Tool Evaluation of SoapUI Automation Testing Tool

Introduction:

Web Services are very popular these days and they can be used to further extend the functionality of your application. Once the application is hosted on a web service, they can be used on other applications as an extended service / framework. The most commonly used Web Services are REST and SOAP. These consist of APIs which assist the application to do specific operations and functionalities. SOAP UI is an automation test tool used to test such REST and SOAP web services. SOAPUI allows testers to execute automated functional, regression, compliance, and load tests on different Web API. SOAP UI has a GUI and Command Line based approach to testing these Web Services which enables Technical and Non-Technical users to use the tool conveniently. SOAP UI is a free software and was released in 2011 by Eviware Software. They also have a Pro version of the tool which mainly aims at making recurring tasks easier.

The commercial use of SOAP UI involves simulating the live environment of the Web Service to check live issues which might arrive. Usually testers use SOAP UI to simulate the scenario reported by a customer to check what might be causing an issue in the Web Service. SOAP UI is developed solely on Java Platform and uses Swing for the User Interaction.

Setup and Usage:

- As SOAP UI is open source, it can be downloaded freely off their website: www.soapui.org.
- The installation is pretty straight forward and following simple instructions presented by the executable should successfully install SOAP UI.
- After the installation is complete, the following screen is visible when SOAP UI is launched.

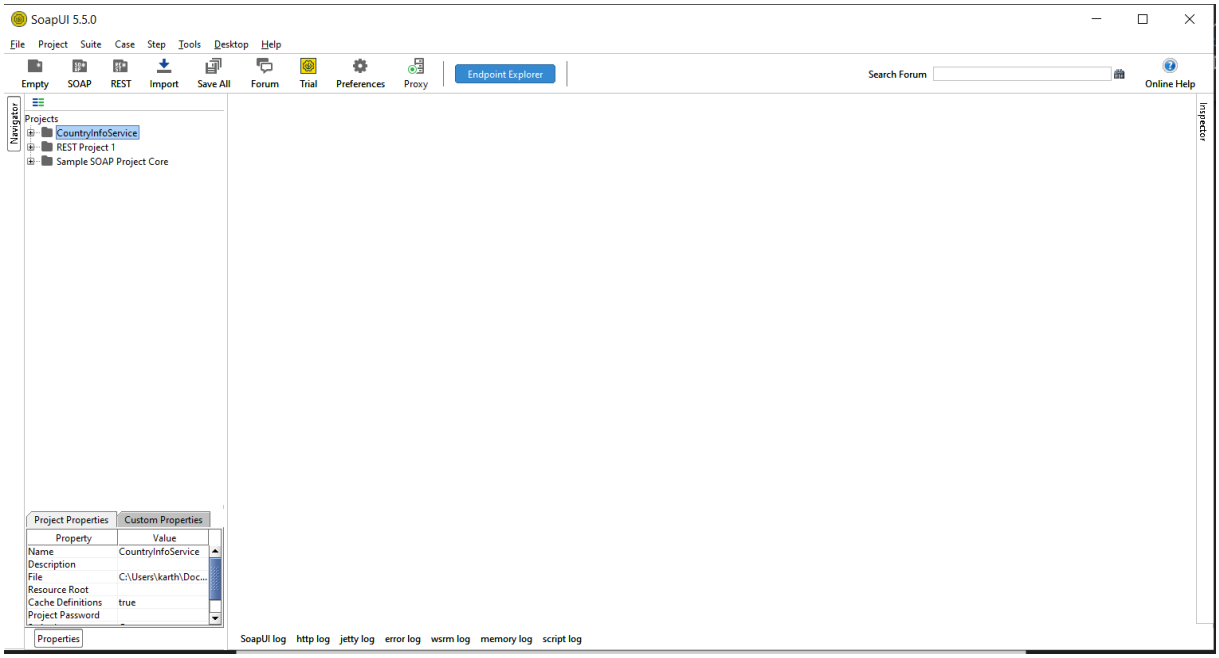


Fig 1: SOAP UI Start Screen

We can test SOAP API or REST API with the tool, we will start with demonstrating the operations with respect to SOAP API first. WSDL(Web Service Description Language) is a description as to what the web service is capable of doing and in order to test a Sample SOAP Web Service, we can import a WSDL in the following manner:

1. Select the SOAP icon on the toolbar and create a new SOAP Project.
2. A sample WSDL has been retrieved, with the following URI http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL
3. The URI is pasted into the Initial URI field and press OK.

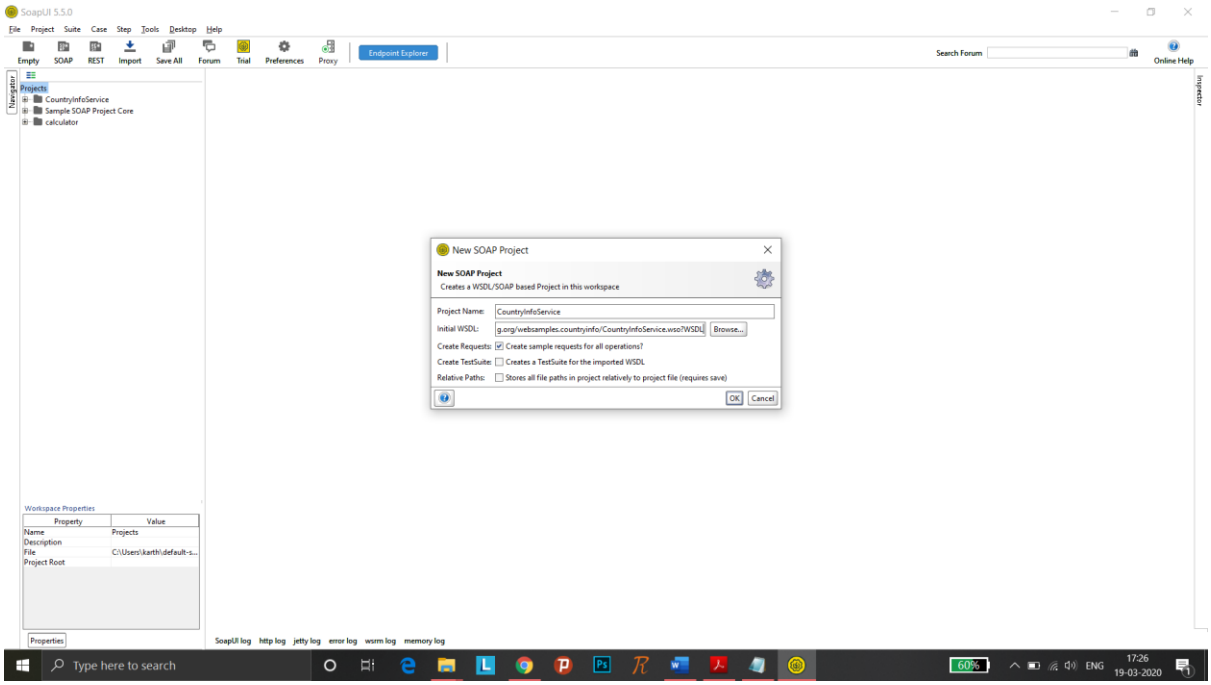
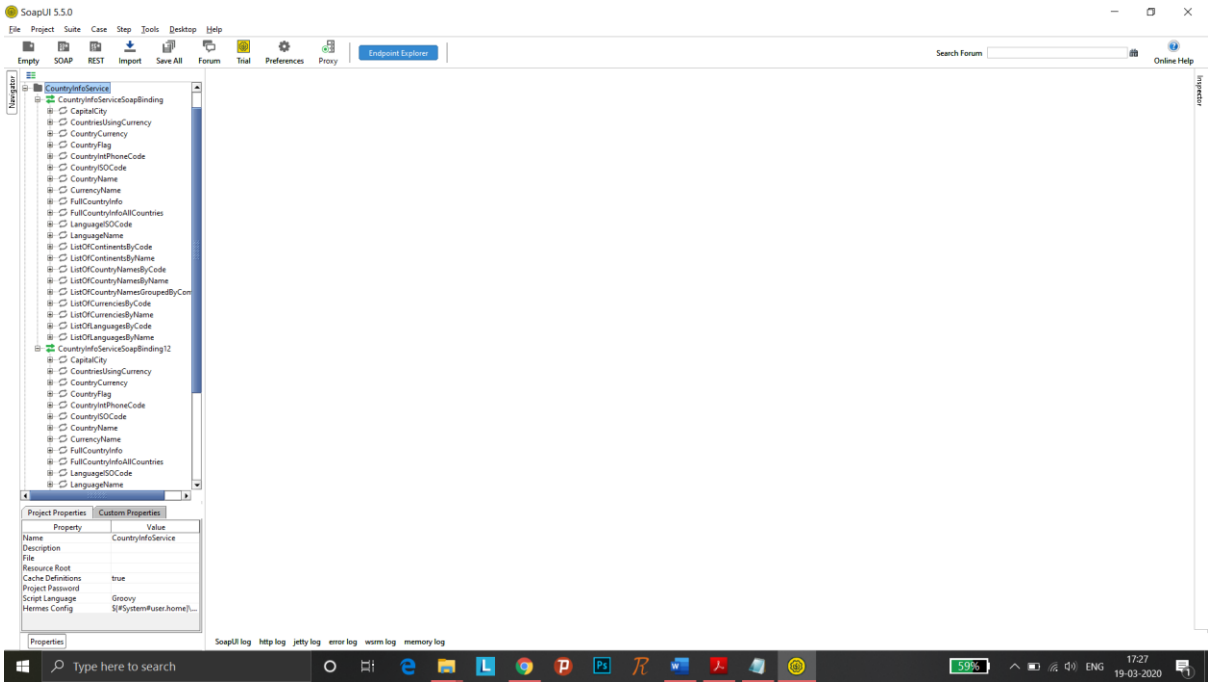


Fig 2: Adding a SOAP Service WSDL

4. Which will import all the APIs used in the used SOAP service.



5. Once done, we can now add Test Cases, Test Suites and Test the functionality of each API and functionality of the Web Service as a whole.

Test Suite, Test Case and Test Step:

A Test Suite is a collection of Test Cases being run against the selected API and in turn a Test Case consists of Test Steps, which are individual APIs which perform actions.
Assertion is something used for validation of test cases, to check if they meet the required output constraints. Basically Assertions validate the output delivered by the TestStep during execution.

1. Right clicking on the imported SOAP service will enable us to create a new Test Suite.

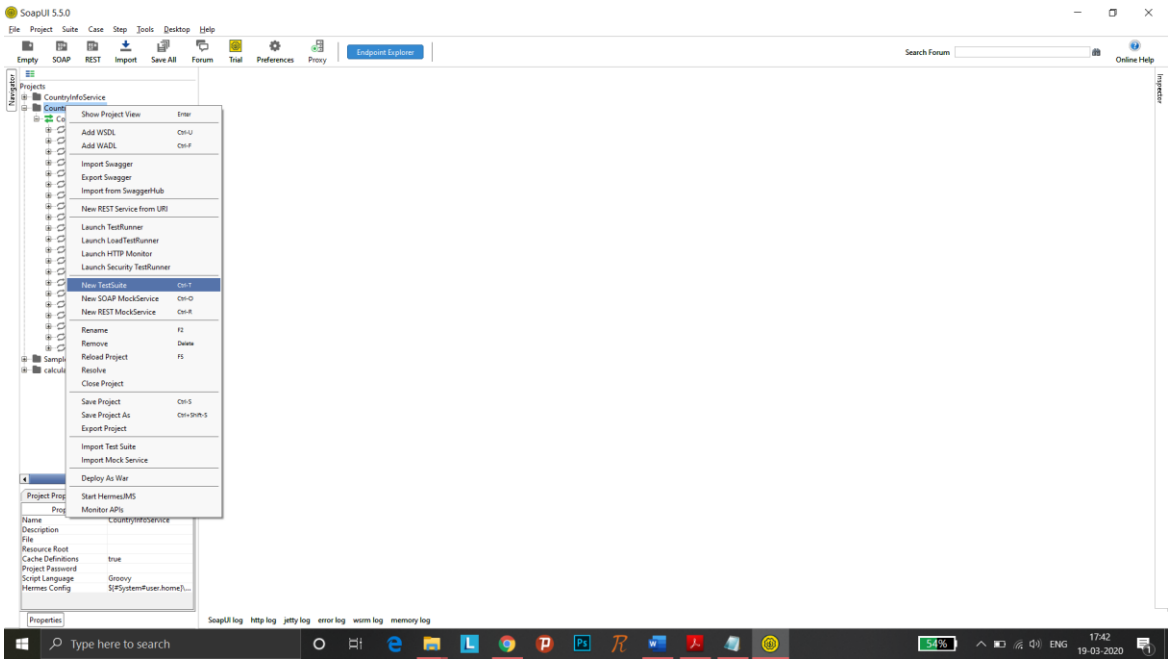


Fig 3: Creating New Test Suite

2. Further a test case can be added into the test suite by right clicking the test suite and selecting add new test case.

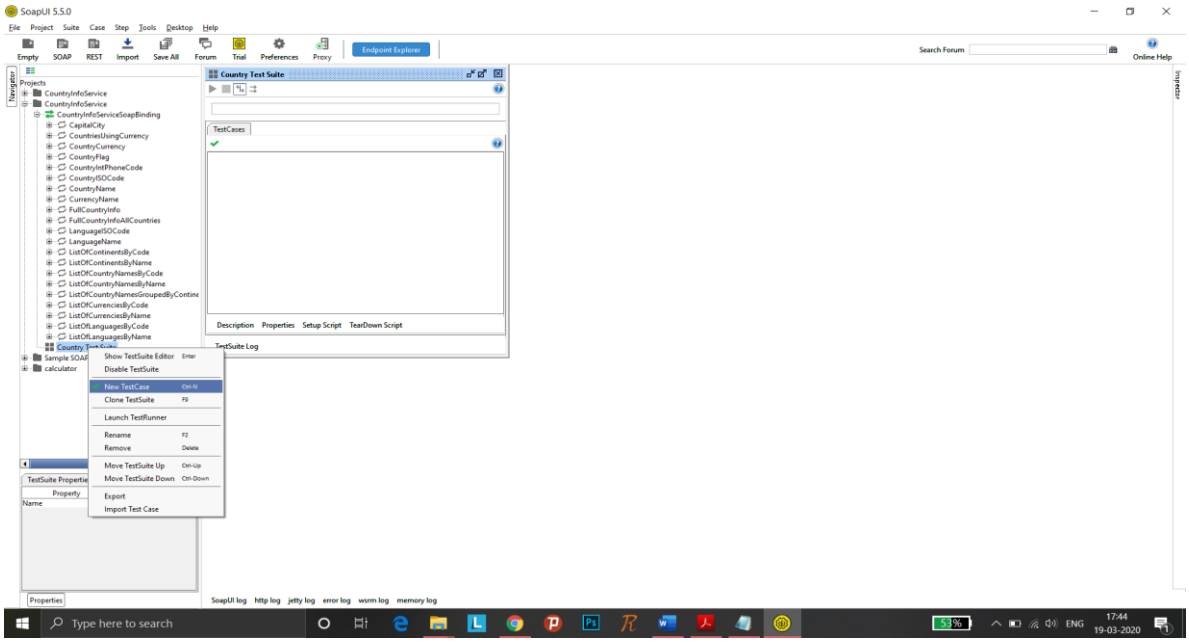


Fig 4: Creating new test case

3. A Test Step can be added from the list of APIs in the service by right clicking on them and using add to test case.

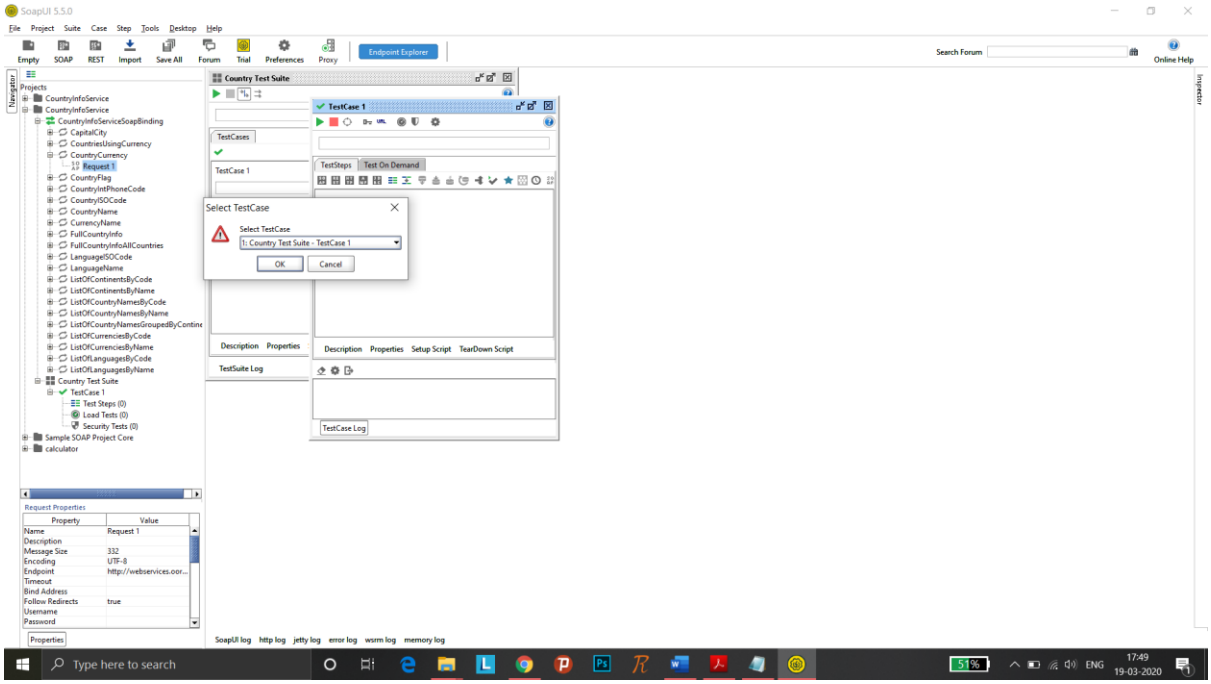


Fig 5: Adding Test Step to test Case

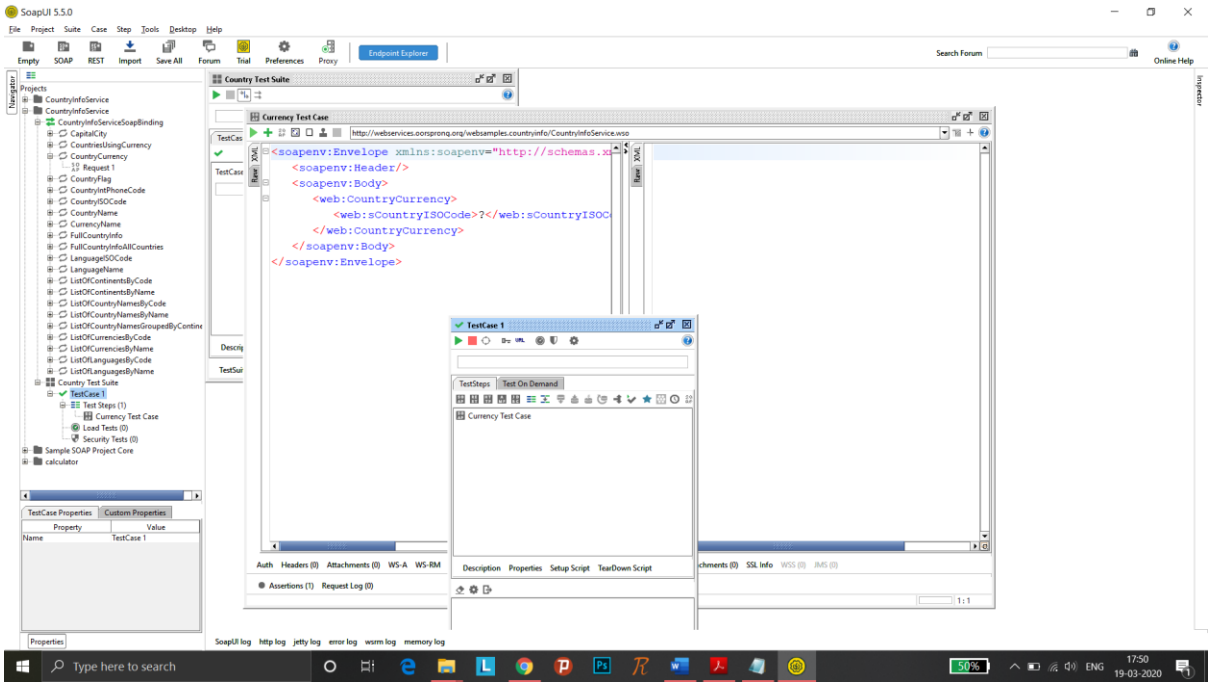


Fig 6: Test Step Execution

4. We will now add a basic assertion to Validate the output of the Test Step. An API, Country Currency is under consideration here, which basically outputs the currency of the country if the ISO Code of the Country is provided. A manual Test can be conducted by replacing the ? in the code and plugging in the desired ISO to observe the output.

We can add an assertion to validate the output in the following manner:

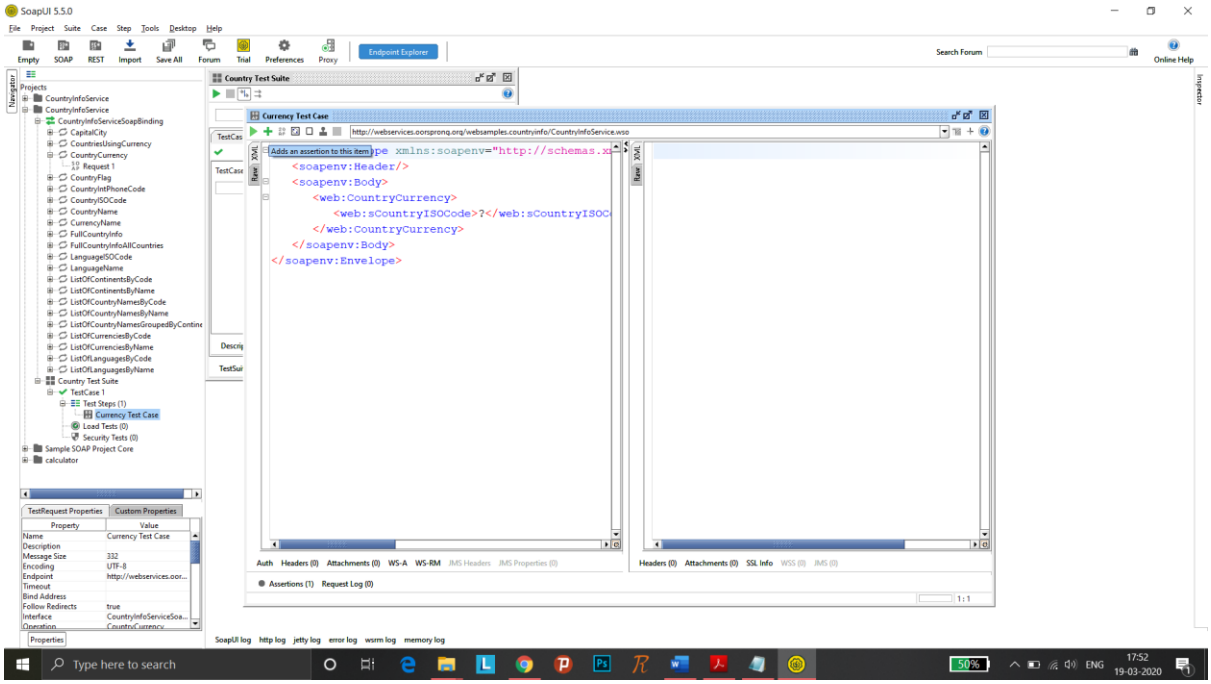


Fig 7: Adding Assertion

We are using the assertion clause CONTAINS, which basically checks if the output of the Test Step contains the provided Content.

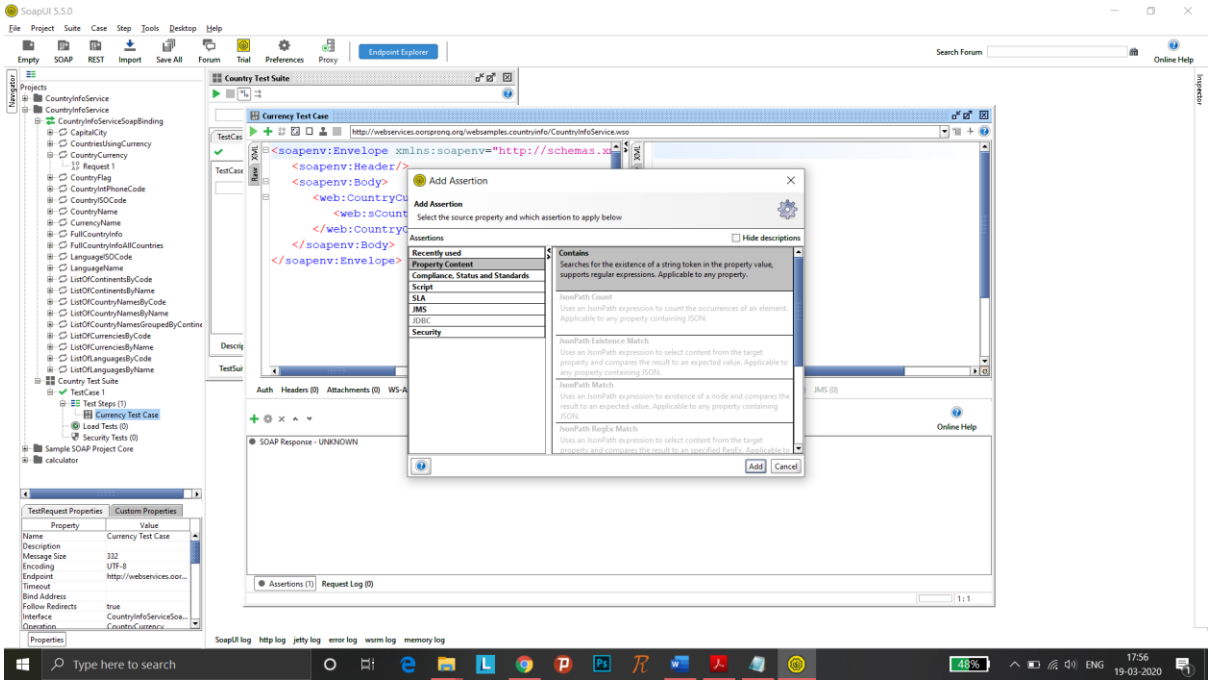


Fig 8: Different Types of Assertions

If a valid ISO is provided and a corresponding currency is present, the API responds in the following manner with the Assertion turning Green and valid response time entry in the Request Log:

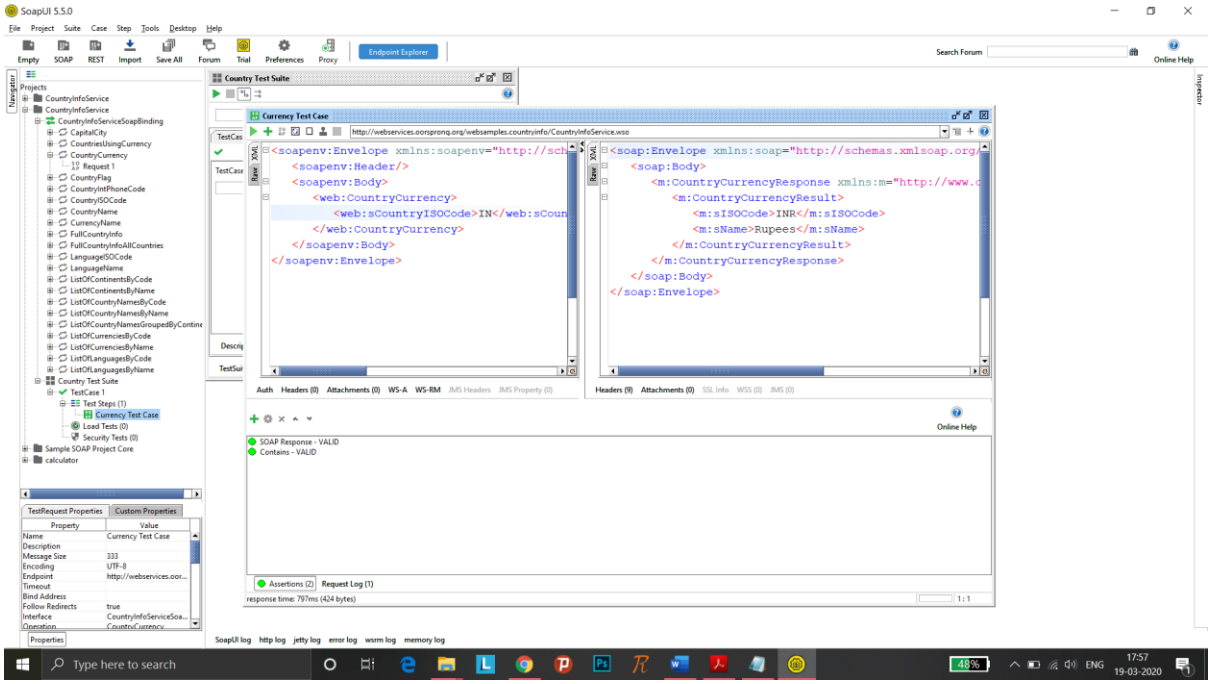


Fig 9: Assertion in action

If an invalid input is provided or there is no corresponding entry for currency, the assertion turns red and throws an error:

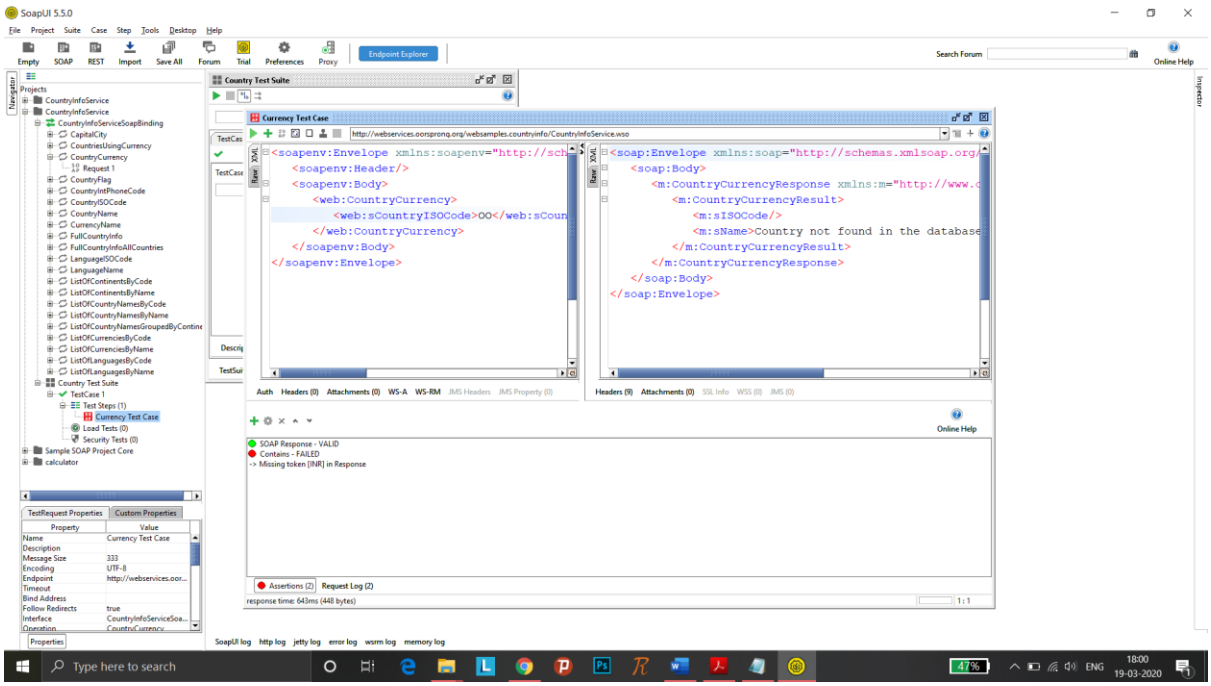


Fig 10: Error case in Assertion

Hence, in this way an Assertion can be used to validate the output and we extensively use this to decide Test Cases. There are multiple other Assertions that can be used such as Not Contains, XPath, or Scripting Assertions.

5. SOAP UI gives the facility to automatically generate Test Case and Test Suites for the added SOAP service. There are two options to do so. Either we can create a new test Case for every

Test Step or Create a single test case for all Test Steps/APIs. We can auto generate test cases by right clicking on the imported Service and selecting “Generate TestSuite”.

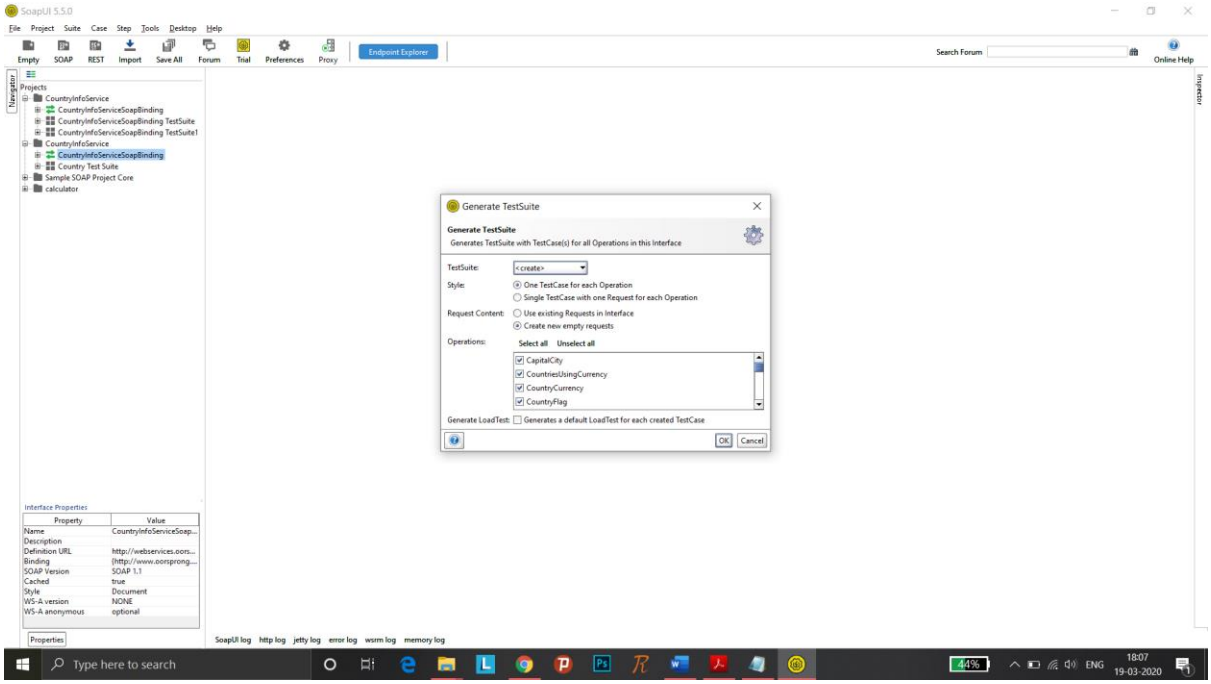


Fig 11: Generating Test Cases for Test Suite

6. A Test Suite can be executed at once to run all the present test cases. SOAP UI provides an option of Parallel or serial execution of Test Cases too. This is in order to prevent the execution of the next test case if the previous case fails, if the execution is set to serial. The execution proceeds and finishes testing all cases if the execution is set to Parallel.
   The option to run parallelly or serially is present when we select a Test Suite.

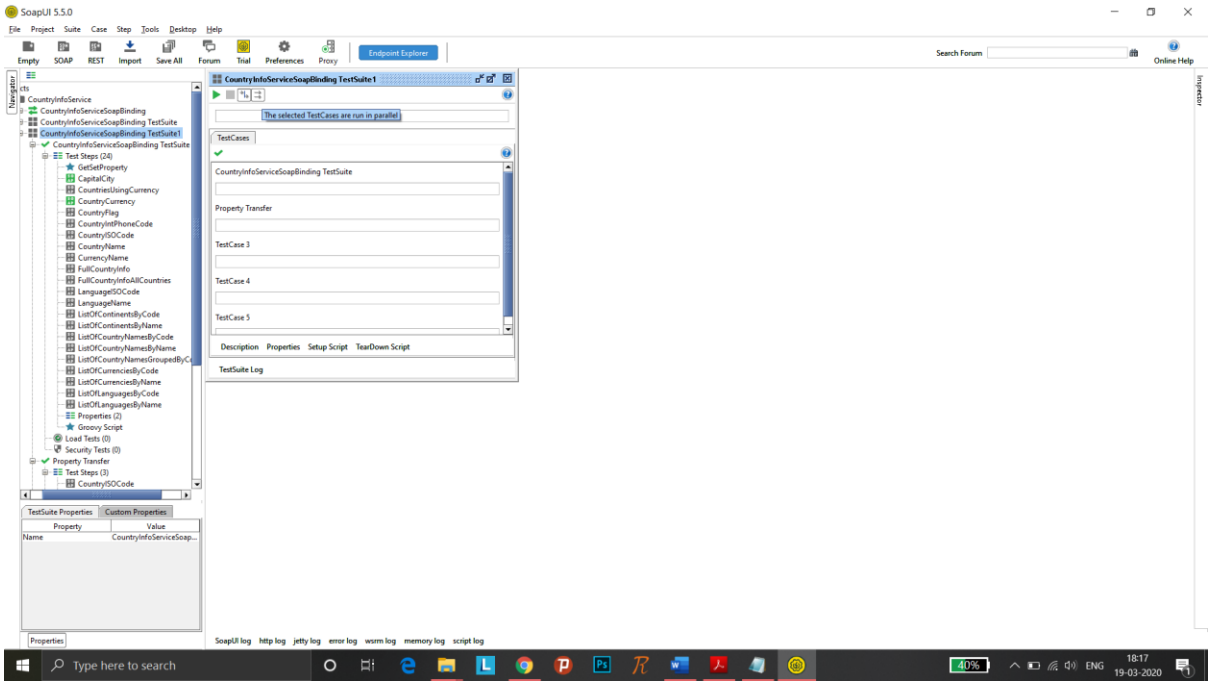


Fig 12: Parallel and Serial Execution of Test Suite

7. SOAP UI has a very innovative feature, where it generates a document(webpage) containing all the functionalities of the added SOAP service.

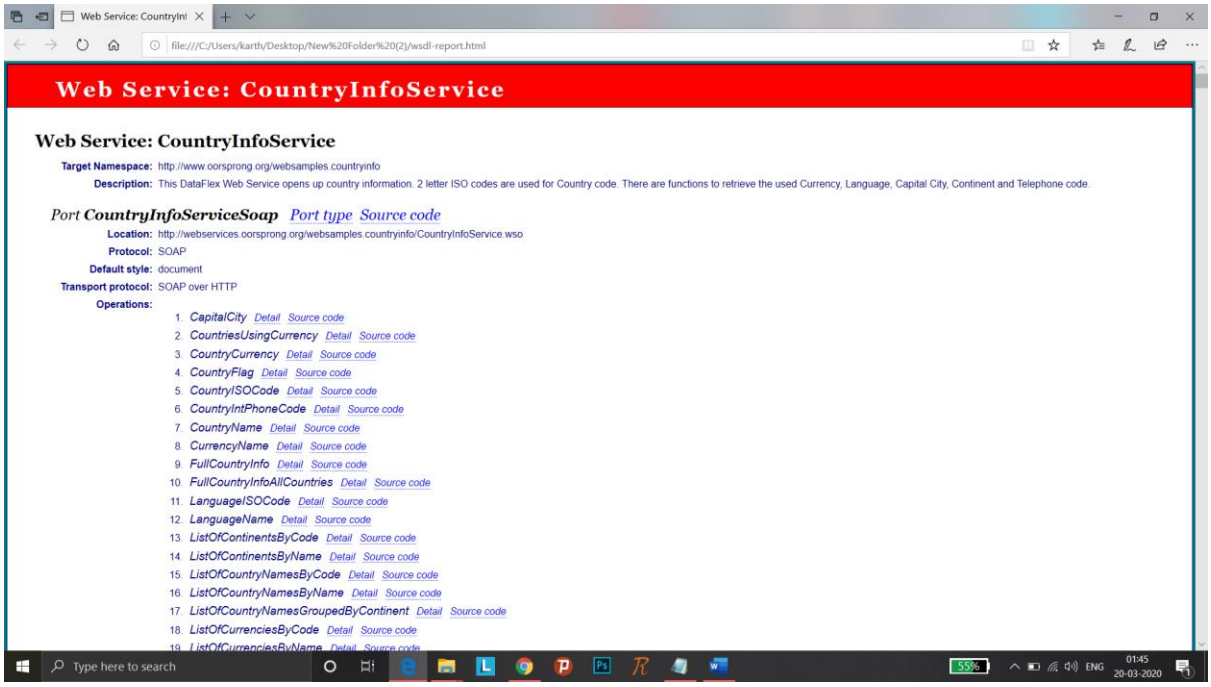


Fig: Generated document of the SOAP Web Service

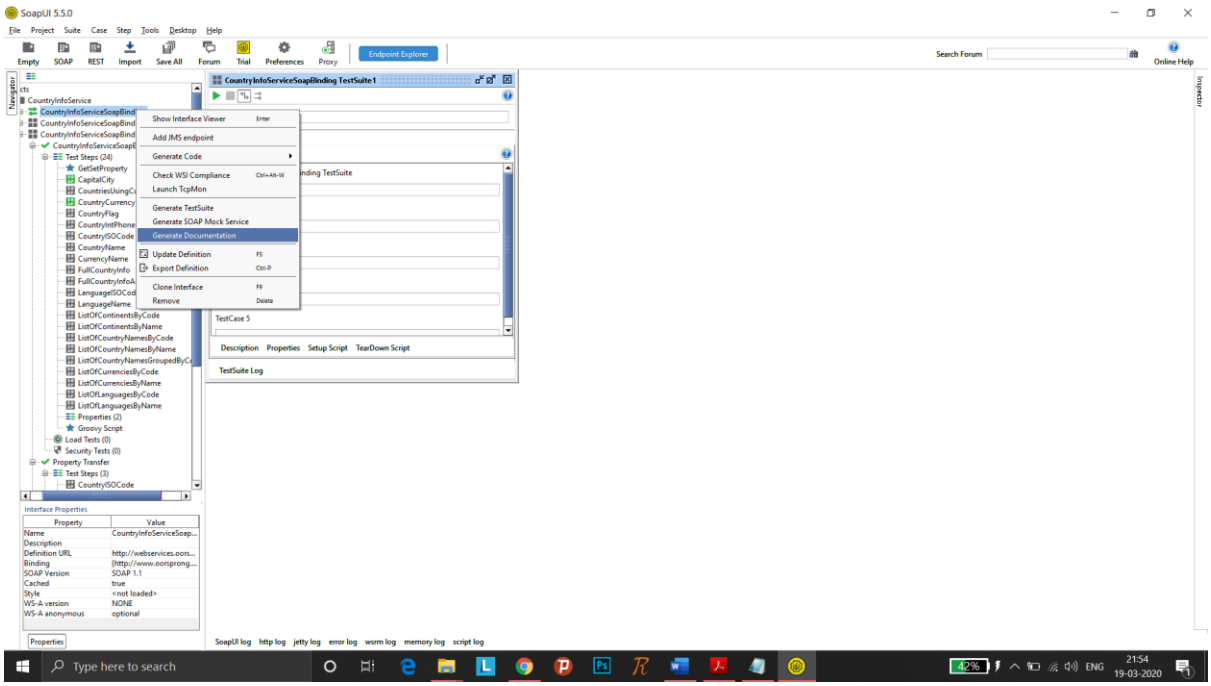


Fig 13: Document Generation

A REST Web Service can also be tested in a similar way that a SOAP Service was shown.

1. The REST Service to be tested can be imported by using the Create a New Rest Project button.

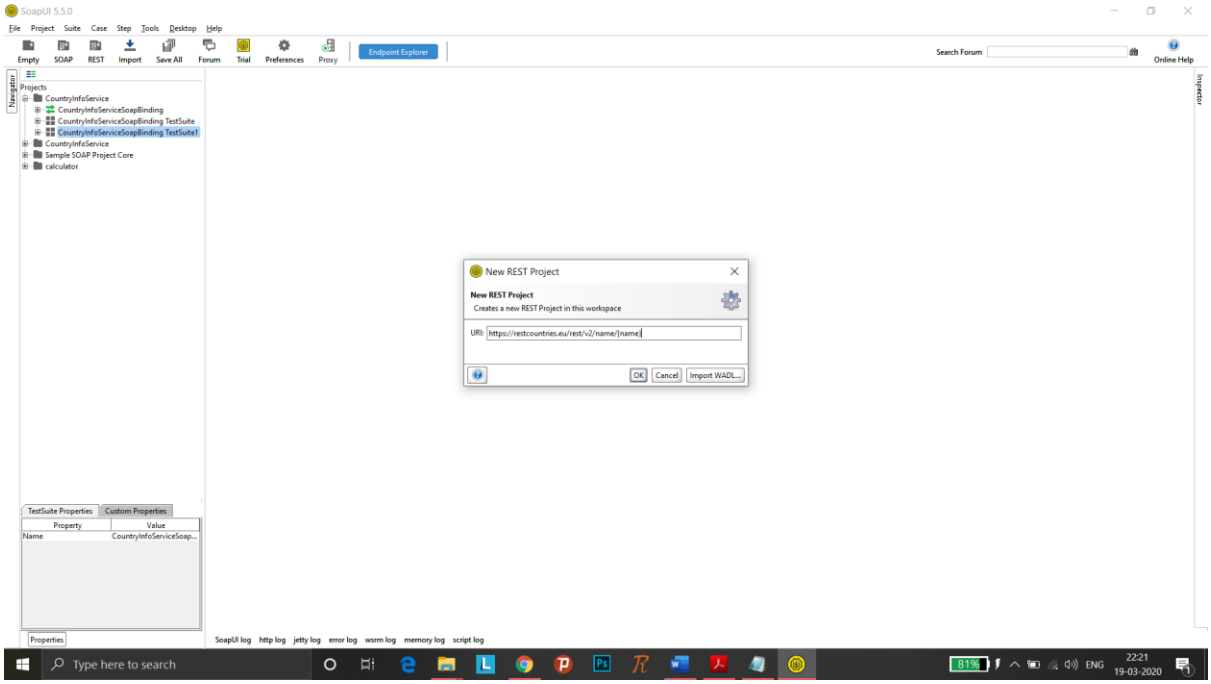


Fig 14: Importing a Rest Web Service

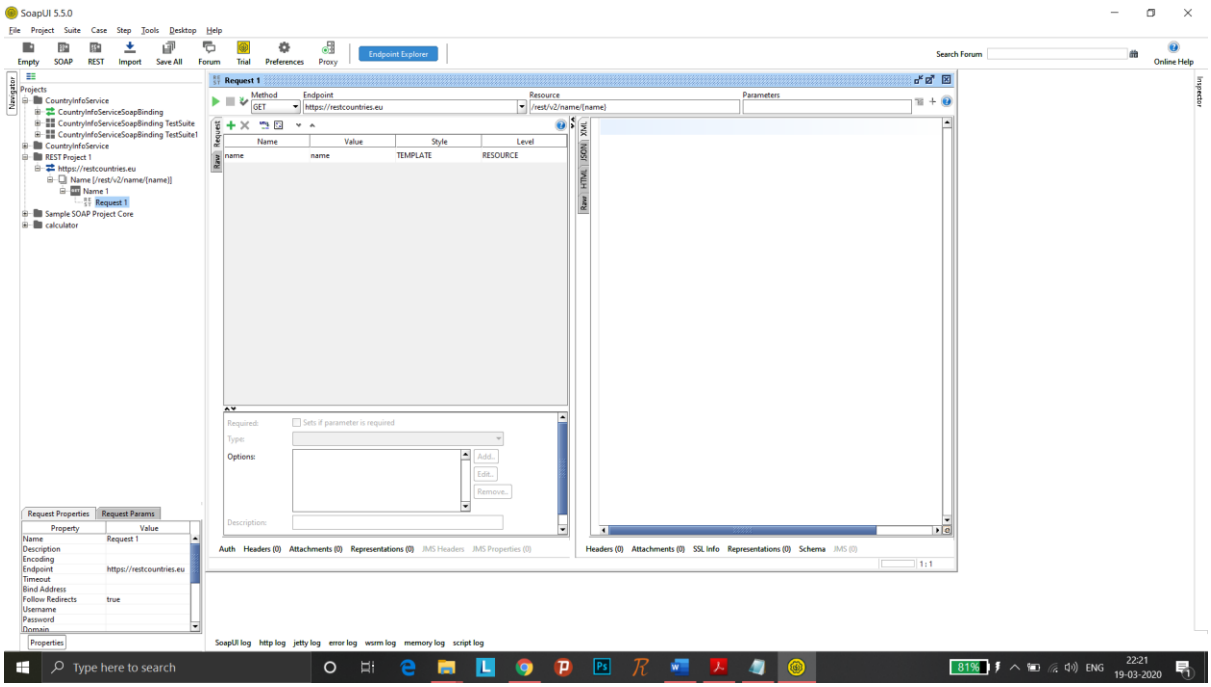


Fig 15: After Importing Rest Service

2. Manual testing for parameters can be done by plugging in values either in the resources field or parameters field. In order to have a more generalized approach, we generalize the parameter being passed to the Resource or Parameters field using a local variable and configuring appropriate value in the Raw table.
   In the following example, a temporary variable countryName is used to generalize the input structure.

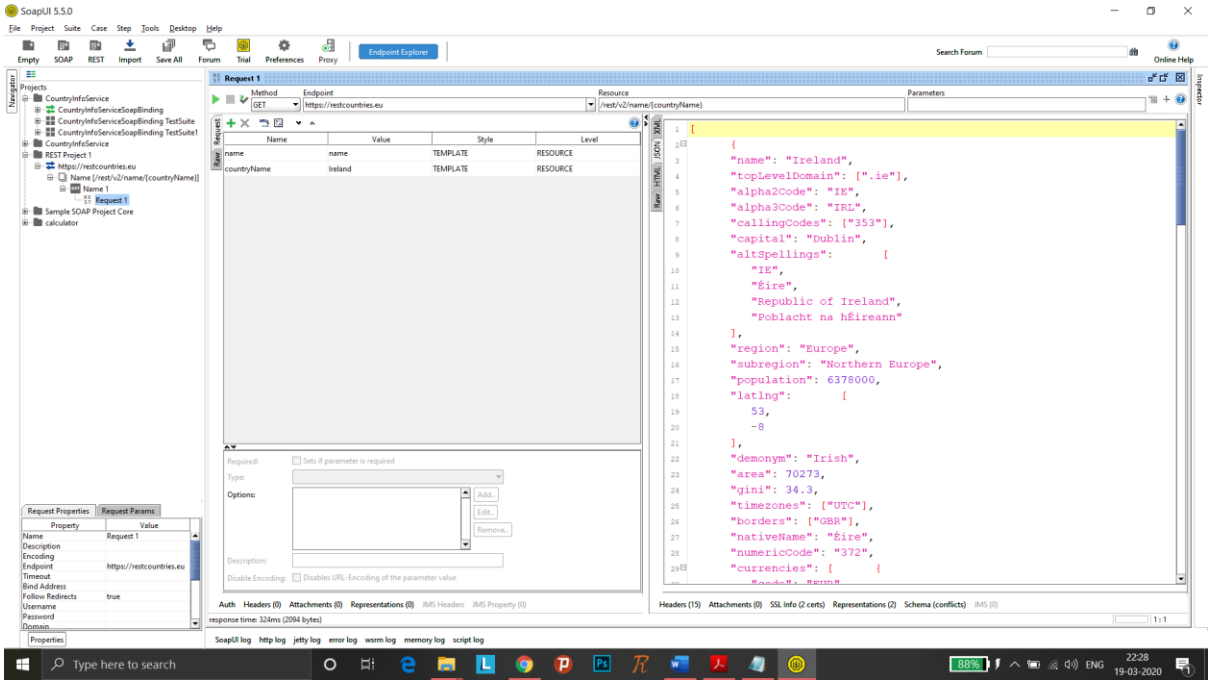


Fig 16: Using RAW table to generalize parameter

An invalid input, gives the following error message:

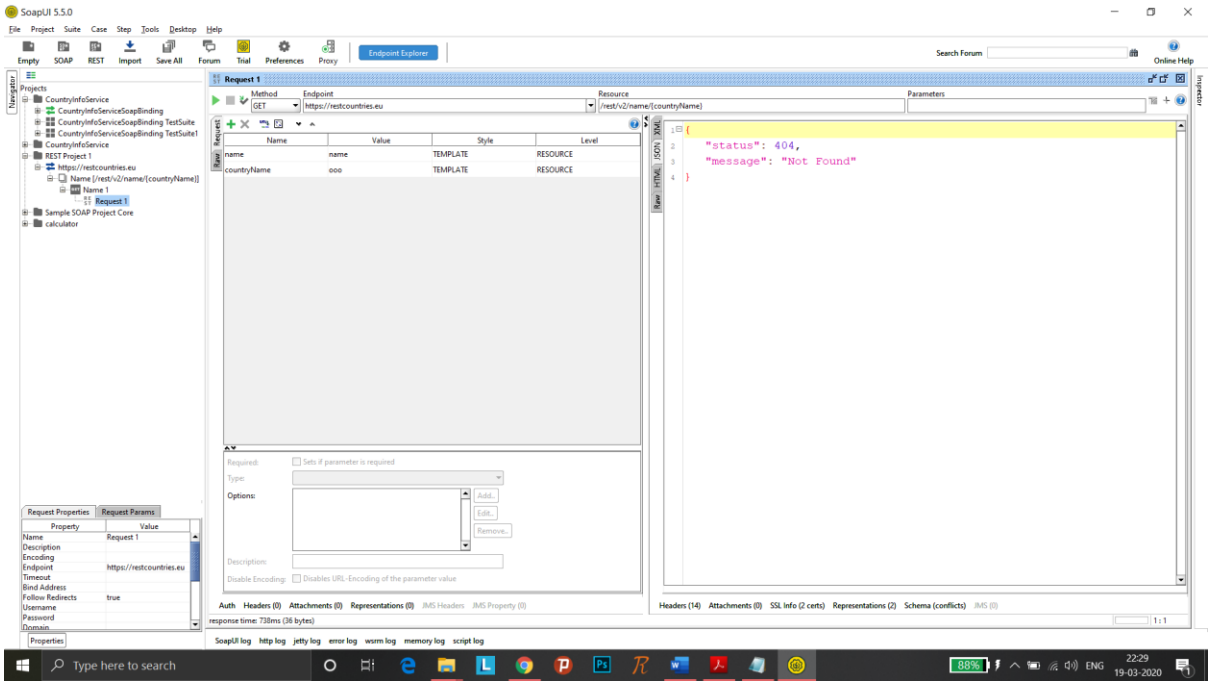


Fig 17: Invalid input in REST API

As previously illustrated for SOAP service, REST Service can also be added with Test Suites, Test Cases and Test Steps. Assertions can be added to validate the output.

Properties:

Properties are used in SOAP UI in order to centralize certain parameters being tested. For example Properties can be used to store credentials being used across the platform or User ID, which will be common across the application.

Properties can be assigned at the Project Level, Test Suite, Test Case or Test Step Level.

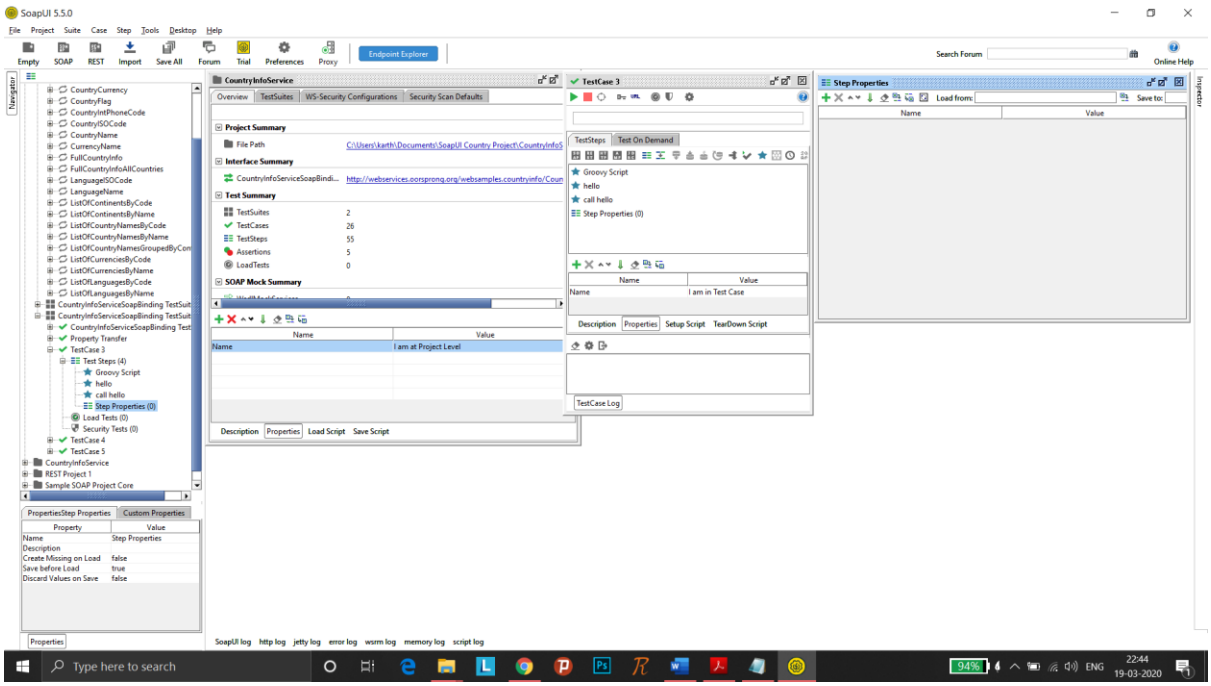


Fig 18: Adding property to a Project

Groovy:

An innovative feature of SOAP UI is that it has its own language to implement object orientation and other concepts alongside testing. Apache Groovy is entirely based on the JAVA platform and can be used as a scripting and programming language.

Groovy scripts can be used in SOAP UI to add scripts alongside the test cases to innovatively provide input and extract output for further processing. It can also be used to make a property transfer between Test Steps. Property setting and getting which was illustrated earlier using the GUI can also be done using the Groovy Scripts.

A groovy script can be added to a test case in the following way:

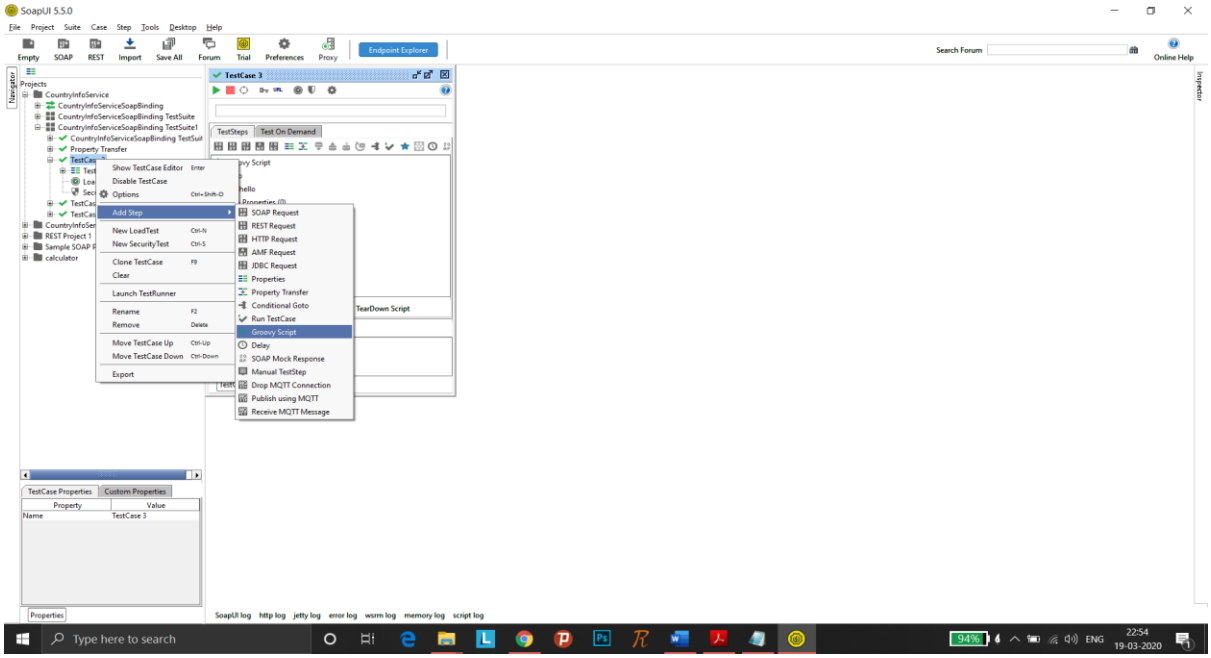


Fig 19: Creating a groovy script

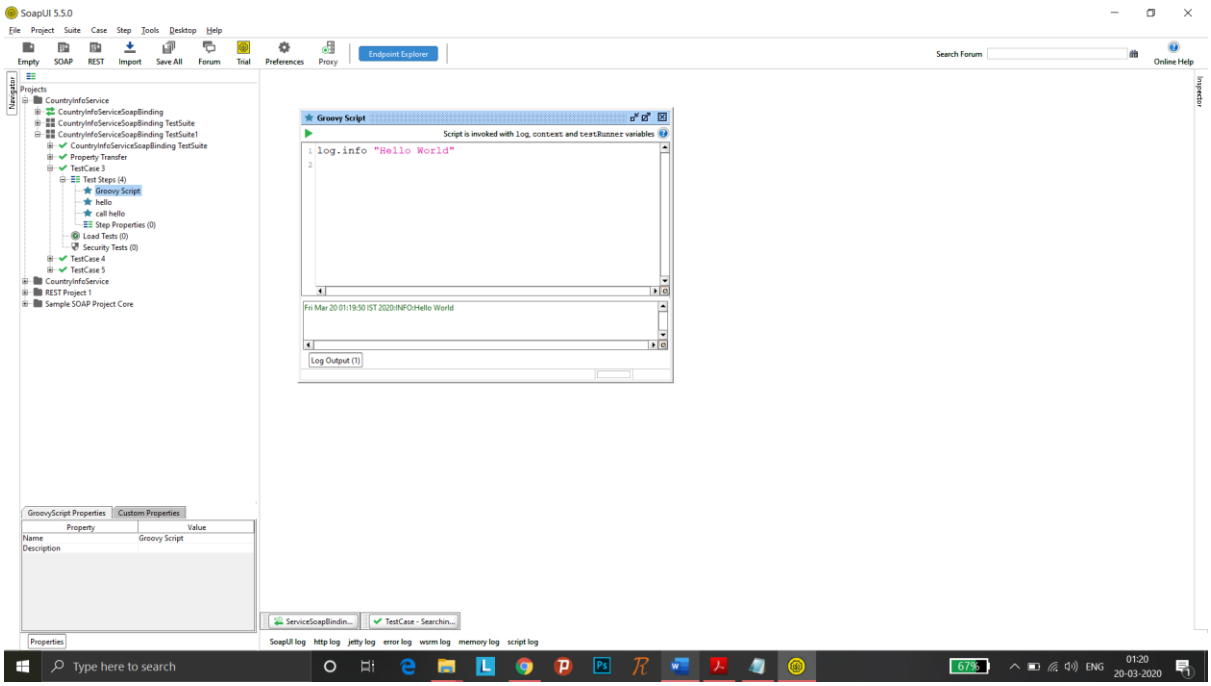


Fig 20: Running a groovy script

The following code can be used to set, get and add properties at Project, Test case and Step level.

Get and Set Property

Project:

```
testRunner.testCase.testSuite.project.getPropertyValue("Name")
testRunner.testCase.testSuite.project.setPropertyValue("Name","I am in Project")
```

TestSuite:

```
testRunner.testCase.testSuite.getPropertyValue("Name")
testRunner.testCase.testSuite.setPropertyValue("Name","I am in TestSuite")
```

TestCase:

```
testRunner.testCase.getPropertyValue("Name")
testRunner.testCase.setPropertyValue("Name","I am in TestCase")
```

TestStep:

```
testRunner.testCase.getTestStepByName("CountryCodes").getPropertyValue("Name")
testRunner.testCase.getTestStepByName("CountryCodes").setPropertyValue("Name","I am in Test Step")
```

Global:

```
com.eviware.soapui.SoapUI.globalProperties.getPropertyValue( "Name")
com.eviware.soapui.SoapUI.globalProperties.setPropertyValue( "Name","I am in Global Prop" )
```

Add Property:

```
testRunner.testCase.testSuite.project.addProperty("DOB")
```

Remove property

Project:

```
testRunner.testCase.testSuite.project.removeProperty("Name");
```

TestSuite:

```
testRunner.testCase.testSuite.removeProperty("Name")
```

Test cases can be run from GUI or from Groovy, which can be coded into a groovy script in the following manner:

Groovy - run request from same TestCase

```
def status = testRunner.runTestStepByName("TestStepName")
def result = status.getStatus().toString();
log.info (" ---- "+result)
```

Groovy - run request from another TestCase or TestSuite

```
project = testRunner.testCase.testSuite.project ;
tcase = project.testSuites["TestSuiteName"].testCases["TestCaseName"] ;
tstep = tcase.getTestStepByName("TestStepName");
def status = tstep.run(testRunner, context)
def result = status.getStatus().toString();
log.info (" ---- "+result)
```

Groovy code to run test case

```
def tCase = testRunner.testCase.testSuite.testCases["TestCaseName"]
runner = tCase.run(new
com.eviware.soapui.support.types.StringToObjectMap(), false);
```

Groovy script to run Test Suite:

```
def suite =
context.testCase.testSuite.project.testSuites["TestSuiteName"]
suite.run(null,false)
log.info (" === "+suite.getName().toString()+" - Executed
successfully")
```

Groovy script to run Project:

```
def
projectName=testRunner.getTestCase().getTestSuite().getProject().getWor
kspace().getProjectByName("REST Project 1")
projectName.run(null,true)
```

Groovy can be used to validate the status of each Test Case and use the return result to display status.

```
def status = testRunner.runTestStepByName("CountryISOCode")
def result = status.getStatus().toString();
if(result == "OK"){
      log.info("Test Passed")
}
else{
      log.info("Test Failed")
}
```

Property Transfer amongst Test Steps:

While testing a service, there might be situations where the output of an API would want to be passed as the input to another API. This can be achieved using Property Transfer, which prevents the manual

plugging in of values. To demonstrate this, we are using a Test case with two test steps, amongst which a transfer has to be made.

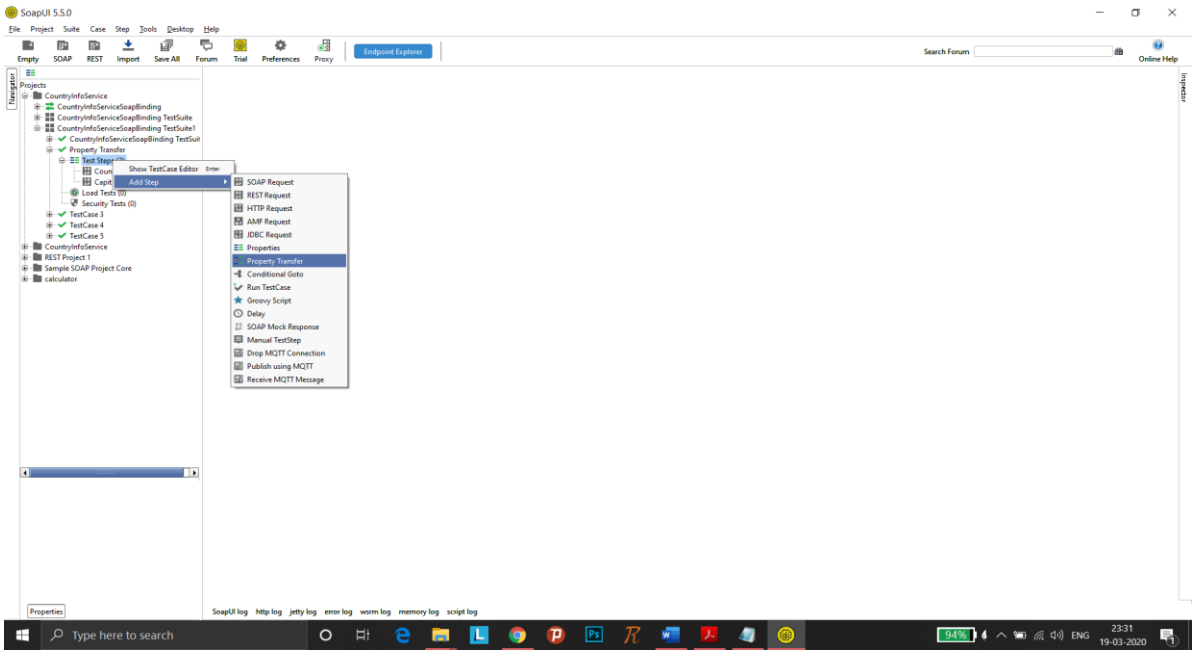


Fig 21: Creating a Property Transfer Step

In our example, we are passing the Country Name as the parameter, which outputs the ISO code of the country and passes it onto the CapitalCity API, which outputs the Capital City of the ISO received from the CountryISOCode API. The following needs to be added into the Property Transfer in order to validate the response and request being received and sent from respective APIs.

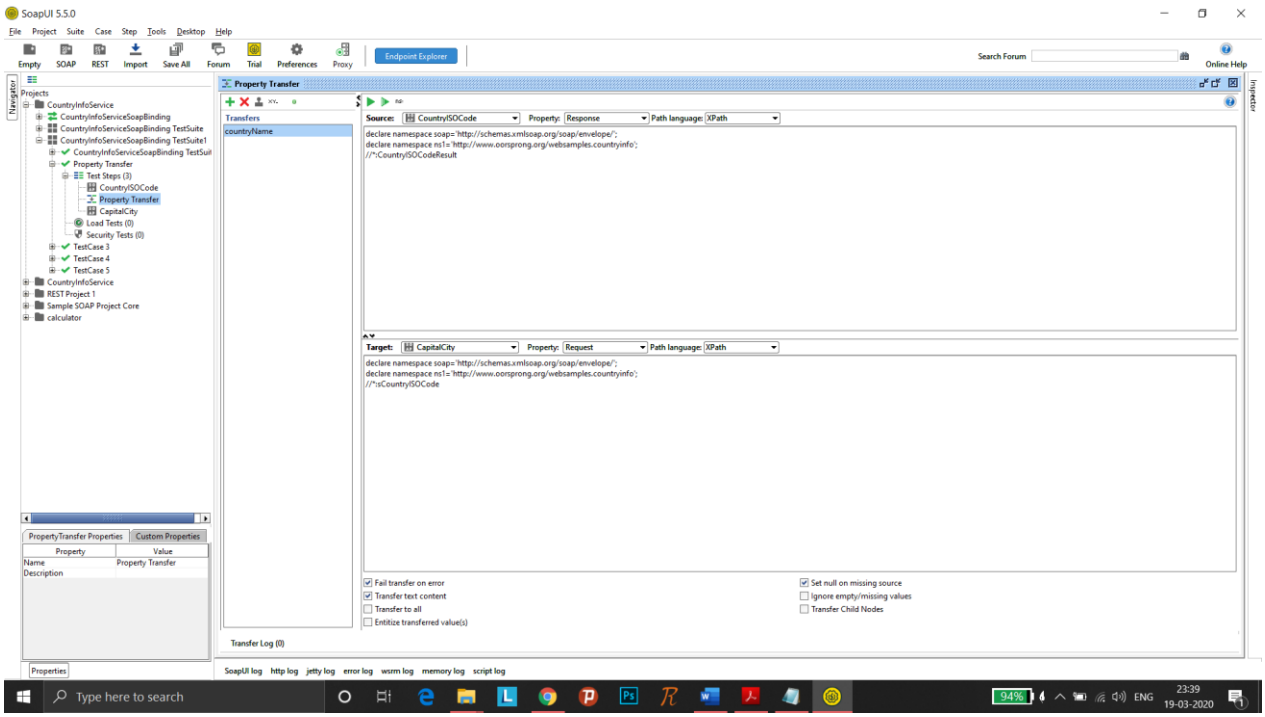


Fig 22: Property Transfer Demo

The following can be coupled along with property from the Test case to provide global input:

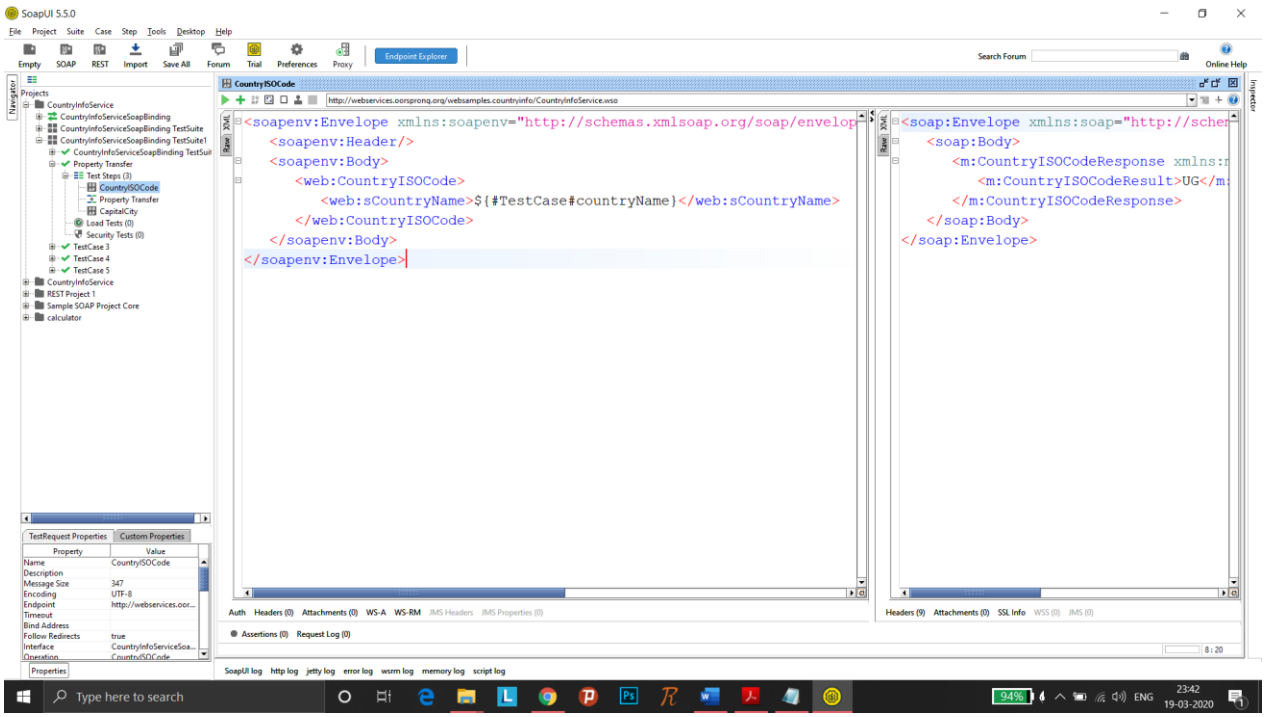


Fig 23: Test Case Property used in Property Transfer

Load tests:

SOAP UI offers the facility of testing the load on each Test Step by providing borderline variables and measures the Service's Quality of Service(QoS) Performance. It displays the minimum value, maximum value(borderline values) being applied onto to the Service to test the load factor which can be withstood by the Service APIs.

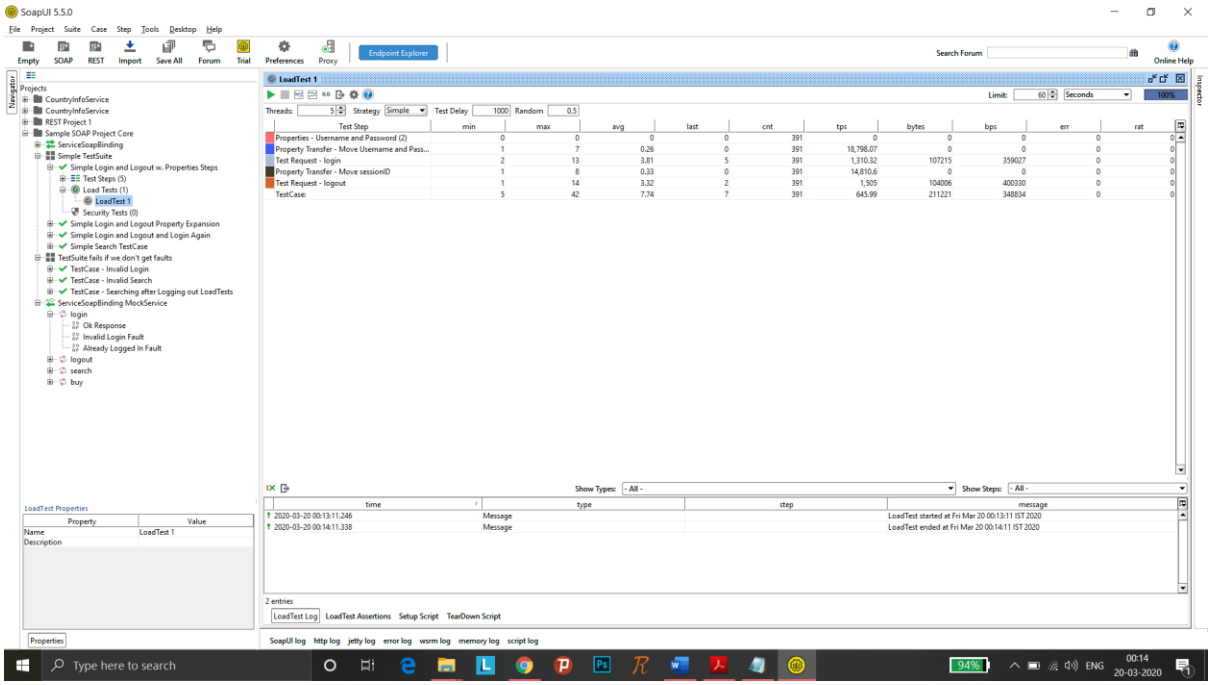


Fig 24: Load Testing

Sample Test Run for a service implementing sample Login Interface:

In order to implement the above-mentioned features of SOAP UI, a simple service is used to login using a random session ID and provide end to end validation. The Service uses a Soap Binding Service, which needs to be initialized before any APIs can be tested.

Test Case 1 – Validation of Username and Password Combination with Session ID

The username and password are being retrieved from the Test Step level property. The Test Request Step accepts the User Login and Password from the property and Validates the same to generate a session id. The Test Step also uses an assertion to validate if the output and check if the generated session id is random and is a numerical.

A property transfer is used here to transfer the username and password from the property to the variable being used in the Test Step. Which is used as:
Loginn${=String.valueOf(Math.random()).substring(0, 5)}

The generated Session ID needs be transferred to the Test Request – Logout Step. The logout step uses the generated session ID to logout the user. The test case also checks if the password supplied into the property and if the supplied password is incorrect, the assertion throws an error. The assertion also checks for redundant users or users who have already logged in and trying to login again, which will set the assertion with an error and throws an error stating “The User has already been logged in” and similarly for the logout API, which will throw an error “User already logged out” in case the session id of already logged out user is passed to the API.

With the following test case, we were able to test an end to end flow scenario for the user login and logout.

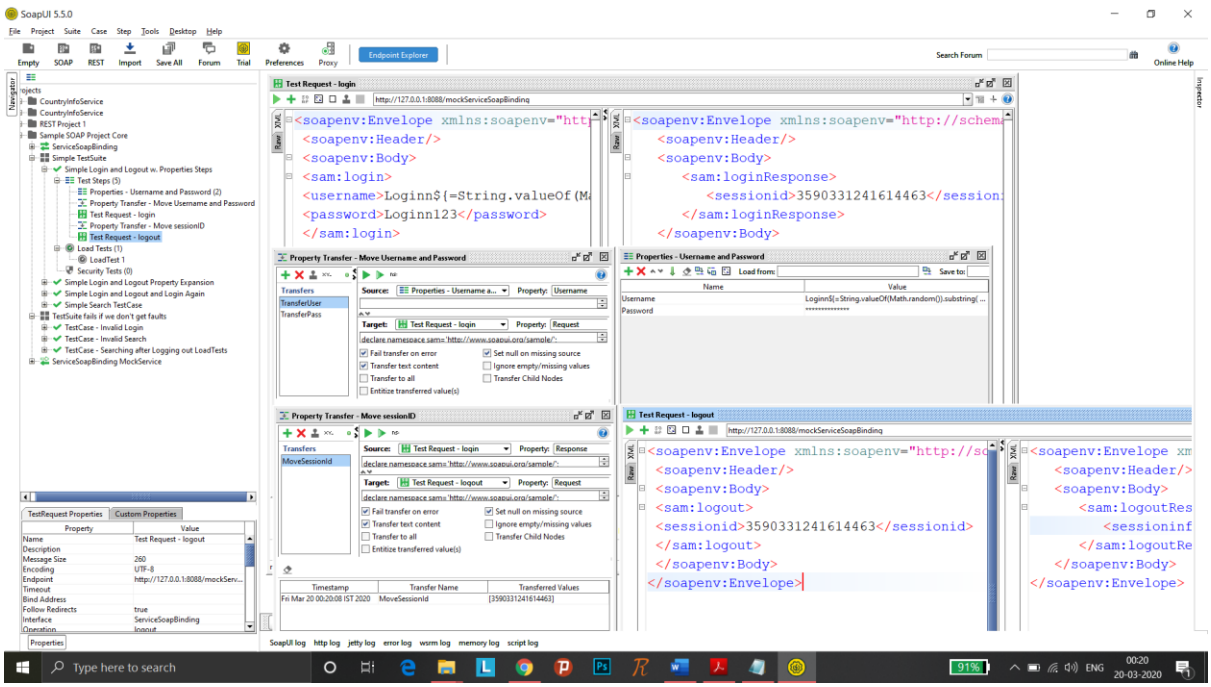


Fig 25: Demonstration of Validation of Username, password and Session ID test case

Test Case 2 – Validation of Username

As per the requirement of the Web Service, the Username has certain requirements and we would want to test this to check if the criteria for the username is being met. We have a test case called Test Case – Invalid Login to validate the entered username. The below image demonstrates each of the test step, which demonstrates the case when the username is wrong, when the password is wrong, when there is a double login attempted, when the username is too long or short. The assertion turns red and indicates an issue in the test case if any of the test case fails. In the following example, a double login was attempted for the same session id and hence the assertion has become red.

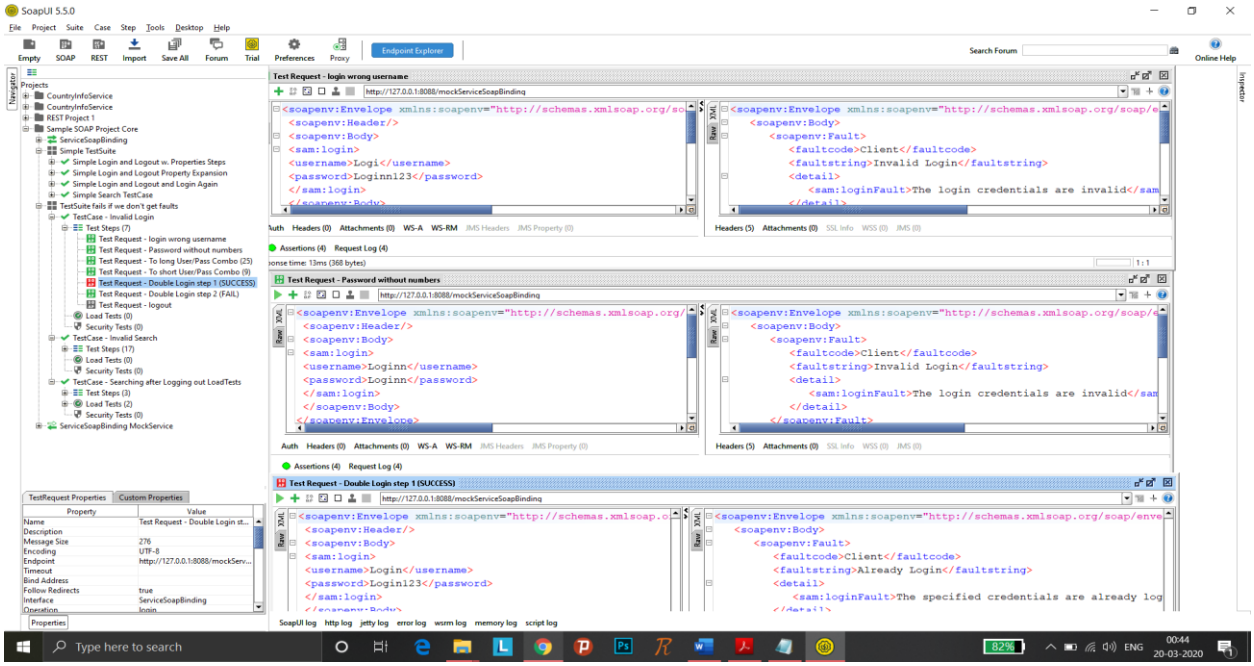


Fig 26: Validation of Username Testcase

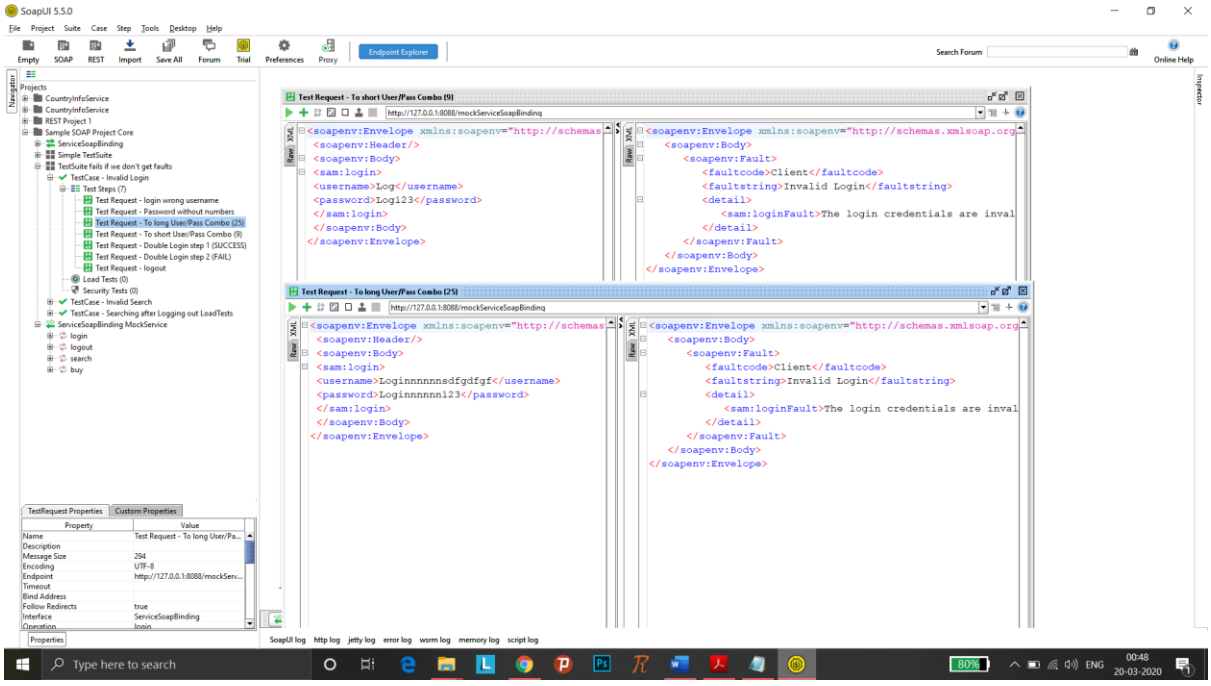


Fig 27: Validation of Username Testcase

Test Case 3 – To check if the login access of user while accessing content

In any webpage, the right user must have access to the right information, and this is crucial. This flow needs to be validated to check that the page does not forward to the information page if the logout has already been generated. We try to check this scenario with our test case “Test Request - Correct Search after Logout”. As per the right flow of the web service, the assertion validates if the assertions on “Correct Search after logout” turn green if the user have already logged out. If we use just the login API and then use the Correct Search after logout, then the content is displayed, and the trigger turns red as it is designed to check for the “Invalid session ID” response.

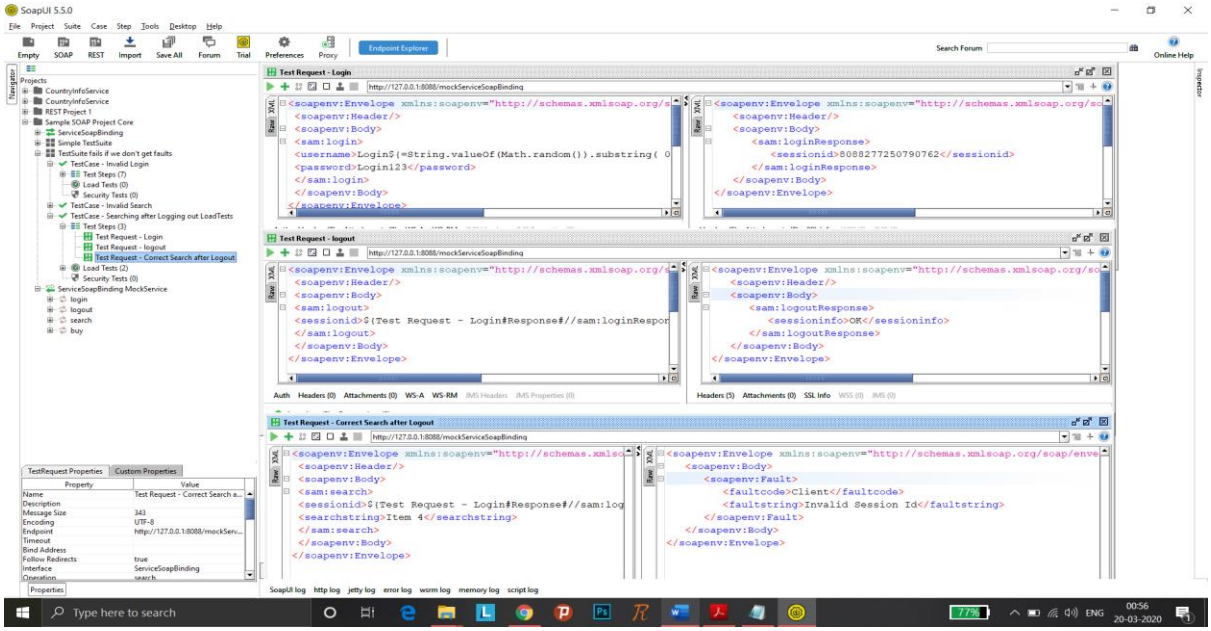
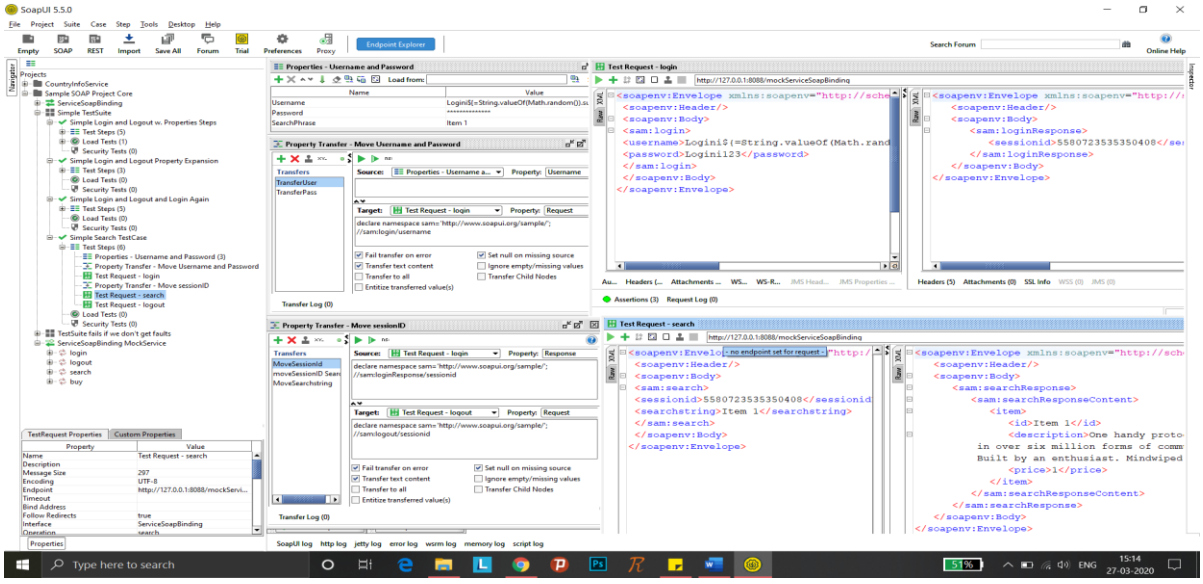


Fig 28: Validation of user access to content

Test Case 4 – Validating sample flow from login -> search -> logout

This test case is tested in order to demonstrate the sample flow of the login, logout and search APIs. The test case also makes use of the property transfer in order to transfer the session id from the login API to the logout and search API as a valid and common session id is required to be used across the API. Any changes in the session ID or if the item we are searching for isn’t available or valid, the test case throws an error stating “Unknown Search String”. The assertion turns red in this case and informs that there is a problem with the test case.



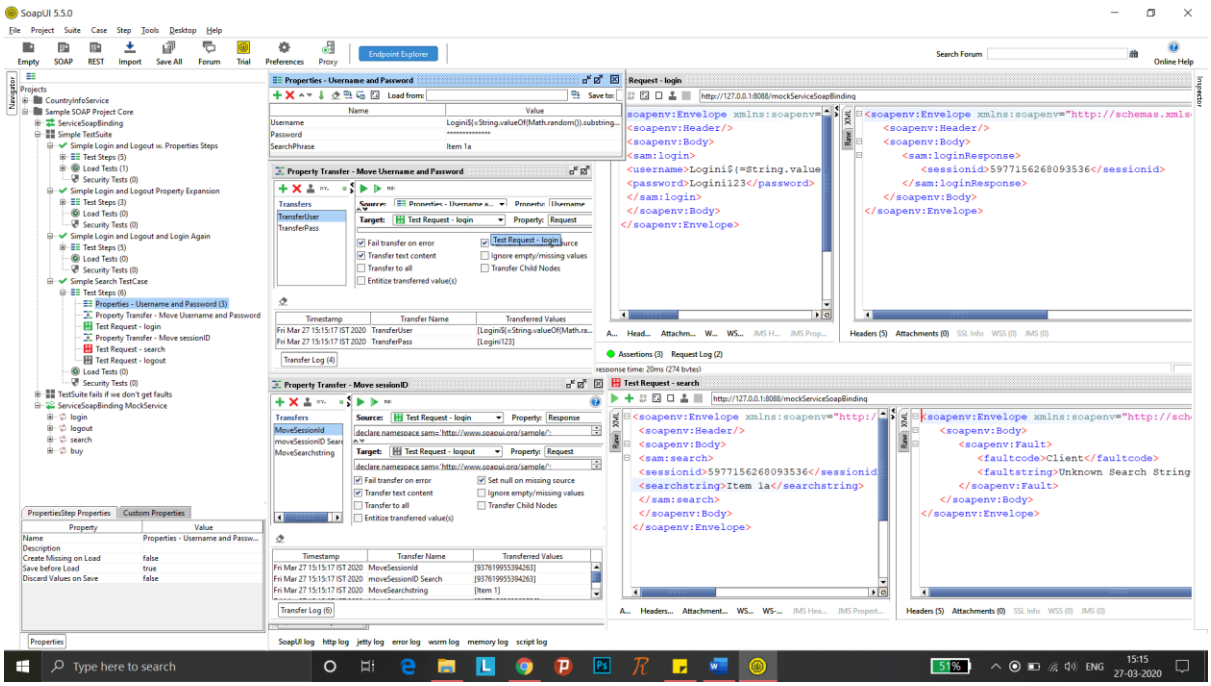


Fig 29 & 30: Test case to validate the sample flow from login -> search -> logout
Fig 29 Illustrates the success flow and Fig 30 illustrates the failure scenario when item isn't found

Coverage Methods:

The **graph coverage** covers all the logical states that can be reached depending upon inputs provided at each node. This can be seen in SOAP UI by using the concept of properties and Test cases. The test cases can be provided with multiple values in order to witness the different behavior of the APIs for differing values of inputs. In addition to the properties, the assertions validate the behavior of the API to further have a better coverage of the Test case. Additionally, SOAP UI provides test debugging which allows us to watch the test execution step by step and hence we can follow the outcome of each input. Also on using the REST service, the RAW table can be used to add user defined variables in the API to generalize the structure so that values can be generalized using user inputs or properties. All of these factors help in aiding the graph coverage of the tool.

The **Input Space** can also be monitored and validated using the properties as the user can chose to pass inputs and also the input values provided can be transferred amongst APIs using the concept of property transfer, which enables the testers to automatically transfer certain values such as username or session ids etc. without manual interfering. In addition to this, we can use Groovy Scripts in order to validate the given inputs and the output received using basic Java syntax. The following input space domains and their partitions could be observed in each of the test cases:

Test Case 1 & 2 - Validation of Username and Password Combination with Session ID

Input Space Domain	Partitions
Username and Password	Matching Username and Password
Username and Password	Incorrect Username and Password Combination
Username and Password	Incorrect Username
Username and Password	Incorrect Password
Session ID	Expected Session ID
Session ID	Incorrect Session ID

Test Case 3 & 4 – Login access and viewing content

Input Space Domain	Partitions
Username and Password	Matching Username and Password
Username and Password	Incorrect Username and Password Combination
Username and Password	Incorrect Username
Username and Password	Incorrect Password
Session ID	Expected Session ID
Session ID	Incorrect Session ID
Search Phrase	Correct Search Phrase
Search Phrase	Incorrect Search Phrase

The **syntax of Groovy Programming** is very simple, and java based and doesn't require much effort to write. In addition to this, object orientation and other concepts are also available in Groovy and the output from the test cases / suites can be used to perform concept operations. As the operation is mostly GUI based, the operation of the tool becomes very easy and even an amateur user can implement all the functionalities of SOAP UI. Advanced test cases in SOAP UI can have Security testing, load testing, Performance testing and other additional coverage methods. SOAP UI also provides a range of assertions which can be used to test APIs and also provides a documentation page for the service which provides the functionality and other details of each API, which makes it easier for the tester to better understand the scope of testing.