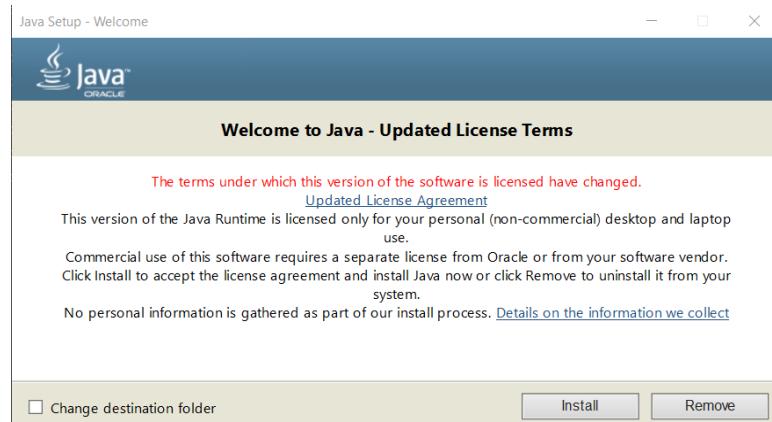
Gambar 9.3 Setting Default

5. Centang *automatically install* untuk menginstal secara otomatis *tools* yang akan dibutuhkan.



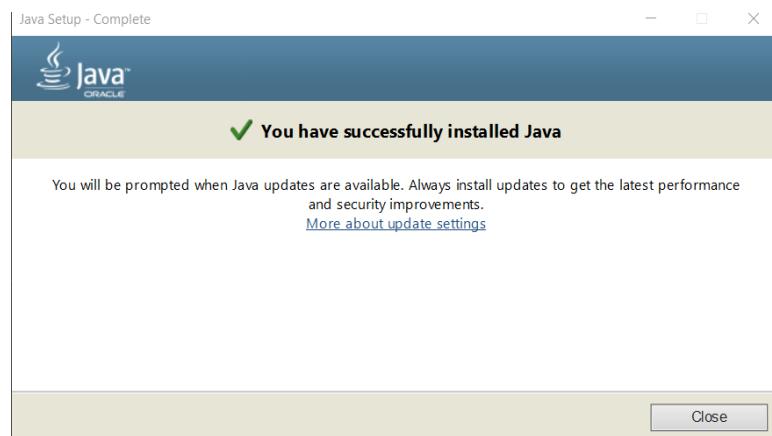
Gambar 9.4 Tool Automatically Install

6. Setelah selesai klik *finish* untuk mengakhiri proses instalasi tersebut.
7. Kemudian download Java pada [https://www.java.com/download/ie\\_manual.jsp](https://www.java.com/download/ie_manual.jsp)
8. Pada tampilan ini akan terdapat *license terms* dari Java, lalu centang *Change destination folder*, jika butuh untuk mengubah lokasi instalasi nya, klik *install*.



Gambar 9.5 Update License Tems

9. Tunggu proses instalasi selesai. Setelah proses instalasi selesai, klik *close*



Gambar 9.6 Tahap Instalasi

10. Untuk mengecek apakah Java sudah terinstall, buka *command prompt* dan masukkan *command* seperti di bawah ini.

```
C:\Users\ASUS>java -version
java version "1.8.0_341"
Java(TM) SE Runtime Environment (build 1.8.0_341-b10)
Java HotSpot(TM) Client VM (build 25.341-b10, mixed mode)
```

Gambar 9.7 Mengecek Instalasi Java

11. Selanjutnya install Appium, buka *command prompt* dan masukkan *command* seperti di bawah ini.

```
C:\Users\ASUS>npm install -g appium
```

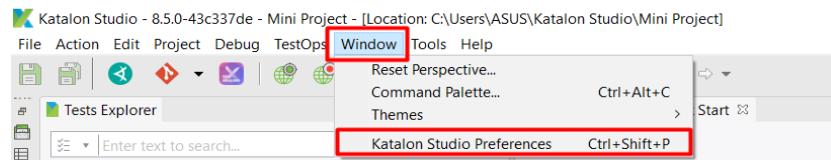
Gambar 9.8 Install Appium

12. Tunggu proses instalasi selesai.

```
C:\Users\ASUS>npm install -g appium
[██████████] \ reify:type-fest: http fetch GET 200 https://registry.npmjs.org/type-fest/-/type-fest-0.20.2.tgz
```

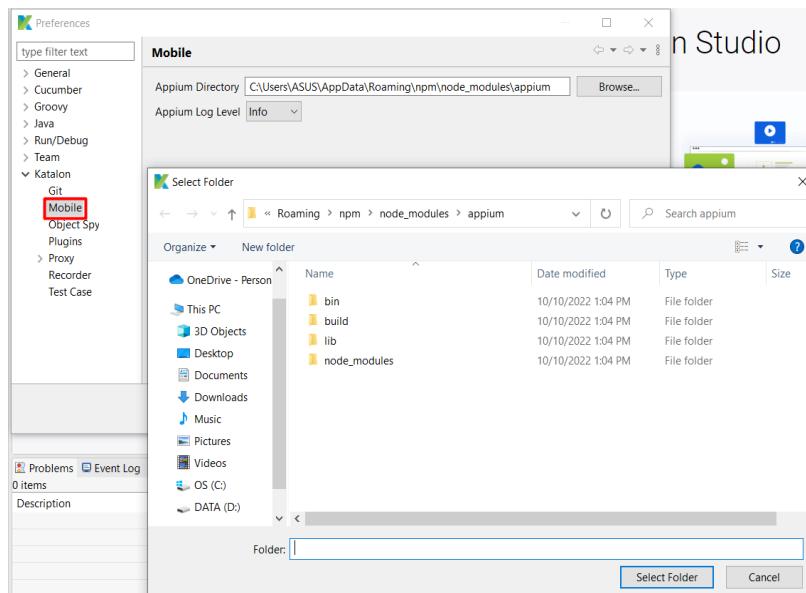
Gambar 9.9 Tahap Instalasi Selesai

13. Setelah melakukan penginstalan buka Katalon Studio dan buat *project* baru dengan pilihan *mobile testing*, kemudian klik pada *window* lalu klik Katalon Studio *Preferences*.



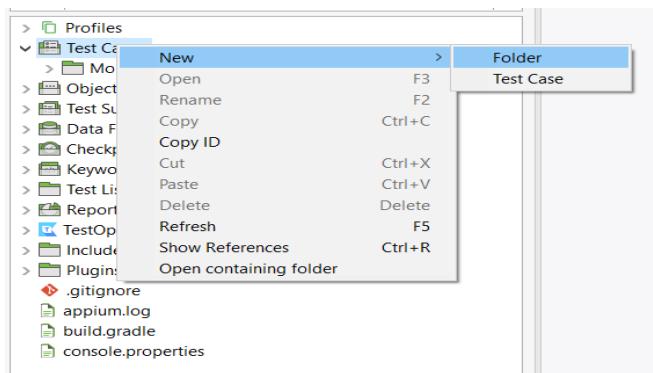
Gambar 9.10 Membuat New Project

14. Pada halaman *preferences*, klik Katalon dan pilih *Mobile*, klik *browse* dan pilih lokasi penginstalan Appium yang sudah diinstal. Setelah memilih lokasi klik *apply and close*.



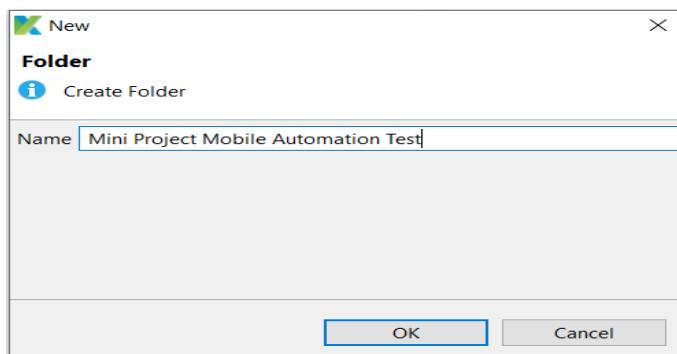
Gambar 9.11 Memilih Destination Folder

15. Pada bagian kiri tampilan Katalon Studio, klik *Test Case* – klik kanan – *new – folder*.



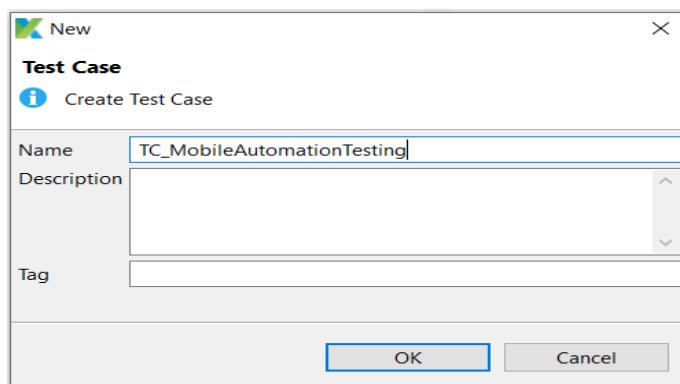
Gambar 9.12 Membuat New Folder Test Case

16. Masukkan nama *folder* dan klik OK.



Gambar 9.13 Create Folder

17. Klik kanan pada *folder* yang sudah dibuat – *new – test case* kemudian masukkan nama *test case*, deskripsi dan tag jika diperlukan.



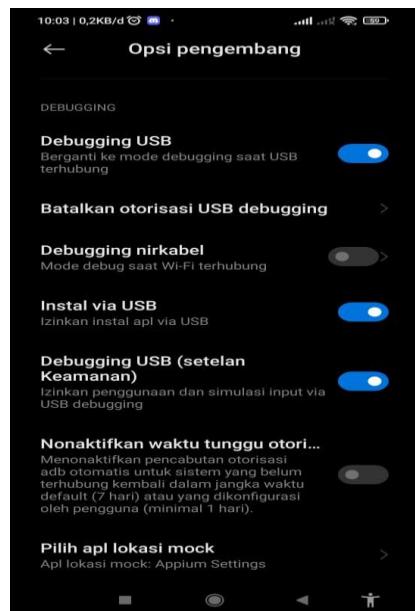
Gambar 9.14 Create Test Case

18. Selanjutnya, sambung Android pada pc/laptop menggunakan USB, Buka tentang telefon (*about phone*) pada setting android, klik 7x pada versi MIUI/*Build Number* (pada android biasanya dituliskan versi OS dengan nama masing-masing) untuk masuk ke mode *developer*.



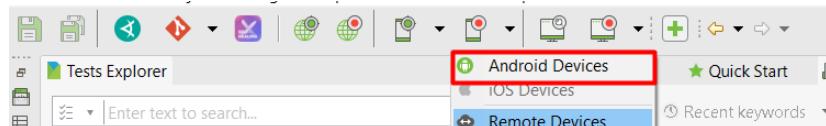
Gambar 9.15 Mengecek About Phone

19. Buka opsi pengembang dan aktifkan opsi pengembang, lalu *Scroll* ke bawah dan aktif *debugging usb* dan instal via usb.



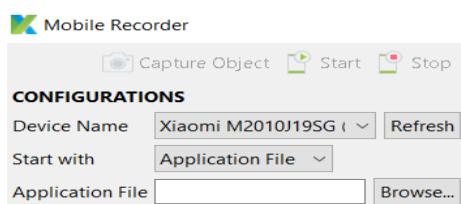
Gambar 9.16 Setting Opsi Pengembang

20. Kembali ke Katalon Studio, klik *record mobile* dan klik *android devices*.



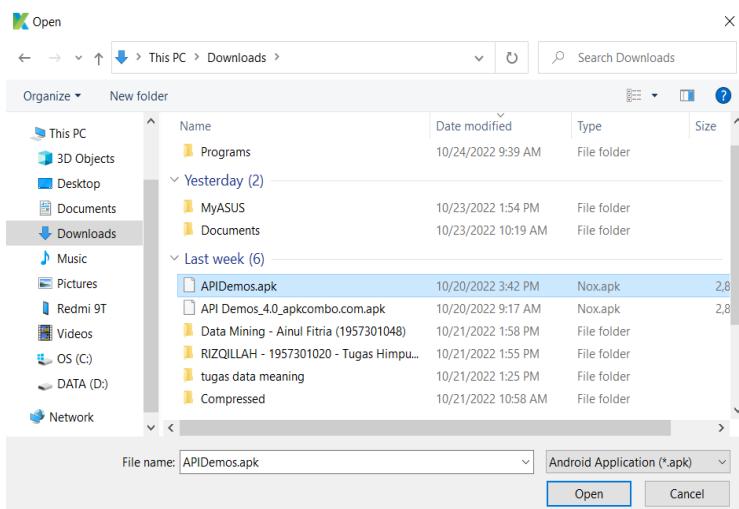
Gambar 9.17 Menjalankan Record Mobile

21. Pada *configurations* klik *refresh* dan klik *browse*.



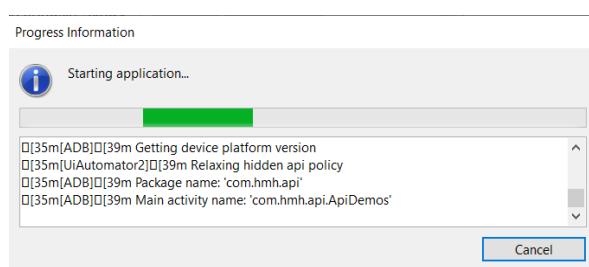
Gambar 9.18 Configuration pada Mobile Recorder

22. Pilih APIDemos.apk (*Download APIDemos terlebih dahulu*) dan klik *open*.



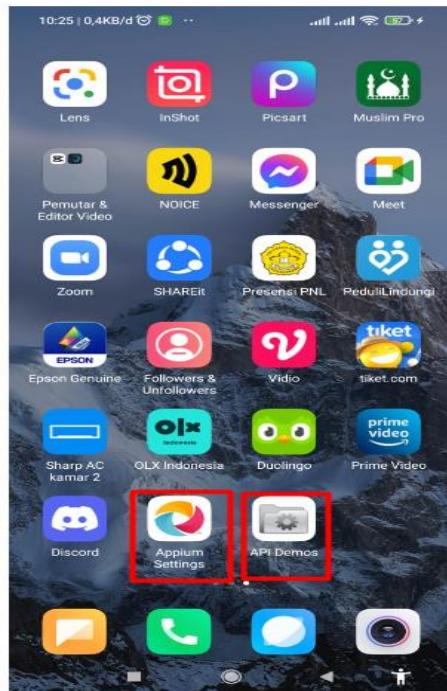
Gambar 9.19 Open APIDemos

23. Selanjutkan klik *start* untuk memulai APIDemos dan tunggu *booting* aplikasi hingga selesai.



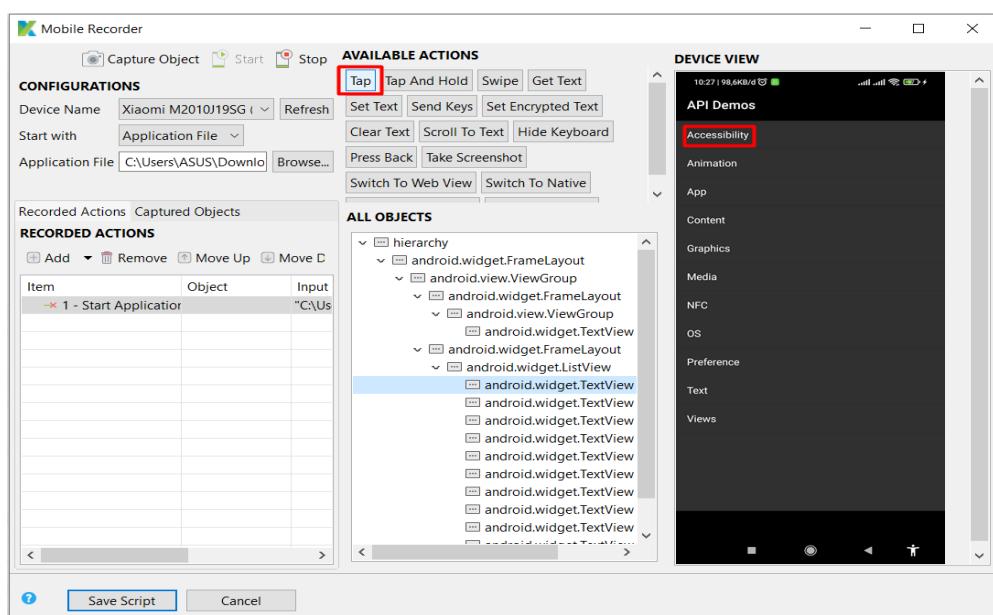
Gambar 9.20 Start application APIDemos

24. Pada saat *booting* aplikasi, android akan diminta untuk menginstal Appium dan API versi android.



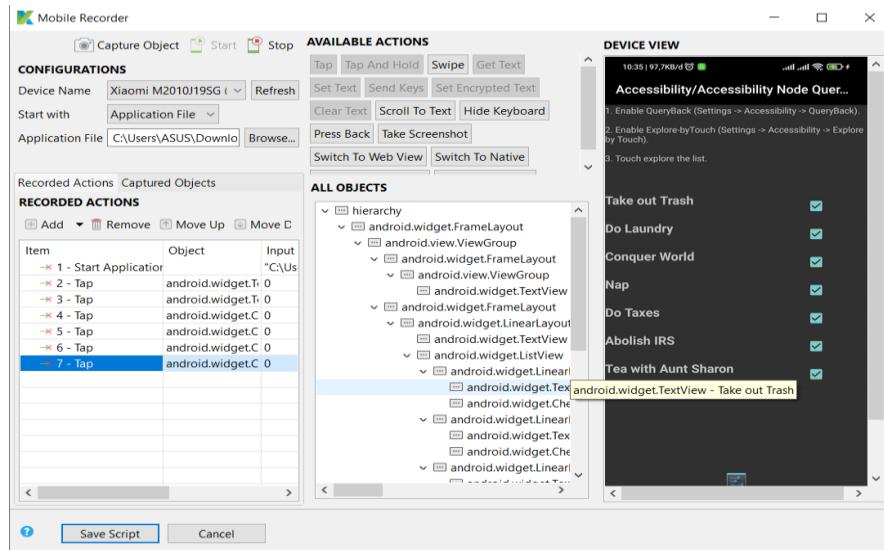
Gambar 9.21 Menginstall Appium dan API

25. Pada tampilan APIDemos pilih *test* yang ingin dilakukan, terdapat gambar di bawah ini pilih *accessibility*, lalu klik *tap* pada *available actions*.



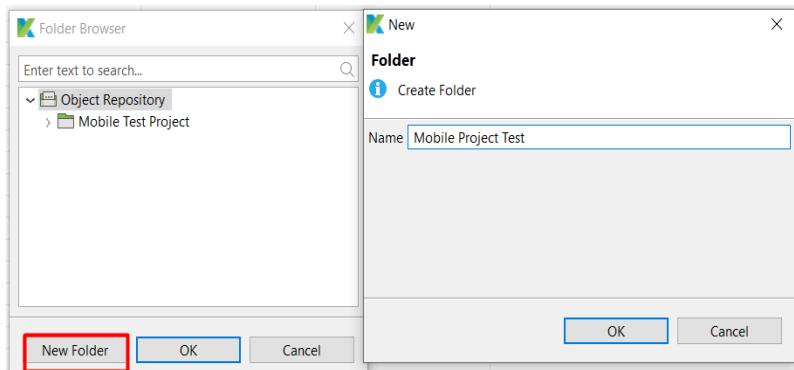
Gambar 9.22 Test pada APIDemos

26. Pilih *accessibility node querying* dan klik tap. Lalu klik *conquer world* dan lalu klik *tap*. Ulangi Langkah di atas untuk ceklis semua pilihan yang ada dan klik *saves script*.



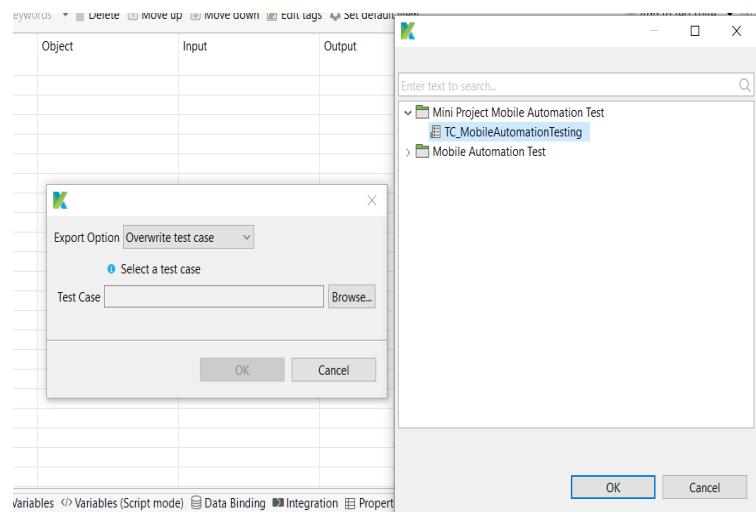
Gambar 9.23 Save Script APIDemos

27. Klik *new folder* dan masukkan nama pada *folder*.



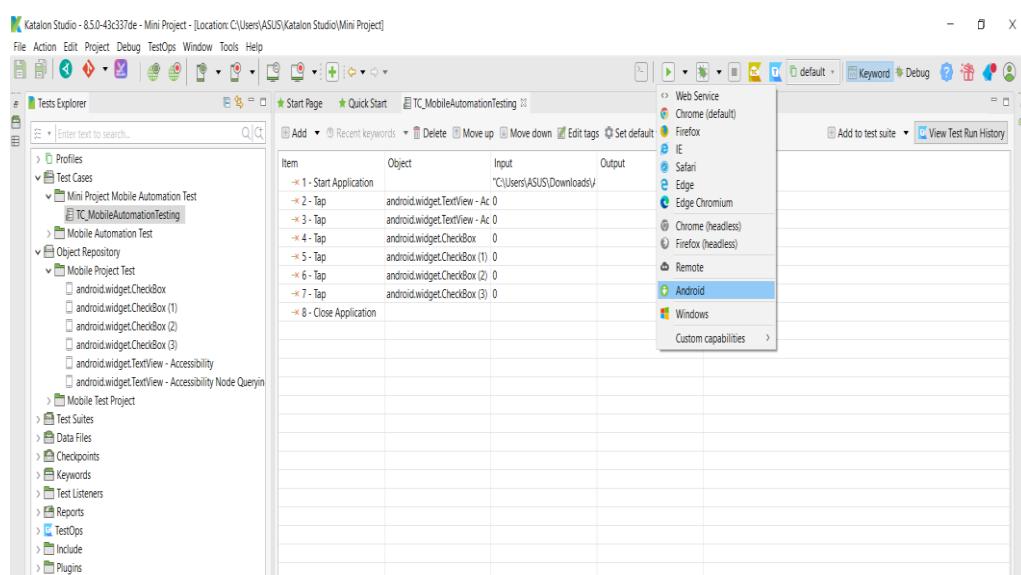
Gambar 9.24 Create Folder APIDemos

28. Pada *export option* pilih *overwrite test case – folder test case* yang sudah dibuat sebelumnya, lalu klik OK.



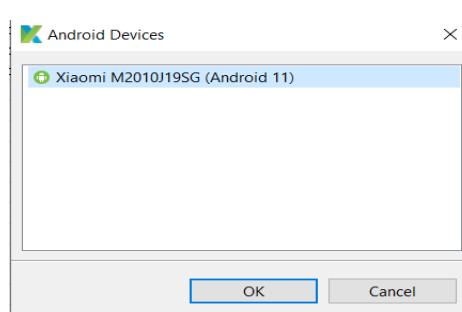
Gambar 9.25 Export Option Mobile Automation Testing

29. Klik *run* dan pilih Android.



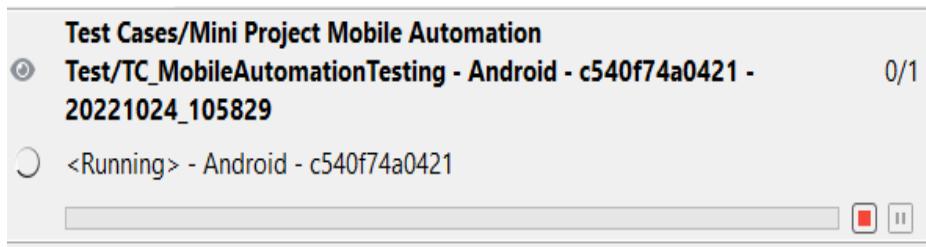
Gambar 9.26 Run Mobile Automation Testing

30. Pilih Android *devices* yang terhubung dan klik OK.



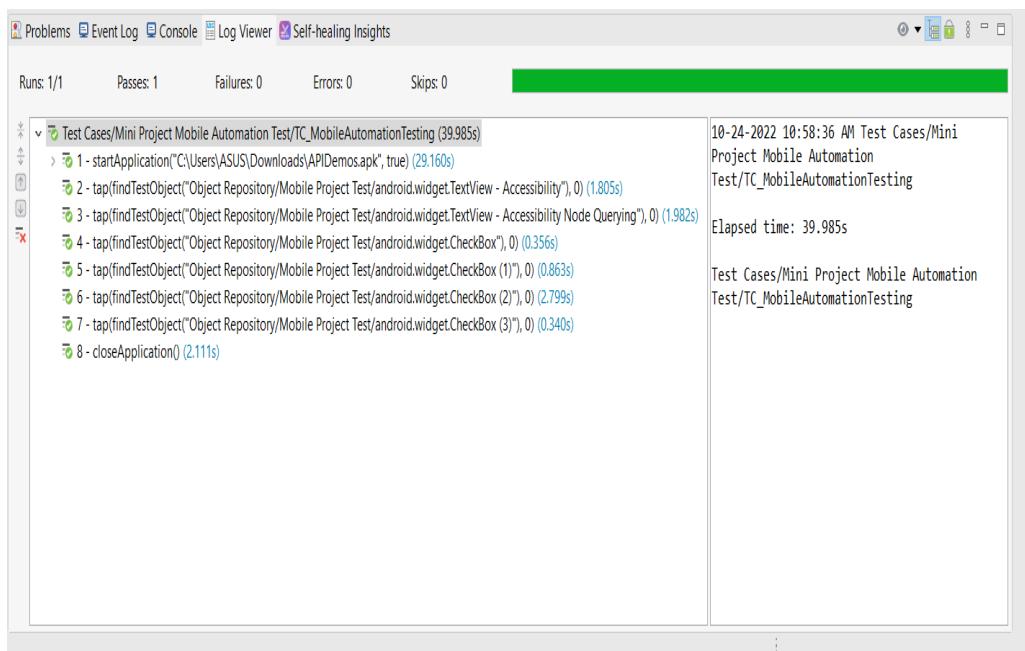
Gambar 9.27 Android Device

31. Tunggu proses *running* hingga selesai.



Gambar 9.28 Proses Running Mobile Automation Testing

32. Setelah proses *running* selesai, akan tampilan pada *log viewer test-test* yang telah diuji, dan berikut adalah tampilan *test* yang telah berhasil dijalankan.



Gambar 9.29 Tampilan Log Viewer Test

## BAB 10

### MEMBUAT ENVIRONMENT

#### 10.1. Tujuan

- 1) Mahasiswa dapat memahami cara membuat Environment dari menu Profiles pada Katalon Studio.
- 2) Mahasiswa dapat memahami cara mengambil nilai dari Profiles.
- 3) Cara mengatur Environment pada saat Test Execution.
- 4) Cara mengatur Environment pada Test Suite dan Test Suite Collection

#### 10.2. Dasar Teori

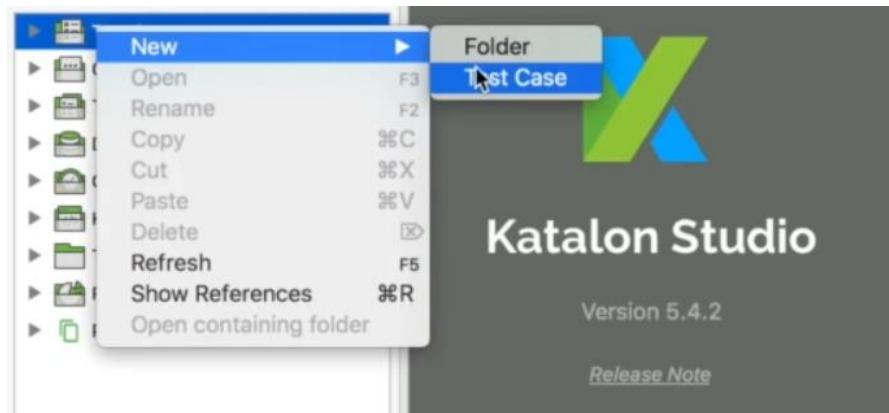
Environment adalah segala hal yang mendukung berkembangnya suatu perangkat lunak. Hal tersebut termasuk system operasi, database, dan tools yang digunakan dalam pembuatan perangkat lunak tersebut.

Test case adalah rangkaian tindakan yang dilakukan oleh tester untuk melakukan verifikasi terhadap fitur atau fungsi tertentu dari sebuah perangkat lunak.

Test Suite adalah kumpulan dari beberapa Test Case untuk menguji program yang memiliki perilaku tertentu.

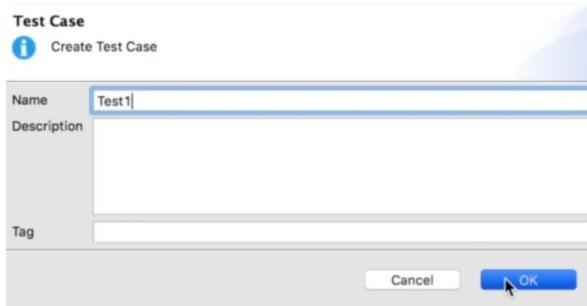
#### 10.3. Percobaan

- 1) Membuat *Test Case* Pada Katalon Studio dengan cara klik kanan>New>*Test Case* pada project yang telah dibuat



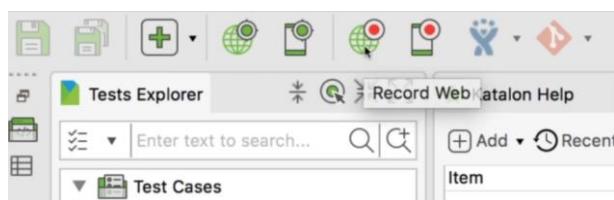
Gambar 10.1 Membuat Test Case

- 2) Memberikan nama pada *Test Case* baru



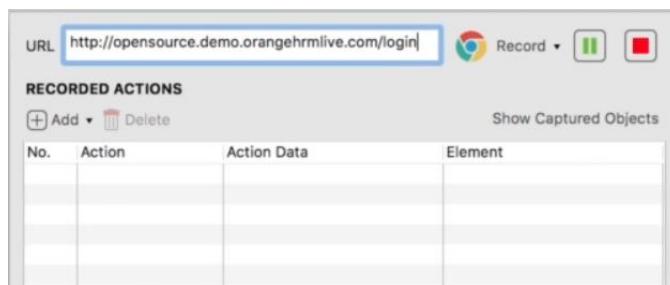
Gambar 10.2 Memberi Nama Test

- 3) Record Pengujian melalui web dengan klik menu *Record Web* pada Katalon Studio



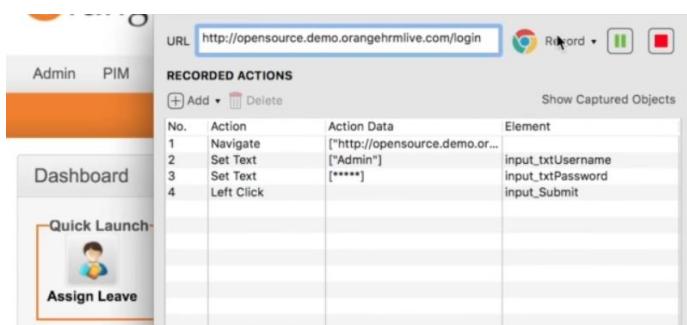
Gambar 10.3 Record Web

- 4) Masukkan link website yang akan dilakukan test



Gambar 10.4 Link Website yang akan Dilakukan Pengujian

- 5) Login dengan memasukkan *Username* dan *Password* lalu klik tombol *Stop Recording*



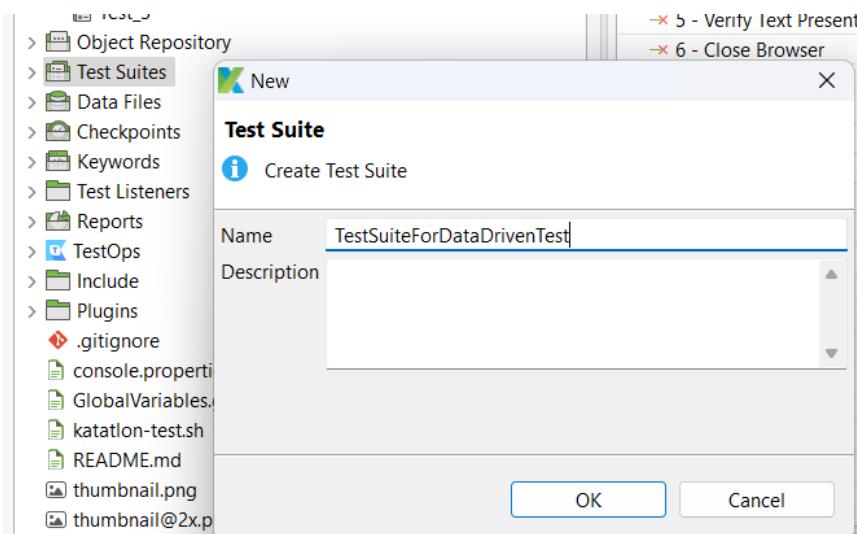
Gambar 10.5 Login

- 6) Setelah klik **OK** kita bisa melihat Test yang telah dibuat meliputi buka browser, membuka URL tertentu, set Text yang merupakan Username dan Password dan klik submit lalu kita menutup Browser semua tercatat. Semua itu tercatat sampai kita melakukan Stop Record

Item	Object	Input	Output	Description
-x 1 - Open Browser				
-x 2 - Navigate To Url		"http://opensource.demo.orangehrmlive.com/login"		
-x 3 - Set Text	input_username	"Admin"		
-x 4 - Set Text	input_password	"admin"		
-x 5 - Click	input_Submit			
-x 6 - Close Browser				

Gambar 10.6 Test Tercatat di Tabel

Di dalam test, kita mungkin punya Environment yang berbeda, seperti QA Environment, PreProd Environment, dan SandBox. Pada setiap Environment, memiliki nilai nilai yang berbeda. Sekarang kita akan menjalankan pengujian yang sama pada environment yang berbeda tersebut.



Gambar 10.7 Membuat Test Suite

- 7) Pada default Profiles, masukkan nilai Url, username dan password dengan menggunakan “Add” lalu simpan.

Name	Value	Description
Url	'http://opensource.demo.orangehrmlive.com/login'	
Username	'Admin'	
Password	'admin'	

Gambar 10.8 Mengisi Default Profiles

- 8) Pada Test Case nilai-nilai pada item di refer dari Profiles.

The screenshot shows the QTP Test Case editor interface. On the left, there's a tree view with 'Test Cases' expanded, showing 'Test1'. Under 'Object Repository', there's a 'Page\_OrangeHRM' entry. Below the tree are buttons for 'Test Suites', 'Data Files', 'Checkpoints', 'Keywords', 'Test Listeners', and 'Reports'. The main area has two tables. The top table, titled 'Item', lists steps: '1 - Open Browser', '2 - Navigate To Url', '3 - Set Text', '4 - Set Text', '5 - Click', and '6 - Close Browser'. The 'Object' column maps these to UI elements: 'input\_txtUsername', 'input\_txtPassword', and 'input\_Submit'. The 'Input' column contains values: an empty string for step 1, 'http://opensource.demo.orangehrmlive.com' for step 2, 'Admin' for step 3, 'admin' for step 4, and an empty string for step 6. The bottom table, titled 'No.', lists a single row with 'Param Name' as 'rawUrl', 'Param Type' as 'String', 'Value Type' as 'Global Variable', and 'Value' as 'Url'. This indicates that the 'Url' variable from the profiles is being used in the test case.

Gambar 10.9 Isi dari Nilai dari Profiles

- 9) Pada Script Mode, ubah nilai variable menjadi nilai yang ada pada default profile.

The screenshot shows the QTP Script Mode editor. It displays a script with several lines of VBA-like code. The first four lines are: 'I.openBrowser()', 'I.navigateToUrl(GlobalVariable.Url)', 'I.setText(findTestObject('Page\_OrangeHRM/input\_txtUsername'), GlobalVariable.Username)', and 'I.setText(findTestObject('Page\_OrangeHRM/input\_txtPassword'), GlobalVariable.Password)'. Below the script is a table with columns 'Item', 'Object', and 'Input'. The 'Item' column lists steps: '1 - Open Browser', '2 - Navigate To Url', '3 - Set Text', '4 - Set Text', '5 - Click', and '6 - Close Browser'. The 'Object' column maps these to UI elements: 'input\_txtUsername', 'input\_txtPassword', and 'input\_Submit'. The 'Input' column contains values: an empty string for step 1, 'GlobalVariable.Url' for step 2, 'GlobalVariable.Username' for step 3, and 'GlobalVariable.Password' for step 4. This indicates that the 'Url', 'Username', and 'Password' variables from the profiles are being used in the script.

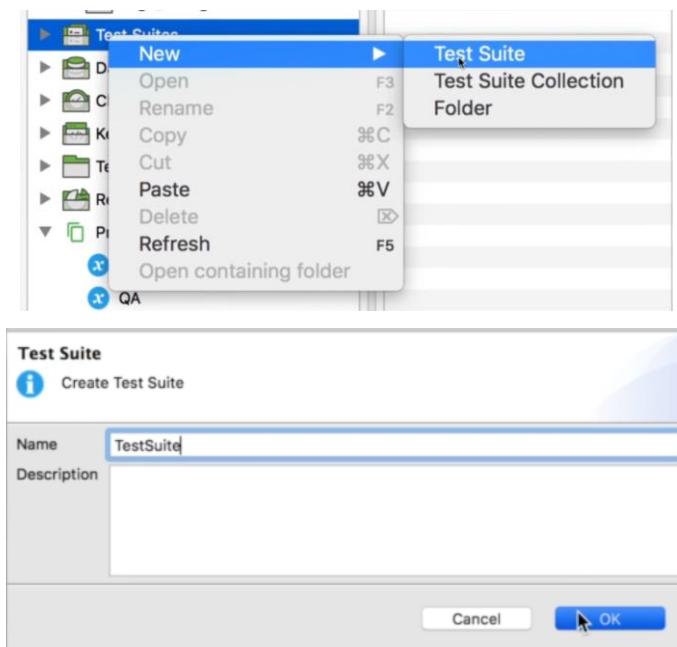
Gambar 10.10 Script Mode

- 10) Salin default Profiles kemudian rename sesuai dengan Environmentnya

The screenshot shows the QTP Profiles manager. A context menu is open over the 'default' profile in the list. The menu options include 'New', 'Open', 'Rename', 'Copy' (which is highlighted with a blue selection bar), 'Cut', 'Paste', 'Delete', 'Refresh', and 'Open containing folder'. The list below shows three profiles: 'default', 'QA', and 'Sandbox'. The 'default' profile is selected and highlighted with a blue bar. At the bottom of the screen, there's a toolbar with various icons and a status bar showing 'default' and 'Debug'.

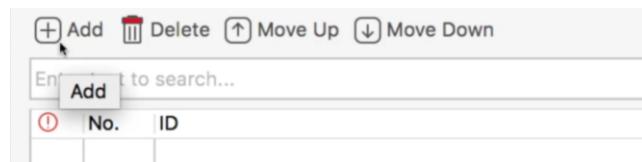
Gambar 10.11 Rename Sesuai Dengan Environment

11) Membuat *Test Suite* baru.



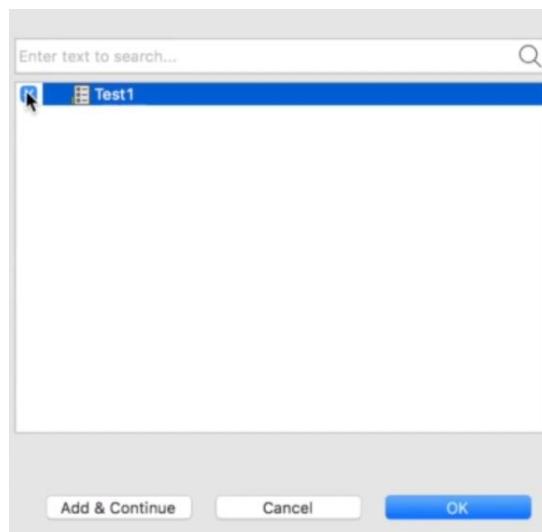
Gambar 10.12 Membuat Test Suite Baru

12) Menambahkan Test Case didalam Test Suite.



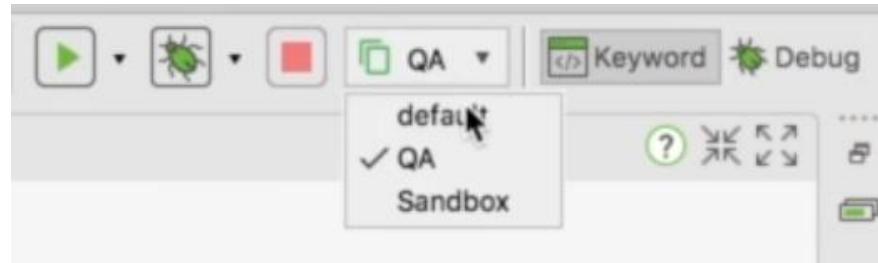
Gambar 10.13 Menambahkan Test Case Dalam Test Suite

13) Klik pada Test case yang akan ditambahkan lalu klik OK



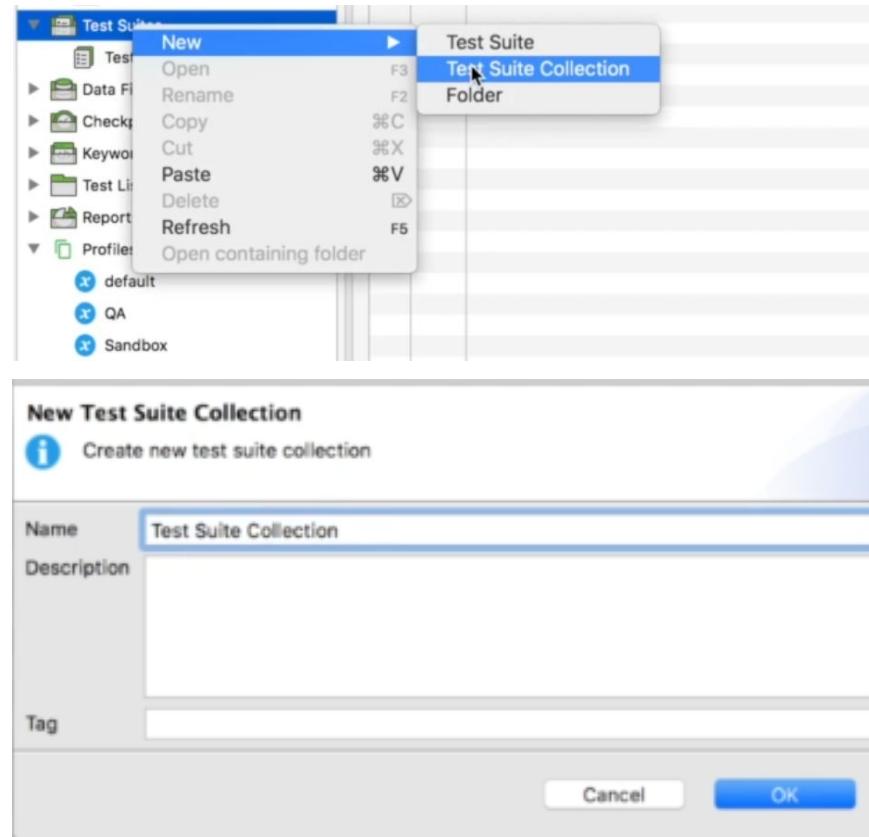
Gambar 10.14 Tambahkan Test Case

- 14) Pada saat Eksekusi *Test Case Profiles* atau Environment diambil dari tombol DropDown di atas.



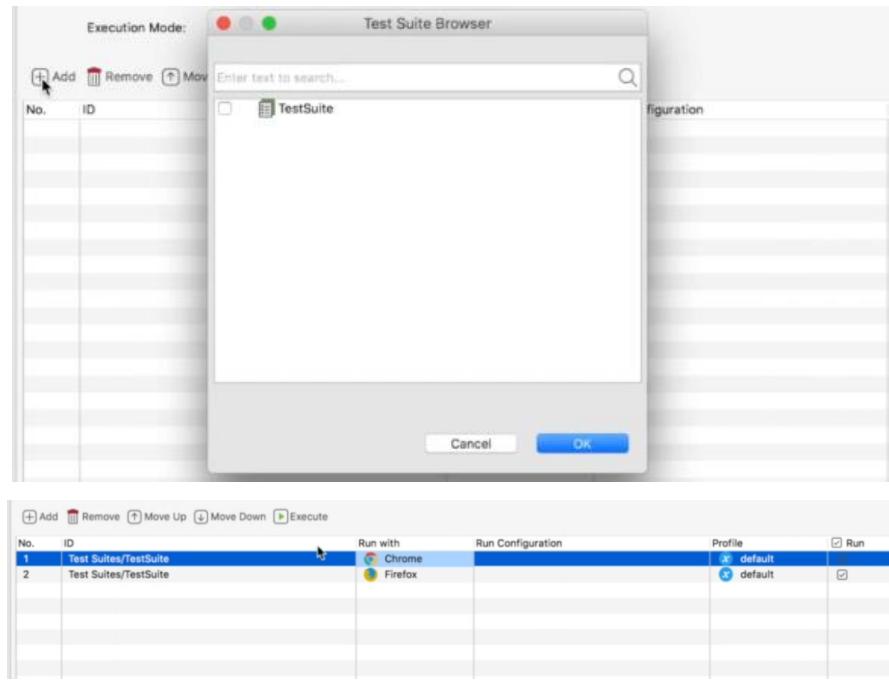
Gambar 10.15 Eksekusi Test Case Profiles

- Namun, pada saat mengeksekusi Test Suite Collection caranya berbeda.
- 15) Membuat Test Suite Collection baru.



Gambar 10.16 Membuat Test Suite Collection

- 16) Tambahkan Test Suite pada Test Suite Collection. Lakukan sebanyak 2 kali.



Gambar 10.17 Tambahkan Test Suite pada Test Suite Collection

Kita bisa memilih Profile atau environment yang akan dieksekusi

No.	ID	Run with	Run Configuration	Profile	Run
1	Test Suites/TestSuite	Chrome		default	<input checked="" type="checkbox"/>
2	Test Suites/TestSuite	Firefox		default	<input type="checkbox"/>

Gambar 10.18 Pilih Environment yang akan dieksekusi

## BAB 11

### CUSTOM KEYWORD

#### 11.1. Tujuan

- 1) Mengetahui apa itu custom keywords pada katalon
- 2) Mengetahui bagaimana cara membuat dan merujuk costum keywords pada katalon studio.

#### 11.2. Dasar Teori

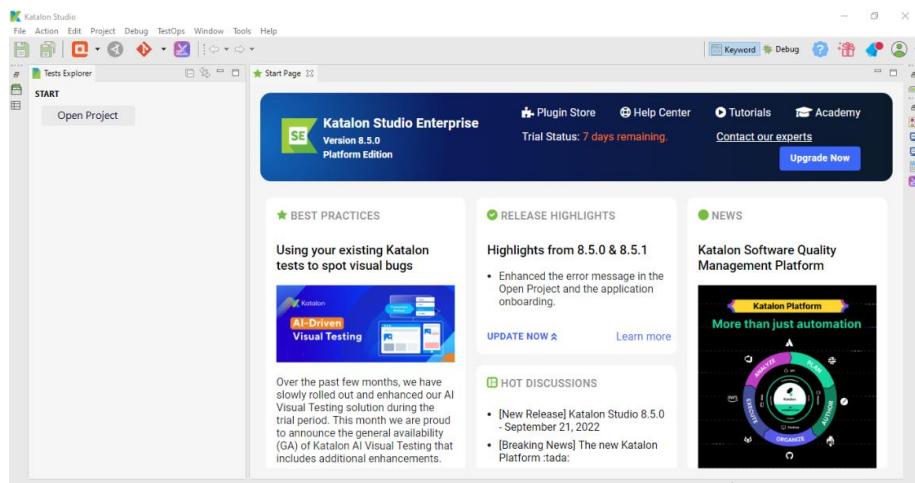
Custom keywords merupakan kata kunci khusus yang dapat anda gunakan untuk memperluas kemampuan Katalon Studio. Kata kunci ini berguna saat kita ingin menerapkan kasus pengujian, sama seperti kata kunci bawaan lainnya.

#### 11.3. Percobaan

Selain kata kunci bawaan, pada katalon kita juga dapat menentukan kata kunci khusus, berikut ini langkah-langkah dalam pembuatan costum keywords pada katalon studio.

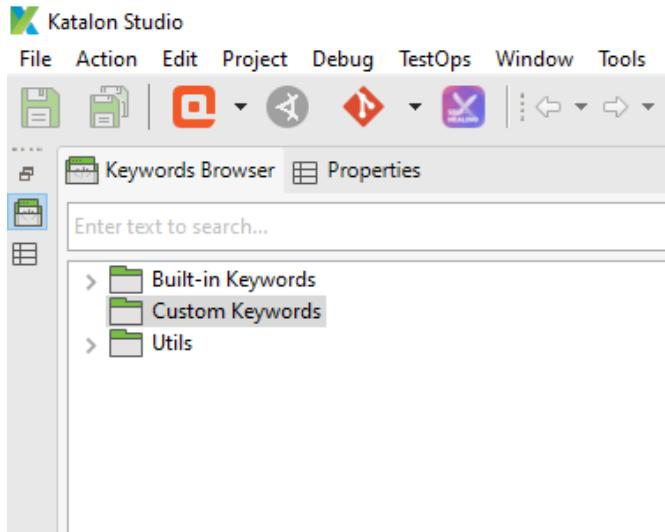
##### a) Bagaimana cara membuat kata kunci khusus

1. Untuk membuat kata kunci khusus pertama buka aplikasi katalon studio.



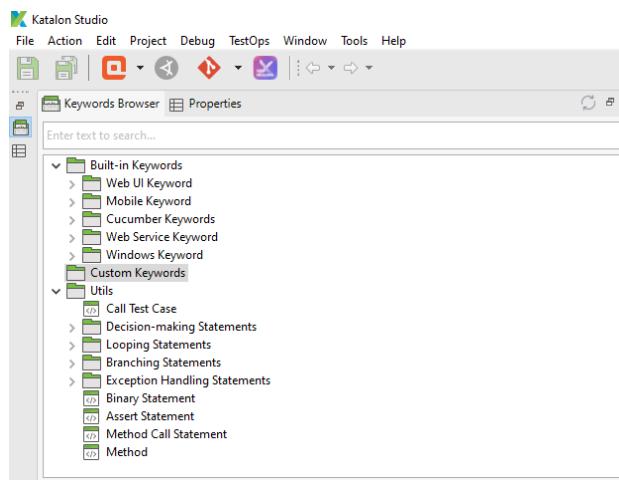
Gambar 11.1 Buka Aplikasi Katalon

2. Kemudian pilih costume keywords.



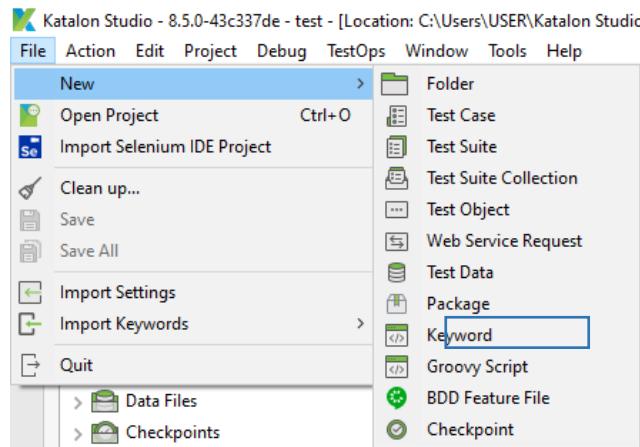
Gambar 11.2 Pilih Custome Keywords

3. Berikut adalah fitur-fitur yang terdapat pada custom keyword dan semua kata kunci yang terdapat pada keyword dibawah ini semua bias dijalankan.



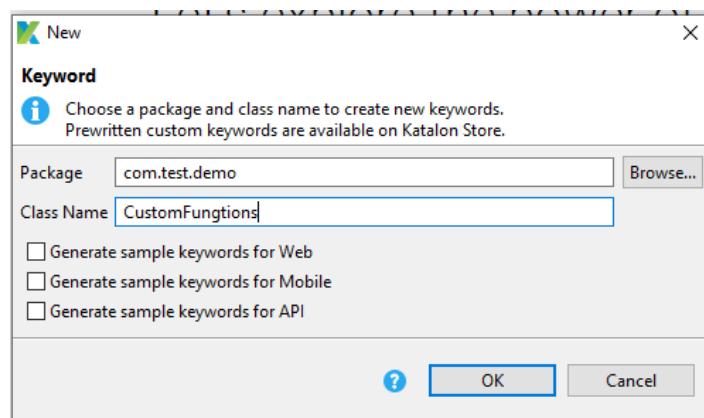
Gambar 11.3 Fitur-fitur Pada Custom Keyword

4. Langkah pertama membuat kata kunci khusus adalah klik file, lalu pilih keyword.



Gambar 11.4 Membuat Keyword

5. Kemudian inputkan package dengan nama *com.test.demo* lalu pada class name inputkan dengan nama *CustomFungsions*.



Gambar 11.5 Masukkan Nama Package dan Nama Class

6. Setelah mengklik OK maka akan muncul tampilan seperti dibawah ini lalu Kata kunci baru dibuat di bawah paket yang ditentukan.

Gambar 11.6 Tampilan dari Keyword

7. Masukkan kode berikut di kelas yang telah ditentukan untuk menentukan kata kunci khusus:

A screenshot of the Katalon Studio interface. The top bar shows the title 'Katalon Studio - 8.5.0-43c337de - test - [Location: C:\Users\USER\Katalon Studio\Healthcare Sample (2)]' and a menu bar with File, Actions, Edit, Project, Debug, TestOps, Window, Tool, Help. Below the menu is a toolbar with various icons for file operations like Open, Save, Find, Copy, Paste, etc. The left side features a sidebar with sections for Profiles, Test Cases, Object Repository, Test Suites, Data Files, Checkpoints, and Keywords, with 'Keywords' currently selected. The main area is a code editor displaying a Gherkin script named 'CustomFuctions.groovy'. The script starts with package declarations and imports, followed by a public class definition named 'CustomFuctions'. Inside the class, there is a keyword definition for 'printHello'.

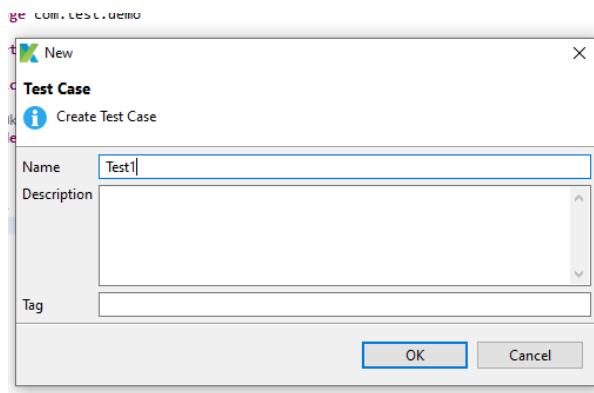
Gambar 11.7 Masukkan Kode

8. Kemudian buka kasus uji dalam tampilan Manual dan pilih Kata Kunci Khusus dari toolbar perintah', pilih ‘‘NEW KLIK TEST CASE’’.

A screenshot of the Katalon Studio interface. The top bar shows the title 'Katalon Studio - 8.5.0-43c337de - test - [Location: C:\Users\USER\Katalon Studio\Healthcare Sample (2)]'. Below the title is a menu bar with File, Action, Edit, Project, Debug, TestOps, Window, Tools, Help. A toolbar follows with icons for Save, Undo, Redo, Run, Stop, etc. The main area has tabs for 'Tests Explorer' and 'CustomFuntions.groovy'. The Tests Explorer shows a tree with 'Profiles', 'Test Cases', and a selected 'Common Test Case'. A context menu is open over the 'Test Case' item, with options 'New', 'Folder', 'Test Case', and 'Script'. The code editor shows Groovy script starting with 'package com.test.demo' and 'import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint'. A tooltip for 'Test Case' is visible at the bottom right.

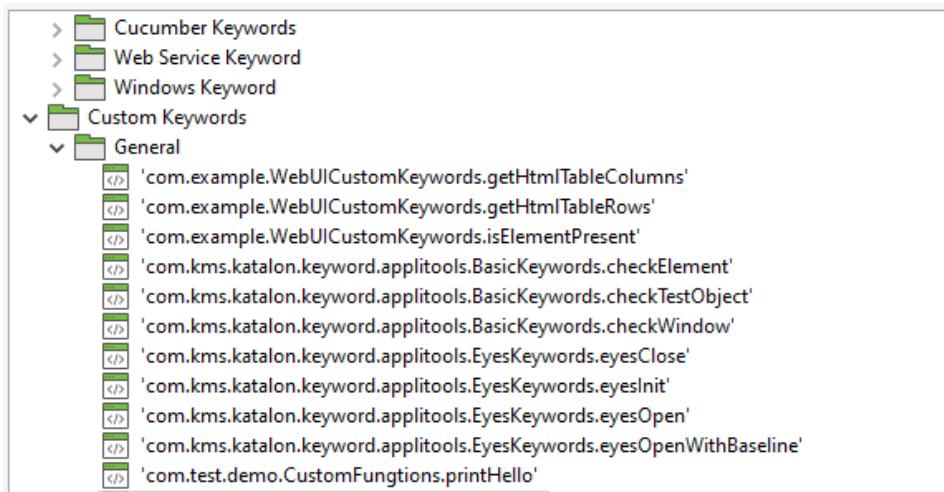
Gambar 11.8 Membuat Test Case Baru di dalam Common Tes Case

9. Inputkan name case sebagai Test1, lalu klik ok.



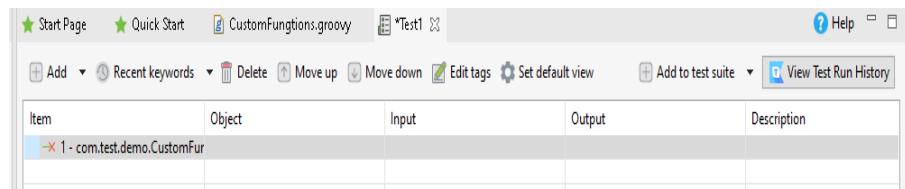
Gambar 11.9 Membuat Test Case Baru “Test1”

10. Selanjutnya Langkah pengujian baru ditambahkan. Pilih salah satu kata kunci khusus.



Gambar 11.10 Tampilan dari Test Case Baru

11. Setelah ditambahkan maka akan muncul tampilan seperti gambar di bawah.



Gambar 11.11 Tampilan dari Test Case Setelah ditambahkan

12. Selanjutnya klik running dan tunggu beberapa saat jika telah selesai klik Console maka muncul inputan Hello word

The screenshot shows the Katalon Studio interface with the 'Console' tab selected. The log output is as follows:

```

<terminated> TempTestCase1665839381023[Katalon] F:\SEMINAR 7\TOERI KPL PAK ARHAMIN\Katalon_Studio_PE_Windows_64-8.5.0\jre\bin\javaw.exe (Oct 15, 2022 6:09:42 AM - 6:09:53 AM)
2022-10-15 06:09:51.946 INFO c.k.katalon.core.main.TestCaseExecutor - -----
2022-10-15 06:09:51.946 INFO c.k.katalon.core.main.TestCaseExecutor - START Test Cases/Common Test Cases/Test1
2022-10-15 06:09:53.251 DEBUG testcase.Test1 - 1: com.test.demo.CustomFungtions.printHello()
    Hello Word .....
2022-10-15 06:09:53.284 INFO k.k.c.m.CustomKeywordDelegatingMetaClass - com.test.demo.CustomFungtions.printHello is PASSED
2022-10-15 06:09:53.315 INFO c.k.katalon.core.main.TestCaseExecutor - END Test Cases/Common Test Cases/Test1

```

Gambar 11.12 Tampilan dari Console

### b) Bagaimana cara merujuk kata kunci khusus

1. Pertama buat terlebih dahulu kata kunci lain, seperti gambar di bawah ini.

The screenshot shows the Katalon Studio code editor with a file named 'CustomFungtions.groovy'. The code contains the following Groovy script:

```

1 package com.test.demo
2
3 @import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
22
23 public class CustomFungtions {
24
25     @Keyword
26     def printHello() {
27         println ("\n      Hello Word .....      \n");
28     }
29
30     @Keyword
31     def printName(String name) {
32
33         println ("\n      Hello "+name+"      ...\\n")
34     }
35 }

```

Gambar 11.13 Membuat Kata Kunci CustomFungtion.groovy

2. Untuk menginputkan file dibawahnya langkah yang dilakukan sama seperti langkah sebelumnya.

Item	Object	Input	Output	Description
1 - com.test.demo.CustomFur				
2 - com.test.demo.CustomFur	**			

Gambar 11.14 Masukkan File

3. Dan saat menu skrip dibuka, maka kita dapat melihat tampilan seperti gambar ini.

```

13 import com.kms.katalon.core.testobject.TestObject as TestObject
14 import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
15 import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
16 import com.kms.katalon.core.windows.keyword.WindowsBuiltinKeywords as Windows
17 import internal.GlobalVariable as GlobalVariable
18 import org.openqa.selenium.Keys as Keys
19
20 CustomKeywords.'com.test.demo.CustomFungtions.printHello'()
21
22 CustomKeywords.'com.test.demo.CustomFungtions.printName'('Raihan')

```

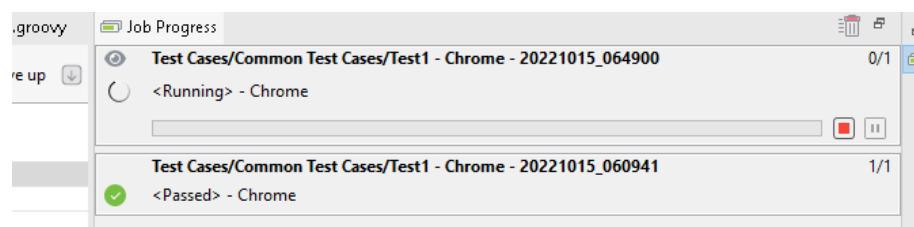
Gambar 11.15 Tampilan Script dari Test1

- Kemudian kembali lagi ke halaman Test1 klik tanda petik “” lalu inputkan kata kunci contohnya “Raihan” lalu klik ok.

No.	Param Name	Param Type	Value Type	Value
1	name	String	String	“Raihan”

Gambar 11.16 Memasukkan Kata Kunci

- Setelah itu klik running, dan tunggu hingga proses running selesai.



Gambar 11.17 Jalankan Test Case Test1

- Jika proses running telah selesai, klik Consule dan akan muncul kata kunci tambahan.

```

2022-10-15 06:49:18.854 INFO c.k.katalon.core.main.TestCaseExecutor - -----
2022-10-15 06:49:18.863 INFO c.k.katalon.core.main.TestCaseExecutor - START Test Cases/Common Test Cases/Test1
2022-10-15 06:49:20.180 DEBUG testcase.Test1 - 1: com.test.demo.CustomFungtions.printHello()

Hello Word .....

2022-10-15 06:49:20.230 INFO k.k.c.m.CustomKeywordDelegatingMetaClass - com.test.demo.CustomFungtions.printHello is PASSED
2022-10-15 06:49:20.231 DEBUG testcase.Test1 - 2: com.test.demo.CustomFungtions.printName("Raihan")

Hello "Raihan" ...
Hello "Raihan" ...

2022-10-15 06:49:20.234 INFO k.k.c.m.CustomKeywordDelegatingMetaClass - com.test.demo.CustomFungtions.printName is PASSED
2022-10-15 06:49:20.271 INFO c.k.katalon.core.main.TestCaseExecutor - END Test Cases/Common Test Cases/Test1

```

Gambar 11.18 Tampilan Console Setelah dijalankan

## **BAB 12**

### **WINDOWS DEKSTOP APPLICATION TESTING**

#### **12.1. Tujuan**

- Mempelajari tentang pengujian terhadap dekstop.
- Mengetahui bagaimana cara melakukan pengujian terhadap dekstop dengan menggunakan katalon studio.

#### **12.2. Dasar Teori**

Aplikasi desktop adalah program yang berjalan secara independen di sistem operasi desktop. Tidak seperti aplikasi web, aplikasi desktop memerlukan sumber daya perangkat keras yang cukup untuk berfungsi.

Pengujian aplikasi desktop adalah praktik pengujian perangkat lunak yang memeriksa fungsionalitas, keamanan, kegunaan, dan stabilitas aplikasi setelah diterapkan. Dalam pengujian aplikasi desktop, kita perlu memperhatikan pemasangan serta pengujian penghapusan instalasi untuk sepenuhnya memenuhi persyaratan pengujian aplikasi. Apa yang bisa diuji pada dekstop dengan menggunakan katalon studio? Katalon studio dapat melakukan beberapa pengujian diantaranya :

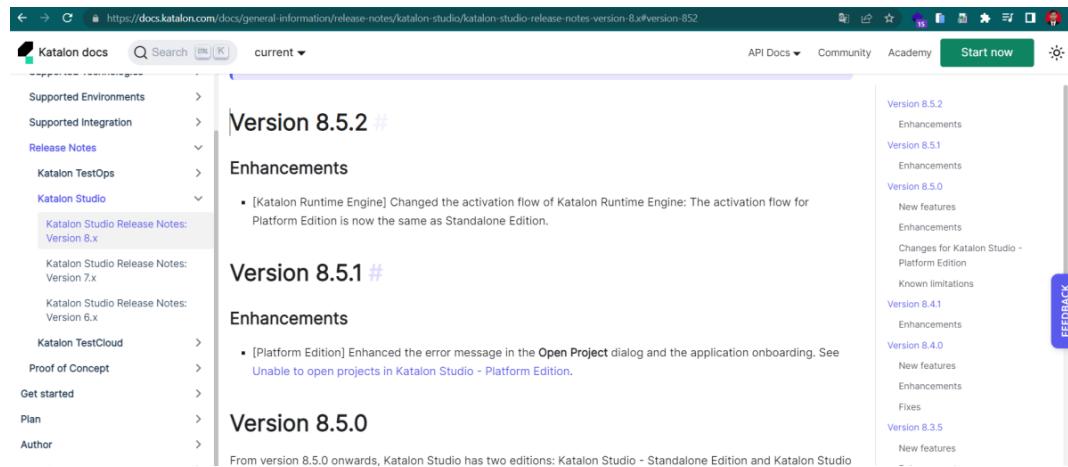
- Melakukan pengaturan dan konfigurasi yang mudah – baik secara lokal maupun jarak jauh
- Mendeteksi dan memata-matai objek Windows
- Merekam tindakan Windows
- Menguji lokasi elemen cerdas
- Menguji kata kunci bawaan dan khusus Windows Perawatan minimal

#### **12.3. Percobaan**

Berikut ini merupakan langkah-langkah untuk melakukan pengujian dekstop dengan menggunakan Katalon Studio.

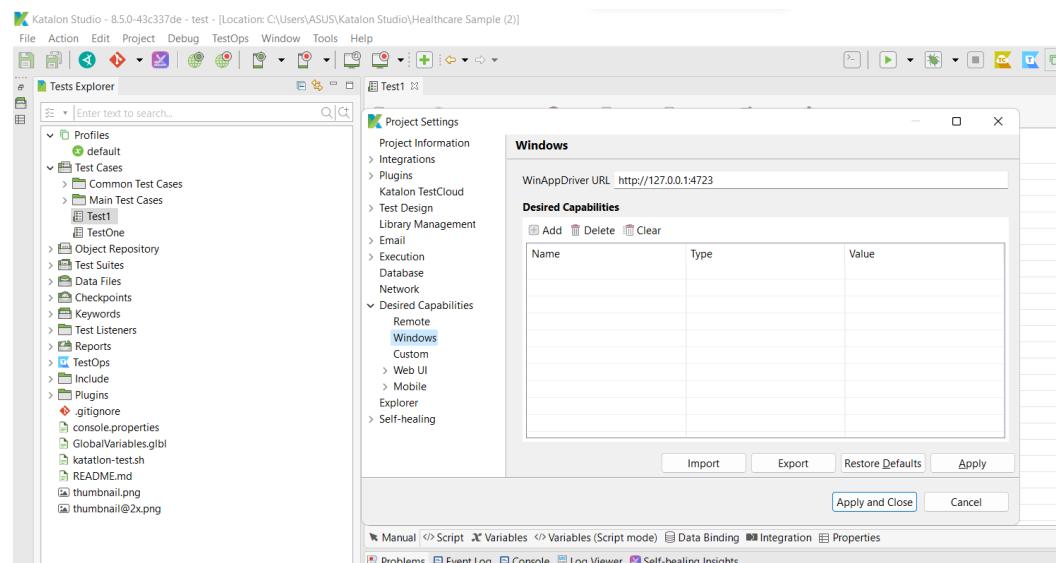
1. Sebelum melakukan pengujian terhadap dekstop pada katalon studio, pastikan katalon studio yang digunakan sudah dalam versi terbaru. Jika anda masih menggunakan katalon studio versi lama, anda dapat mendownload

versi terbaru katalon studio pada web resmi katalon studio.com. Hal ini dilakukan untuk memastikan bahwa menu atau tools pengujian terhadap dekstop telah tersedia.



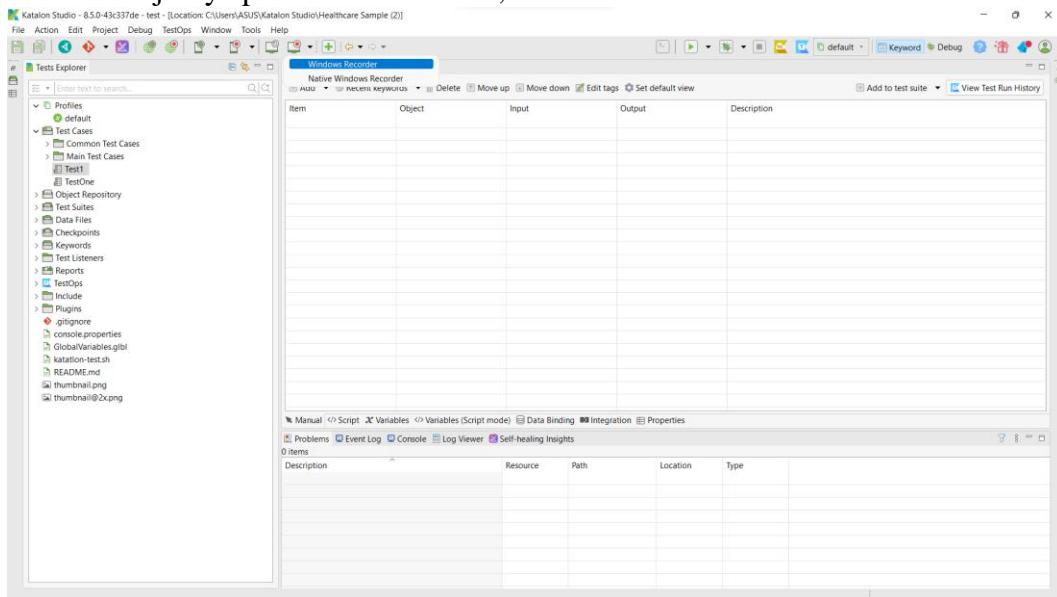
Gambar 12.1 Tampilan Halaman Utama Katalon

2. Setelah katalon studio berhasil di download, buka katalon studio kemudian untuk memastikan apakah katalon studio yang dipakai bisa melakukan pengujian terhadap dekstop, pada menu utama buka projek, lalu pilih Desired Capabilities, pada menu tersebut bisa dilihat bahwa pilihan untuk windows telah tersedia.



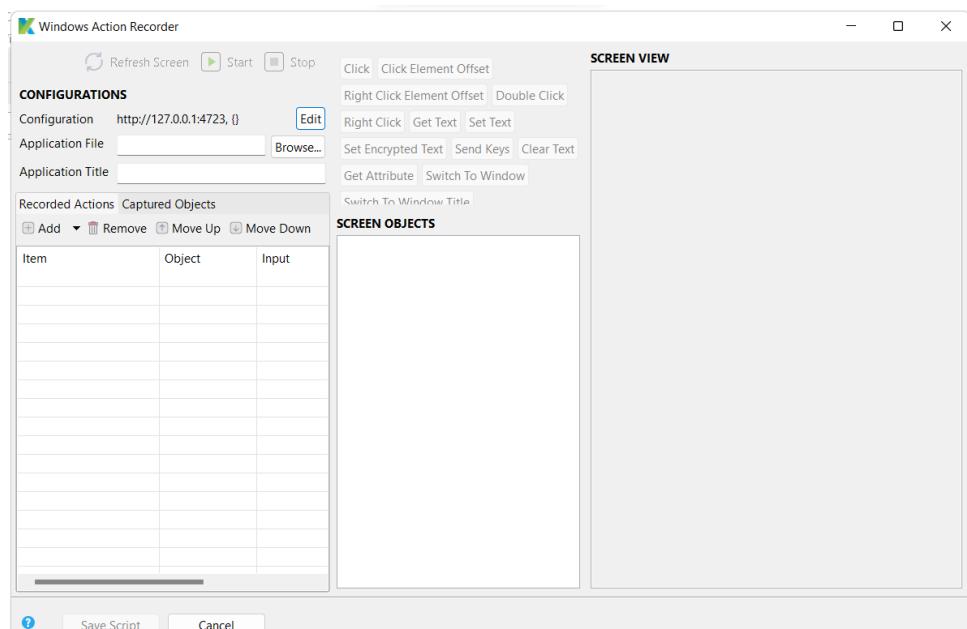
Gambar 12.2 Tampilan Hasil Katalon Setelah Didownload

### 3. Selanjutnya pada menu test case, buat satu test case baru.



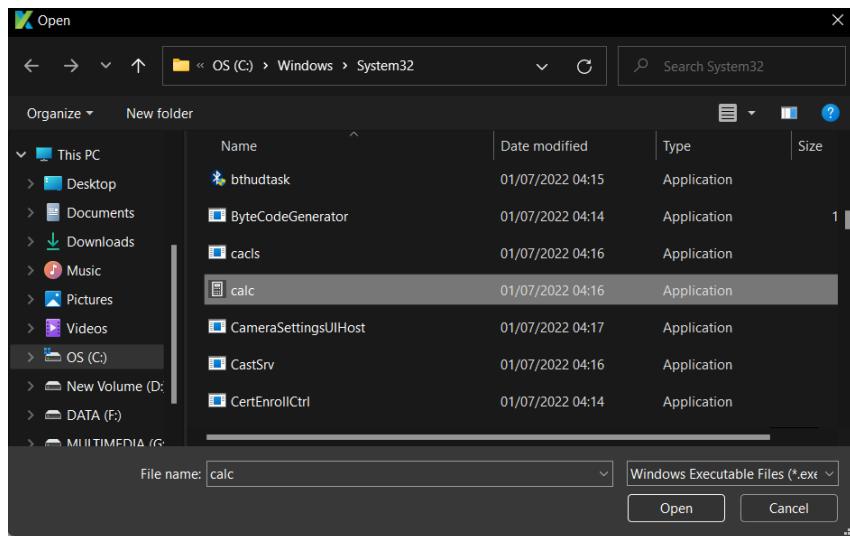
Gambar 12.3 Tampilan Menu test Case

4. Klik tombol windows recorder. Setelah tombol di klik, maka user akan disuguhkan dengan tampilan jendela yang berisi rekaman tindakan windows.



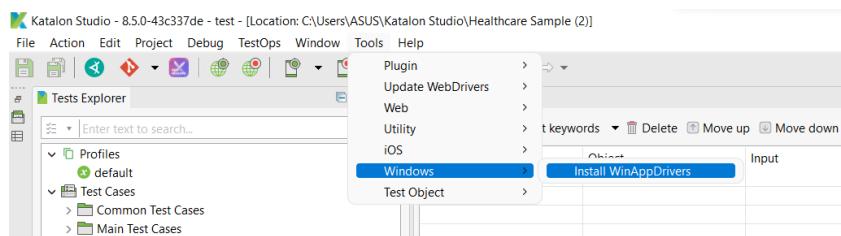
Gambar 12.4 Tampilan Jendela Rekaman

5. Pada application file, pilih file yang ingin diuji, dan biasanya file yang diuji berformat .exe. Pada kasus ini saya menggunakan file calc.exe untuk pengujinya. File calc.exe bisa didapat pada folder AdvancedInstallers yang berada di dalam folder System32 pada localdisc C. Setelah file dipilih, lalu klik ok.



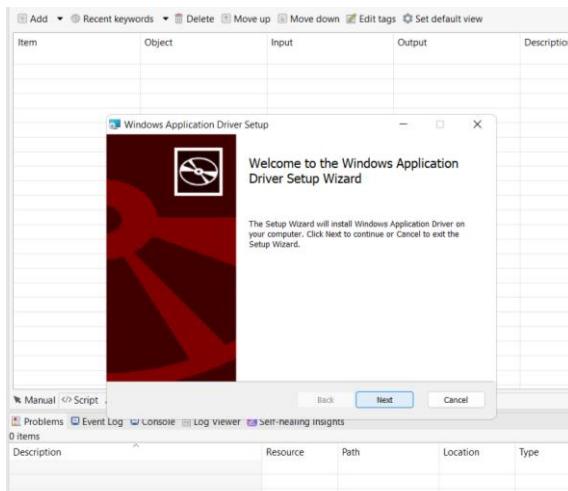
Gambar 12.5 Tampilan application file Yang Akan Diuji

6. Sebelum melakukan proses recording, ada beberapa hal yang harus dilakukan. Terlebih dahulu kita harus menginstall winAppDriver, winAppDriver ini digunakan sebagai library agar bisa berinteraksi dengan aplikasi windows. Proses penginstallan dapat dilakukan dengan tools pada menu utama, pilih windows, lalu klik Install WinAppDrivers.



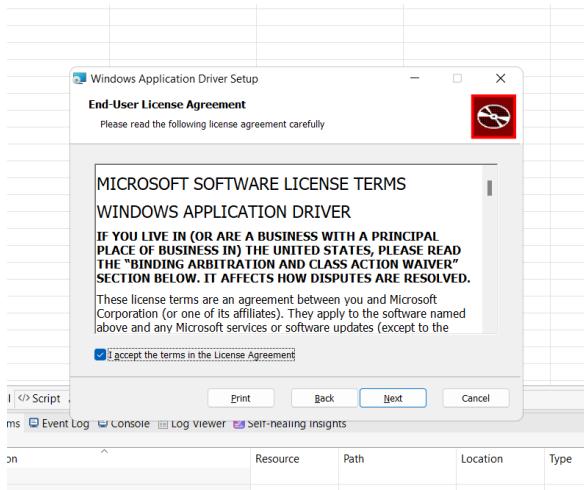
Gambar 12.6 Tampilan Proses Recording

7. Pada instalasi wizard klik next saja.



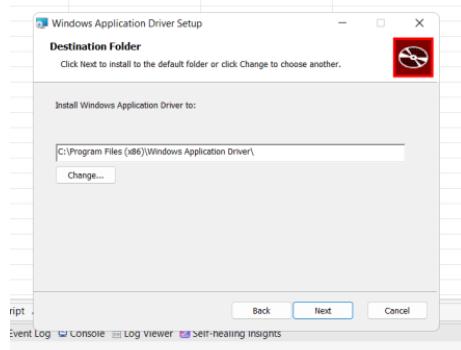
Gambar 12.7 Tampilan instalasi wizard

8. Pilih I accept untuk license agreement, kemudian next.



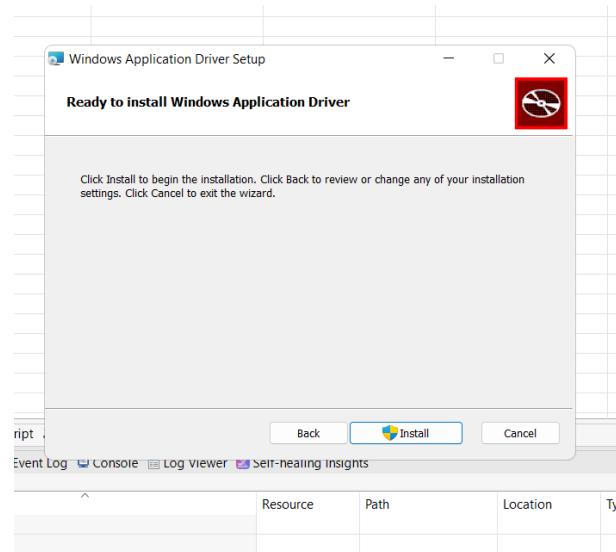
Gambar 12.8 Tampilan License agreement

9. Pilih lokasi penyimpanan file install, lalu klik next.



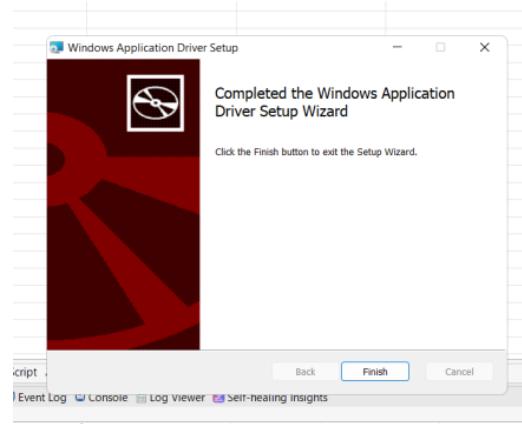
Gambar 12.9 Tampilan Lokasi Penyimpanan File Install

10. Kemudian klik install.



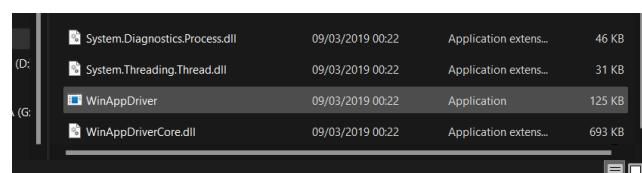
Gambar 12.10 Tampilan Hasil Install

11. Tunggu sampai proses installasi selesai, lalu klik finish.



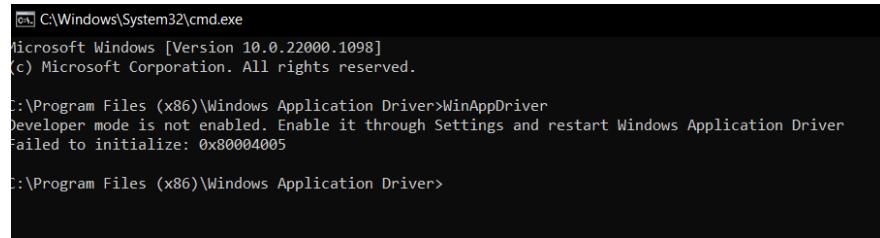
Gambar 12.11 Tampilan Hasil Install Berhasil

12. Berikut merupakan tampilan WinAppDriver yang sudah di install, dan tersimpan pada directory komputer. Langkah selanjutnya adalah melakukan start terhadap WinAppDriver.



Gambar 12.12 Tampilan WinappDriver

13. Start WinAppDriver bisa dilakukan dengan menggunakan CMD. Copy file path WinAppDriver, lalu paste ke CMD, kemudian klik enter.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

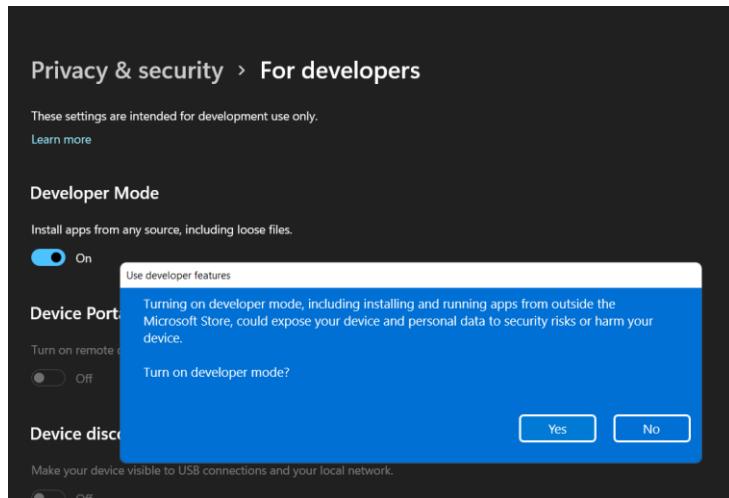
C:\Program Files (x86)\Windows Application Driver>WinAppDriver
Developer mode is not enabled. Enable it through Settings and restart Windows Application Driver
Failed to initialize: 0x80004005

C:\Program Files (x86)\Windows Application Driver>
```

Gambar 12.13 Tampilan Start WinappDriver

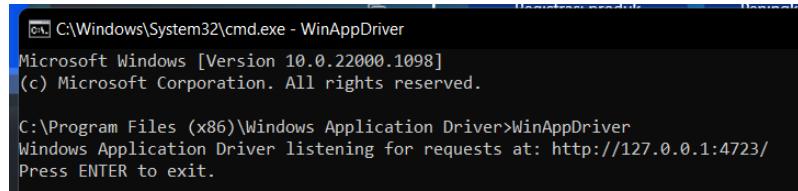
Hasil di atas menunjukkan bahwa WinAppDriver belum berhasil di start, anda dapat mengatasinya dengan mengaktifkan terlebih dahulu developer mode yang ada pada windows anda.

14. Anda dapat mengaktifkan developer mode pada developer setting, seperti yang terlihat pada gambar di bawah ini.



Gambar 12.14 Tampilan Mengaktifkan Developer Mode

15. Setelah developer mode berhasil di aktifkan, kembali ke halaman cmd kemudian lakukan kembali proses start pada WinAppDriver. Pada kasus ini WinAppDriver sudah berhasil di start, dengan hasil yang ditampilkan seperti gambar di bawah ini.

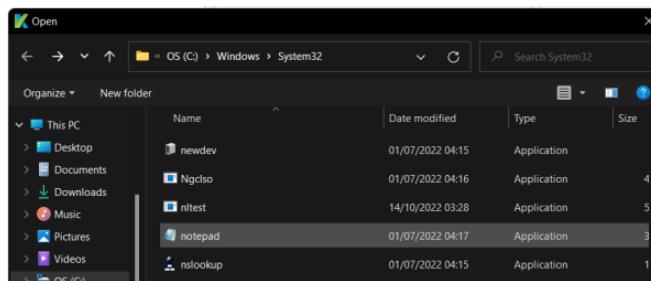


```
C:\Windows\System32\cmd.exe - WinAppDriver
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files (x86)\Windows Application Driver>WinAppDriver
Windows Application Driver listening for requests at: http://127.0.0.1:4723/
Press ENTER to exit.
```

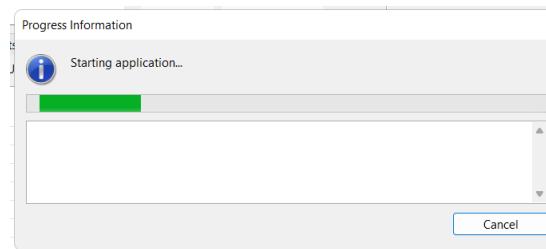
Gambar 12.15 Tampilan Developer Mode Berhasil Di Aktifkan

16. Selanjutnya pada katalon studio pilih application dengan notepad.exe, lalu klik open.



Gambar 12.16 Tampilan notepad.exe

17. Setelah itu klik tombol start, dan tunggu sampai proses start selesai.



Gambar 12.17 Tampilan tombol Start

18. Kembali ke halaman cmd, kemudian klik enter. Setelah tombol enter di klik, maka user akan disuguhkan dengan tampilan jendela notepad yang ada pada sistem kita.

```

C:\Windows\System32\cmd.exe - WinAppDriver
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files (x86)\Windows Application Driver>WinAppDriver
Windows Application Driver listening for requests at: http://127.0.0.1:4723/
Press ENTER to exit.

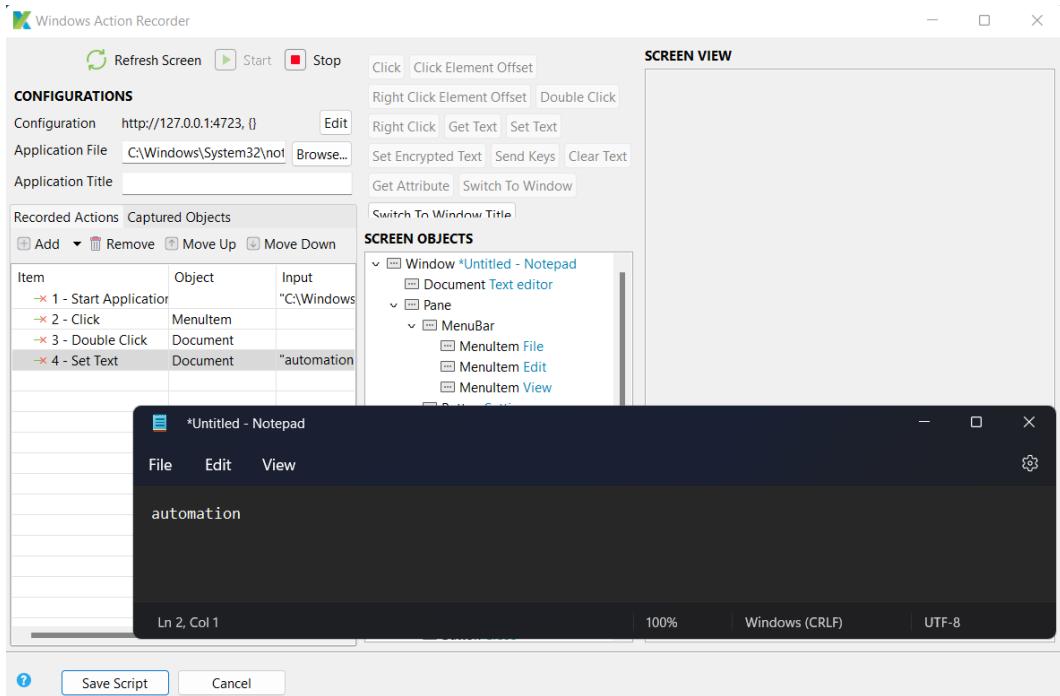
=====
POST /session HTTP/1.1
Accept-Encoding: gzip
Connection: Keep-Alive
Content-Length: 274
Content-Type: application/json; charset=utf-8
Host: 127.0.0.1:4723
User-Agent: selenium/3.141.59 (java windows)

{
  "desiredCapabilities": {
    "app": "C:\\Windows\\System32\\notepad.exe",
    "platformName": "windows"
  },
  "capabilities": {
    "firstMatch": [
      {
        "appium:app": "C:\\Windows\\System32\\notepad.exe",
        "platformName": "windows"
      }
    ]
  }
}

```

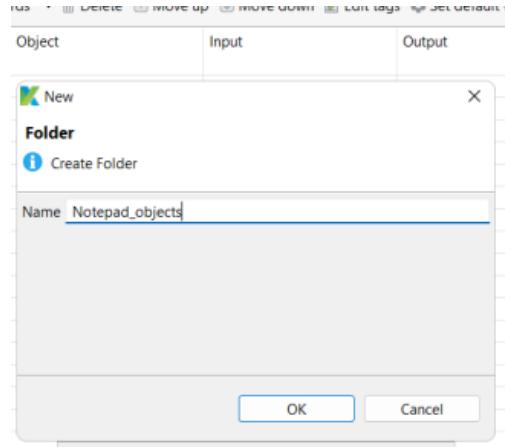
Gambar 12.18 Tampilan Halaman CMD Pada Notapad

19. Selanjutnya, anda sudah dapat melakukan pengujian terhadap dekstop. Pada menu possible action, user bebas untuk melakukan proses yang ada didalamnya. Untuk kasus ini user melakukan proses click, double click, dan proses untuk menambahkan text dengan kata “automation”. Jika proses pengujian selesai dilakukan, anda dapat klik tombol stop.



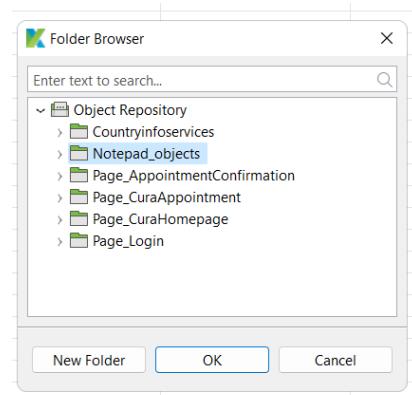
Gambar 12.19 Tampilan Pengujian Terhadap Desktop

20. Kemudian, pada objek repository buat satu folder baru dengan nama Notepad\_objects, lalu klik ok.



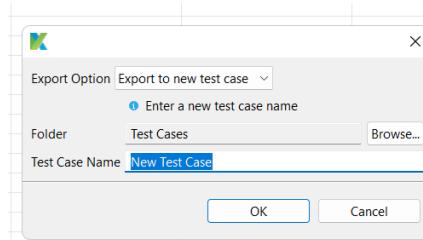
Gambar 12.20 Tampilan Objek Repository Baru

21. Folder Notepad\_objects berhasil dibuat.



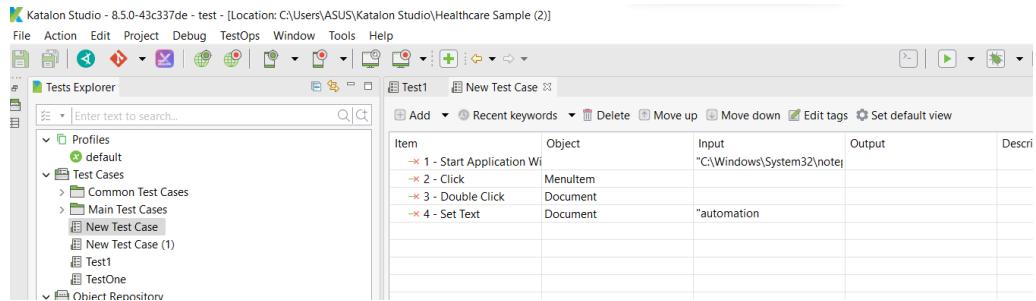
Gambar 12.21 Tampilan Folder Notepad Objects berhasil

22. Pada folder Notepad\_object dapat dilihat objek dekstop yang telah dibuat, kemudian, buat satu test case baru, lalu klik ok.



Gambar 12.22 Membuat Satu Test Case Baru

23. Pada test case tersebut dapat dilihat kasus uji yang telah direkam dan telah dilakukan sebelumnya. Seperti yang terlihat pada gambar di bawah ini.



Gambar 12.23 Tampilan Test Case Pada Kasus Uji

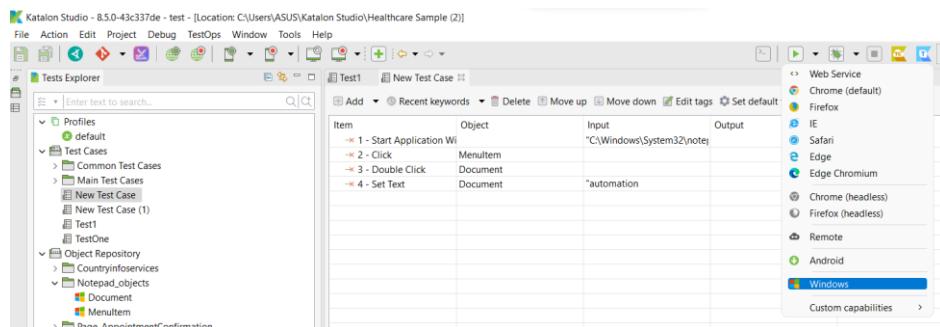
24. Kemudian jika kita lihat pada jendela script juga tertera proses-proses apa saja yang telah dilakukan saat melakukan pengujian.

```

import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
import static com.kms.katalon.core.testdata.TestDataFactory.findTestData
import static com.kms.katalon.core.testobject.ObjectRepository.findTestObject
import static com.kms.katalon.core.keyword.CucumberBuiltinKeywords as CucumberKW
import static com.kms.katalon.core.mobile.keyword.MobileBuiltInKeywords as Mobile
import static com.kms.katalon.core.model.FailureHandling as FailureHandling
import com.kms.katalon.core.testcase.TestCase as TestCase
import com.kms.katalon.core.testdata.TestData as TestData
import com.kms.katalon.core.testing.keyword.TestNGKeyword as TestNGKW
import com.kms.katalon.core.testobject.TestObject as TestObject
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
import com.kms.katalon.core.keyword.MobileBuiltInKeywords as WebUI
import com.kms.katalon.core.keyword.WindowsBuiltInKeywords as Windows
import internal.GlobalVariable as GlobalVariable
import org.openqa.selenium.Keys as Keys
Windows.startApplicationWithTitle('C:\Windows\System32\notepad.exe', '')
Windows.click(findWindowsObject('Object Repository/Notepad_objects/MenuItem'))
Windows.doubleClick(findWindowsObject('Object Repository/Notepad_objects/Document'))
Windows.setText(findWindowsObject('Object Repository/Notepad_objects/Document'), 'automation')
    
```

Gambar 12.24 Tampilan Jendela Script

26. Selanjutnya running test case untuk melihat proses eksekusinya.



Gambar 12.25 Tampilan Running Test Case

27. Berikut merupakan hasil running yang telah berhasil dilakukan, dan tidak terjadi error.

## **BAB 13**

### **CARA MENGGUNAKAN PLUGIN XPATH DAN PLUGIN VISUAL VALIDATION**

#### **13.1. Tujuan**

Bagian ini membahas apa itu plugin xpath serta bagaimana cara menggunakan plugin xpath pada aplikasi katalon. Pada akhir pembahasan, diharapkan praktikan dapat:

1. Mengetahui bagaimana cara menggunakan Plugin Xpath
2. Mengetahui fungsi dari plugin Xpath
3. Mempraktekkan cara menggunakan plugin Xpath
4. Praktikan memahami cara-cara untuk membuat akun di applitools
5. Praktikan bisa menjalankan test dan check applitools di dashboard secara teratur dan benar

#### **13.2. Dasar Teori**

##### **A. Pengertian Plugin Xpath**

Plugin adalah kode software dengan fungsi tertentu yang memungkinkan aplikasi atau program untuk menjalankan fitur tambahan di aplikasi atau program tersebut. Kode software tersebut jamak ditemukan di program atau aplikasi seperti Mozilla, Google Chrome, Opera Browser, dan WordPress.

Umumnya setiap plugin memang didesain untuk satu aplikasi atau program tertentu. Misalnya, add ons Mozilla tidak dapat diinstall untuk Google Chrome. Begitu juga sebaliknya, extension Google Chrome tidak dapat dipasang di Mozilla.

XPath (XML Path Language) merupakan bahasa query kepada memilih anggota - anggota (nodes) dari sebuah dokumen XML. XPath juga bisa dipergunakan untuk menghitung nilai (contoh : string, angka atau boolean) dari inti dari sebuah dokumen XML. XPath distandarisasi oleh World Wide Web Consortium.

## B. AppliTools

AppliTools menghadirkan otomatisasi pengujian generasi berikutnya yang didukung oleh teknologi visi komputer berbantuan AI yang dikenal sebagai Visual AI. Visual AI membantu Pengembang, Insinyur Otomasi Pengujian, dan profesional QA merilis aplikasi web dan seluler berkualitas tinggi lebih cepat dan lebih efisien.

AppliTools terintegrasi dengan lebih dari 60 kerangka kerja otomatisasi pengujian modern seperti Selenium, WebdriverIO, dan Cypress untuk meningkatkan efisiensi dan mengurangi pemeliharaan pengujian otomatis.

Apa yang dilakukan AppliTools?

AppliTools sedang membangun layanan cloud untuk pengujian visual otomatis. AppliTools Eyes secara otomatis memvalidasi kebenaran tata letak, konten, fungsionalitas, dan tampilan UI di semua browser, perangkat, dan resolusi layar, serta memungkinkan untuk mengotomatiskan pengujian yang hanya dapat dilakukan secara manual tanpa itu.

1. Uji semua aplikasi Anda secara visual : Temukan bug visual di seluruh browser, ukuran layar, perangkat seluler, dan sistem operasi
2. Visual AI adalah Intelligent Computer Vision : Gunakan AI untuk menghilangkan kesalahan positif yang membuat tim Anda frustrasi. Dapat memvalidasi seluruh halaman aplikasi, mendeteksi masalah tata letak, dan memproses halaman paling kompleks dan dinamis tanpa kalibrasi atau pelatihan.
3. Pemeliharaan Otomatis Bertenaga AI : Selesaikan perbedaan serupa secara instan dengan memanfaatkan algoritme canggih yang secara otomatis menganalisis perbedaan di semua pengujian Anda untuk menghasilkan laporan ringkas yang hanya menunjukkan perbedaan yang berbeda. Setujui atau tolak perubahan yang akan diterapkan secara otomatis di semua perubahan serupa di seluruh rangkaian pengujian Anda. Tunjukkan elemen yang diizinkan untuk dipindahkan atau

diabaikan dan secara otomatis mendeteksinya di semua layar dalam semua pengujian Anda.

4. Pengujian Lintas Browser Sangat Cepat : pengujian fungsional & visual otomatis berjalan 30 hingga 70 kali lebih cepat, pada berbagai browser dan perangkat yang diemulasi termasuk Chrome, Firefox, Safari, Edge, dan IE, iPhone, iPad, perangkat Galaxy, dan Kindle.
5. Manajemen & Analisis Uji Visual : semua laporan ringkas dan mudah dibaca. Setiap laporan menyertakan tangkapan layar dengan perbedaan visual yang ditandai dengan jelas. Semua layar dapat diperbesar dan secara otomatis dikelompokkan berdasarkan perbedaan serupa untuk menghemat waktu yang berharga. Secara opsional, Anda memiliki kemampuan untuk membuat laporan khusus menggunakan API AppliTools.
6. Berkolaborasi Tentang Kualitas : Buka masalah, bertukar pikiran, dan berikan umpan balik tentang UI aplikasi Anda langsung dari laporan pengujian. Masalah dan diskusi tetap terlihat dalam pengujian berjangka selama masih terbuka di pelacak masalah Anda. Pilih untuk gagal dalam pengujian fungsional berdasarkan status masalah atau tunda kegagalan hingga perbaikan diharapkan.
7. Percabangan & Penggabungan Baseline
8. Integrasi Berkelanjutan & Pelacakan Cacat : Buka masalah Jira langsung dari manajer pengujian AppliTools.
9. Manajemen Pengguna : Kelola pengguna dan tim dengan mudah menggunakan AppliTools.
10. Tersedia di Cloud Publik atau Khusus
11. Hubungkan Bisnis Dengan Teknologi: Pelaporan visual AppliTools memungkinkan Anda berkolaborasi antara tim bisnis dan teknologi di

perusahaan Anda dengan menggunakan bahasa yang sama seputar aspek fungsional dan visual aplikasi atau situs web Anda.

### C. Fungsi Plugin

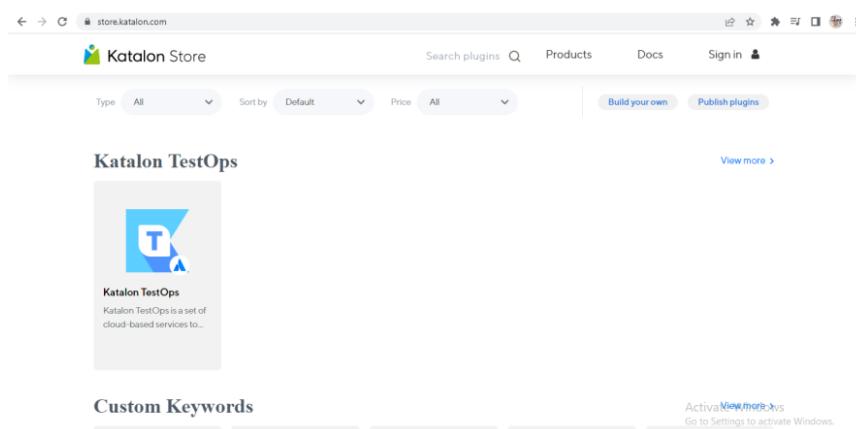
Plugin adalah alat atau tool yang bisa merubah dan mentransformasi web simpel menjadi web yang fungsional dan memiliki banyak feature.

#### 13.3. Langkah-langkah dalam Menggunakan Plugin

##### A. Plugin Xpath

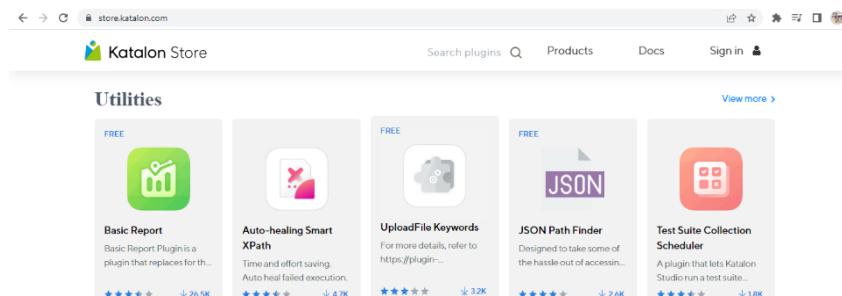
###### a) Add File

1. Buka website katalon studio, lalu Add plugin



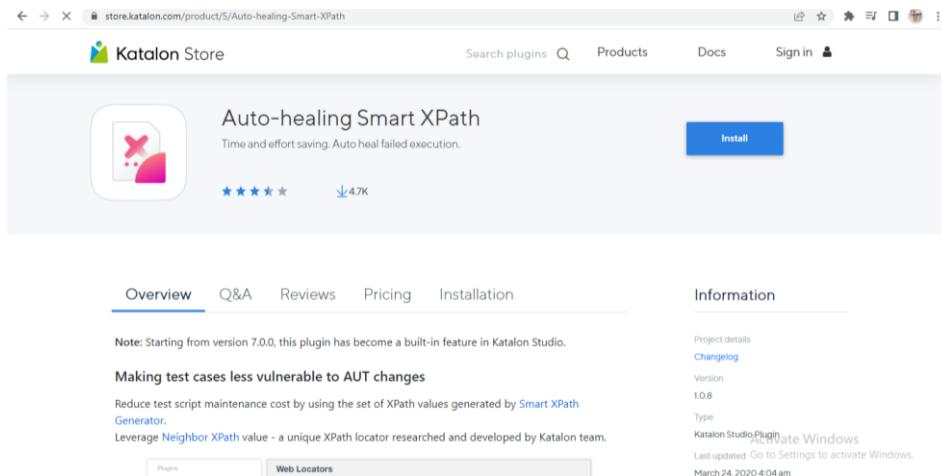
Gambar 13.1 Website Katalon Store

2. Disana terdapat beberapa kategori plugins, lalu pilih yang free dan kemudian klik Auto healing smart Xpath



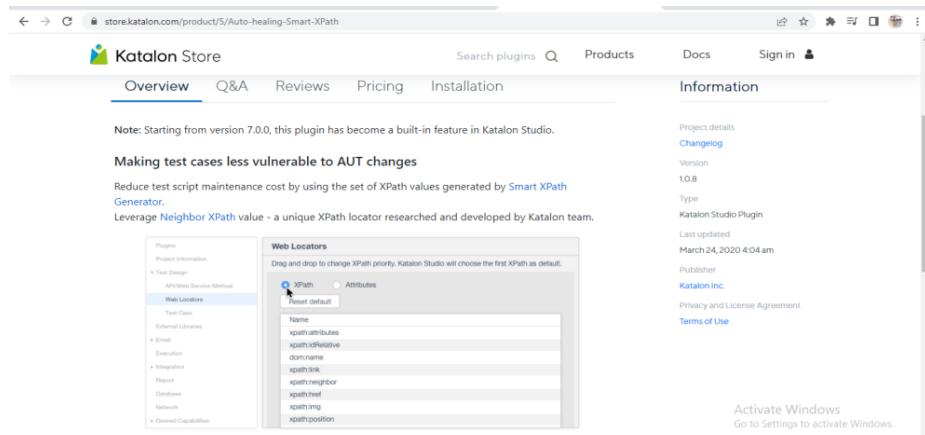
Gambar 13.2 Utilities Katalon Store Websites

- Setelah kita klik nantinya akan terlihat beberapa dokumentasi untuk plugin ini, yang pada dasarnya plugin ini akan membantu kita jika web mengalami perubahan.



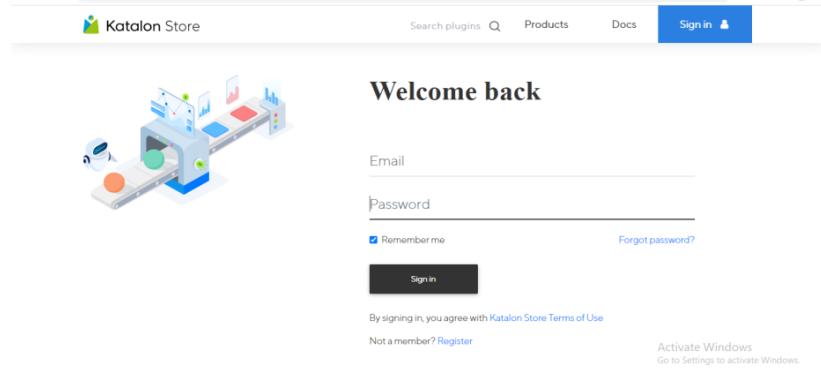
Gambar 13.3 Menginstall Auto-healing Smart Xpath

- Setelah kita klik nantinya akan terlihat beberapa dokumentasi untuk plugin ini, yang pada dasarnya plugin ini akan membantu kita jika web mengalami perubahan.



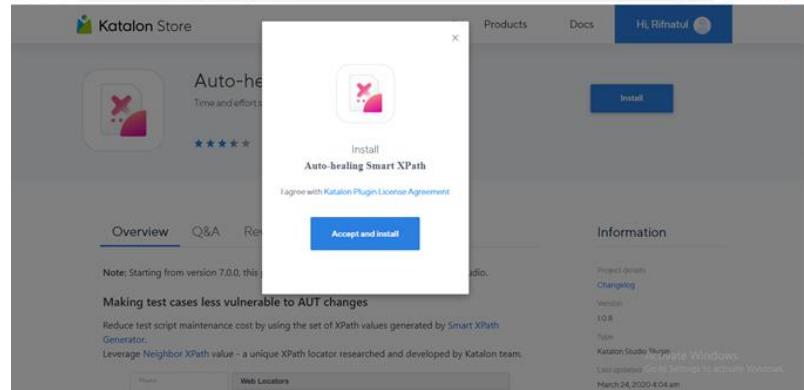
Gambar 13.4 Dokumentasi Plugin Xpath

- Untuk menginstall plugin pastikan bahwa kita sudah login pada website katalon store, untuk login masukkan email dan password akun yang telah terdaftarkan



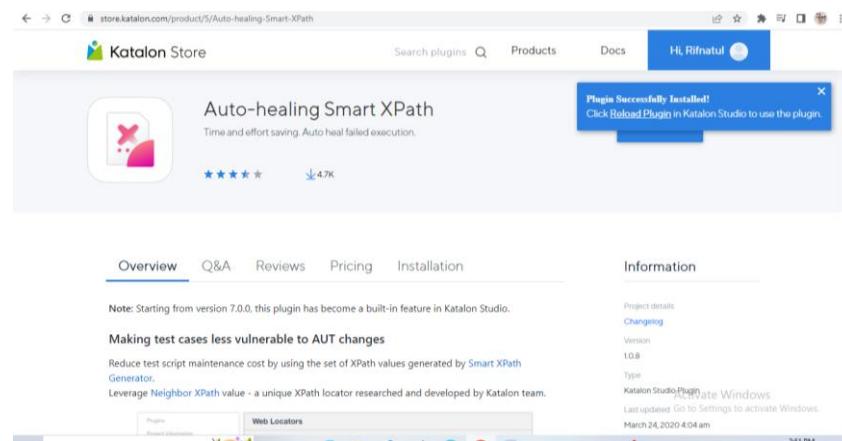
Gambar 13.5 Halaman Login Katalon Store

6. Klik install pada, auto healing smart xpath, lalu akan muncul pop up. Klik tombol accept and install.



Gambar 13.6 Pop up Accept and Install Xpath

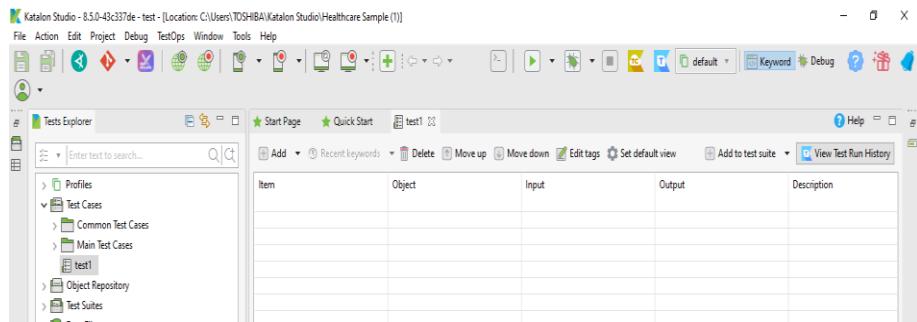
7. Setelah berhasil terinstall maka akan muncul notifikasi pesan pemberitahuan *plugin successfully installed.*



Gambar 13.7 Notifikasi Plugin Xpath Successfully Instaled

### b) Record Web Test

- Untuk memulai membuat project baru test case, maka kita bisa klik kanan pada test case, lalu pilih *new*, pilih test case, kemudia beri nama project yang dibuat, misalnya “test 1”



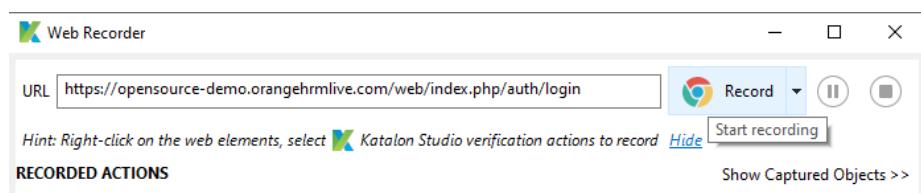
Gambar 13.8 Case test 1 Katalon Studio Project

- Selanjutnya pilih record web,



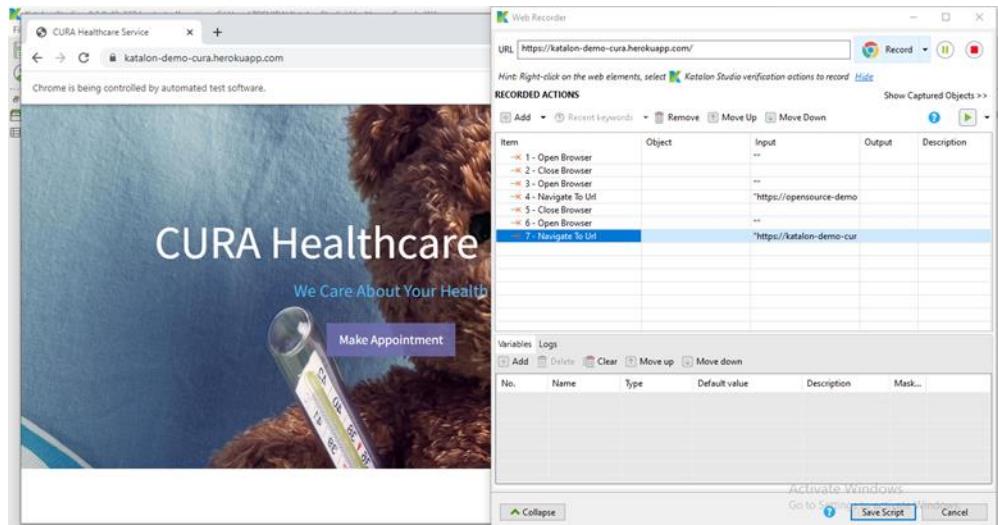
Gambar 13.9 Record Web Tools Katalon Studio

- Berikut merupakan contoh bagaimana untuk menjalankan sebuah test menggunakan aplikasi demo, masukkan link aplikasi web yang akan di test, pilih chrome untuk memulai *recording*.



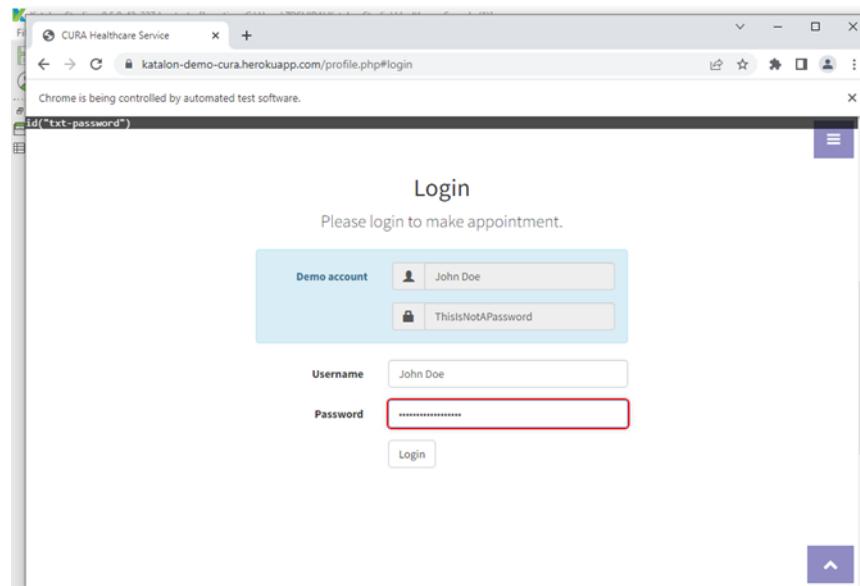
Gambar 13.10 Memulai Recording Action Katalon Studio

- Kemudian jendela chrome akan dibuka dan kita akan diarahkan ke sebuah halaman web demo yang menjadi test case, klik make appointment.



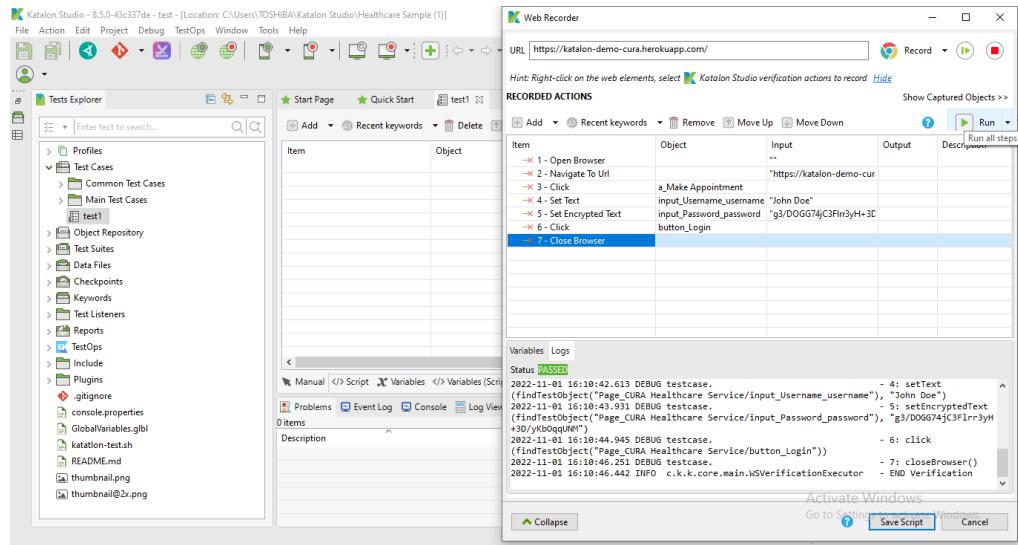
Gambar 13.11 Pengujian Website CURA Healthcare Service

- Kemudian masukkan username dan password milik akun demo, klik tombol login.



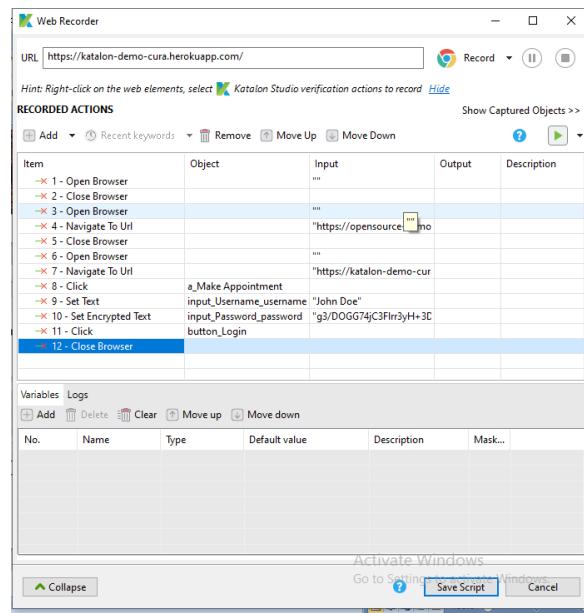
Gambar 13.12 Memasukkan Username dan Password akun Demo

- Semua aktivitas yang dilakukan pada web test akan di record, pada jendela web recorder aplikasi katalon.



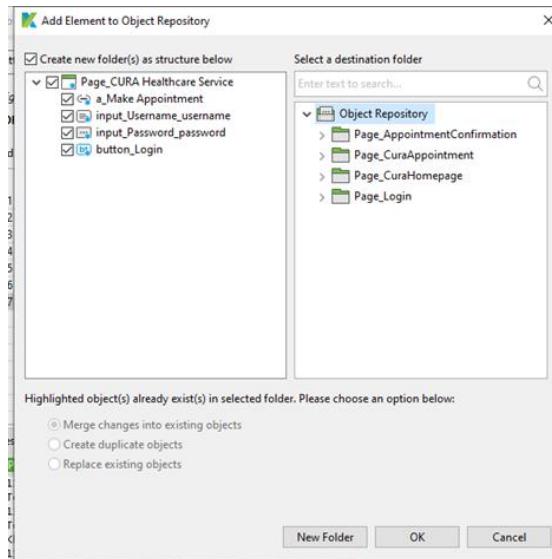
Gambar 13.13 Jendela Web Recorder Katalon Studio

- Setelah itu klik stop record. Kemudian kita dapat klik run kembali untuk melihat apakah test web dilakukan berjalan dengan benar.



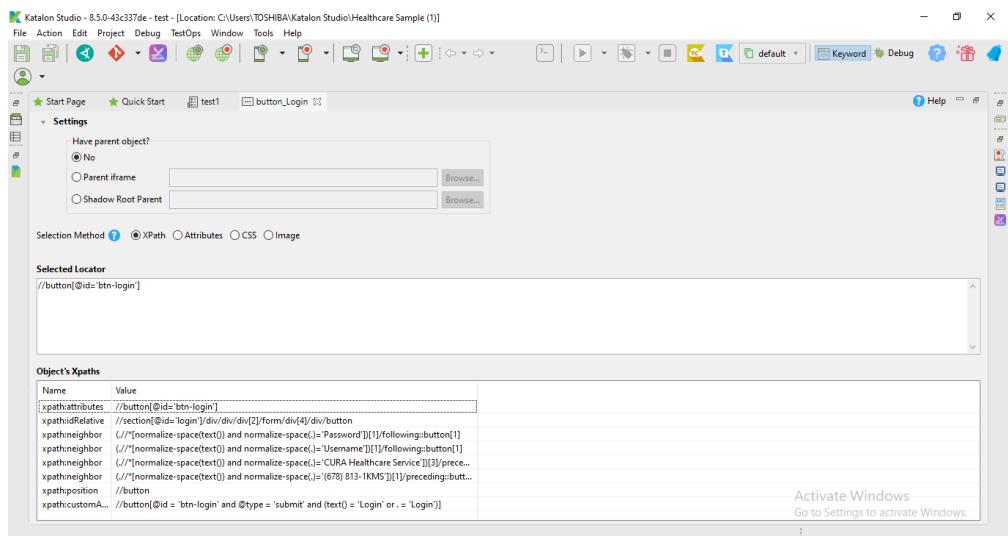
Gambar 13.14 Menjalankan Kembali Test Recording Web

- Klik oke pada tampilan add elemen to object repository, untuk menyimpan object yang sudah dilakukan test.

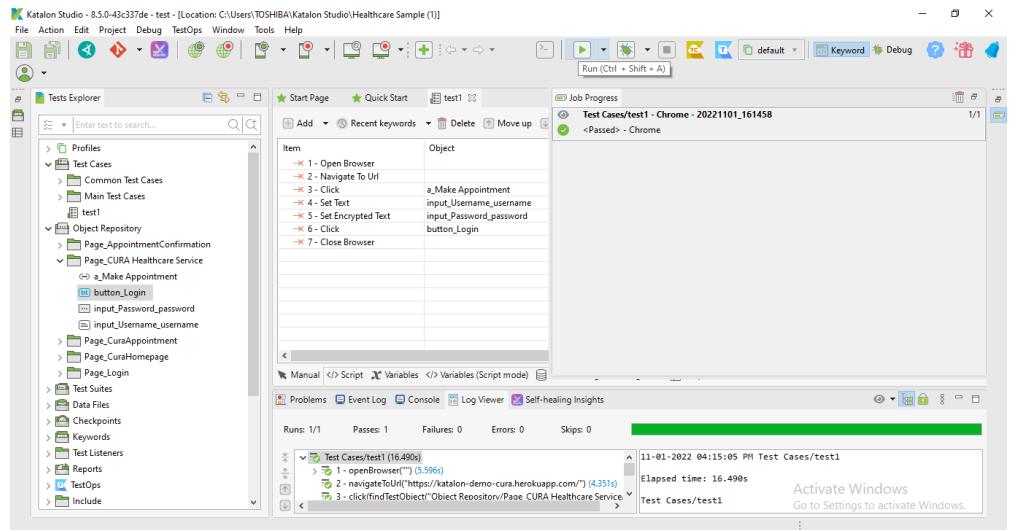


Gambar 13.15 Menyimpan Object Yang Sudah Dilakukan Test

- Pada bagian object repositori kini kita dapat melihat object beserta properties yang telah di record oleh aplikasi katalon studio

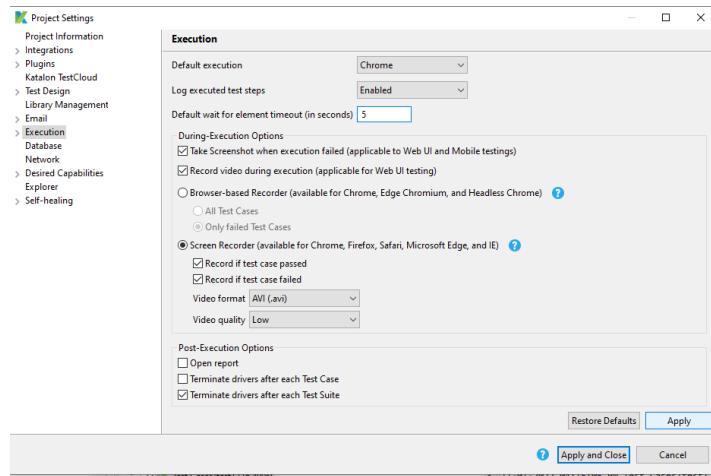


Gambar 13.16 Informasi Object Beserta Properties Web Recorder test1



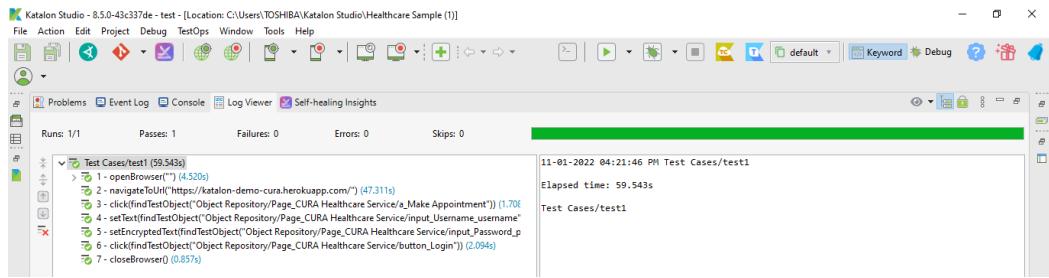
Gambar 13.17 Informasi Object Beserta Properties Web Recorder Katalon Studio

10. Kemudian, klik project setting, ke bagian execution, ubah nilai default wait element for timeout menjadi 5, klik apply, kemudian apply and close.



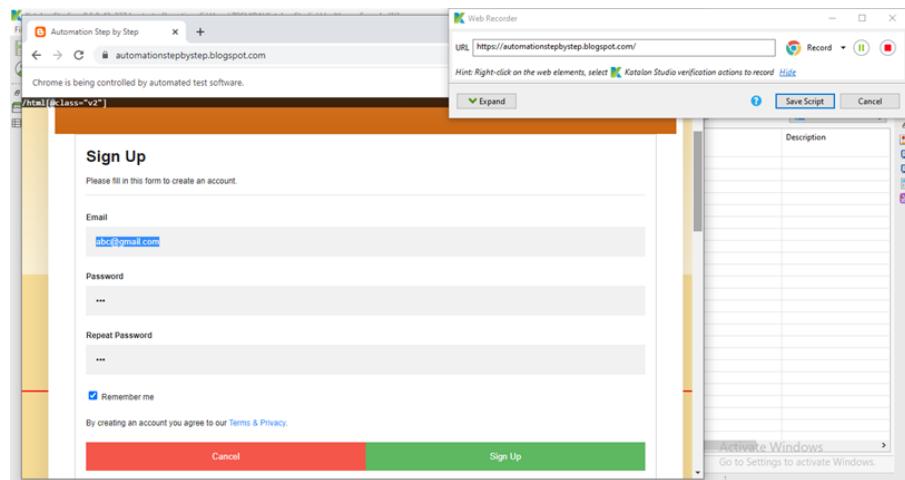
Gambar 13.18 Mengubah wait element for timout menjadi 5 detik

11. Kemudian cobalah klik run Kembali web test yang dimiliki, lalu lihatlah hasil identified dari proses login button.



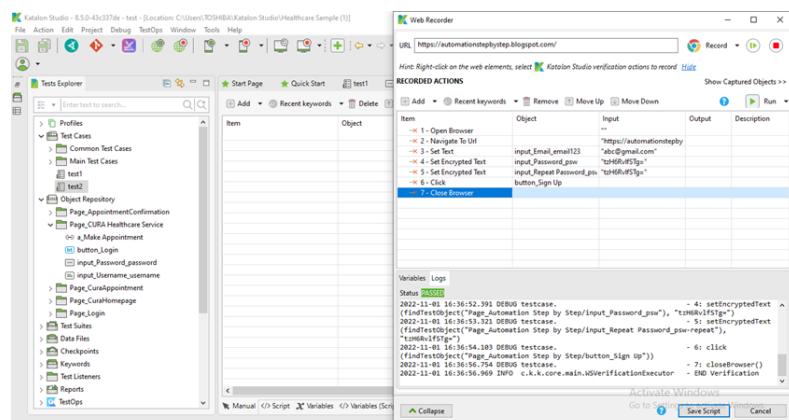
Gambar 13.19 Menjalankan kembali web recording katalon studio

12. Untuk mencoba test web pada object lain, mari lihat dan masukkan link test web berikut dengan memberi nama test2, kemudian klik run.



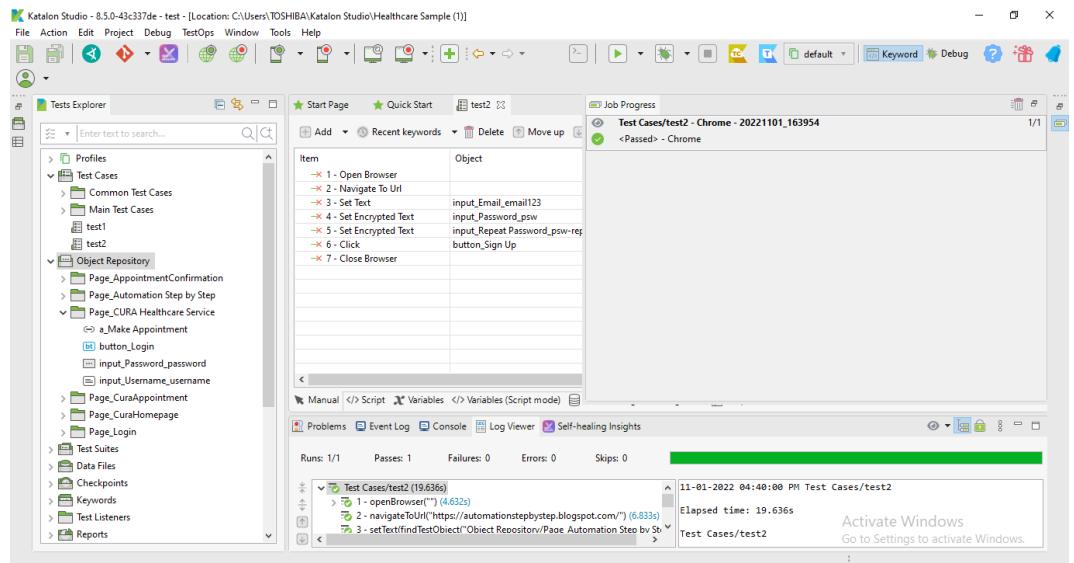
Gambar 13.20 Menjalankan Web Recorder Dengan Website Yang Lainnya

13. Berikut merupakan record yang disimpan oleh aplikasi katalon untuk test2



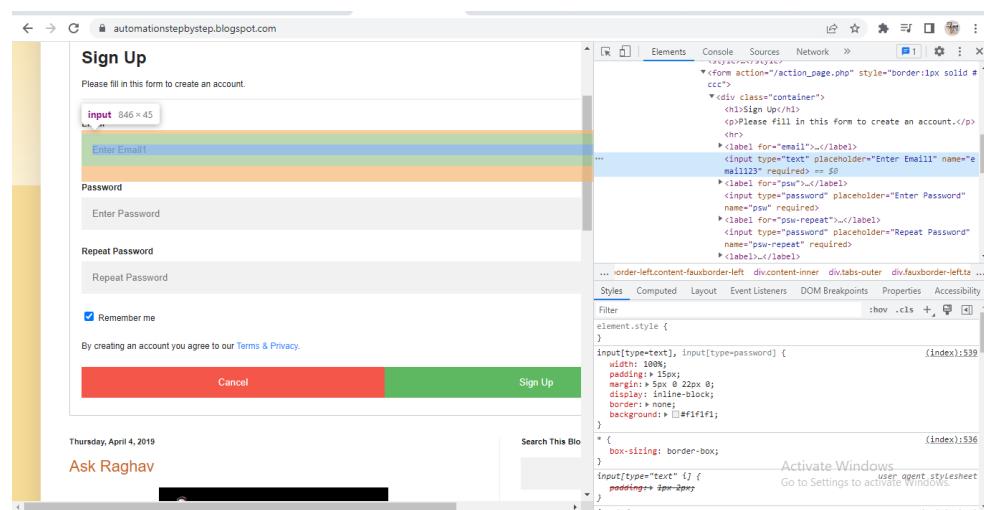
Gambar 13.21 Jendela Informasi Web Recorder

14. Berikut merupakan object repositori yang dicatat pada test2, beserta properties yang dimilikinya. Berikut adalah tampilan apabila hasil running berjalan dengan benar.



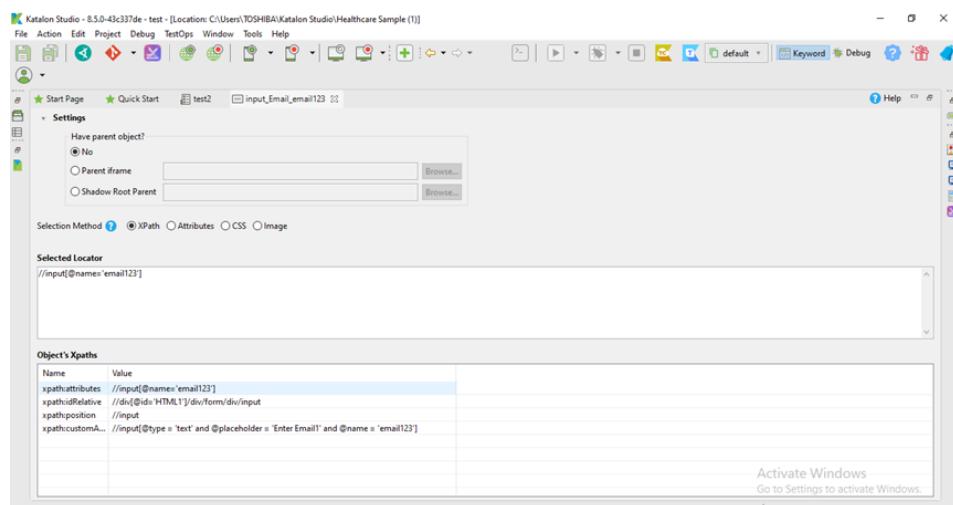
Gambar 13.22 Informasi Object Beserta Properties Web Recorder test2

15. Berikut adalah inspect yang diidentifikasi dari web test2, dimana katalon akan mengidentifikasi sesuai code yang dilakukan oleh developer, contoh field email yang diidentifikasi adalah email123, hal ini akan sesuai dengan code yang dimiliki oleh web yang dilakukan tes.



Gambar 13.23 Inspect dari web test2

16. Berikut merupakan value, dari identifikasi object yang terdapat pada web test2.

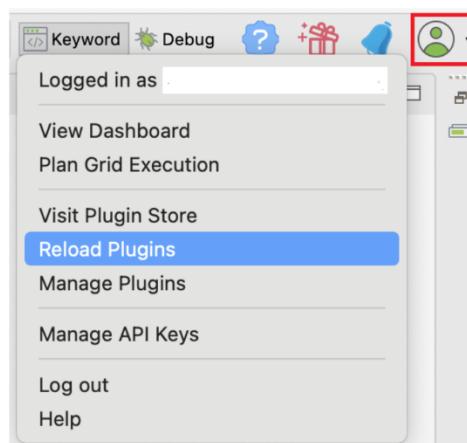


Gambar 13.24 Value Identifikasi Obejct Pada Web Test2

## B. Plugin Visual Validation

### a) Add applitools plugin

1. Pergi ke website katalon studio store
2. Pastikan anda telah login terlebih dahulu
3. Kemudian cari lah plugin API TOOLS:  
<https://store.katalon.com/product/44/Applitools-Integration>
4. Kemudian reload plugin



Gambar 13.25 Reload Plugins Pada Icon Profile

- Setelah itu klik manage plugin untuk melihat plugin anda yang telah terinstall

The screenshot shows a table with four columns: Name, Category, Type, and Status. There is one row of data:

Name	Category	Type	Status
Applitools Integration	Integration	Enterprise	ACTIVE

Gambar 13.26 Status Active Plugin Applitools Intergration

#### b) Create Account Applitools

- Buka Website Applitools <https://applitools.com/>
- Setelah itu lakukanlah registrasi
- Berikut setelah kita login di applitools

The screenshot shows the Applitools Eyes dashboard. On the left, there's a sidebar for 'Last 1 batch runs' with a 'Demo batch' entry. The main area displays 'Test results of batch: Demo batch'. It shows 1 test, 1 step, and 3 screenshots. One screenshot is labeled '1/3 Login Screen' and another is '2/3 Customer Overview'. A message from 'Keith from Applitools' is visible: 'Hi 😊 Do you have any questions?'. The top right has a 'Demo with an expert' button and other navigation icons.

Gambar 13.27 Tampilan Utama Applitools

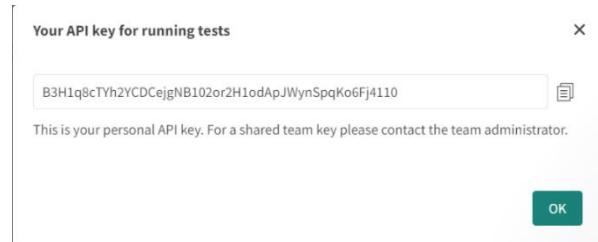
#### c) Add Applitools API Key In Katalon

- Buka applitools kemudian klik icon orang lalu klik My API Key

The screenshot shows the user profile page. It includes sections for 'USER (ADMIN)' with email 'ahmadwalialchalidi22@gmail.com', 'TEAM' with email 'ahmadwalialchalidi22@gmail.com', and a 'My API key' input field containing 'Admin'. At the bottom are 'Log out' and 'Logout' buttons.

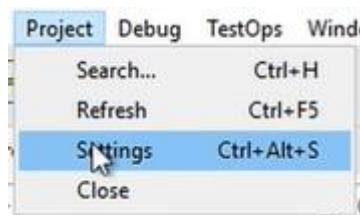
Gambar 13.28 My API key pada icon profile

2. Kemudian salin API keynya



Gambar 13.29 Menyalin API Key

3. Kemudian klik project lalu klik setings

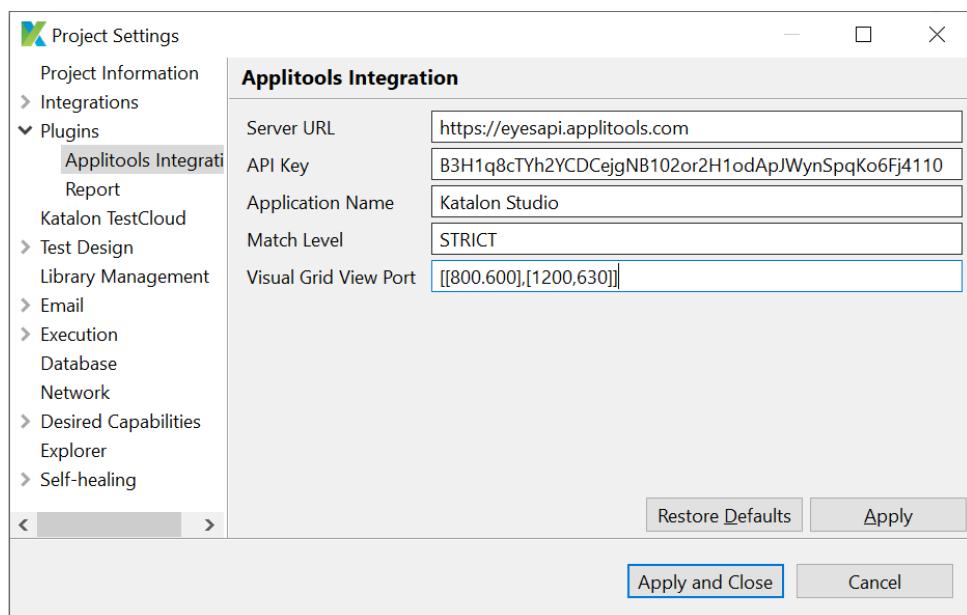


Gambar 13.30 Membuka Setting pada tab Project

4. Kemudian klik plugin lalu klik applitools integration, berikut pengertian dari colomn seperti gambar di bawah

- **Server URL:** URL server Applitools untuk diintegrasikan dengan Katalon Studio. URL ini dapat berupa URL Private Cloud Applitools. (tersedia dari Katalon Studio versi 8.4.0 dan seterusnya).
- **API Key :** Kunci API Applitools Anda (wajib). Anda dapat mempelajari cara mendapatkan Kunci API Applitools di dokumentasi Applitools: [Cara mendapatkan Kunci API Applitools](#) .
- **Name Application:** Nama aplikasi di Applitools. Anda tidak perlu membuat aplikasi di Applitools terlebih dahulu.
- **Match level :** Tingkat pertandingan di Applitools. Untuk mempelajari lebih lanjut tentang tingkat kecocokan, Anda dapat merujuk ke dokumentasi Applitools: Tingkat Kecocokan Applitools.

- **Visual Grid View Port** : [[width1, height1], [width2, height2], ...] (mis., [[800,600],[1200,630]]). Daftar viewports termasuk lebar dan tinggi untuk grid visual AppliTools. Setelah Anda menentukan semua gambar browser, viewports tersebut akan ditangkap dan dibandingkan. Jika Anda membiarkan Visual Grid View Port kosong, Katalon Studio menggunakan viewport browser pada titik eksekusi.

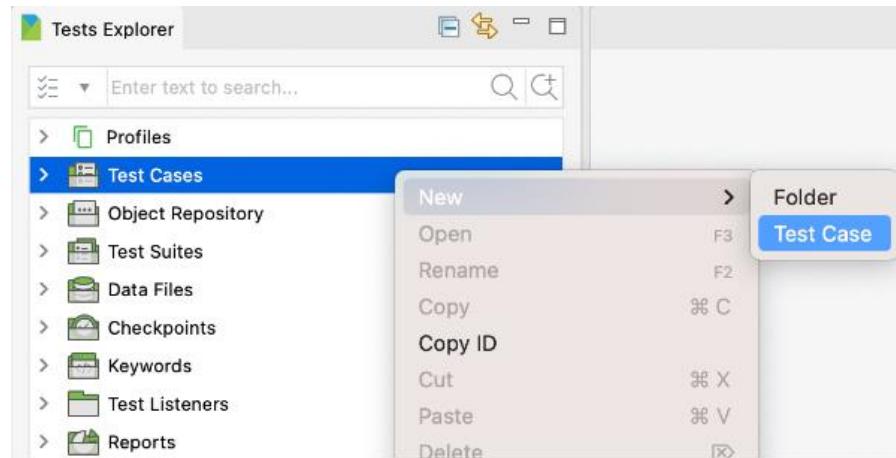


Gambar 13.31 Menyesuaikan Isian AppliTools Intergration

5. Kemudian klik apply

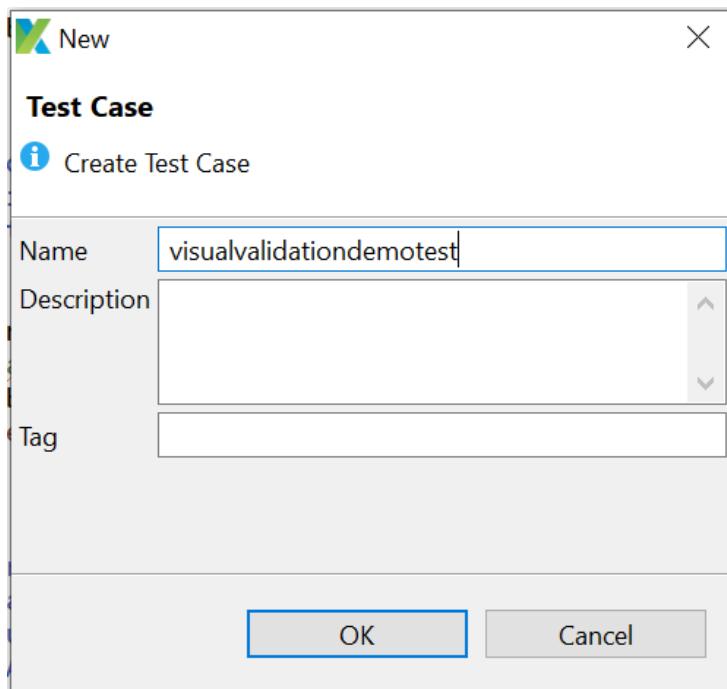
d) **Record Test And Add AppliTools Custom Keyword For Visual Validation**

1. Kemudian kita akan membuat new test case untuk membuat tempat uji coba kita.
2. Pada panel Tests Explorer , klik kanan pada folder Test Case, pilih New > Test Case.



Gambar 13.32 Membuat test case baru pada aplikasi katalon studio

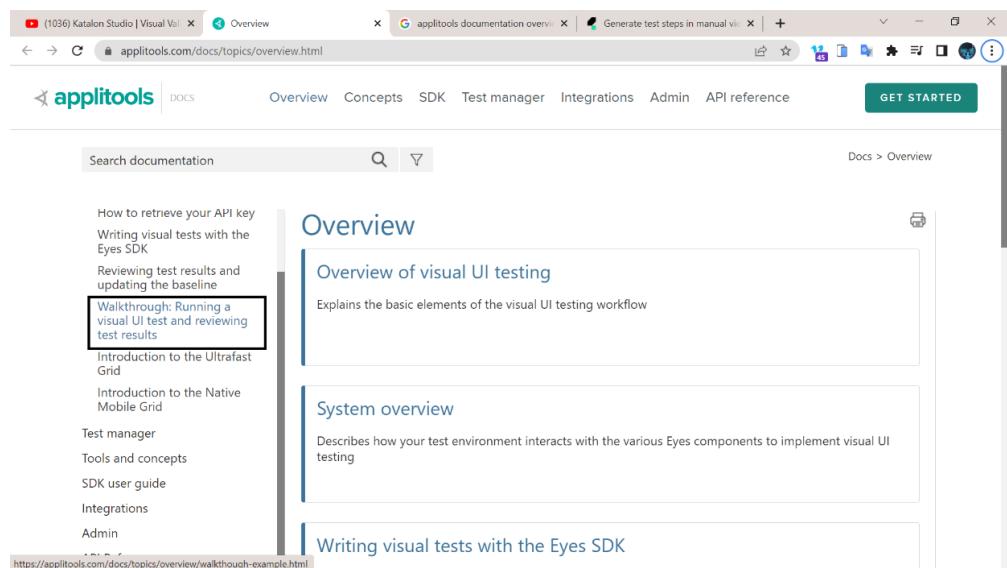
3. Pop up Test Case muncul. Masukkan nama kasus pengujian Anda, deskripsi, dan tag. Anda dapat mengatur proyek Anda secara lebih efisien dengan membuat nama kasus uji yang relevan, tag, dan memberikan deskripsi mendetail.



Gambar 13.33 Memberi label/nama test casenya

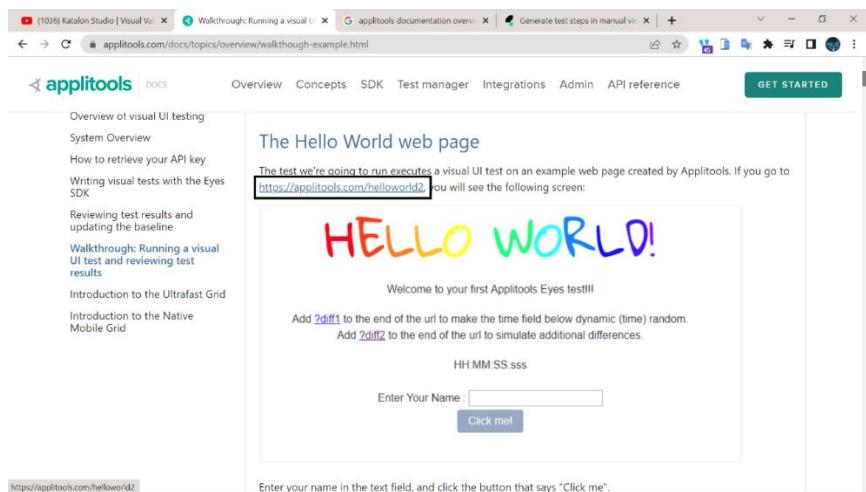
4. Kemudian untuk melakukan recording Test aplikasi kita akan mengambil sample yang telah di sediakan di applitools.

5. Kita pergi ke halaman web nya applitools kemudian ke documentation > overview, atau anda bisa klik link yang saya tautkan. <https://applitools.com/docs/topics/overview.html>
6. Kemudian kita akan mencari sample aplikasi yang akan kita gunakan untuk uji coba.
7. Kemudian klik pada Walkthrough



Gambar 13.34 Mengklik Link Walktrough pada website applitools

8. Berikut adalah aplikasi simple yang ingin kita coba, kemudian klik link yang ditandakan untuk menuju ke aplikasi simple yang di sediakan tersebut.



Gambar 13.35 Mengklik link The Hello World Web Page

9. Kemudian klik [diff1](#) untuk ke halaman yang berbeda.



Gambar 13.36 Mengklik link diff1

10. Sekarang kita bisa coba click me dan lihat apa yang terjadi.

Enter Your Name :

**Click me!**

A screenshot of a web page with a form. It has a text input field with the placeholder 'Enter Your Name :'. Below it is a blue button labeled 'Click me!'. The button has a slight shadow and rounded corners.

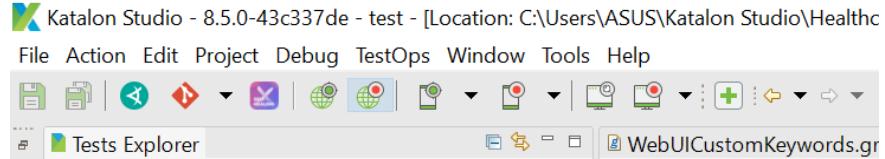
Gambar 13.37 Mengklik Tombol Click me!

11. Maka angka yang di atasnya nanti akan berubah ubah dan hal itu lah yang nanti akan kita record, uji dan test pada tahap pengujian ini..

19:46:34.186

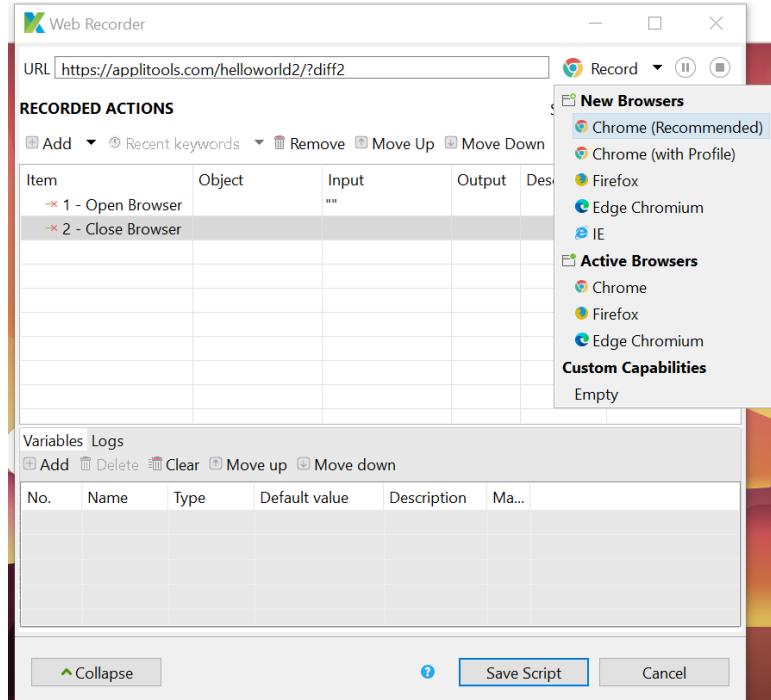
Gambar 13.38 Perubahan angka

12. Kemudian kita mengcopy URL web sederhana tersebut.
13. Kemudian kita pergi ke katalon lalu klik web record



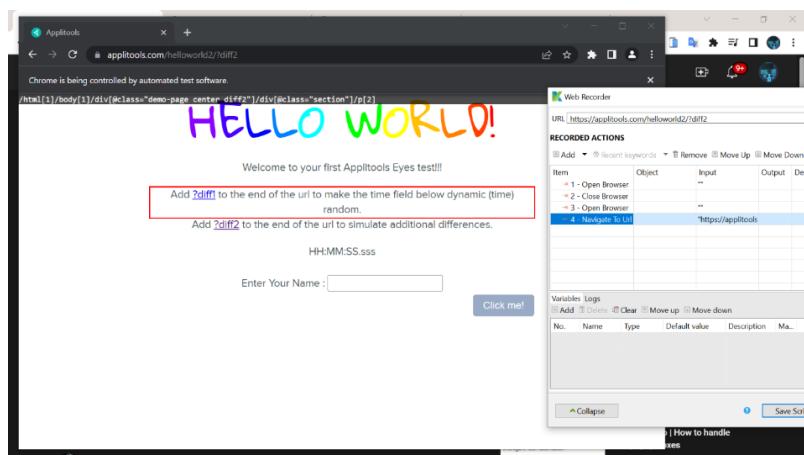
Gambar 13.39 Membuka Web Record Tools Aplikasi Katalon Studio

14. Kemudian pada URL kita pastekan link tadi, dan pada web browsernya kita menggunakan Chrome (Recommended)



Gambar 13.40 Masukkan Link The Hello World Page ke Web Recorder

15. Kemudian kita akan melihat web browsernya akan terbuka

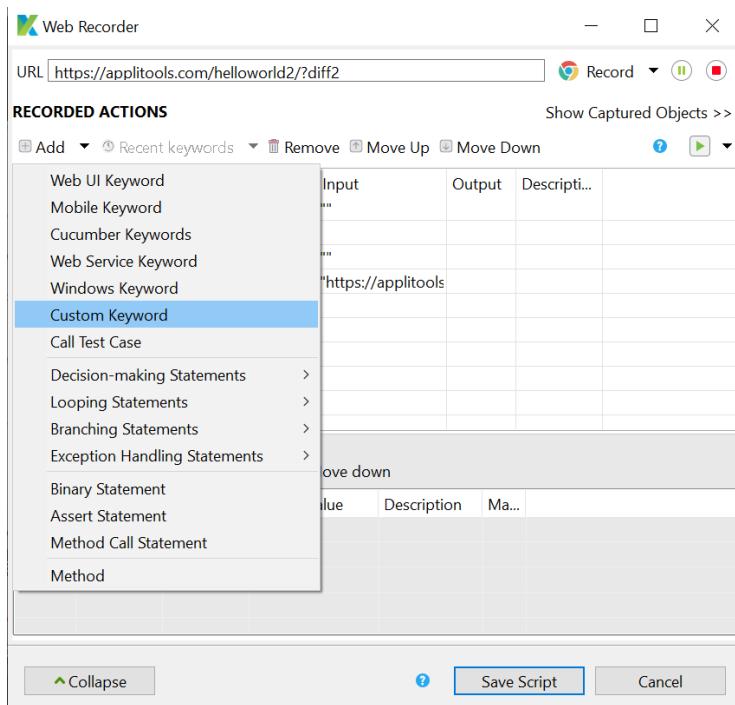


Gambar 13.41 Proses Pengujian The Hello World Page Website

16. Langkah selanjut nya adalah kita akan menambahkan Costum keyword.

17. Klik pada add yang ada di Web Recorder

18. Kemudian klik Custom Keyword



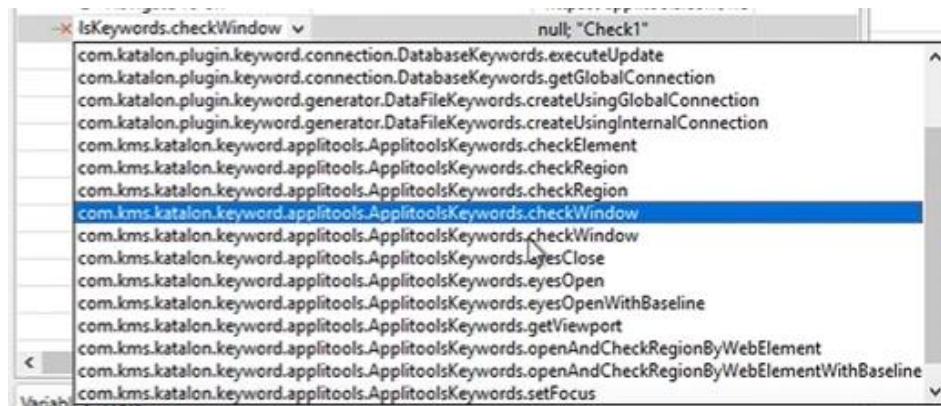
Gambar 13.42 Menambahkan Custom Keyword Pada Web Recorder

19. Kemudian klik pada input lalu pada value tuliskan lah check1 itu untuk menamai proses check nantinya jadi kita akan tau bahwasanya capture dari halaman itu yang akan di check namanya check1 dan value type nya pilih variable valuenya null saja karena nanti akan di isi otomatis ke ‘a’.

No.	Param Name	Param Type	Value Type	Value
1	eyes	Eyes	Variable	null
2	message	String	String	"Check1"

Gambar 13.43 Menuliskan check1 pada Input Web Recorder

20. Kemudian klik pada iskeywords dan klik pada tools check windows untuk memakai tools tersebut.



Gambar 13.44 Memilih Keywords.checkWindows pada IsKeywords

21. Kemudian kita akan menuju ke capture selanjut nya pada halaman aplikasi web sederhana tadi
22. Lalu klik [?diff1](#) untuk melakukan perubahan yang akan record

**HELLO !**

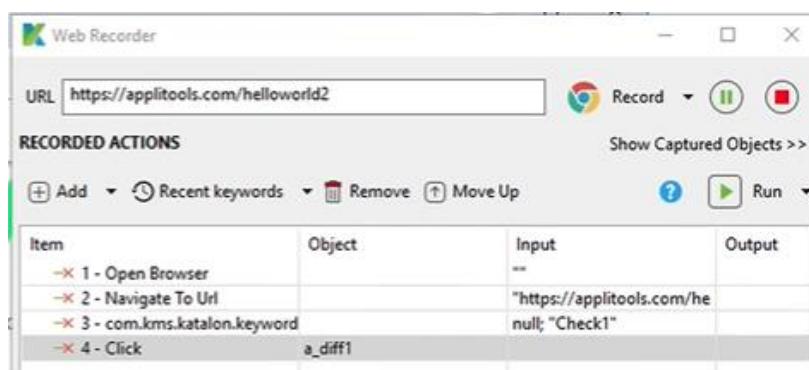
Welcome to your first Ap

Add [?diff1](#) to the end of the url to make t  
random  
Add [?diff2](#) to the end of the url to si

HH:MM:SS

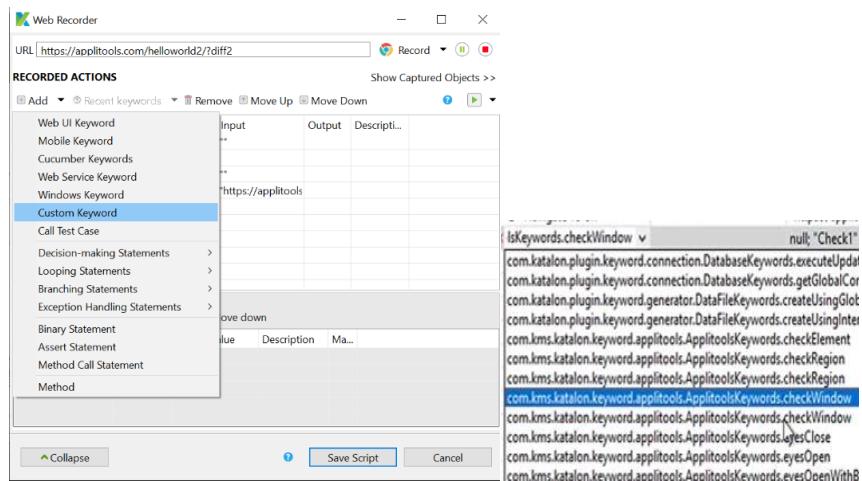
Gambar 13.45 Mengklik link ?diff1

23. Kemudian kita bisa melihat bahwasannya yang kita klik tadi sudah ter record



Gambar 13.46 Melihat Informasi hasil record

24. Kemudian untuk manamai dan menambahkan toolsnya maka kita akan klik add dan custom keyword lagi dan memilih tools yang sama.



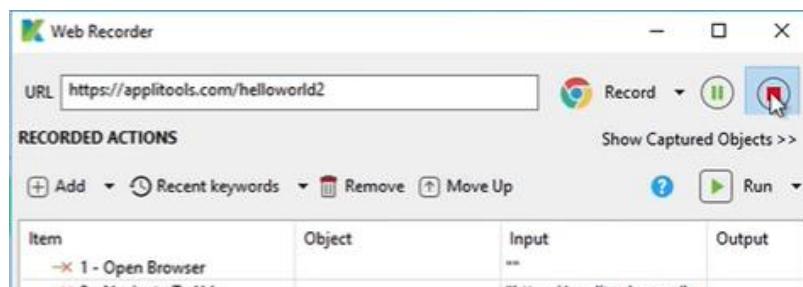
Gambar 13.47 Menambahkan Custom Keyword untuk kedua kalinya

25. Kemudian pada input kita akan menamai nya check 2 dan value nya adalah b.

No.	Param Name	Param Type	Value Type	Value
1	eyes	Eyes	Variable	b
2	message	String	String	"Check2"

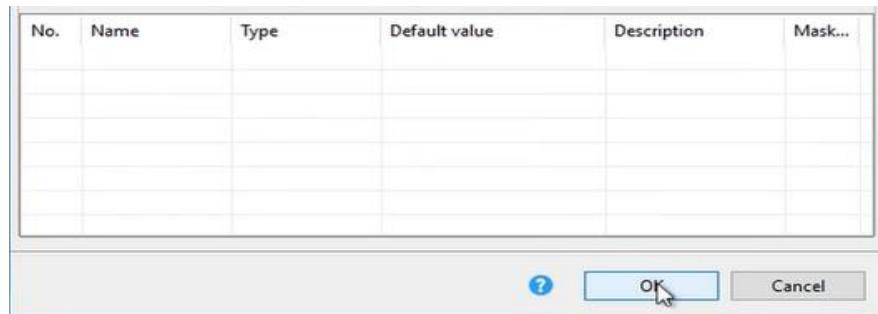
Gambar 13.48 Menuliskan check2 pada Input Web Recorder

26. Kemudian kita klik stop.



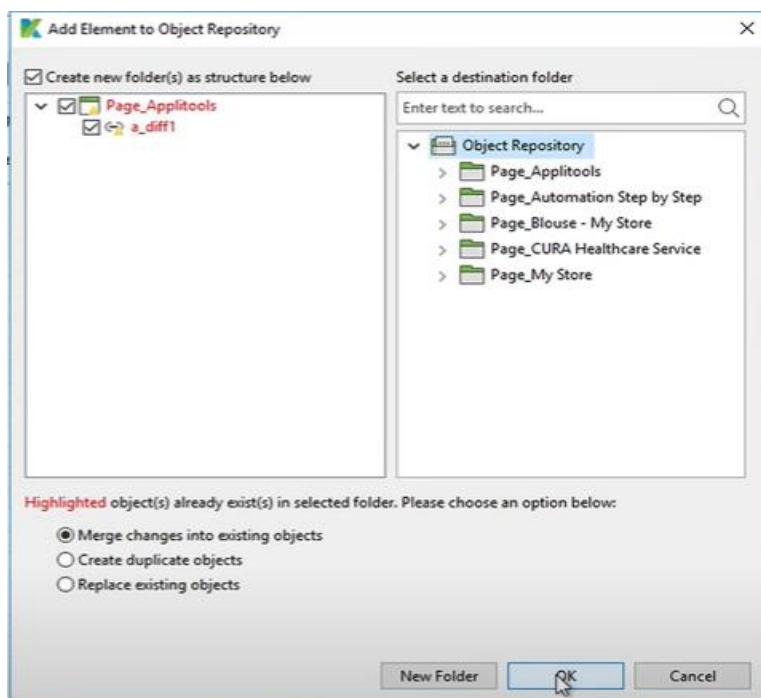
Gambar 13.49 Mengklik tombol stop recording

27. Kemudian kita klik oke lagi.



Gambar 13.50 Klik OK pada web recorder

28. Dan klik oke.



Gambar 13.51 Klik OK pada jendela Add Element to Obejct Repository

29. Dan sekarang kita memiliki test yang akan di uji

Item	Object	Input	Output
1 - Open Browser		""	
2 - Navigate To Url		"https://applitools.com/helloworld2"	
3 - com.kms.katalon.keyword.applitools.ApplitoolsKeywords.checkWindow		a; "Check1"	
4 - Click	a_diff1		
5 - com.kms.katalon.keyword.applitools.ApplitoolsKeywords.checkWindow		b; "Check2"	
6 - Close Browser			

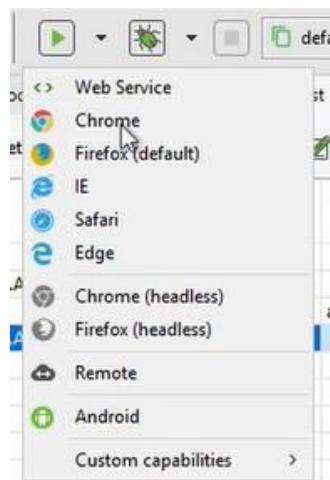
Gambar 13.52 Test Check2

30. Dan itu juga berlaku seperti algortimanya yang pertama pengecekan akan open browser kemudian akan melakukan navigate ke url nya atau masuk ke link nya, kemudian test akan mengklik diff1 yang seperti kita click tadi.

31. Yang di cek yaitu UI nya sama atau tidak dengan yang kita record sebelumnya, seperti yang saya katakana di atas yang kita cek adalah perubahan nomor pada saat tombol click me! Di tekan.

e) **Run Test and Check AppliTools Dashboard**

1. Sekarang kita akan menjalankan aplikasinya dari katalon untuk di test dan output test nya akan keluar di AppliTools Dashboard akun kita
2. Kita klik run dan pilih opsi Chrome



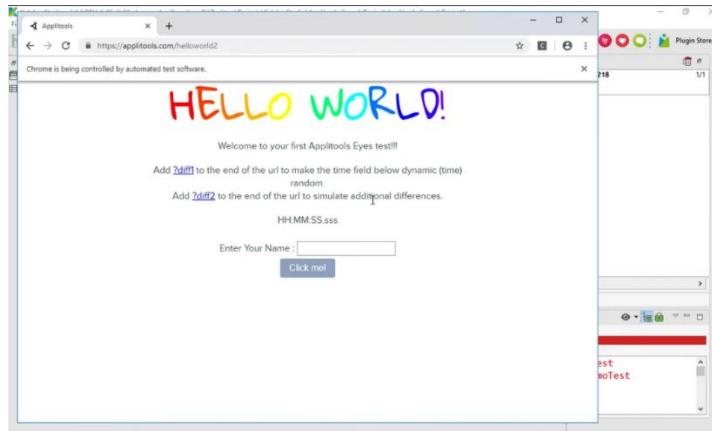
Gambar 13.53 Opsi Chrome Pada Web Service

3. Jika ada error maka coba pergi ke script dan hapus variabel a dan b, pada baris tersebut

```
    &ords.'com.kms.katalon.keyword.appliTools.AppliToolsKeywords.checkWindow'(@, 'Check1')
    &findTestObject('Page_AppliTools/a_diff1'))
    &ords.'com.kms.katalon.keyword.appliTools.AppliToolsKeywords.checkWindow'(@, 'Check2')
    &eBrowser()
```

Gambar 13.54 Menghapus Variabel a dan b pada script

4. Maka aplikasi sederhananya akan di jalankan



Gambar 13.55 Aplikasi The Hello World Page Berjalan

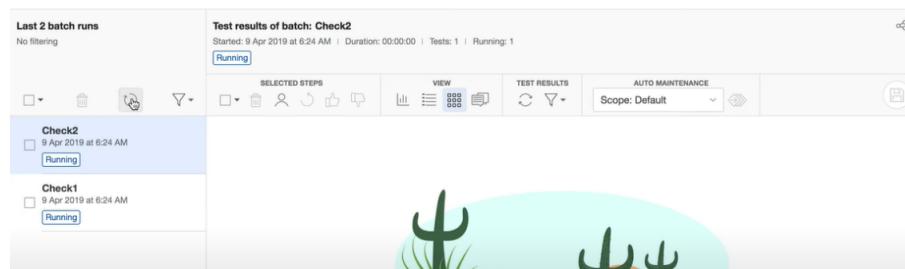
5. Kemudian kita pergi ke applitools web dan login ke akun kita

6. Kemudian klik referesh



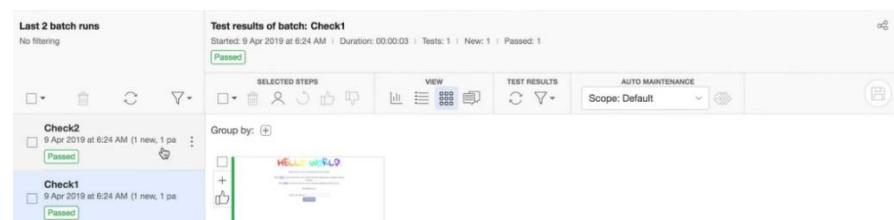
Gambar 13.56 Merefresh Applitools

7. Maka nanti akan terlihat bahwasanya check1 dan check2 running yakni sedang berjalan dan dilakukan test.



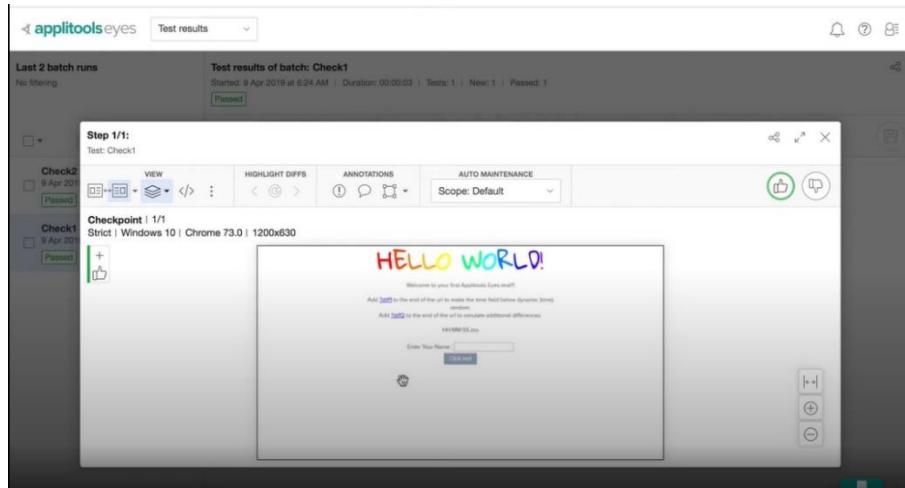
Gambar 13.57 Status berjalannya Check1 dan Check2 pada Applitools

8. Kemudian kita refresh lagi sampai kedua check tersebut passed, contohnya seperti berikut.



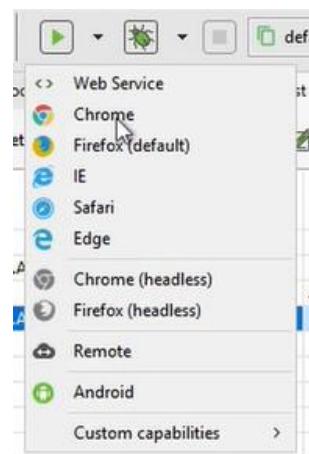
Gambar 13.58 Merefresh kembali Applitools

## 9. Kemudian kita bisa melihat capturenya



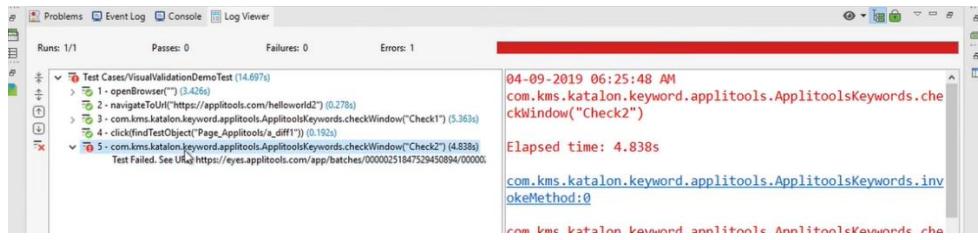
Gambar 13.59 Hasil Caputure The Hello World Page Website

10. Check1 dan check2 passed karena itu hanya untuk mengambil acuan pertama yang nantinya akan dibandingkan dengan test-test selanjutnya dan yang kita check adalah nomor yang ada di check dua akan berubah dan akan menghasilkan valuenya unresolved.
11. Kemudian kita akan running lagi aplikasi sederhananya dari katalon



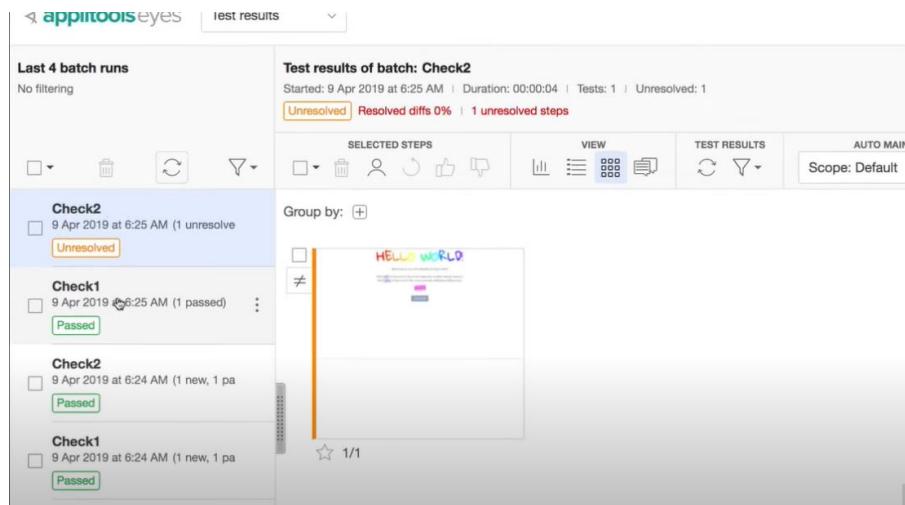
Gambar 13.60 Menjalankan kembali aplikasi sederhana dari katalon

12. Kemudian klik log viewer kita bisa lihat ada error dikarenakan ada perbedaan capture pada check2 dari acuan sebelumnya.



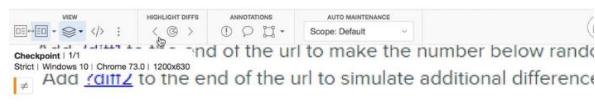
Gambar 13.61 Pesan Error Test Case pada Log Viewer

13. Kemudian kita refresh lagi di halaman web appli tools
14. Kemudian kita bisa melihat check1 passed dikarenakan gambarnya sama dengan gambar acuan yang sebelumnya.



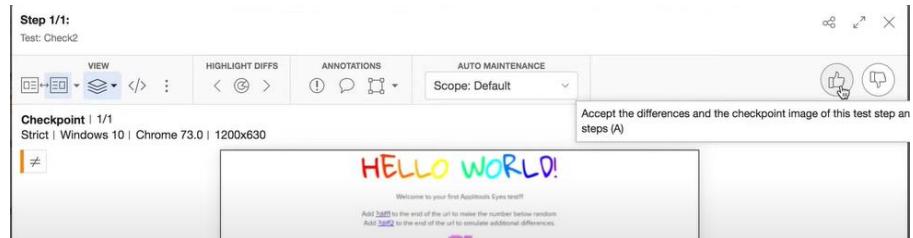
Gambar 13.62 Check 1 yang telah Passed

15. Kemudian kita pergi ke check2 hasilnya adalah unresolved dikarenakan nomornya berbeda dengan acuan sebelumnya.
16. Yang kita lakukan selanjutnya adalah kita harus meluluskan ada menggagalkannya dikarenakan untuk memastikan apakah perubahan itu memang valid ataupun tidak dan gambar tersebut akan dijadikan acuan yang baru



Gambar 13.63 Meluluskan Test Case The Hello World Page

17. Jika anda ingin menerima gambar ini sebagai acuan baru dan perubahan yang wajar maka anda harus klik button icon like jika sebaliknya (tidak setuju) maka anda klik tombol icon dislike.



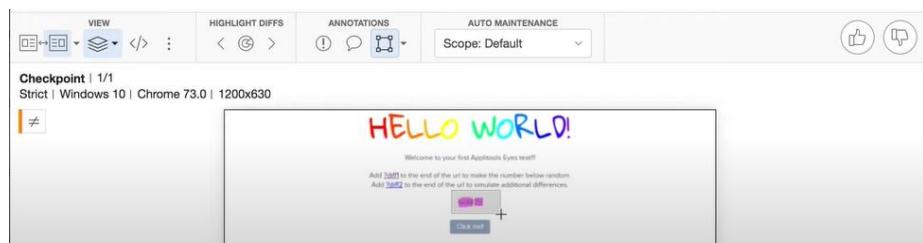
Gambar 13.64 Tombol Like Untuk Menerima Gambar ini

18. Apabila anda ingin mengabaikan perubahan di tempat tertentu saja atau yang seperti di atas yakni di nomor saja maka anda harus klik tools tersebut.



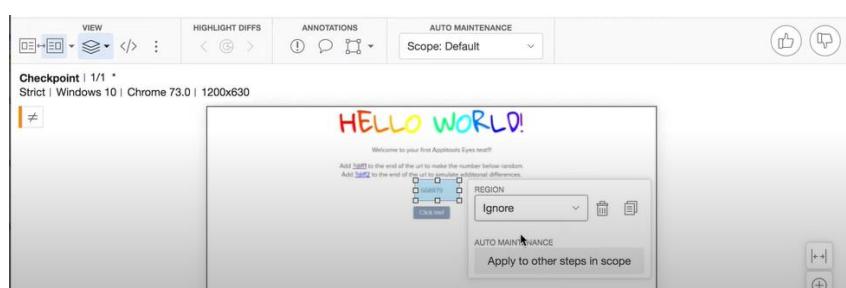
Gambar 13.65 Tools untuk mencrop area

19. Lalu anda bisa menandai area mana yang ingin anda abaikan.



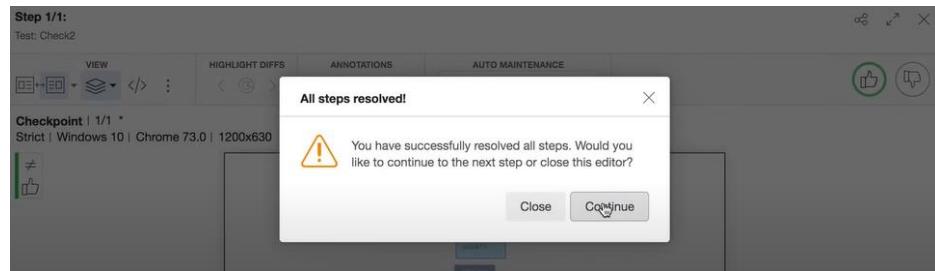
Gambar 13.66 Mengcrop area yang akan diabaikan

20. Kemudian pilih Ignore



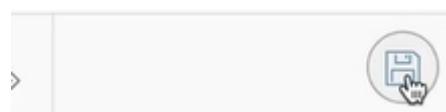
Gambar 13.67 Memilih Region Ignore

21. Kemudian klik icon like di pojok atas kanan lalu pop up akan muncul dan klik continue.



Gambar 13.68 Pop Up Ketika Di Klik tombol like

22. Lalu klik Save



Gambar 13.69 Menyimpan Preset

23. Kemudian kita bisa melihat bahwasannya check 2 telah passed atau lulus.

Gambar 13.70 Check2 telah berstatus Passed

## **BAB 14**

### **BDD CUCUMBER TEST**

#### **14.1. Tujuan**

Setelah menyelesaikan praktikum ini, Mahasiswa diharapkan mampu:

1. Mahasiswa memahami definisi BDD
2. Mahasiswa dapat melakukan BDD Cucumber *Test* di Katalon Studio
3. Mahasiswa memahami Bagaimana Cucumber Runner
4. Mahasiswa dapat *melakukan Create Cucumber Runner* di Katalon Studio

#### **14.2. Landasan Teori**

Dalam pengujian sebuah produk, pihak perusahaan mengambil Langkah yang maju agar mendapatkan keuntungan yang lebih baik dalam pengujian perangkat lunak dengan cara mereka menerapkan skenario uji penerimaan yang penting pada saat pengembangan dilakukan secara langsung. Umumnya metode yang mereka lakukan ini dikenal sebagai *Behaviour Driven Development* atau BDD.

*Behaviour Driven Development* adalah metode untuk memberikan para penguji untuk membuat skrip pengujian dari perspektif pengembang dan pelanggan. Pada awalnya pengembang (*developer*), manajer proyek (*Project Manager*), QA, Penguji penerimaan pengguna (*User acceptance tester*), dan pemilik produk (*stockholder*) berkumpul dan berdiskusi tentang bagaimana skenario pengujian yang harus dilewati untuk memanggil sebuah perangkat lunak ini.

##### **14.2.1. Example**

Jika para pengembang sedang membuat fitur otentikasi pengguna, maka untuk dikatakan sukses jika dapat mengikuti beberapa skenario pengujian utama sebagai berikut.

- a) Pengguna atau user dapat masuk menggunakan *username* dan *password* yang benar
- b) Pengguna atau user tidak dapat masuk menggunakan *username* yang salah dan *password* yang benar

- c) Pengguna atau user tidak dapat masuk menggunakan *username* yang benar dan *password* yang salah

#### 14.2.2. Bagaimana cara ini dapat bekerja?

Pada saat kode ini telah siap, skrip pengujian juga telah siap. Kode yang telah dibuat harus lulus skrip pengujian yang telah ditentukan oleh BDD. Jika tidak terjadi, maka *code refactoring* akan diperlukan. Kode akan dibekukan bila sudah dilakukan-nya eksekusi pada skrip pengujian yang ditentukan telah berhasil.



Gambar 14.1 BDD Cucumber Test proses

Ini merupakan gagasan yang dibuat sangatlah sederhana, tetapi apa yang akan kita butuhkan untuk menerapkan konsep ini. Jawaban-nya adalah *Framework Behaviour Driven Development* (BDD). Cucumber adalah salah satu *tools* yang bersifat *open source* yang mendukung *Behaviour driven development*. Lebih tepatnya Cucumber dapat mendefinisikan sebagai *testing framework*, *driven* dengan teks *plain English*. Ini berfungsi sebagai dokumentasi, *automated tests*, dan *development*, semua-nya dalam satu.

Jadi, apa yang akan dilakukan oleh Cucumber? Berikut akan dijelaskan langkah-langkah sebagai berikut.

- a) Cucumber membaca kode yang dibuat dalam teks *plain English text* yang terdapat di *feature file*.
- b) Setelah itu akan menemukan kecocokan yang tepat dari setiap Langkah dalam definisi langkah.
- c) Beberapa potongan kode yang akan dieksekusi berupa *software framework* yang berbeda seperti **Selenium**, **Ruby on Rails**, dan lain-lain. Tidak semua BDD *Framework tool* mendukung setiap *tool*.
- d) Ini juga menjadi alasan mengapa Cucumber begitu populer dan menjadi teratas pada *framework* lain, seperti **JBehave**, **JDave**, **Easyb**, dan lain-lain.

Cucumber dapat mendukung lebih dari selusin *software platform* yang berbeda, antara lain :

- Ruby on Rails
- Selenium
- PicoContainer
- Spring Framework
- Watir

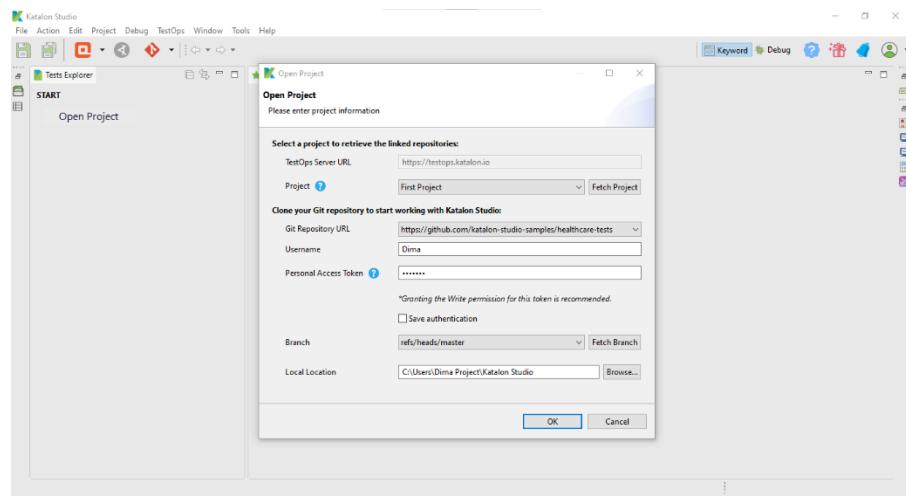
#### **14.2.3. Keuntungan Cucumber dibandingkan dengan *tools* lainnya?**

- a) Cucumber mendukung berbagai Bahasa seperti Java.net dan Ruby.
- b) Dapat bertindak sebagai penghubung antara bisnis dan Bahasa teknis. Selain itu kita dapat membuat *test case* dalam *plain English text*.
- c) Memungkinkan skrip pengujian ditulis tanpa mengetahui kode apapun, demikian juga memungkinkan keterlibatan-nya *non-programmers*.
- d) Dapat melayani tujuan dari *test framework* dari ujung ke ujung, tidak seperti *tools* lainnya.
- e) Dari segi arsitektur skrip pengujian-nya yang sederhana, Cucumber menyediakan penggunaan kode yang bersifat *reusability* atau dapat digunakan kembali.

## 14.3. Pembahasan

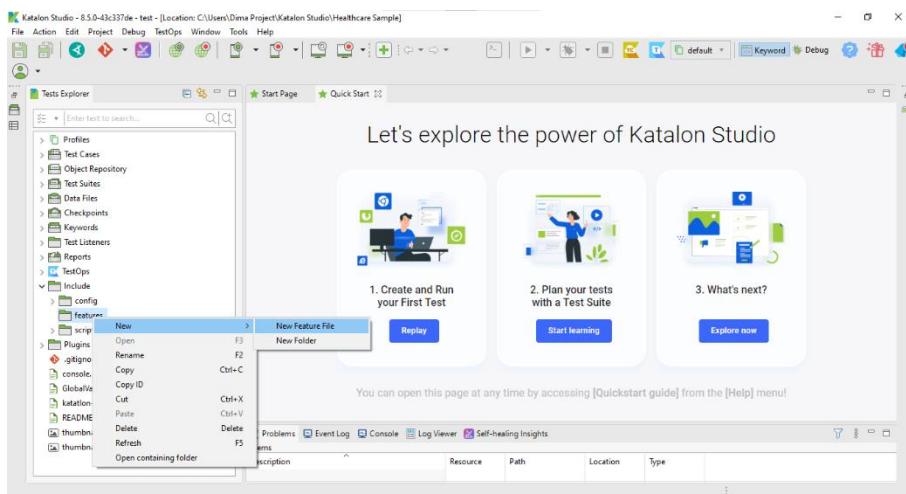
### 14.3.1. Membuat BDD Cucumber tests di Katalon Studio

1. Buka Katalon Studio, Lalu *Open Project*. Pada bagian ini, isikan *Username* dan *Personal Access Token*. Jika sudah, klik OK



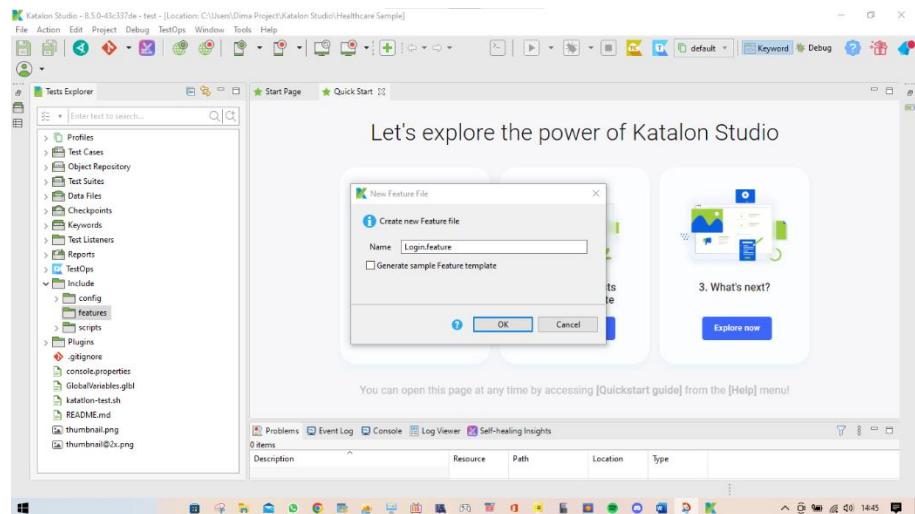
Gambar 14.2 open project

2. Selanjutnya pilih *folder Include* pada bagian kiri, klik kanan, pilih *New* dan pilih *New Feature File*



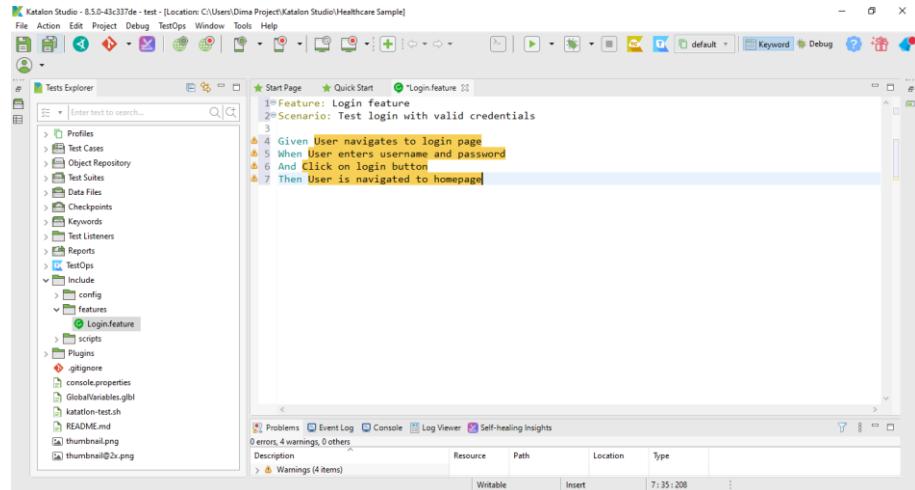
Gambar 14.3 pilih folder include

3. Pada bagian ini silahkan isikan nama *file*-nya. Disini kita akan buat fitur untuk *login* dengan nama **Login.feature**. Kemudian klik OK



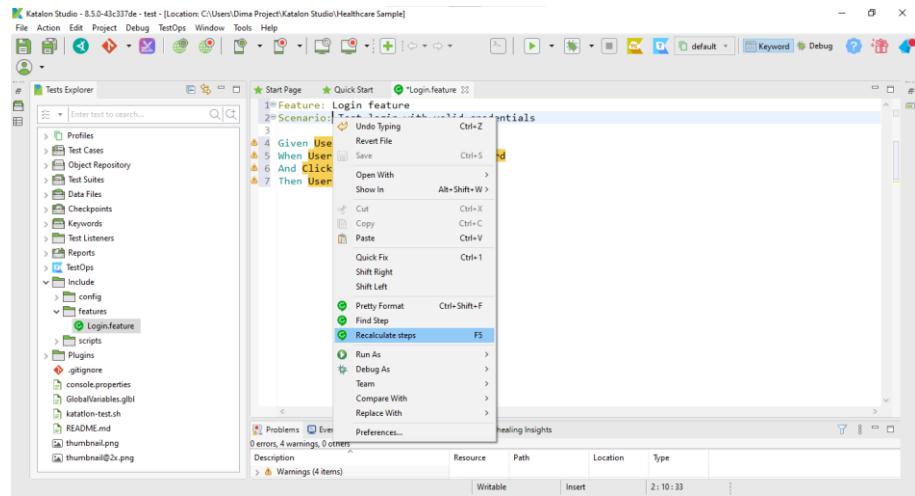
Gambar 14.4 buat fitur baru

4. Dan ini adalah tampilan dari *file Login.feature* yang sudah tertera dengan *script template* bawaan. Kemudian hapus *script* bawaan hingga kosong, lalu isikan *script* baru seperti yang tertera pada gambar dibawah



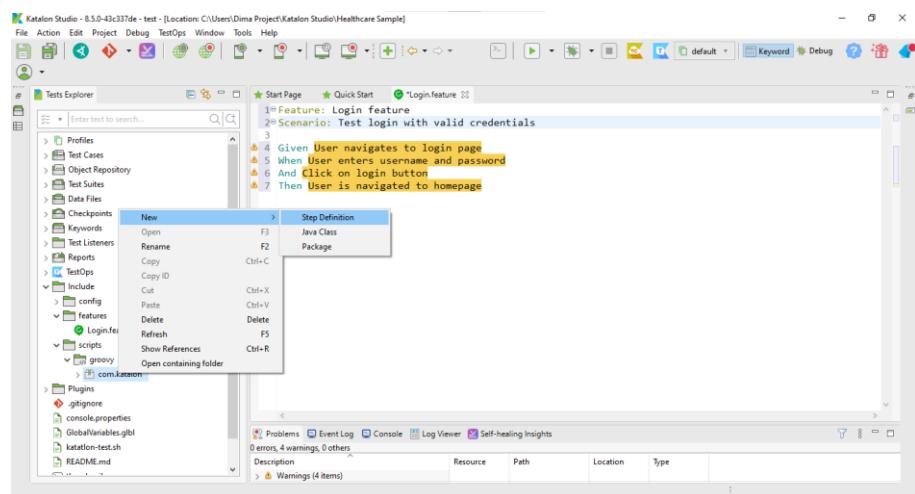
Gambar 14.5 tampilan file fitur login

5. Pada bagian ini, terdapat *warning* pada bagian *script* yang di highlight oleh warna kuning dikarenakan *script* ini masing kosong atau tidak ada statement-nya. Untuk mengatasi kasus ini kita akan menggunakan fitur dari Katalon Studio yaitu ***Recalculate steps***.



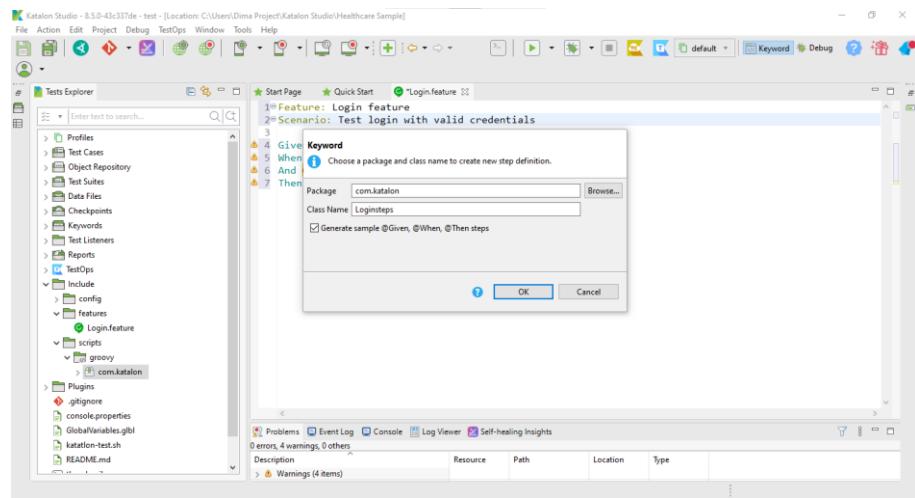
Gambar 14.6 pilih recalculate steps

6. Kita akan membuat file baru dengan cara pilih folder `scripts\groovy` dan klik kanan pada `com.katalon`, pilih *New* dan pilih *Step Definition*.



Gambar 14.7 membuat new step definition

7. ada bagian *Class Name*, isi nama-nya dengan `Loginsteps`. Kemudian klik OK.



Gambar 14.8 isi nama class name

8. Setelah membuat file baru, pada line 48, hapus dan buat dengan script baru

```

package com.katalon
import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()

class Loginsteps {
    /*
     * The step definitions below match with Katalon sample Gherkin steps
     */
    @Given("I want to write a step with (.*)")
    def I_want_to_write_a_step_with_name(String name) {
        println name
    }

    @When("I check for the (\d+) in step")
    def I_check_for_the_value_in_step(int value) {
        println value
    }

    @Then("I verify the (.*) in step")
    def I_verify_the_status_in_step(String status) {
        println status
    }
}

```

Gambar 14.9 hapus dan buat script baru

9. Pada bagian line 49, buat untuk fitur navigasi *login page*. Silahkan ikuti script yang tertera Digambar bawah

Gambar 14.10 buat fitur navigasi

- Kembali lagi ke file **Login.feature**. Lalu untuk mengecek apakah *script* di line 4 pada file tersebut sudah ada *statement* yang telah dibuat atau tidak dengan cara klik kanan, pilih *recalculate steps*

Gambar 14.11 cek login fitur

- Hasil dari kita menggunakan fitur *Recalculate steps* adalah bahwa line 4 yang sebelumnya memiliki *warning* atau teks-nya terhighlight warna kuning sekarang tidak ada

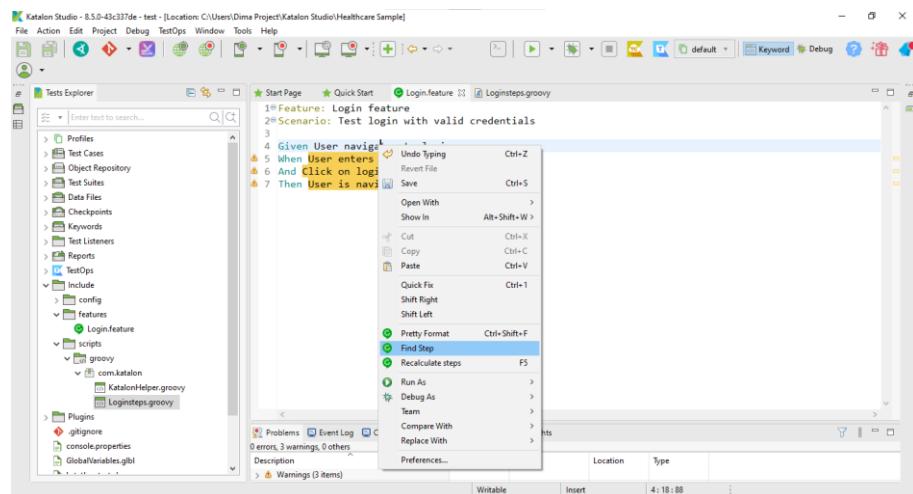
```

1 Feature: Login feature
2 Scenario: Test login with valid credentials
3
4 Given User navigates to login page
5 When User enters username and password
6 And Click on login button
7 Then User is navigated to homepage

```

Gambar 14.12 hasil recalculate

12. Untuk mengetahui lebih lagi apakah *script* pada *line 4* sudah ada statement-nya atau tidak, silahkan klik kanan kemudian pilih *Recalculate steps*



Gambar 14.13 cek script dengan cara recalculate

13. Maka akan menuju ke *script* yang terdapat di *line 49* pada *file Loginsteps.groovy* yang berarti *script* ini mempunyai statement-nya

```

1 package com.katalon
2 import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
4
45
46
47 class Loginsteps {
48
49@ Given("User navigates to login page")
50    def navigateToLoginPage() {
51        println ("\n I am inside navigateToLoginPage ")
52    }
53 }

```

Gambar 14.14 menuju ke script

14. Selanjutnya silahkan ikuti lagi Langkah-langkah yang terdapat pada nomor 9 dan buat *script* lagi untuk membuat fitur ***enter username and password***, ***login button*** dan ***navigated to homepage*** pada gambar dibawah.

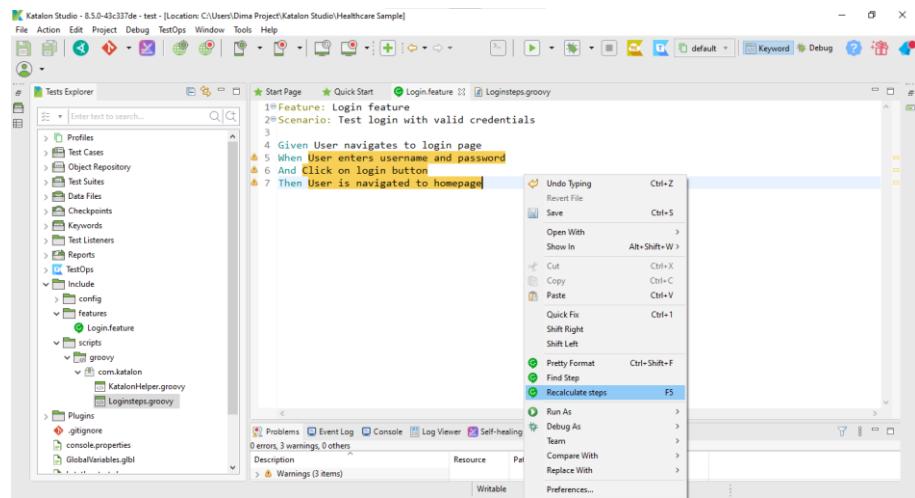
```

package com.katalon
import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
44
45
46
47 class Loginsteps {
48
49@ Given("User navigates to login page")
50    def navigateToLoginPage() {
51        println ("\n I am inside navigateToLoginPage ")
52    }
53
54@ When("User enters username and password")
55    def enterCredentials() {
56        println ("\\n I am inside Credentials")
57    }
58
59@ Then("Click on login button")
60    def clickLogin() {
61        println ("\n I am inside clickLogin")
62    }
63
64@ Then("User is navigated to homepage")
65    def verifyHomePage() {
66        println ("\n I am inside home page")
67    }
68

```

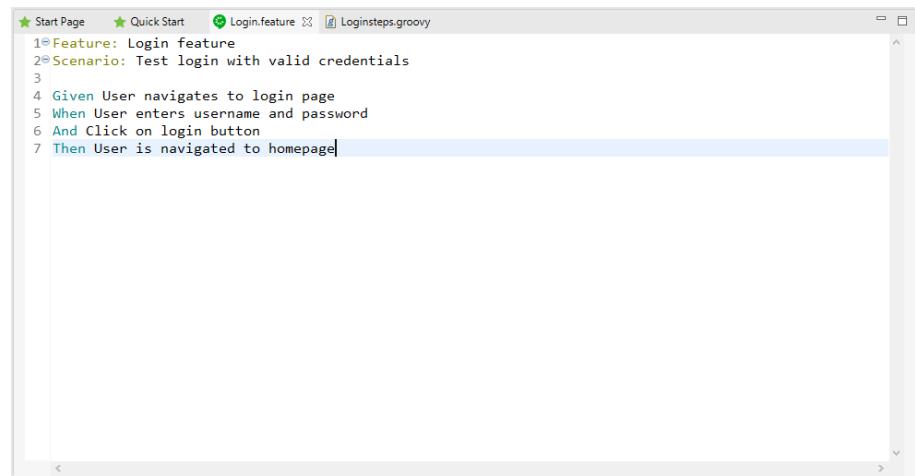
Gambar 14.15 buat script untuk membuat fitur

15. Kemudian cek Kembali apakah *script* ini sudah ada *statement* atau tidak dengan cara klik kanan, pilih *Recalculate steps*



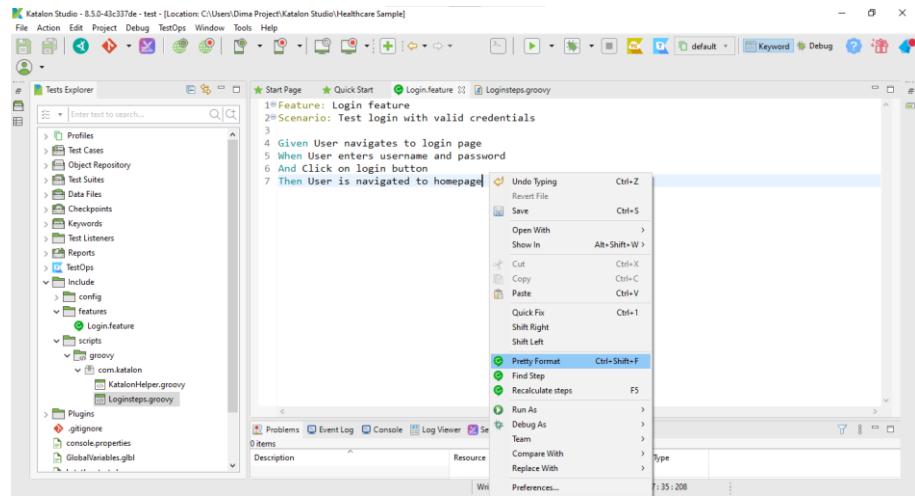
Gambar 14.16 cek statement script

16. Maka hasil-nya adalah *script* ini sudah memiliki *statement*



Gambar 14.17 hasil script yang sudah memiliki statement

17. Untuk merapikan *script* yang tidak rapi atau berantakan, klik kanan, pilih *Pretty Format*



Gambar 14.18 pilih pretty format untuk merapikan script

18. Hasil-nya adalah *script* ini yang semula tidak rapi sekarang sudah rapi

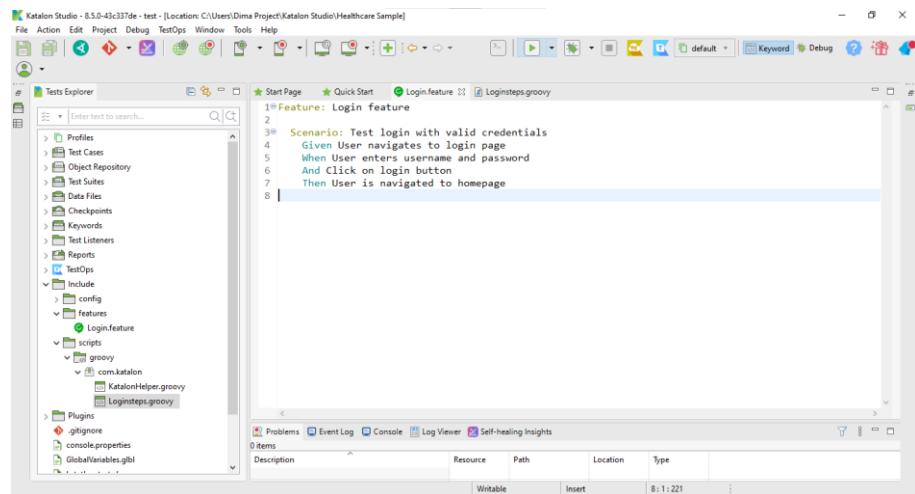
```

1@ Feature: Login feature
2
3@ Scenario: Test login with valid credentials
4 Given User navigates to login page
5 When User enters username and password
6 And Click on login button
7 Then User is navigated to homepage
8

```

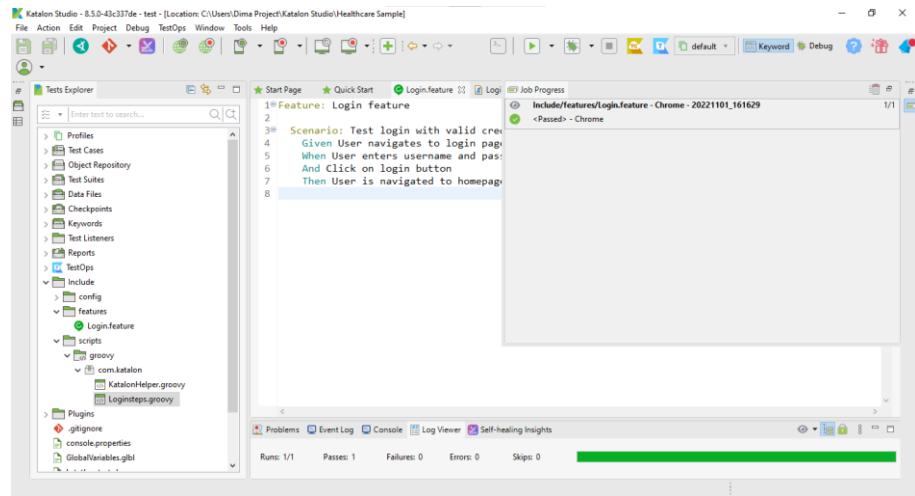
Gambar 14.19 hasil script yang sudah rapi

19. Untuk menjalankan *script* ini, klik *button play* pada bagian atas



Gambar 14.20 klik button play untuk menjalankan script

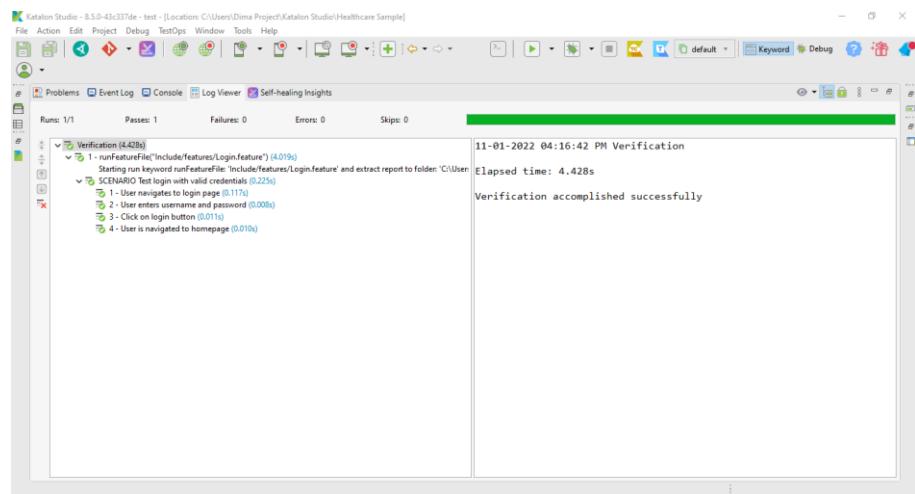
20. Tunggu hingga mendapatkan hasil-nya apakah dapat dijalankan atau terdapat eror. Hasil-nya adalah *script* ini berhasil dijalankan



Gambar 14.21 script berhasil dijalankan

21. Untuk melihat detail dari *script* yang telah dijalankan, klik *Log Viewer*.

Maka akan menampilkan detail dari setiap *step by step*



Gambar 14.22 klik log viewer untuk melihat detail script

22. Pada bagian *Console*, juga terdapat detail secara lengkap

```

Katalon Studio - 8.5.0-43c337de - test - [Location: C:\Users\Dimas Project\Katalon Studio\Healthcare Sample]
File Action Edit Project Debug TestOps Window Tools Help
# Problems Event Log Console Log Viewer Self-healing Insights
<terminated> Temp\TempCase166794189818 [Katalon] E:\Kuliah\Semester 7\Kualitas Perangkat Lunak\File installer\Katalon_Studio_PE_Windows_64-8.5.0\Katalon_Studio_PE_Windows_64-8.5.0\jet\bin\java.exe (Nov 1, 2022 4:16:31 PM -4:16:47 PM)
2022-11-01 16:16:42.683 INFO c.k.k.core.main.WSVerificationExecutor - -----
2022-11-01 16:16:42.693 INFO c.k.k.core.main.WSVerificationExecutor - START Verification
2022-11-01 16:16:43.164 DEBUG testcase.
2022-11-01 16:16:43.164 DEBUG c.k.k.core.main.WSVerificationExecutor - Starting run Keyword runFeatureFile: 'Include/features/Login.feature'
2022-11-01 16:16:43.164 DEBUG c.k.k.core.main.WSVerificationExecutor - -----
2022-11-01 16:16:46.515 INFO c.runtime.formatter.CucumberReporter - START SCENARIO Test login with valid credentials
2022-11-01 16:16:46.516 INFO c.runtime.formatter.CucumberReporter - STEP User navigates to login page
2022-11-01 16:16:46.524 DEBUG c.runtime.formatter.CucumberReporter - -----
I am inside navigateToLoginPage
2022-11-01 16:16:46.589 DEBUG c.runtime.formatter.CucumberReporter - STEP User navigates to login page
2022-11-01 16:16:46.688 DEBUG c.runtime.formatter.CucumberReporter - -----
I am inside Credentials
2022-11-01 16:16:46.687 DEBUG c.runtime.formatter.CucumberReporter - STEP User enters username and password
2022-11-01 16:16:46.694 DEBUG c.runtime.formatter.CucumberReporter - -----
I am inside clickLogin
2022-11-01 16:16:46.698 DEBUG c.runtime.formatter.CucumberReporter - STEP Click on login button
2022-11-01 16:16:46.709 DEBUG c.runtime.formatter.CucumberReporter - -----
I am inside home page
2022-11-01 16:16:46.712 DEBUG c.runtime.formatter.CucumberReporter - STEP User is navigated to homepage
2022-11-01 16:16:46.735 DEBUG c.runtime.formatter.CucumberReporter - -----
2022-11-01 16:16:46.740 INFO c.runtime.formatter.CucumberReporter - END SCENARIO Test login with valid credentials
1 Scenarios (0[32m passed[0m)
4 Steps (0[32m passed[0m)
0m1.317s

```

Gambar 14.23 lihat detail script pada bagian console

23. Selanjutnya pada bagian ini, kita akan membuat fitur untuk *enter username and password* untuk beberapa kasus pengujian. Untuk contohnya kita akan membuat dua *username* dan dua *password*. Untuk itu kita tambahkan “< >” pada bagian *username* dan *password*. Hasilnya akan muncul *warning* yang terhighlight oleh warna kuning. Hal ini dikarenakan kita telah mengubah bagian itu dan di file lainnya tidak kita ubah.

```

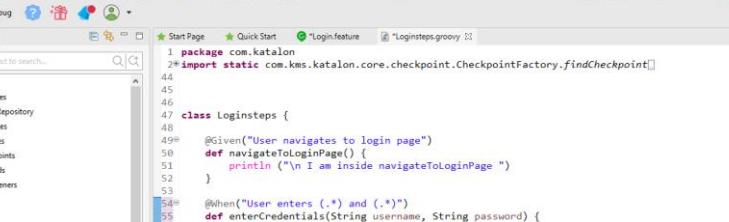
Katalon Studio - 8.5.0-43c337de - test - [Location: C:\Users\Dimas Project\Katalon Studio\Healthcare Sample]
File Action Edit Project Debug TestOps Window Tools Help
# Tests Explorer Start Page Quick Start "Login.feature" Loginssteps.groovy
1# Feature: Login feature
2#
3# Scenario Outline: Test login with valid credentials
4# Given User navigates to login page
5# When User enters <username> and <password>
6# And Click on login button
7# Then User is navigated to homepage
8#
9# Examples:
10| username | password |
11| ABC      | 123   |
12| DEF      | 456   |

```

Gambar 14.24 membuat fitur enter

24. Buka file *Loginsteps.groovy*, kemudian tambahkan script ““(User enters (.\*) and (.\*))” di line 54, kemudian tambahkan di “(String username, String password)” pada line 55 dan tambahkan script 57 dan 58

dikarenakan kita membuat 2 *username* dan 2 *password* atau ikuti *script* yang telah dibuat pada gambar dibawah.



Katalon Studio - B.5.0-42c37de - test - Location: C:\Users\Dimas Project\Katalon Studio\Healthcare Sample

File Action Edit Project Debug TestOps Window Tools Help

Keyword Debug

Tests Explorer

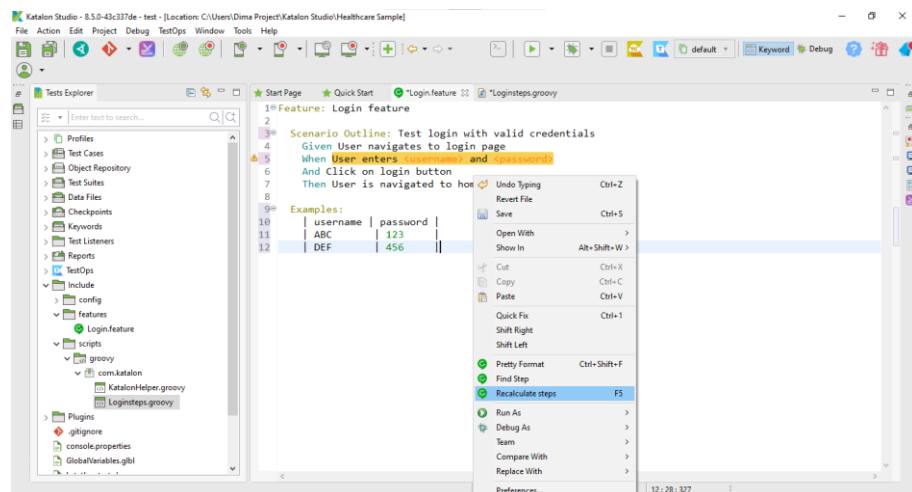
Enter text to search...

Start Page Quick Start "Login.feature" "Loginteststeps.groovy"

```
1 package com.katalon
2 import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
3
4
5
6
7 class Loginteststeps {
8
9     @Given("User navigates to login page")
10    def navigateToLoginPage() {
11        println ("\\n I am inside navigateLoginPage ")
12    }
13
14
15    @When("User enters (.*) and (.*)")
16    def enterCredentials(String username, String password) {
17        println ("\\n I am inside Credentials")
18        println [Username : "+username"]
19        println ("Password : "+password)
20    }
21
22
23    @And("Click on login button")
24    def clickLogin() {
25        println ("\\n I am inside clickLogin")
26    }
27
28
29    @Then("User is navigated to homepage")
30    def verifyHomePage() {
31        println ("\\n I am inside home page ")
32    }
33
34 }
```

Gambar 14.25 tambahkan script

25. Kemudian klik kanan, pilih *Recalculate steps* untuk mengkalkulasi Kembali *script* yang dibuat di *file* satu-nya



Gambar 14.26 recalculate script

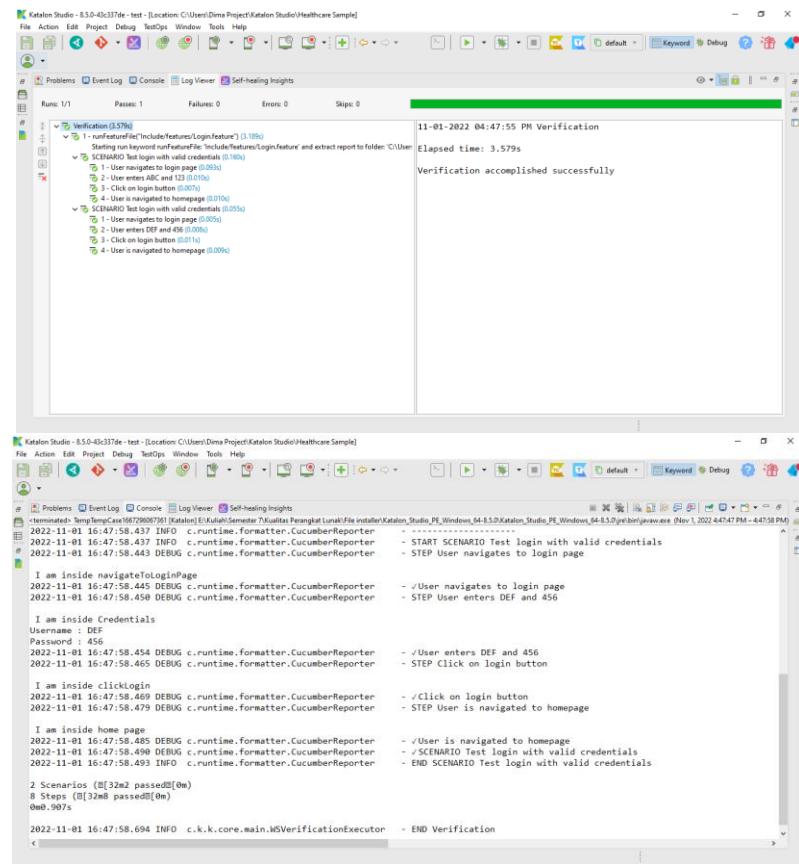
26. Maka hasil-nya berhasil dan tidak ada *warning*. Selanjut-nya jalankan Kembali *script* yang telah kita ubah

Gambar 14.27 hasil recalculate tidak ada warning

27. Tunggu hingga proses selesai dijalankan. Hasil-nya adalah *script* yang dibuat berhasil pada kasus pengujian dua *username* dan dua *password* ini

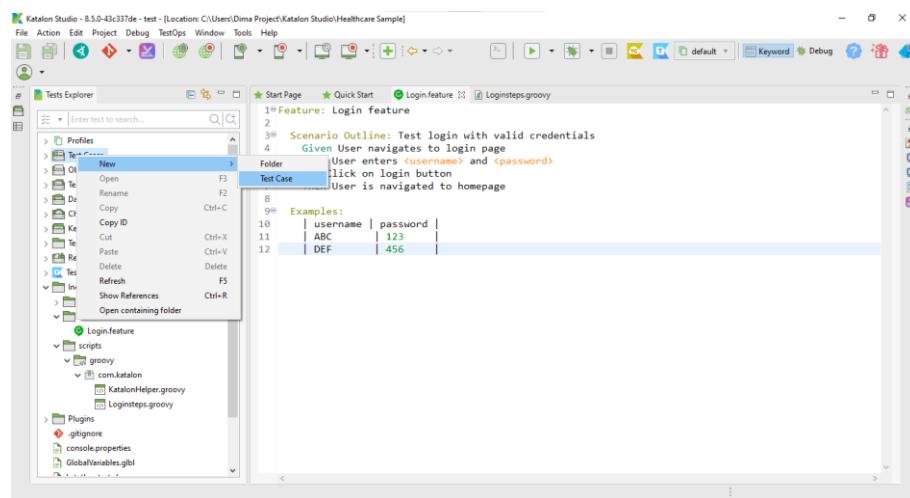
Gambar 14.28 script yang dibuat berhasil

28. Kita dapat melihat Kembali *Log viewer* dan *console* untuk melihat hasil *script* yang telah dijalankan



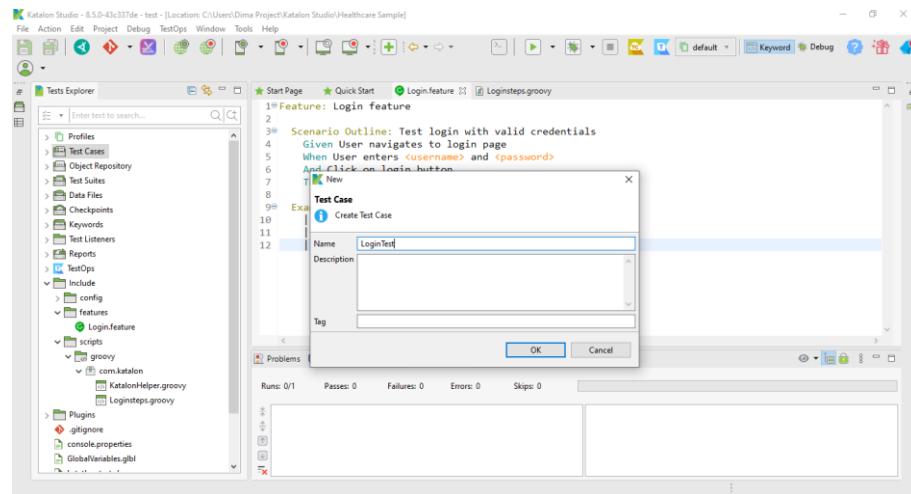
Gambar 14.29 melihat detail script dengan log viewer

29. Untuk menguji sebuah fitur dari sebuah *web* lain atau yang lainnya perlu untuk membuat *file Test Case*. Cara-nya adalah klik kanan pada *folder Test Cases*, pilih *New* lalu pilih *Test Case*



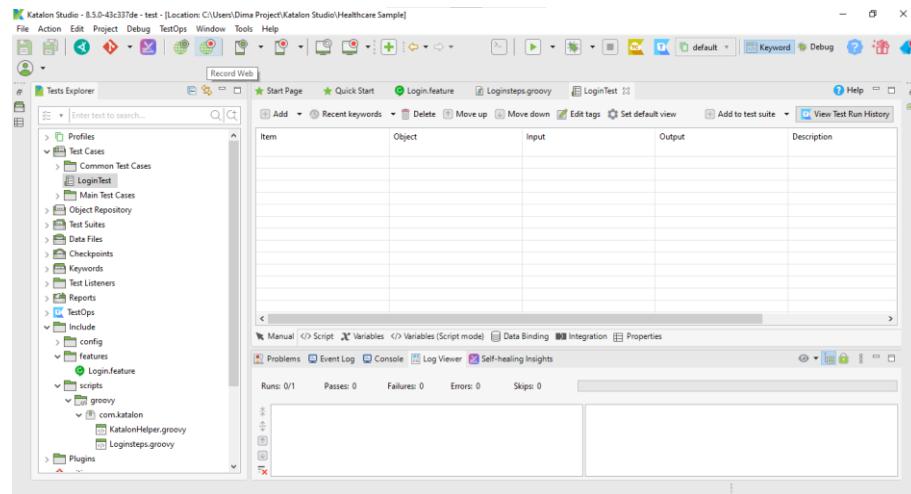
Gambar 14.30 pilih test case

30. Pada bagian ini, Buat nama-nya **LoginTest**. Lalu klik OK



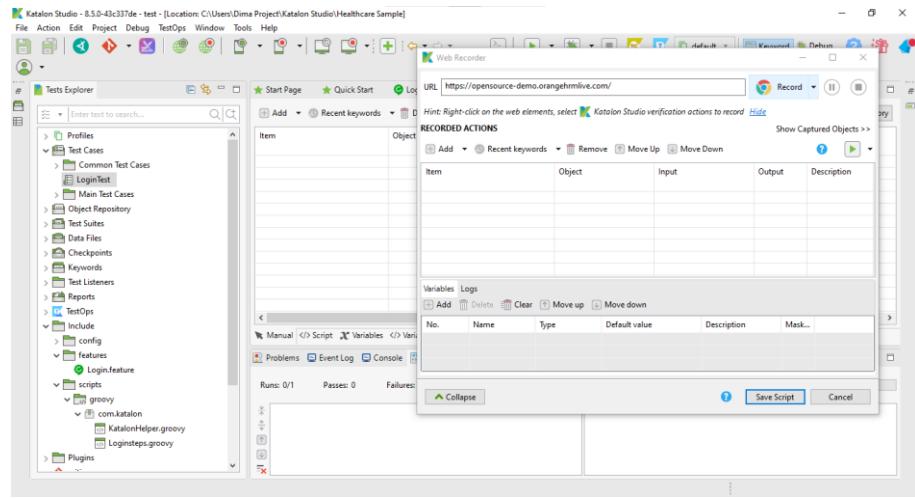
Gambar 14.31 isi nama test case yang ingin dibuat

31. Selanjutnya klik icon *Record Web*



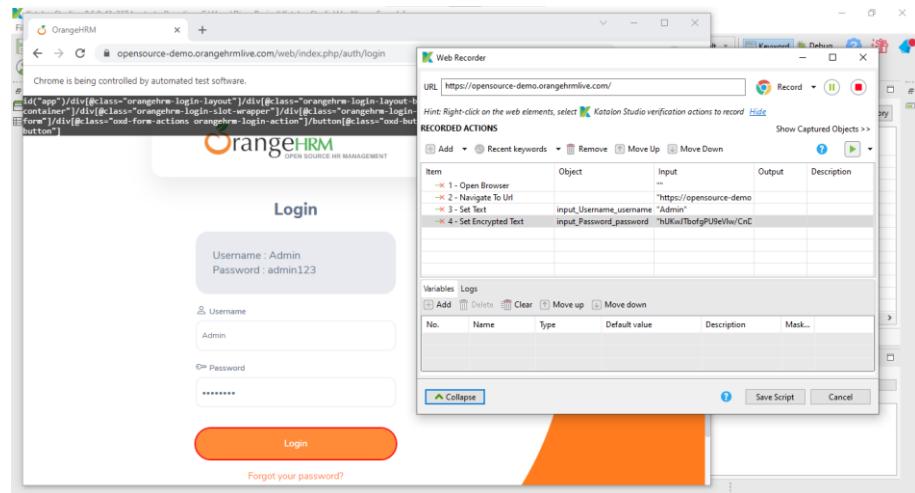
Gambar 14.32 klik icon record web

32. Pada bagian *Web recorder*, isikan URL yang akan kita uji. Kita akan mengisi URL dengan <https://opensource-demo.orangehrmlive.com/>, lalu gunakan web Chrome sebagai media pengujian-nya

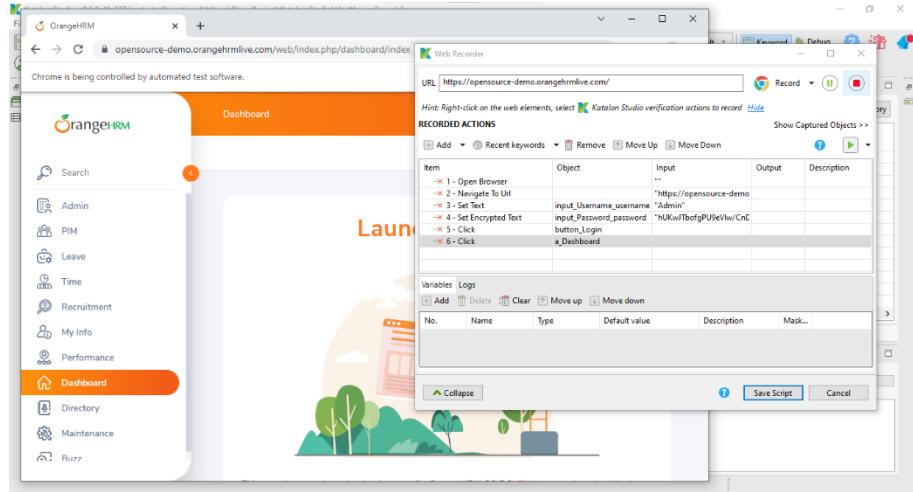


Gambar 14.33 isi URL yang akan diuji

33. Setelah itu web-nya telah dibuka, lalu klik *action* pada setiap web pengujian-nya, klik bagian kolom *username* dan *password* untuk pengujian ***enter username and password***, tombol *login* untuk pengujian ***login button*** dan tombol *Dashboard navigated to homepage*

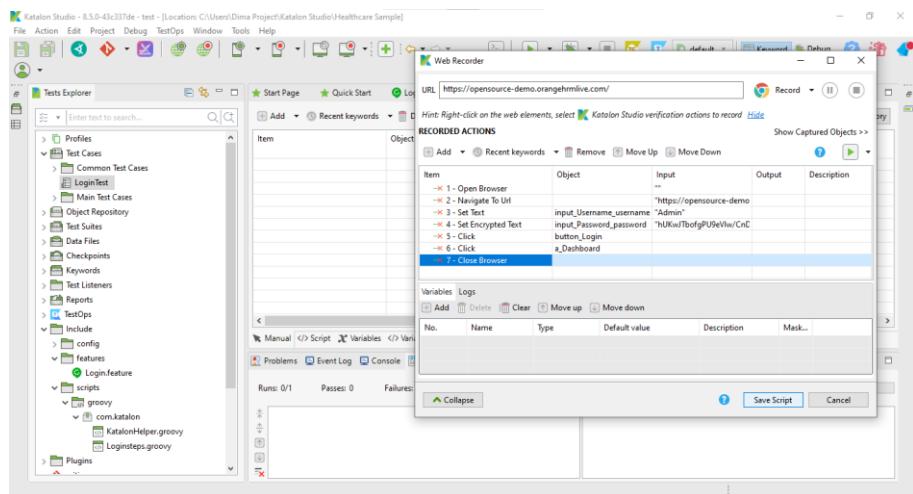


Gambar 14.34 klik action pada setiap pengujiannya



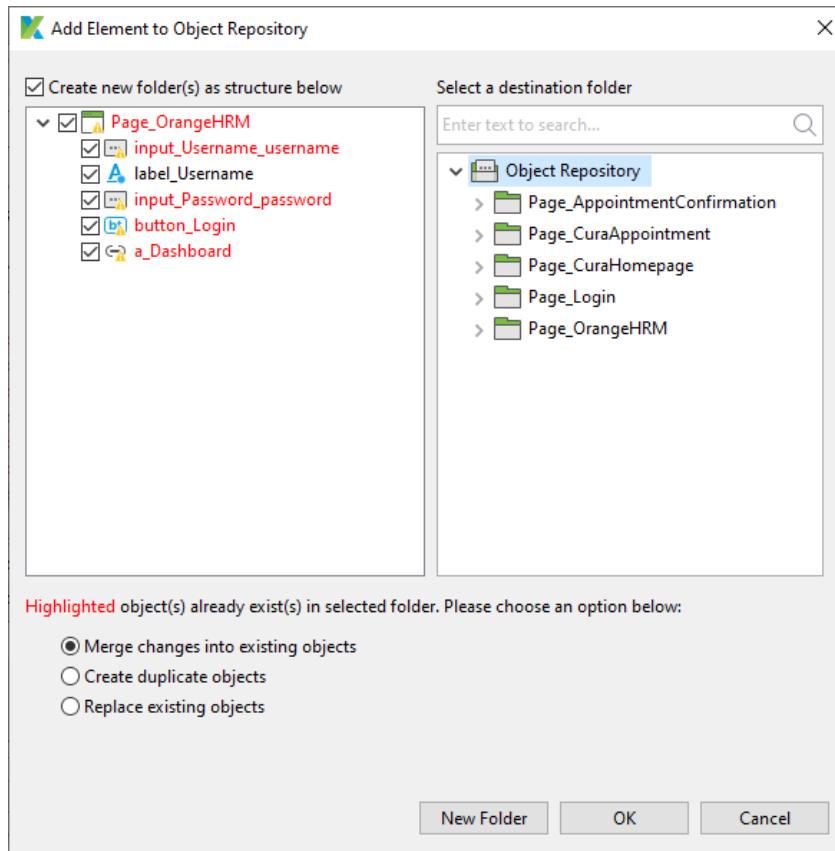
Gambar 14.35 save script

34. Setelah itu klik stop record untuk menghentikan pengujian-nya



Gambar 14.36 klik stop record

35. Selanjutnya klik OK untuk membuat folder baru untuk menyimpan file yang telah kita uji



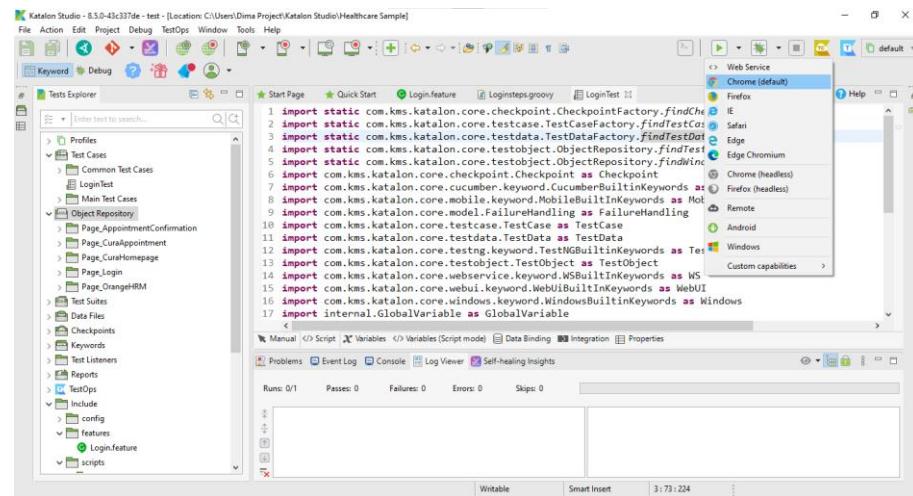
Gambar 14.37 klik OK untuk membuat folder baru penyimpanan file

36. Pada tampilan ini, kita akan mengubah tabel nomor 6 dimulai dari *Item* yang awalnya adalah Click kita ubah menjadi *Verify Element Present*, lalu untuk *Input* yang awalnya kosong kita tambah menjadi 5 seperti pada gambar dibawah

Item	Object	Input	Output	Description
1 - Open Browser		...		
2 - Navigate To UI				
3 - Set Text	input_Username_username	"Admin"		
4 - Set Encrypted Text	input_Password_password	"hJKw!TbofgPUjeVlw/CnDQ=="		
5 - Click	button_Login			
<b>6 - Verify Element Present</b>	<b>a_Dashboard</b>	<b>5</b>		
7 - Close Browser				

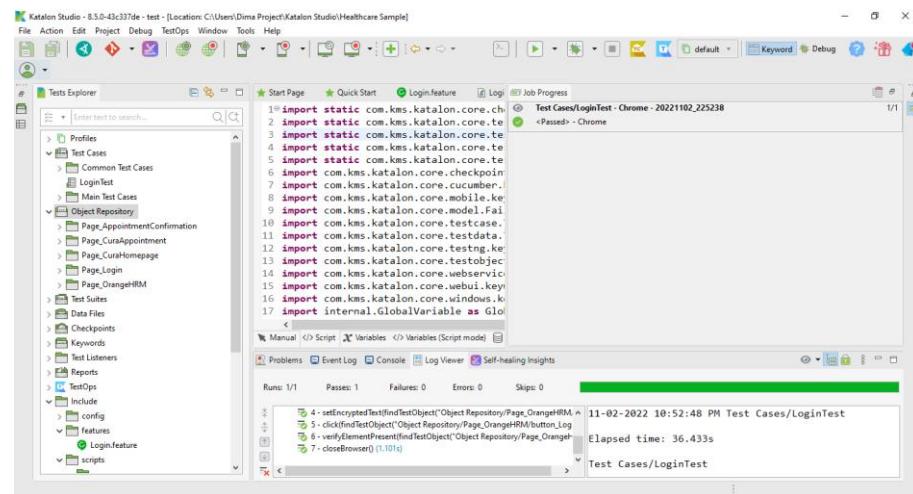
Gambar 14.38 klik ubah menjadi verify element

37. Buka Tab Script di bagian bawah. Kemudian jalankan lagi dengan Chrome sebagai default browser-nya



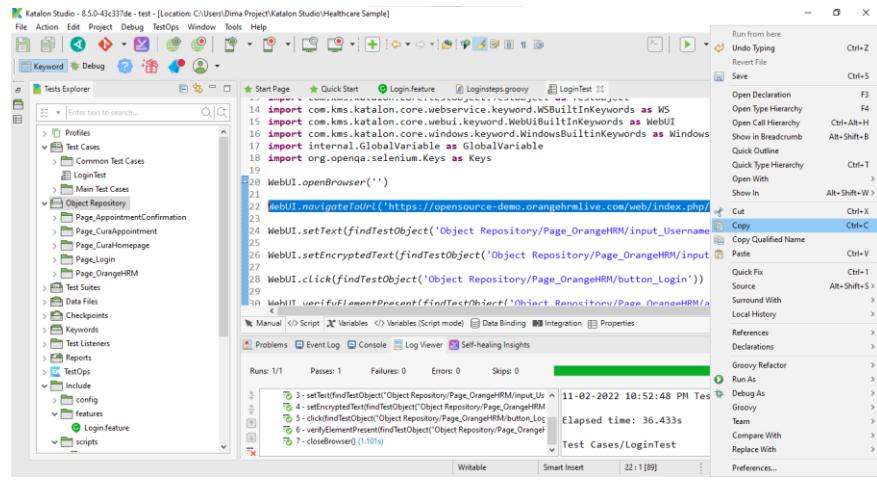
Gambar 14.39 buka tab script

38. Maka hasil-nya adalah file pengujian berhasil dijalankan



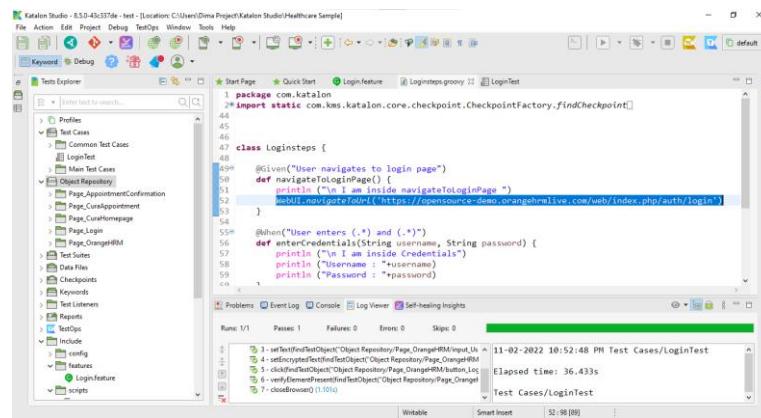
Gambar 14.40 pengujian yang berhasil dijalankan

39. Selanjutnya pada line 22, kita salin script ini di file **LoginTest**



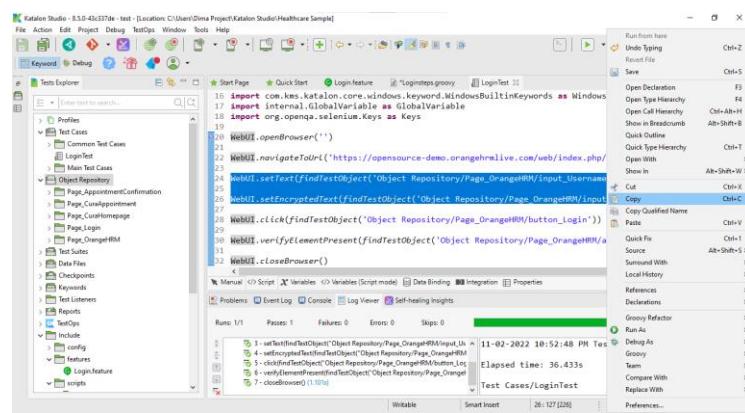
Gambar 14.41 salin script file LoginTest

40. Kemudian *Paste* kan script ini ke *line 52* pada file **Loginsteps.groovy** pada bagian ***navigate to login page***

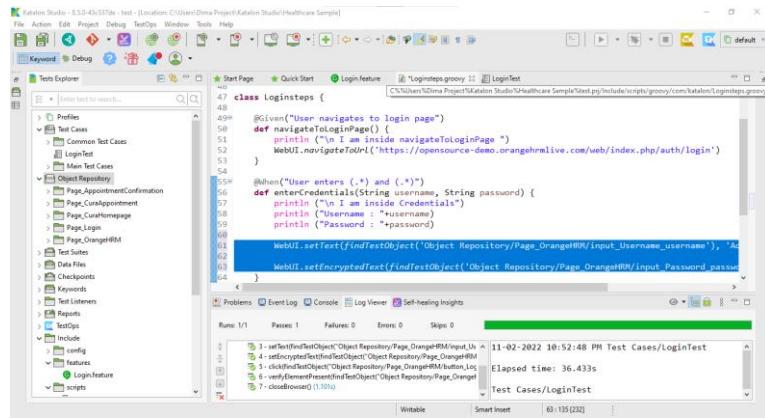


Gambar 14.42 pastekan script pada bagian navigate to login page

41. Kemudian salin *script* pada *line* 24 dan 25 dan lakukan hal sama pada langkah nomor 39 untuk bagian ***enter username and password***

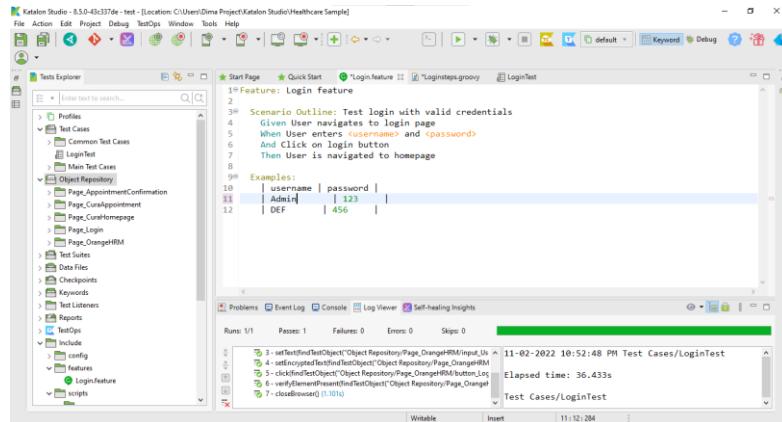


Gambar 14.43 salin script



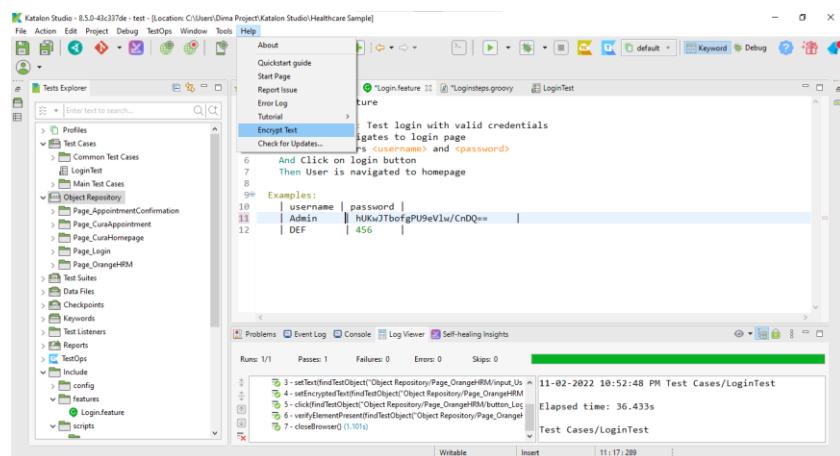
Gambar 14.44 enter username dan password

42. Kemudian pada *line 11* pada file **Login.feature**, kita ubah *username* dan *password* sesuai dengan *web* pengujian yang telah dilakukan



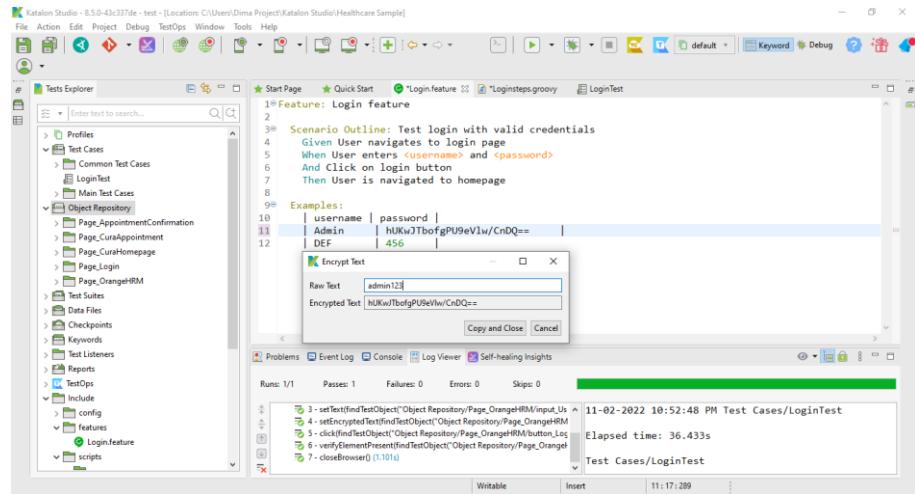
Gambar 14.45 ubah username dan password

43. Untuk mengubah teks *password* biasa menjadi *password* yang terenkripsi, dengan cara klik *help* pada *tab menu*, kemudian pilih *Encrypt Text*



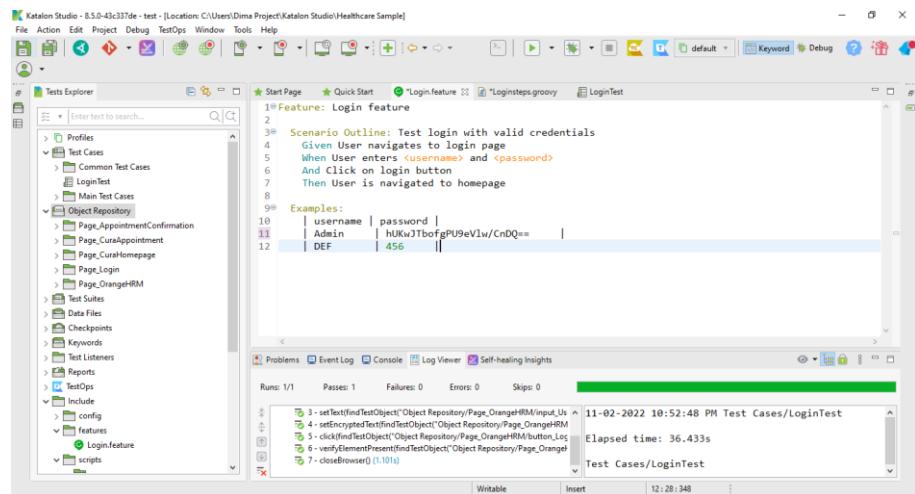
Gambar 14.46 mengubah text password biasa menjadi password enkripsi

44. Untuk bagian ini pada *Raw Text*, masukan password yang sesuai sama web pengujian, lalu salin pada bagian *Encrypted Text* dengan mengklik *button Copy and Close*



Gambar 14.47 masukkan password yang sesuai dengan web pengujian

45. Lalu *paste* kan pada bagian *password*



Gambar 14.48 paste pada bagian *password*

46. Kemudian salin *script* pada *line 28* di file *LoginTest*

```

16 import com.kms.katalon.core.windows.keyword.WindowsBuiltinKeywords as Windows
17 import internal.GlobalVariable as GlobalVariable
18 import org.openqa.selenium.Keys as Keys
19
20 WebUI.openBrowser('')
21
22 WebUI.navigateToUrl("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")
23
24 WebUI.setText(findTestObject('Object Repository/Page_OrangeHRM/input_Username'), "Admin")
25
26 WebUI.setEncryptedText(findTestObject('Object Repository/Page_OrangeHRM/input_Password'), "e10adc37adbe938a18d4c94f3a68495")
27
28 WebUI.click(findTestObject('Object Repository/Page_OrangeHRM/button_Login'))
29
30 WebUI.verifyElementPresent(findTestObject('Object Repository/Page_OrangeHRM/div_Error'), 3)
31
32 WebUI.closeBrowser()

```

Gambar 14.49 salin script dari file login test

47. lalu *paste* kan script ini ke *line 69* di file ***Loginsteps.groovy*** pada bagian ***login button***

```

16 println('Username : ' + username)
17 println('Password : ' + password)
18
19 WebUI.setText(findTestObject('Object Repository/Page_OrangeHRM/input_Username_username'), username)
20
21 WebUI.setEncryptedText(findTestObject('Object Repository/Page_OrangeHRM/input_Password_password'), password)
22
23 @And("Click on login button")
24 def clickLogin() {
25     println("I am inside clickLogin")
26     WebUI.click(findTestObject('Object Repository/Page_OrangeHRM/button_Login'))
27 }
28
29 @Then("User is navigated to homepage")
30 def verifyHomePage() {
31     println("\nI am inside home page")
32 }
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76

```

Gambar 14.50 paste kan script pada bagian login button

48. Kemudian salin script pada *line 30* dan *32* di file ***LoginTest***

```

import
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34

```

Copy Qualified Name

Paste

Quick Fix

Source

Surround With

Local History

References

Declarations

Groovy Refactor

Run As

Ignore

Debug As

Groovy

Team

Compare With

Replace With

Preferences...

Gambar 14.51 salin script dari file LoginTest

49. Kemudian *Paste* kan script ini ke *line 52* pada file **Loginsteps.groovy** pada bagian **home page**

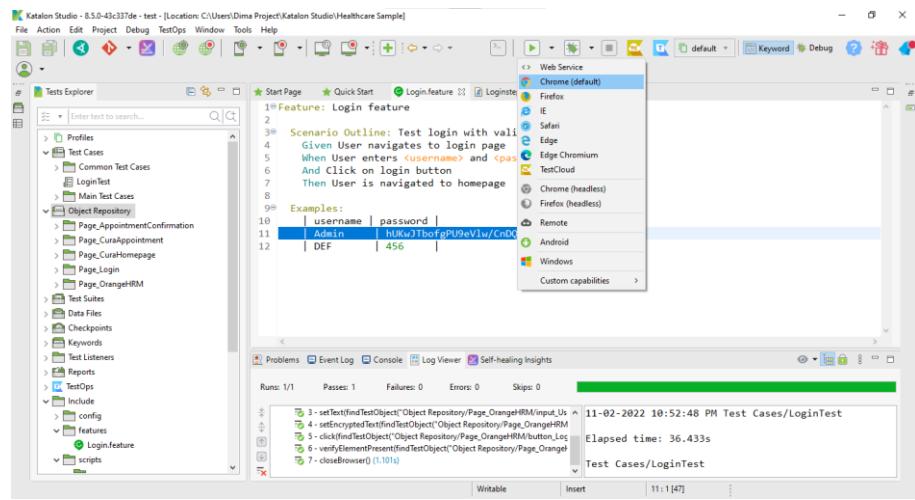
```

princin("password : " +password)
WebUI.setText(findTestObject('Object Repository/Page_OrangeHRM/input_Username_username'), use
WebUI.setEncryptedText(findTestObject('Object Repository/Page_OrangeHRM/input_Password_password'))
@And("Click on login button")
def clickLogin() {
    println ("\n I am inside clickLogin")
    WebUI.click(findTestObject('Object Repository/Page_OrangeHRM/button_Login'))
}
@Then("User is navigated to homepage")
def verifyHomePage() {
    println ("\n I am inside home page")
    WebUI.verifyElementPresent(findTestObject('Object Repository/Page_OrangeHRM/a_Dashboard'), 5)
    WebUI.closeBrowser()
}

```

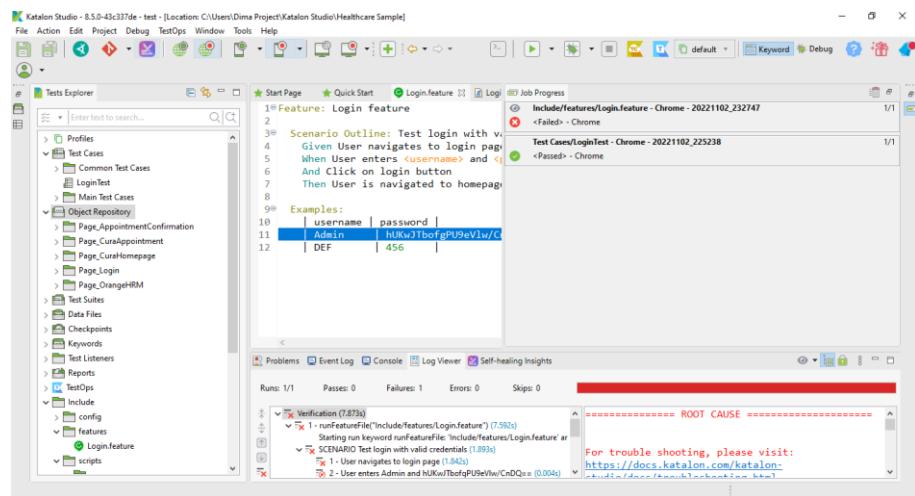
Gambar 14.52 pastekan script pada bagian home page

50. Kemudian jalankan file **Login.feature** dengan menggunakan Chrome



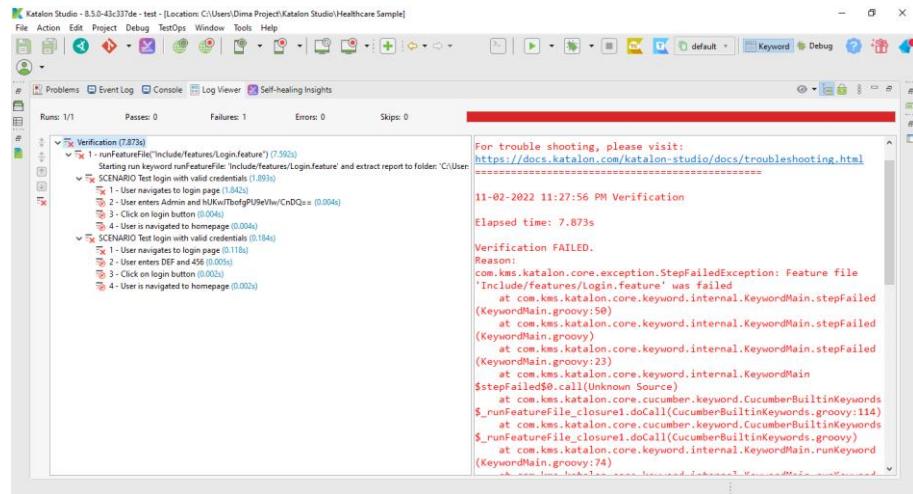
Gambar 14.53 jalankan file login fitur dengan chrome

## 51. Hasilnya adalah *file* ini mengalami eror

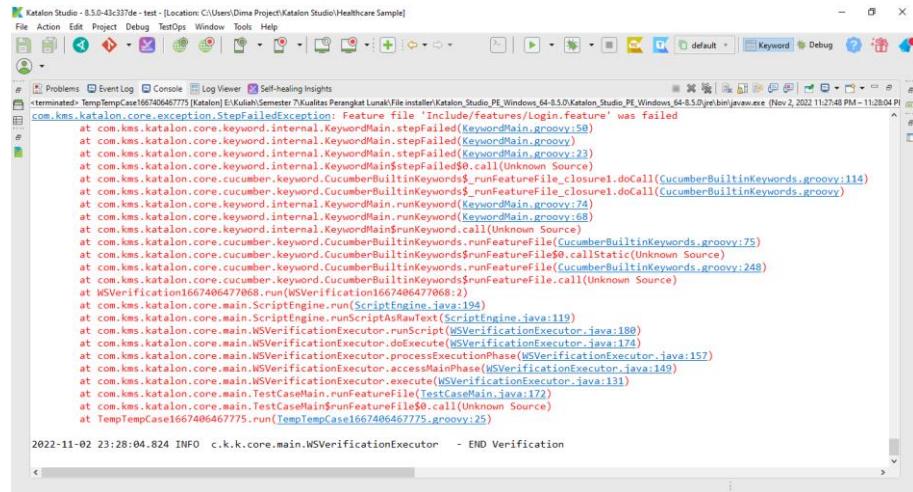


Gambar 14.54 hasil setelah dijalankan

52. Kita dapat melihat eror di bagian *Log Viewer* dan kita dapat mengetahui bahwa penyebab eror adalah kita tidak menambahkan satu *script* lagi. Jadi kita Kembali lagi *file LoginTest*

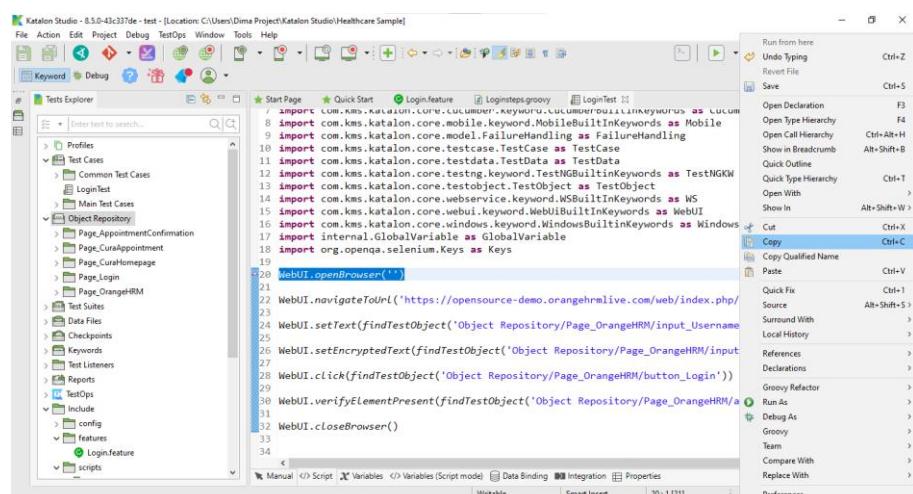


Gambar 14.55 lihat detail error pada log viewer



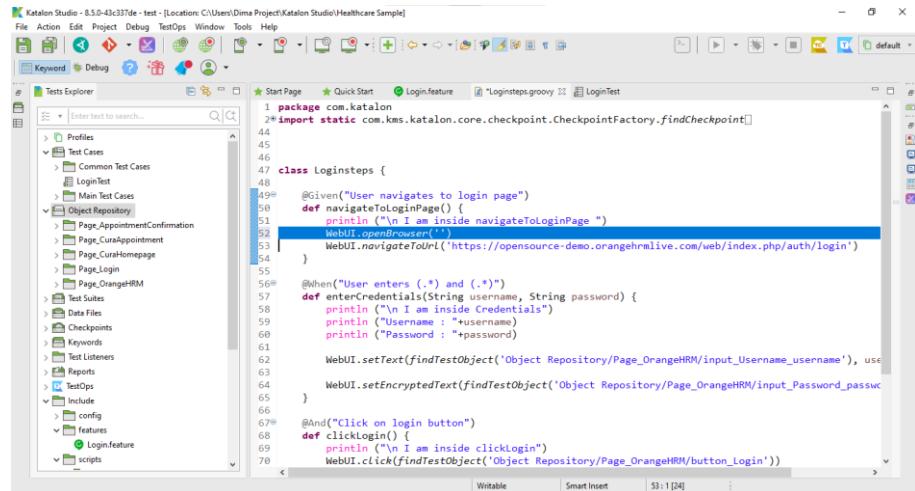
Gambar 14.56 error ditemukan

53. Pada bagian ini, kita salin *script* pada *line 20*



Gambar 14.57 kembali ke file LoginTest

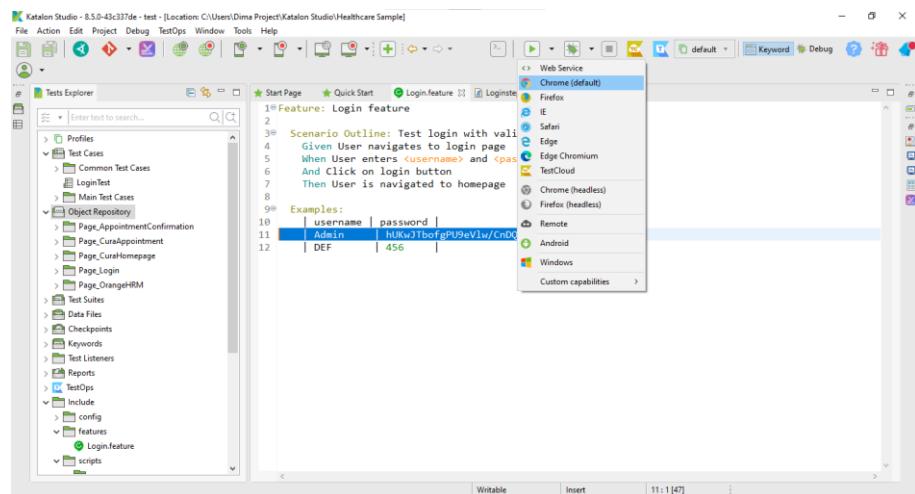
54. Kemudian paste kan script ini ke *line 52* pada file *Loginsteps.groovy*



```
package com.katalon
import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
class Loginsteps {
    @Given("User navigates to login page")
    def navigateToLoginPage() {
        println ("\\n I am inside navigateToLoginPage ")
        WebUI.openBrowser('')
        WebUI.navigateToUrl('https://opensource-demo.orangehrmlive.com/web/index.php/auth/login')
    }
    @When("User enters (.*) and (.*)")
    def enterCredentials(String username, String password) {
        println ("\\n I am inside Credentials")
        println ("Username : "+username)
        println ("Password : "+password)
        WebUI.setText(findTestObject('Object Repository/Page_OrangeHRM/input_Username_username'), user)
        WebUI.setEncryptedText(findTestObject('Object Repository/Page_OrangeHRM/input_Password_password'), pass)
    }
    @And("Click on login button")
    def clickLogin() {
        println ("\\n I am inside clickLogin")
        WebUI.click(findTestObject('Object Repository/Page_OrangeHRM/button_Login'))
    }
}
```

Gambar 14.58 pastekan script yang kurang pada file LoginTest

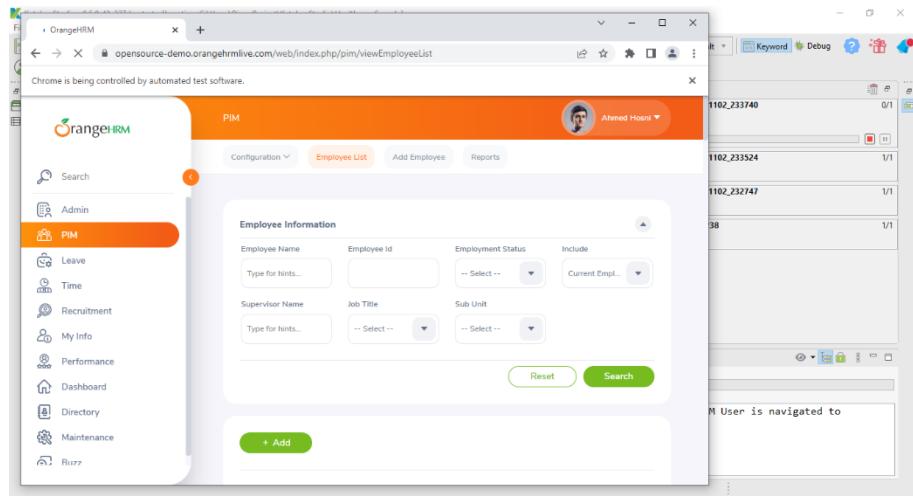
55. Lalu jalankan lagi file tersebut



The screenshot shows the Katalon Studio interface with the code editor open. A dropdown menu is visible in the top right corner, titled "Web Service". It lists various browser options: Chrome (default), Firefox, IE, Safari, Edge, Edge Chromium, TestCloud, Chrome (headless), Firefox (headless), Remote, Android, Windows, and Custom capabilities.

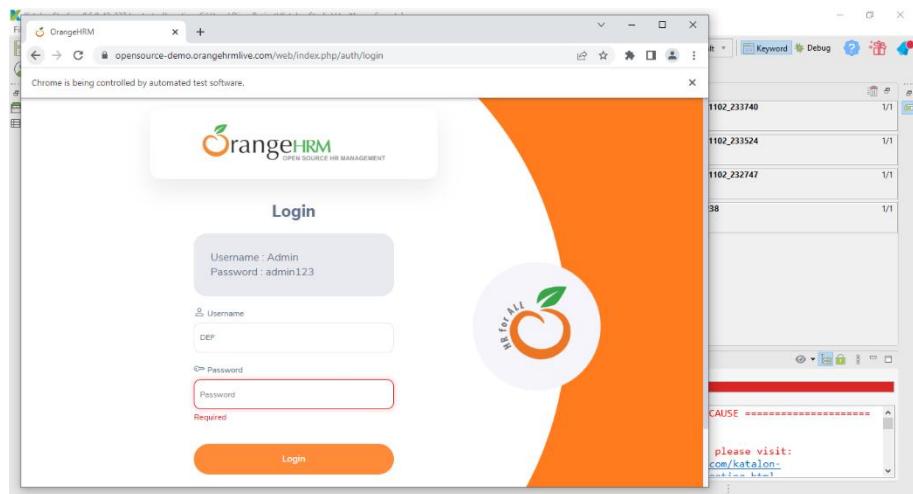
Gambar 14.59 jalankan lagi file tersebut

56. Hasil-nya adalah untuk *run* pada *scenario* pertama berhasil dengan *username* dan *password* dengan benar dan dapat masuk ke menu *dashboard*



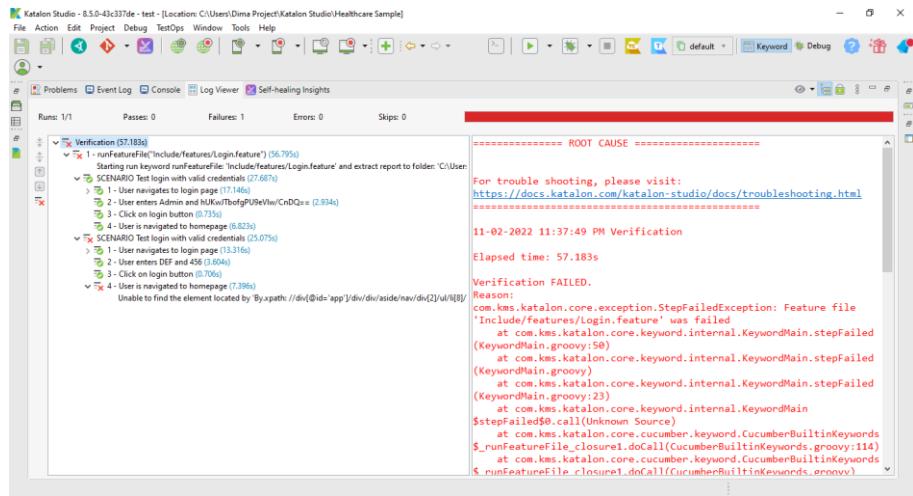
Gambar 14.60 hasil saat dijalankan

57. Untuk *run* pada *scenario* kedua akan mengalami eror dikarenakan *username* dan *password* yang dimasukan salah dan tidak dapat melanjutkan masuk ke website tersebut



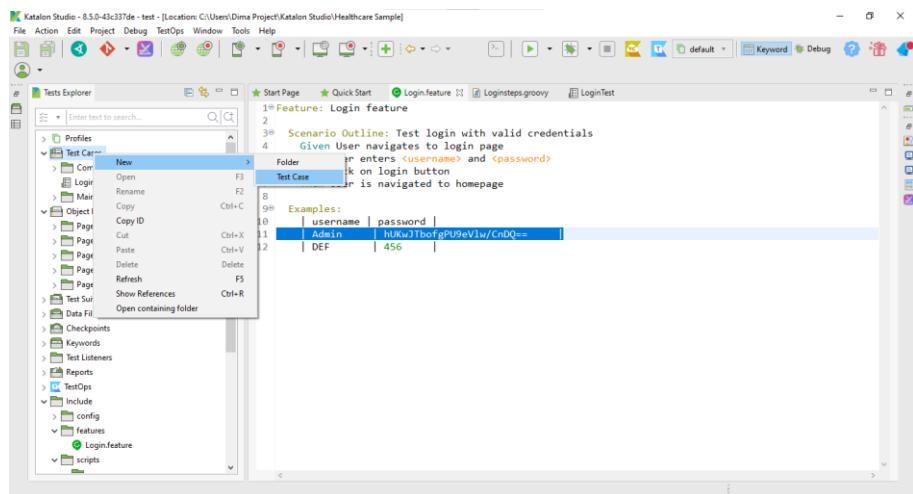
Gambar 14.61 Terjadi error pada skenario kedua

58. Pada bagian *Log Viewer* kita dapat melihat penyebab eror pada *scenario* kedua



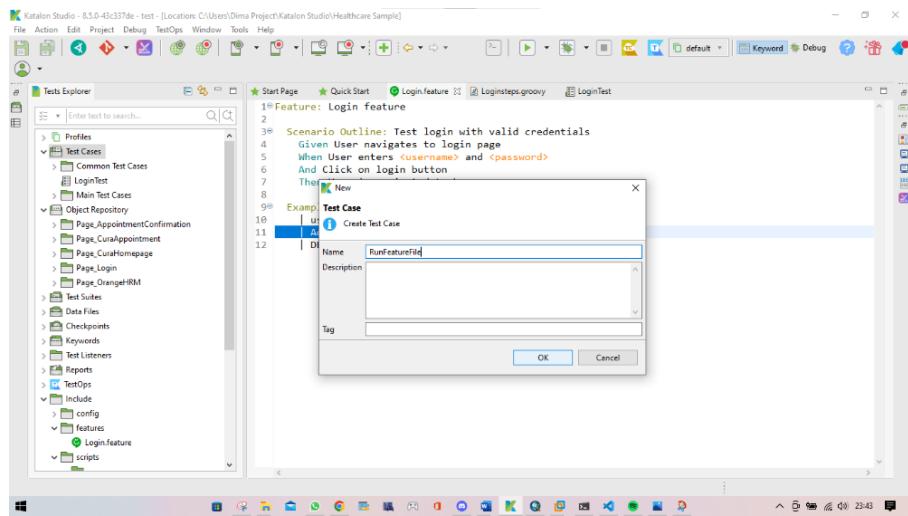
Gambar 14.62 lihat skenario error pada log viewer

59. Sekarang kita akan mencoba membuat *Test Case* yang baru dengan cara pilih klik kanan pada *folder Test case*, pilih *New* dan pilih *Test Case*



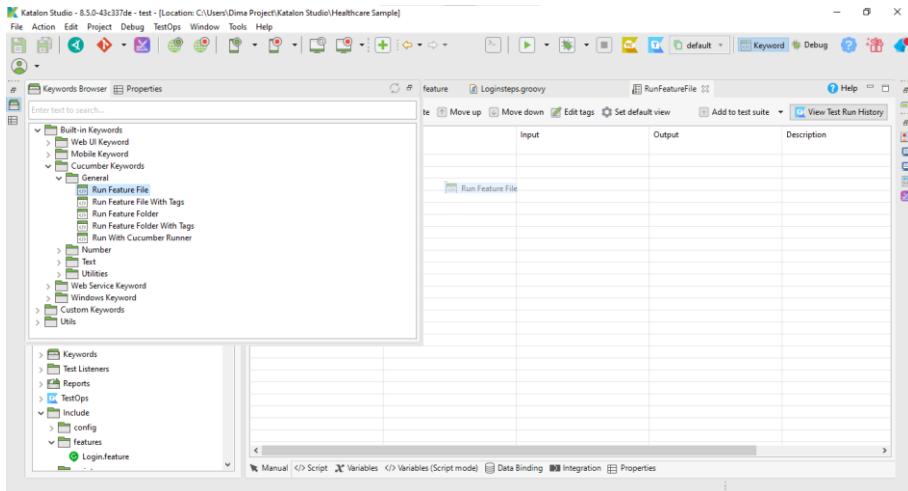
Gambar 14.63 buat test case baru pada folder Test Case

60. Buat dengan nama ***RunFeatureFile*** untuk file *Test Case* baru. Kemudian klik OK

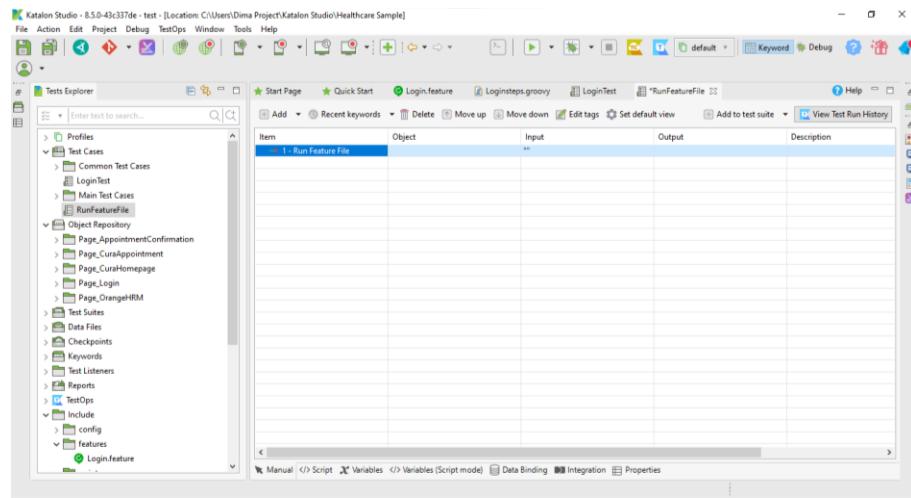


Gambar 14.64 beri nama file test case baru

61. Selanjutnya klik *Keyword Browser* yang terdapat di pojok kiri. Pilih *folder Cucumber Keyword\General* kemudian drag *Run Feature File* kedalam file *RunFeatureFile*

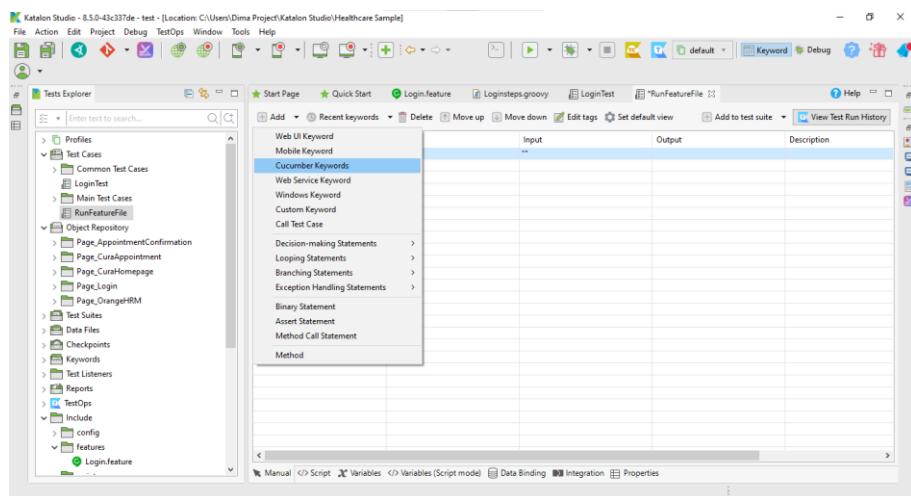


Gambar 14.65 klik keyword browser



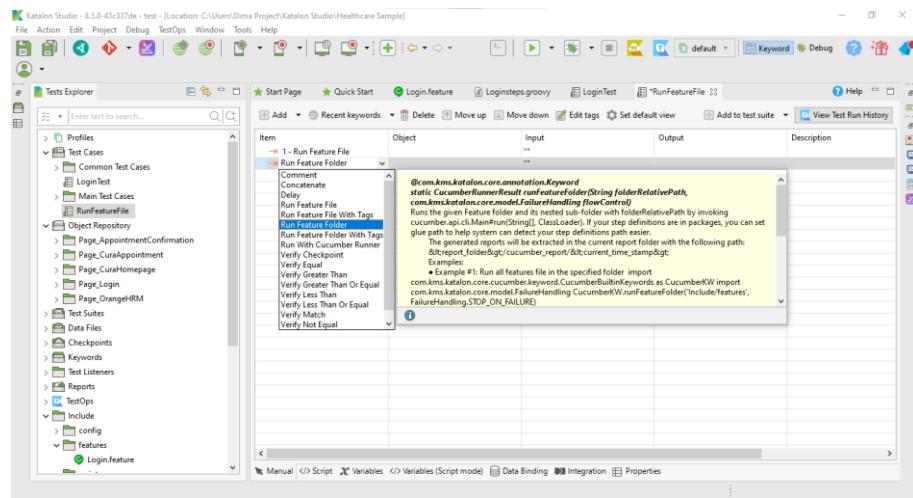
Gambar 14.66 klik RunFeaturFile

62. Untuk menambah item baru pada file **RunFeatureFile** kita tambahkan *item* baru dengan cara klik *Add*, pilih *Cucumber Keywords*



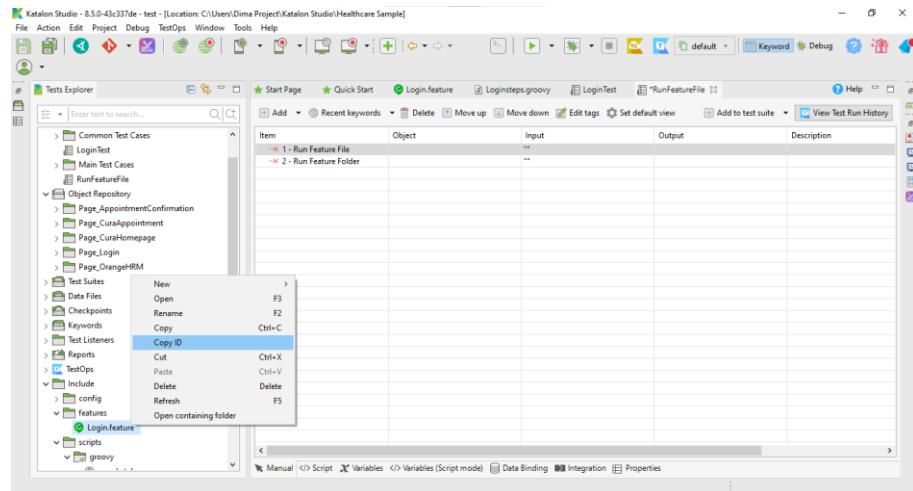
Gambar 14.67 tambahkan item dengan klik cucumber keyword

63. Pada bagian kolom *item*, kita beri nama *item* pada kolom tabel nomor 2-nya adalah *Run Feature Folder*



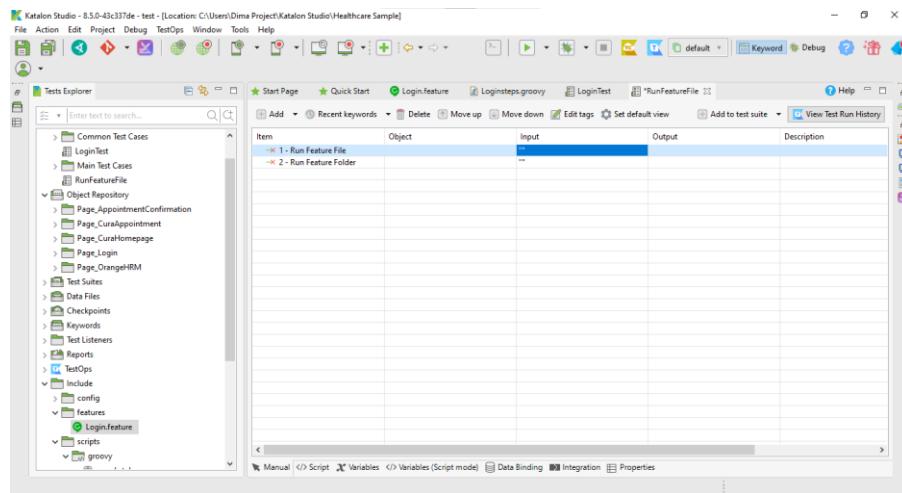
Gambar 14.68 beri nama item yang ditambahkan

64. Kemudian kita salin ID dari *item* pada kolom tabel nomor 1 dengan cara klik kanan *Run Feature File*, kemudian pilih *Copy ID*

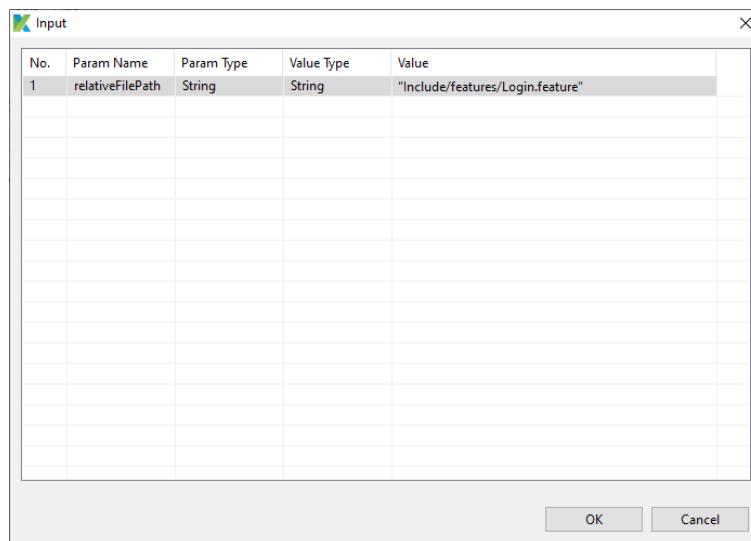


Gambar 14.69 salin ID item

65. Pada kolom *Input*, masukan ID yang telah kita salin, lalu *paste* kan kedalam bagian *value*. Kemudian klik OK untuk menutup jendela ini

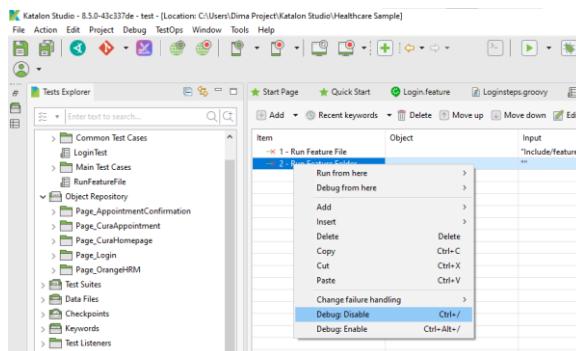


Gambar 14.70 masukkan ID pada bagian value



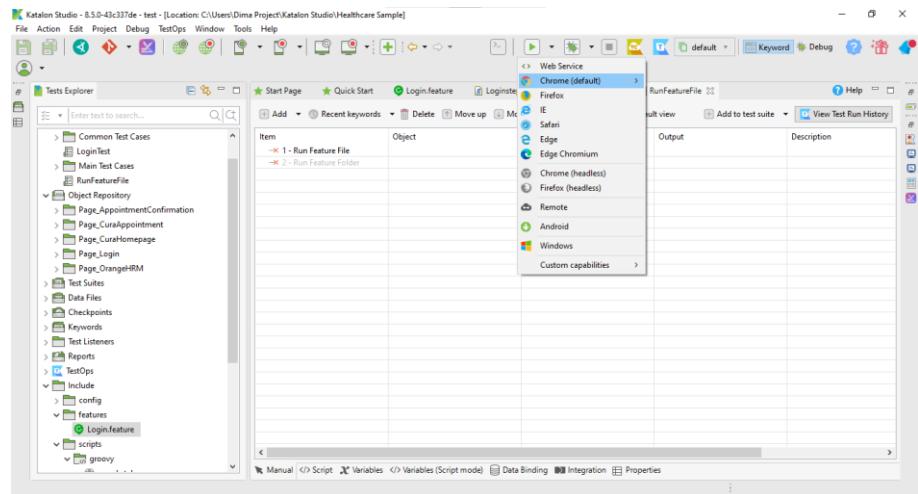
Gambar 14.71 id yang telah ditambahkan

66. Kemudian kita akan men *disable item* nomor 2 dengan cara klik *item* nomor 2, lalu pilih *Debug: Disable*. Maka yang akan jalan hanyalah *item* nomor 1



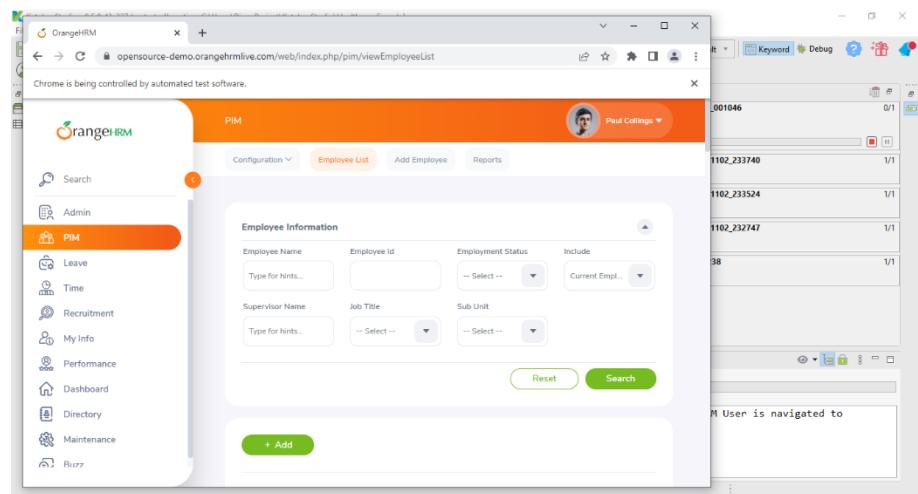
Gambar 14.72 disable kan debug file

67. Selanjutnya *run file* nya dengan Chrome



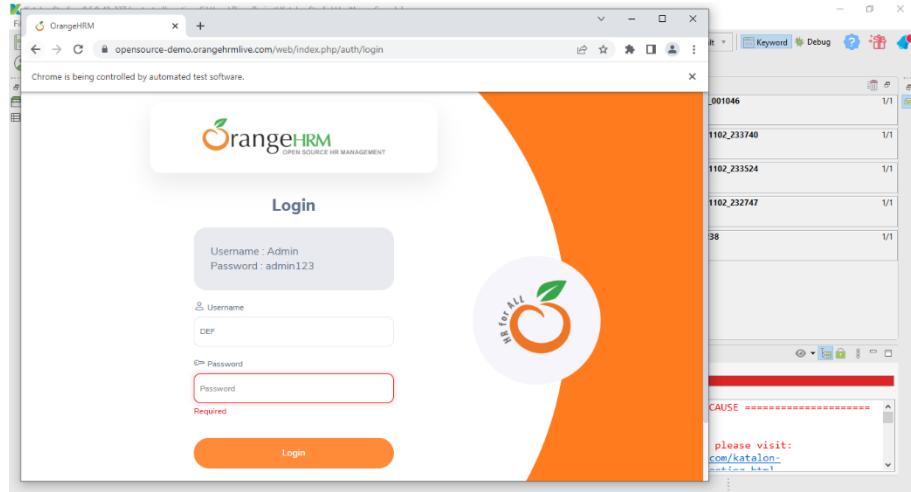
Gambar 14.73 jalankan file dengan chrome

68. Hasil-nya adalah pada *scenario* pertama berhasil dijalankan



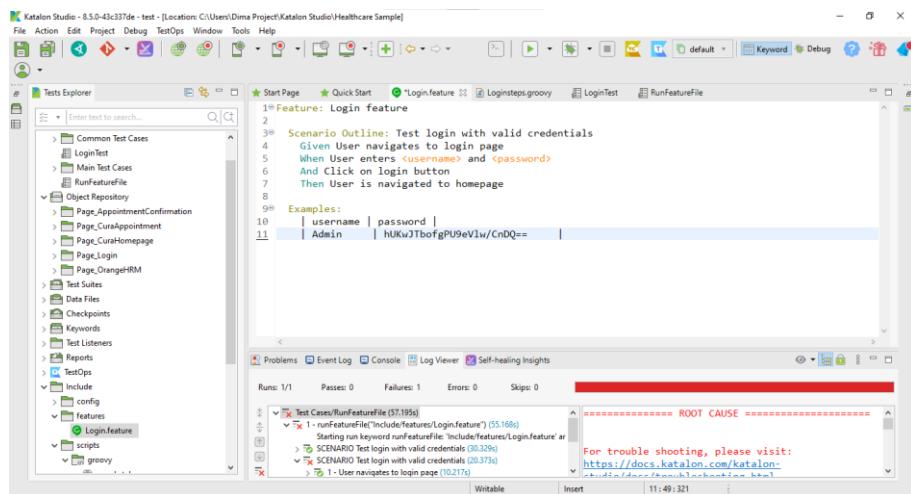
Gambar 14.74 skenario pertama berhasil dijalankan

69. Ketika menjalankan *scenario* kedua, hasil-nya adalah eror karena *username* dan *password* kedua tidak terdapat di web pengujian-nya



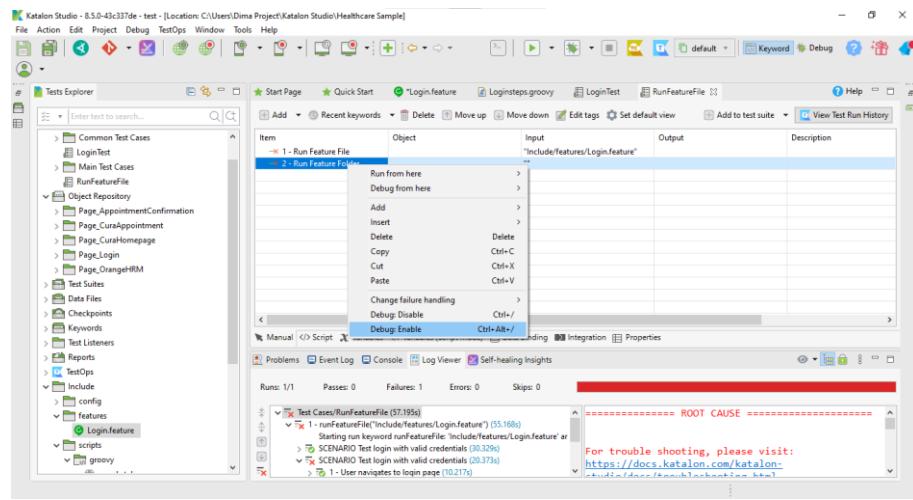
Gambar 14.75 skenario kedua error

70. Kembali lagi ke file **Login.feature**, hapus script di line 12 dan hanya menyisakan satu script, yaitu script pada line 11



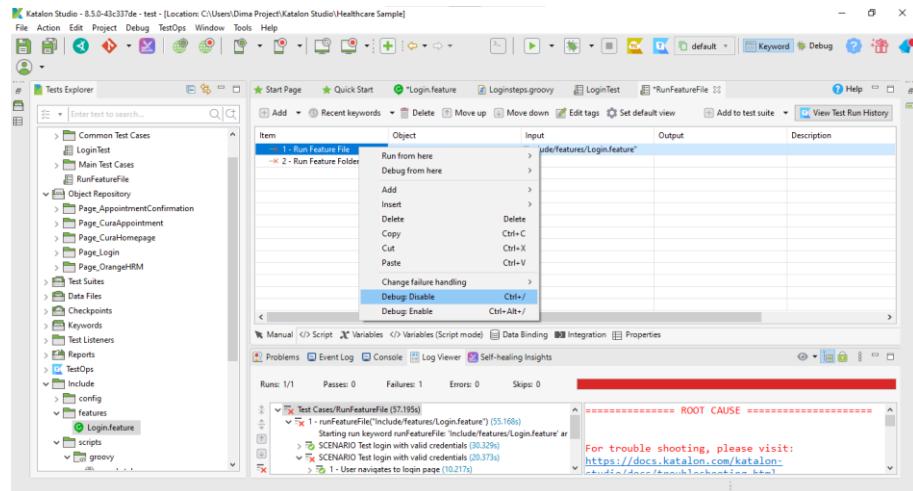
Gambar 14.76 kembali ke Login Feature

71. Kemudian kita enable item kedua dengan cara klik kanan pada item nomor 2, lalu pilih *Debug: Enable*



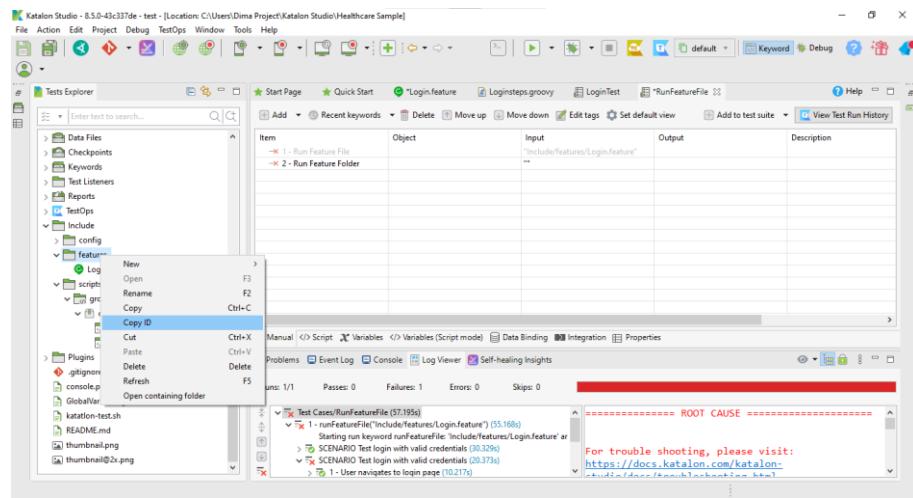
Gambar 14.77 pada item kedua enable-kan debug

72. Selanjutnya kita *disable* item nomor 1 dengan cara klik *item* nomor 1, lalu pilih *Debug: Disable*. Maka yang akan jalan hanyalah *item* nomor 2



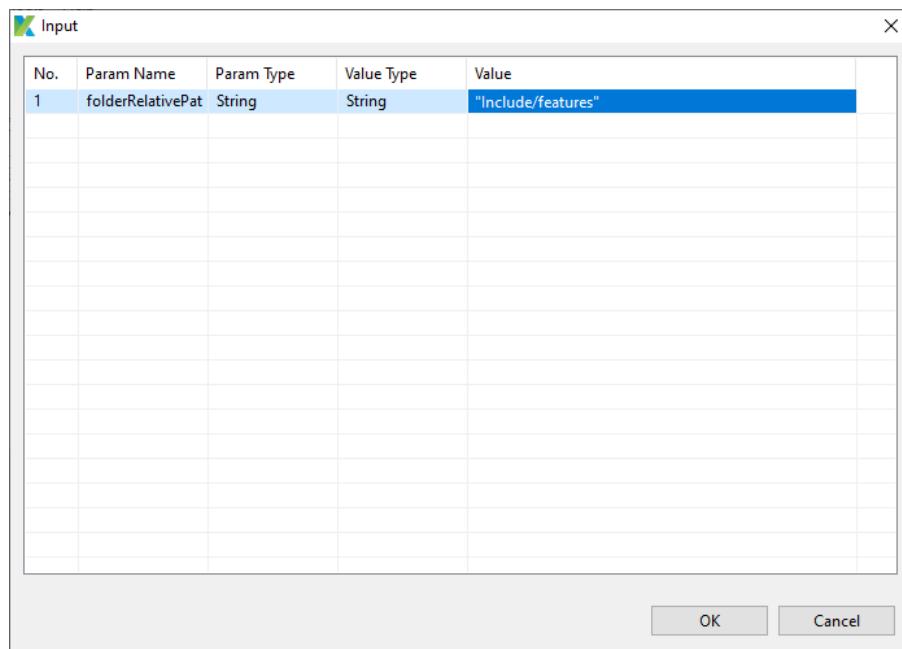
Gambar 14.78 disable kan item nomor 1

73. Kemudian kita salin ID dari *item* pada kolom tabel nomor 2 dengan cara klik kanan *Run Feature Folder*, kemudian pilih *Copy ID*



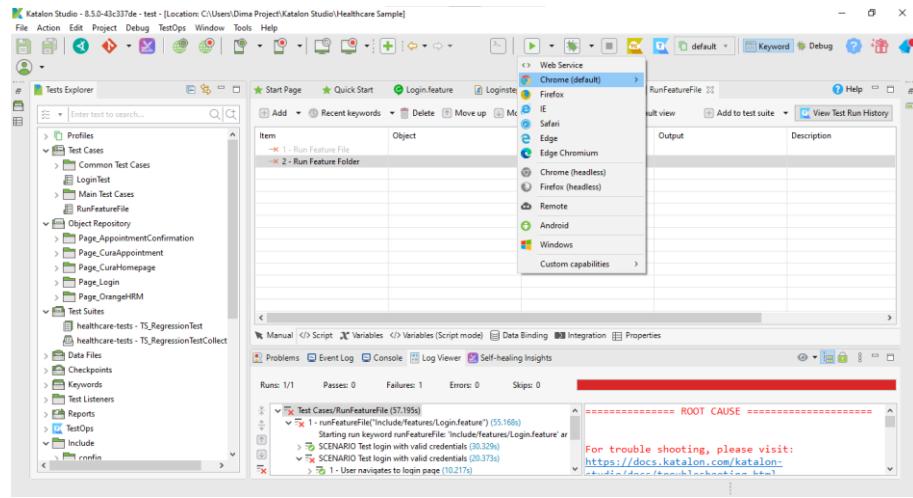
Gambar 14.79 salin ID

74. Pada kolom *Input*, masukan ID yang telah kita salin, lalu paste kan kedalam bagian value. Kemudian klik OK untuk menutup jendela ini



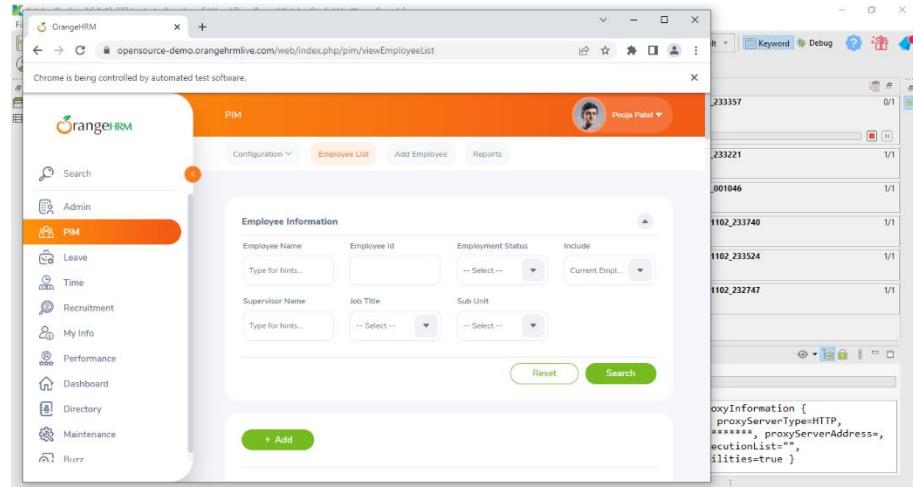
Gambar 14.80 salin pada kolom paste

75. Selanjutnya *run file* nya lagi dengan chrome



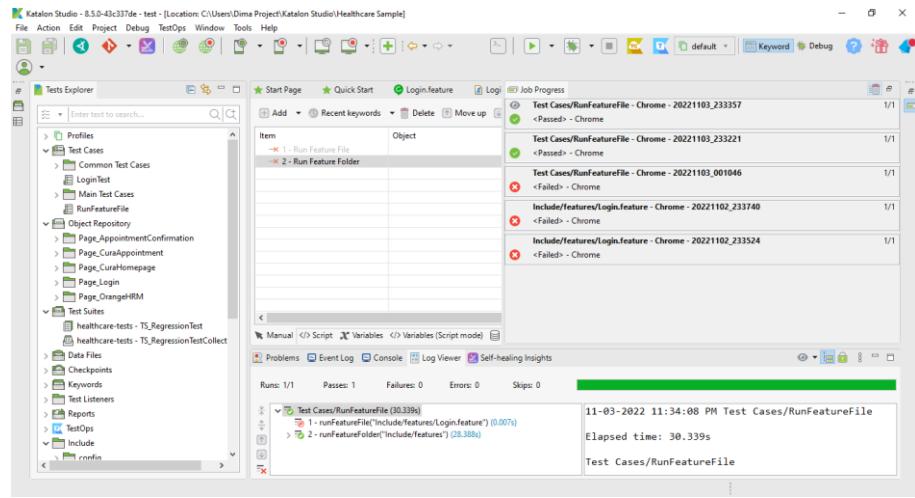
Gambar 14.81 jalankan file dengan chrome

76. Hasilnya adalah pada *scenario* pertama berhasil dijalankan. Hal ini dikarenakan pada *web* pengujian yang kita gunakan hanya mempunyai satu *username* dan *password*. Jadi lebih dari 1 *username* dan *password* akan mengalami eror, untuk itu kita hapus satu *username* dan *password* pada *file Login.feature*



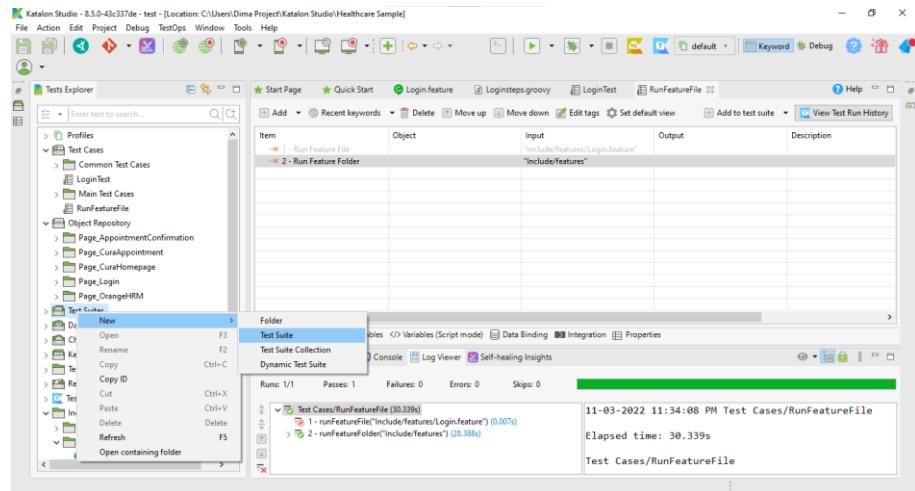
Gambar 14.82 hasil skenario pertama

77. Pengujian yang kita lakukan telah berhasil dijalankan. Bisa kita lihat pada tab *Job Progress* bahwa *test* nya berhasil dengan tanda ceklis berwarna hijau



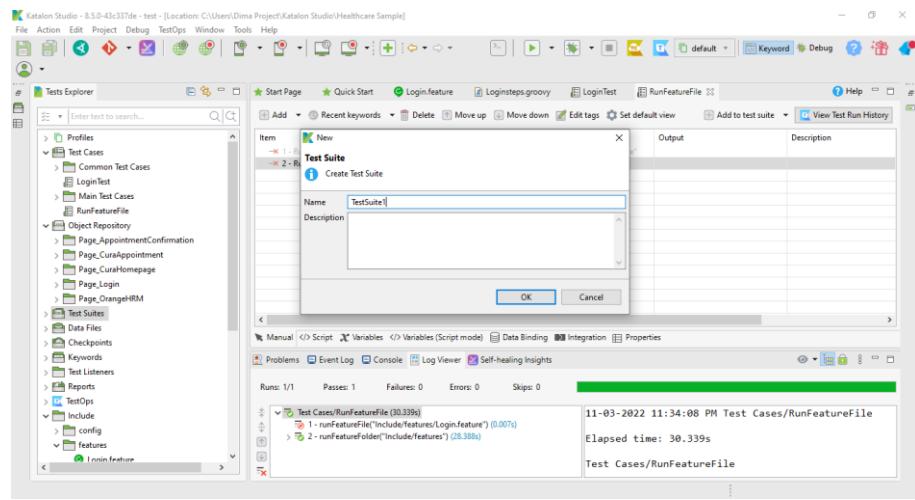
Gambar 14.83 keterangan berhasil dijalankan pada job progress

78. Untuk membuat *Test Suite* baru dengan cara klik kanan pada *folder Test Suites*, pilih *Test Suite*



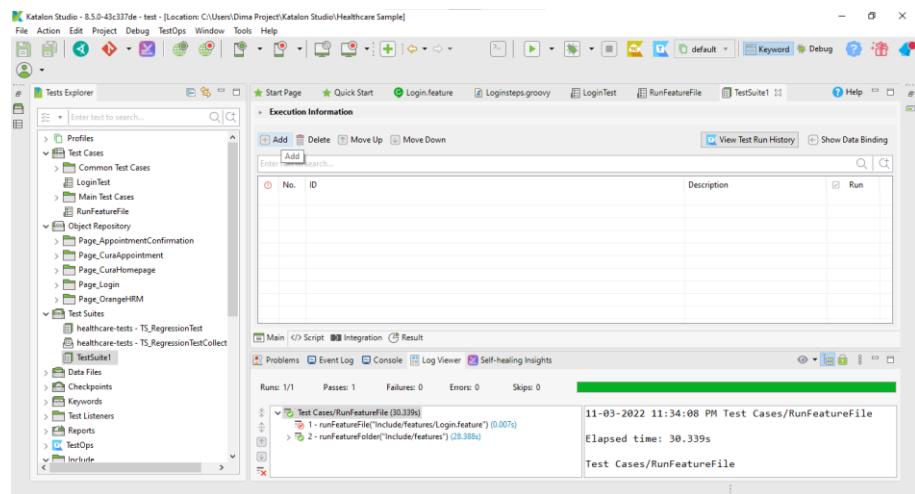
Gambar 14.84 buat Test Suite baru

79. Beri nama untuk file *Test Suite*. Jika sudah, klik OK



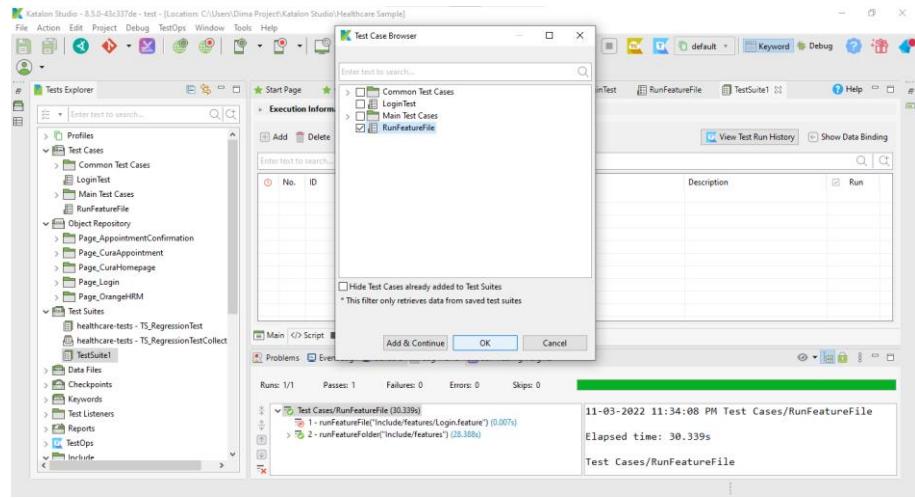
Gambar 14.85 berikan nama file Test Suite

80. Berikut adalah tampilan setelah kita membuat *file test suite*. Untuk menambah *item* baru pada tabel *Test Suite*, klik *button Add*



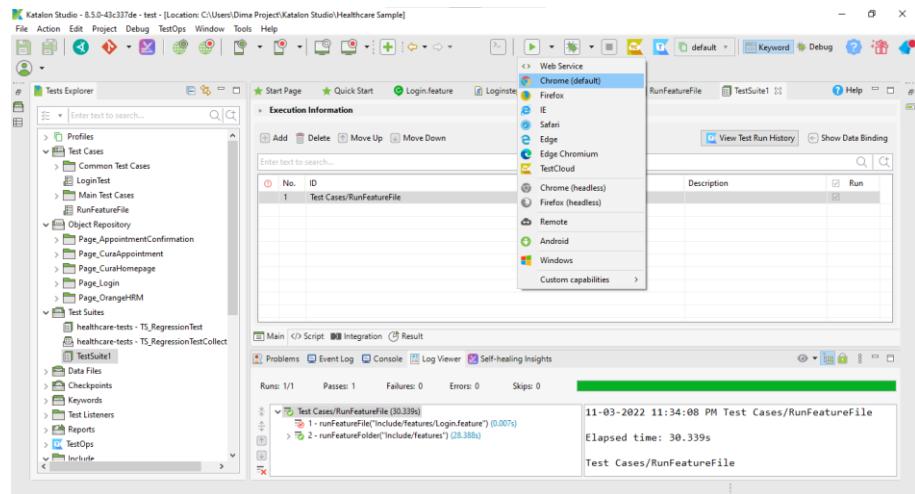
Gambar 14.86 tampilan setelah file Test Suite berhasil dibuat

81. Pilih *RunFeatureFile*, lalu klik OK untuk menutup jendela ini



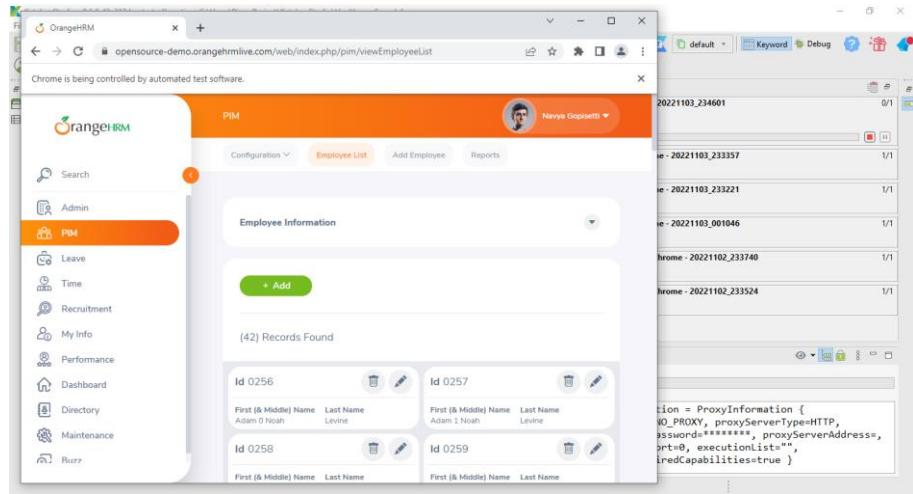
Gambar 14.87 pilih RunFeatureFile

82. Jalankan lagi file *Test Suite* nya untuk melihat apakah berhasil atau tidak



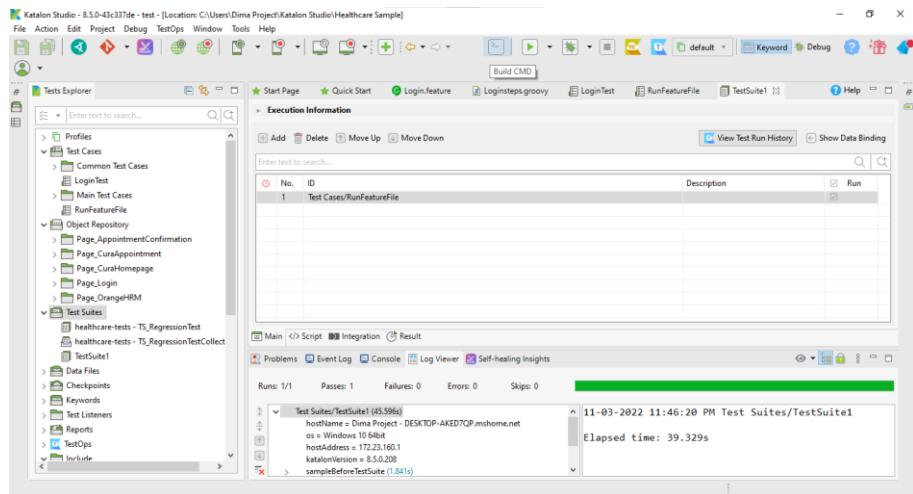
Gambar 14.88 jalankan file Test Suite

83. Hasil-nya adalah file ini berhasil



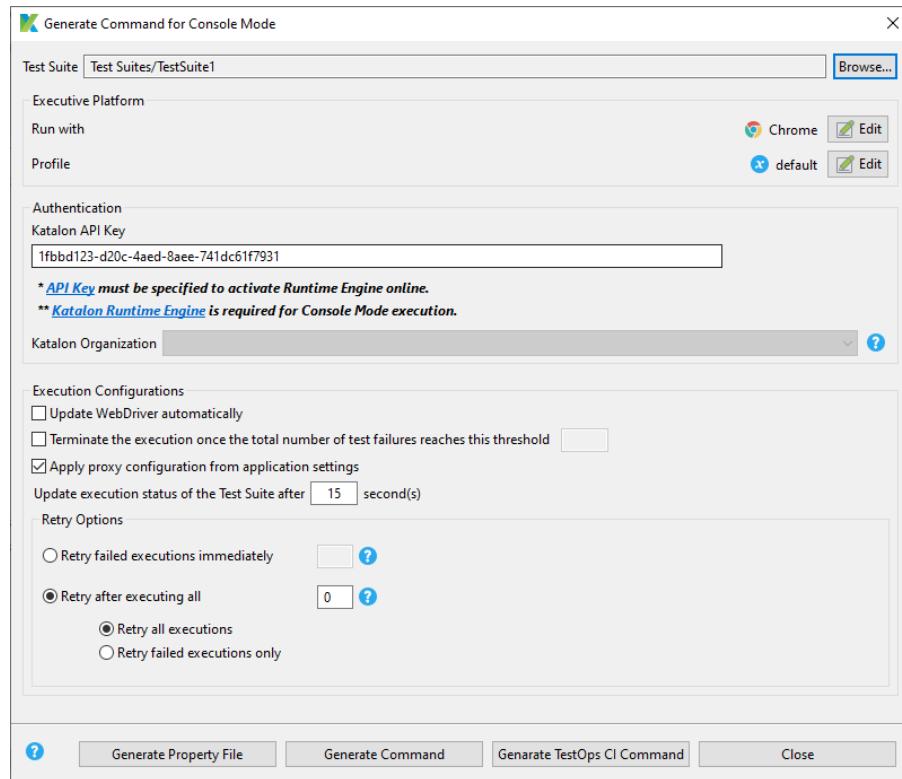
Gambar 14.89 berhasil dijalankan

#### 84. Selanjutnya klik button *Build CMD*



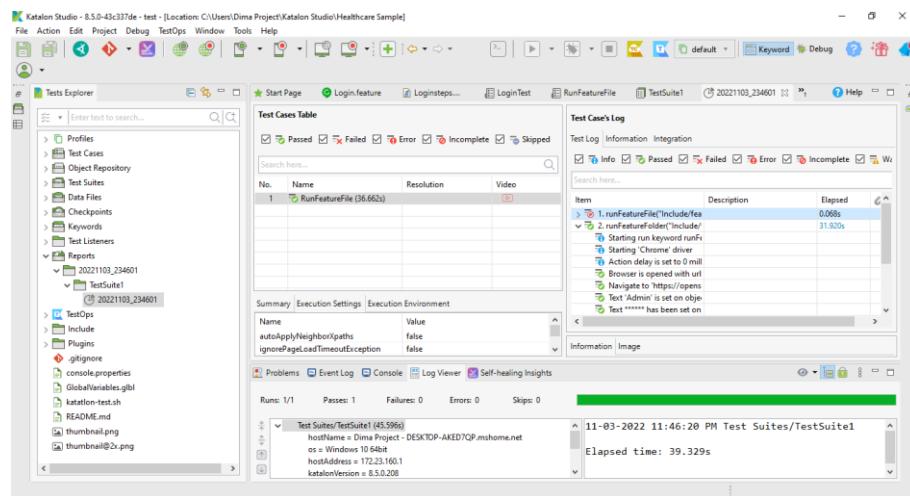
Gambar 14.90 klik button build CMD

#### 85. Setelah itu pada bagian *Test Suite*, pilih lokasi dari file *Test Suite* yang telah dibuat. Dibawahnya kita biarkan saja *default*. Kemudian klik *Close* dan jalankan lagi



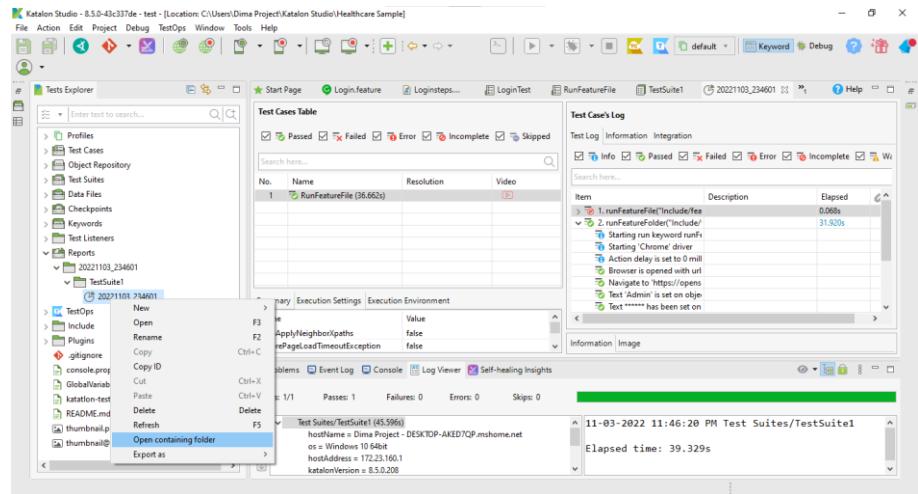
Gambar 14.91 pilih lokasi file Test Suite

86. Selanjutnya Buka *folder Reports\20221103\_234601\TestSuite1\* klik file **20221103\_234601**. Maka akan menampilkan *report* dari file *TestSuite1* beserta *Log*-nya



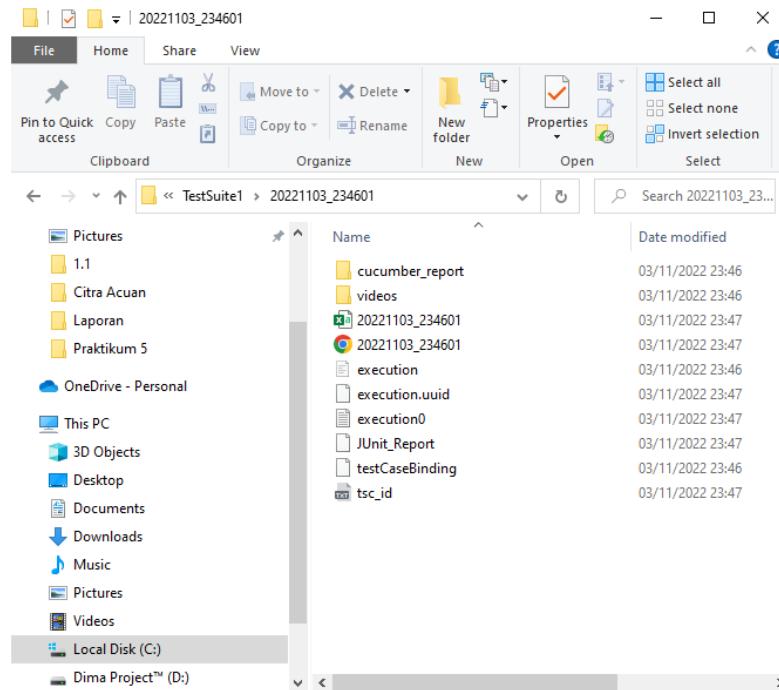
Gambar 14.92 buka folder reports untuk menampilkan report file Test Suite

87. Untuk mengetahui lokasi penyimpanan file ini adalah dengan cara klik kanan file **20221103\_234601**, lalu pilih *Open containing folder*



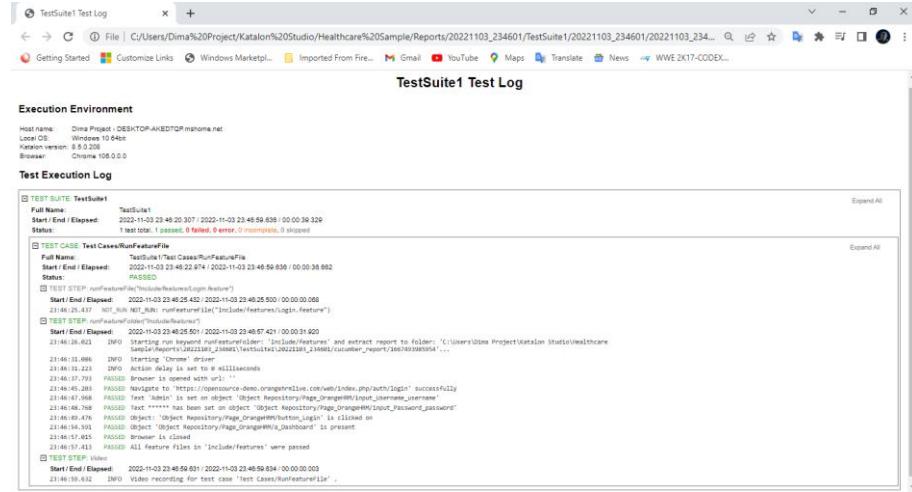
Gambar 14.93 cari lokasi penyimpanan

88. Maka kita akan dibawa ke tempat lokasi penyimpanan. Untuk membuka *report* dari file *HTML Test Suite1* melalui *web browser*, klik file *HTML*-nya.



Gambar 14.94 pada lokasi penyimpanan pilih file HTML

89. Maka file *HTML*-nya sudah terbuka di Chrome. Kita bisa melihat *log* dari pengujian yang telah kita lakukan

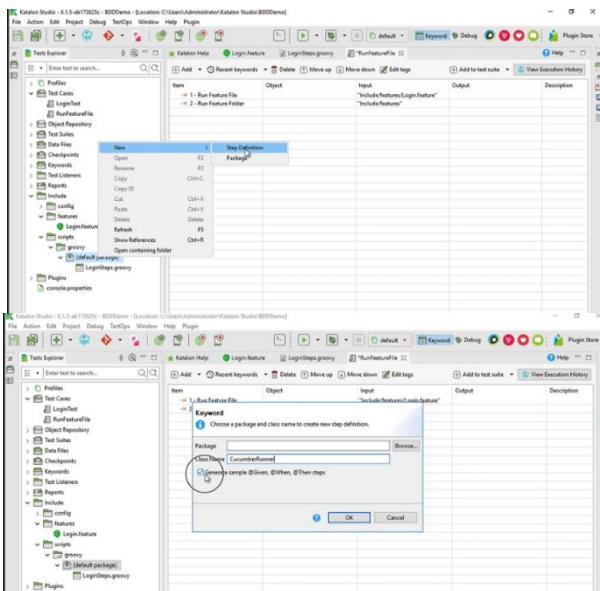


Gambar 14.95 log dari pengujian yang dilakukan

### 14.3.2. How To Create Cucumber Runner

Apa itu Cucumber Test Runner Class? Dalam istilah yang sangat sederhana, Cucumber test runner class adalah salah satu dari banyak mekanisme yang dapat digunakan untuk menjalankan file fitur Cucumber.

## 1. Membuat class baru didalam folder groovy



Gambar 14.96 membuat class baru dalam folder groovy

## 2. Tambah code didalam class

- Import org.junit.runner.RunWith
  - Import cucumber.api.CucumberOptions

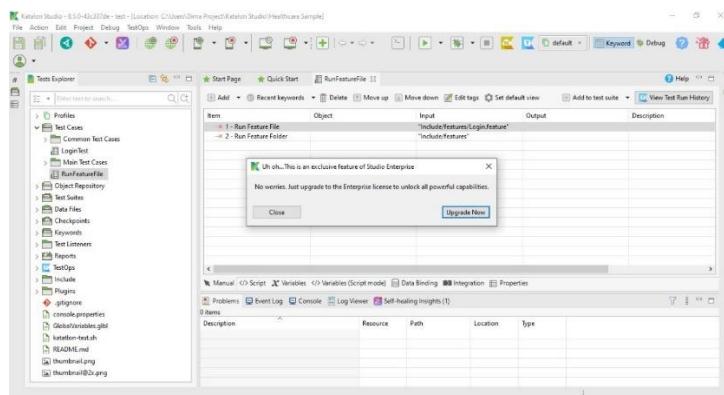
- Import cucumber.api.junit.Cucumber



```
18 import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
29
30 @RunWith(Cucumber.class)
41 @CucumberOptions(features="Include/features", glue="", plugin=
52 ["pretty"], "html:ReportsFolder", "json:ReportsFolder/cucumber.json")
63 public class CucumberRunner {
74
85 }
```

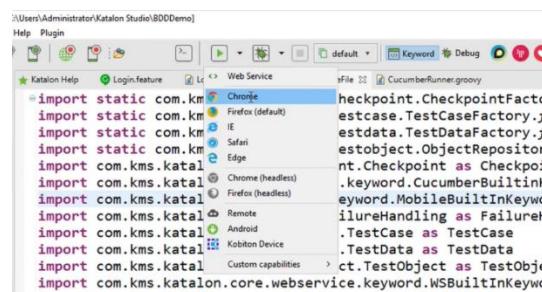
Gambar 14.97 tambahkan code di dalam class

3. Didalam test tersebut kita akan meng add cucumber



Gambar 14.98 add cucumber didalam test

4. Masuk ke script Runfeaturefile lalu jalankan



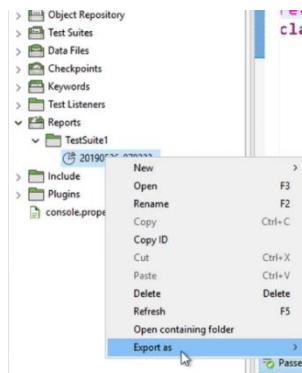
Gambar 14.99 masuk ke script Runfeaturefile dan jalakan

5. Tunggu proses running sampai selesai



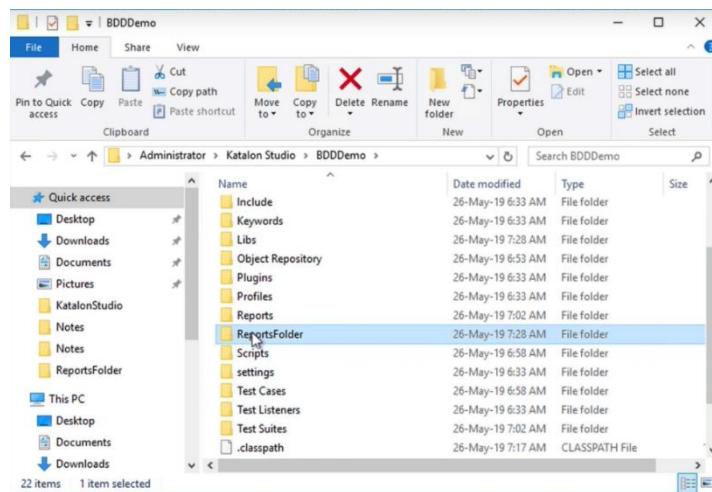
Gambar 14.100 proses running file

6. Lalu pilih reports dan



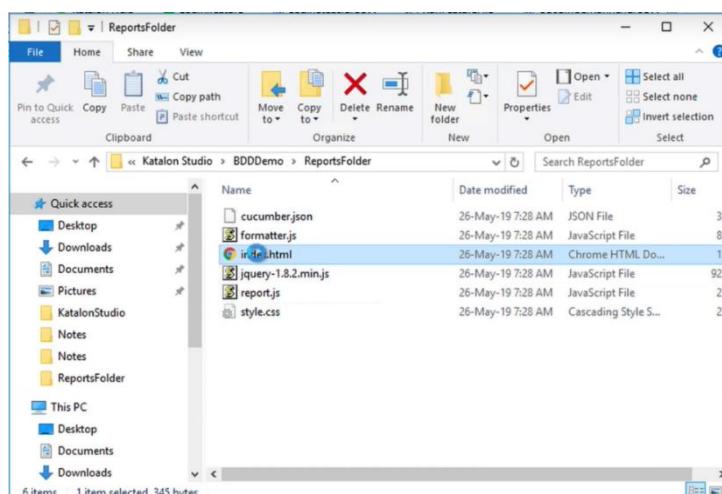
Gambar 14.101 pilih reports

7. Pilih Reports folder



Gambar 14.102 pilih reports folder

8. Klik dua kali di index.html



Gambar 14.103 klik 2 kali pada index.html

9. Maka Hasil Nya akan seperti ini

▼ Feature: Login feature				
▼ Scenario Outline: Test login with valid credentials				
Given User navigates to login page				
When User enters <username> and <password>				
And Click on login button				
Then User is navigated to homepage				
▼ Examples:				
<table border="1"><tr><td>username</td><td>password</td></tr><tr><td>Admin</td><td>hUKwJTbofgPU9eVlw/CnDQ==</td></tr></table>	username	password	Admin	hUKwJTbofgPU9eVlw/CnDQ==
username	password			
Admin	hUKwJTbofgPU9eVlw/CnDQ==			
▼ Scenario Outline: Test login with valid credentials				
Given User navigates to login page				
When User enters Admin and hUKwJTbofgPU9eVlw/CnDQ==				
And Click on login button				
Then User is navigated to homepage				

Gambar 14.104 hasil reports folder

#### 14.4. Penutupan

##### 14.4.1. Simpulan

BDD adalah suatu metode yang menjelaskan mengenai behavior mengenai sudut pandang pengguna, mendeskripsikan kebutuhan fungsional dan dari pengguna bedasarkan behaviornya, tujuannya menyamakan persepsi dari semua pihak tersebut.

Jadi bisa dikatakan kalau BDD ini sangat penting dilakukan, karena ini bisa menghindari berbagai persepektif yang sama-sama baiknya namun kadang tidak tepat sasaran dikarenakan masing-masing posisi memiliki pola pikir yang berbeda terhadap ekspektasi calon usernya.

## **DAFTAR PUSTAKA**

**There are no sources in the current document.**