# Task 1:

*WEB UI test:*

*The web application allows users to add politicians to a database through a web form. The following fields are mandatory: name, country (country code or full name), date of birth (mm/dd/yyyy), position (string), url (where the info was extracted from, can be a fake url in valid format), risk level (list selection). The data format is also validated. If data was successfully added, a popup will confirm it, showing the name of the politician. This process is asynchronous, it might take a few seconds.*

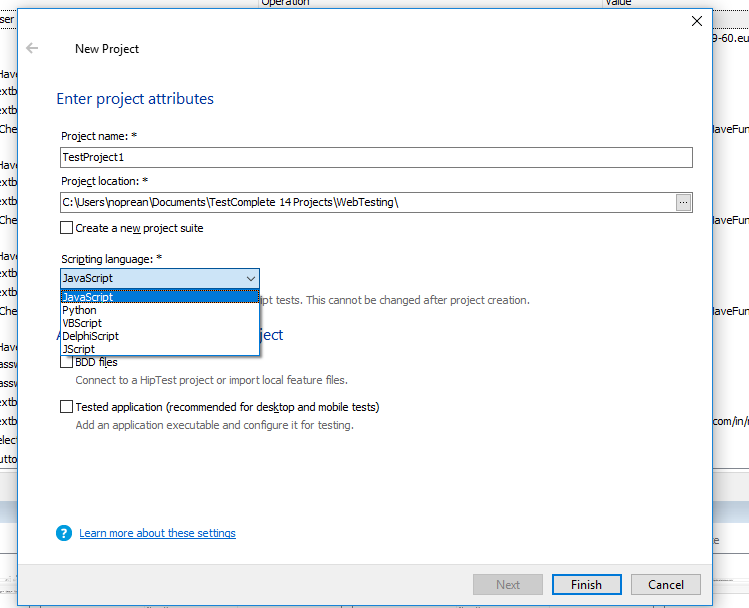
*Write most relevant test scenarios with the automation tools of your choice, please also describe the scenarios as simple as possible in human-readable format, report any bugs you encounter. Extra points are added if the automated tests are reporting bugs (in console, html format, text file, test runner, etc.) as test failures.*

*Please explain how the test suite can be run.*

*The UI can be accessed at*

[*http://ec2-34-250-139-60.eu-west-1.compute.amazonaws.com/*](http://ec2-34-250-139-60.eu-west-1.compute.amazonaws.com/)

For this task I selected Test Complete tool that can easily be integrated with Web UI Testing. As alternatives I looked also over **Selenium Web driver** and **Webdriver.io**. I selected Test Complete with a trial 30 days Web Testing license, because I was familiar with it and it allows you to choose between 5 different scripting languages. I Selected Java Script:

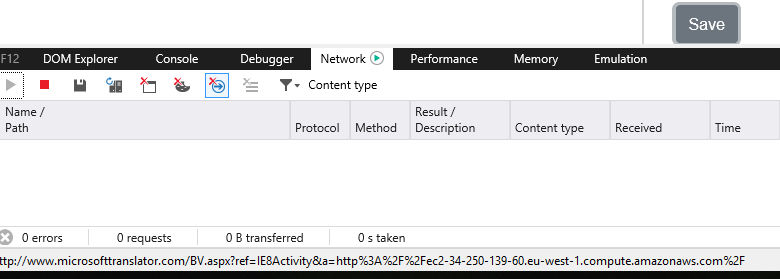


After creating the project I started creating test scenarios based on the requirements from the task.

1. I started with the basic flow scenario: fill in all the fields with correct data and submit the data and check if the data is sent to the server. Even from this point I noticed **(Bug 1)** that no request is sent to the server when the user fills in the correct data. Right after that I noticed **(Bug 2)** that the data displayed on the popup did not mention anything about the Name Entered ***aka. My name*** but the default John Doe.

2. After the first automated scenario I said I should focus on each field apart so I started creating second automated test scenario where I did basic checks like **(Bug 3)** if not all the fields are entered the Save button should be disabled. Tried to add multiple checks and raise bug reports on warnings/errors. As a note Test Complete is integrated with Zephyr for Jira and allows you to raise bugs automatically. As I did not have full platform only a trial license and taking into account the short amount of time I had I was not able to make also the integration with Jira part but this can be done easily.

Moving on I figured out that **(Bug 4)** you are allowed to leave the **Full name** field blank but still be able to proceed with the form submit or **(Bug 5)** there is no check made on data entered on Full name field or any other field. You can enter any character you want on the Full name field or Country even inside the Year of birth. There is no limit on the number of characters entered **(Bug 6)** on Full name field or Country or anything else. Even if all the data you enter is correct the data is not transmitted to the server.



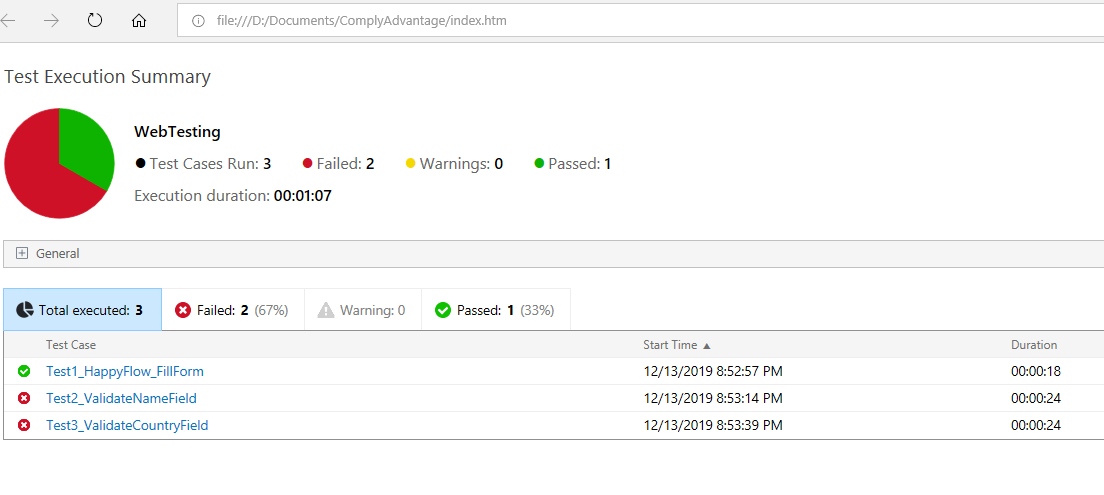
3. I then moved to next field (Country) where I spotted **(Bug 7)** that the selection of the Country should be done from drop down or the risk of having wrong data inside the database is very high. There is no check on the data entered **(Bug 8)** allowing you to enter any data you want. Also **(Bug 9)** you are allowed to submit the form even if the country is not filled in or with incorrect data on it.

4. I did not have time to continue with the rest of automated test scenarios because the time was very short but you got the idea that this is doable (I only had 3 days for research and implementation), but we can discuss on this if you want how I was thinking to develop further and integrate with Jenkins and create common Java Script library to be reused inside multiple tests, and create reusable Business logic specific to each functionality apart, however spotted many other issues:

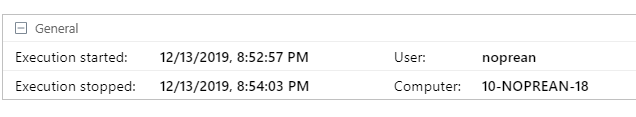
* **Bug 10**: year of birth is not validated with integer data;
* **Bug 11**: year of birth can be left blank;
* **Bug 12**: year of birth data is not submitted;
* **Bug 13**: Position should not be a password field and data should be visible;
* **Bug 14**: Position cam be left blank;
* **Bug 14**: Source URL does not validate the data if you do not enter the https schema
* **Bug 15**: Source URL should allow you to enter any link with http or ftp or any other schema. The rule should be that it contains a valid domain name: testdomain.extension where the extension is more than 2 chars long.
* **Bug 16**: Source URL should not allow you to submit the form if the data entered is “Invalid URL format” and the data apparently is submitted as it is entered.
* **Bug 17**: Risk level Drop down does not contain consistent data and the ***HUGE*** option is not similar to other 3 options.

I grouped the automated tests into a smoke test suite that allows you better handling, you can parametrize tests specific to your will. Also you have the option to easily integrate with Jenkins and run only certain Test Suite groups organized as virtual folders but referenced as Jenkins job parameters. Also they can be scheduled and spitted into multiple jobs and functionalities.

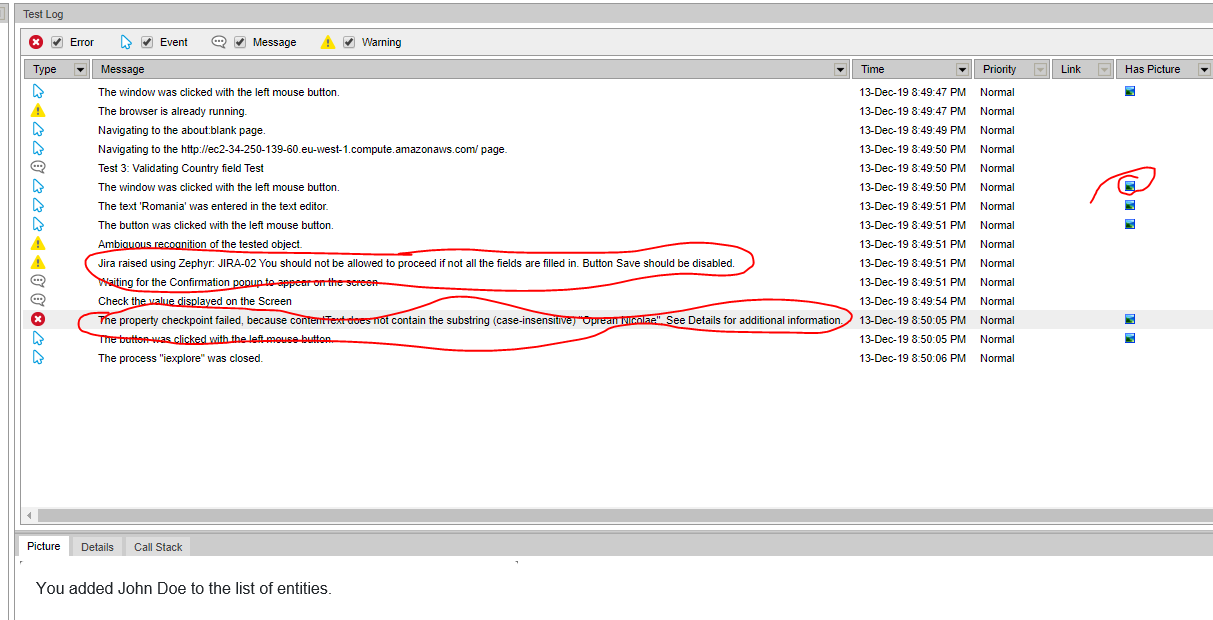
The test report generated will be attached to the e-mail with the results that I will send to you but basically if you just click on the index.htm you can see full report with the test steps performed.



You can expand the General section and you will notice what was the running time and more details about the execution machine and the User running them.



You can then click on every test and you can see the steps performed and where it was throwing warnings or errors. Normally on a real test scenario I would throw error and stop the execution if you did not fill in all the required fields or if data validation is wrong but here I raised warnings so that I can do multiple checks with the same test.



This preliminary project was built today in a couple of hours but it can be easily extended, enhanced with Jenkins integration, performance testing, coverage and we can discuss more around it. In order to be able to run this project you will need free test complete trial with platform and web modules on it: <https://smartbear.com/product/testcomplete/free-trial/>

You can ask for one license by filling in the form on the link I sent you or you can give it a try with my trial license for TestComplete Web with its platform but it might be seen as already activated.

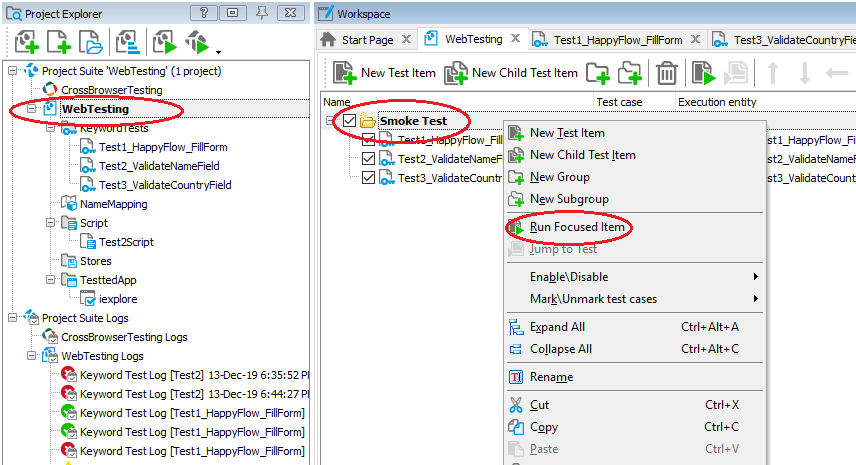
- Product key for the new license: b2a78b54-ba29-4c69-a191-8ab93c955d99

- Product key for the new license: d1a74a57-b169-45eb-935b-cd26400bcc8a

PS: You can tell me when you want to test and I think I can disabled it on my machine and you can enable on yours. There is only one restriction for the license to only be activated on a single machine at a time and it is only valid for 30 days.

The link of the project and the test results can be found at this location: <https://github.com/nickoa2001/HaveFunTesting>

You just needs to be download the test project or the test results. If you want to open the test project you need to open WebTesting.pjs using Test Complete. Once you opened the project if you double click on the WebTesting project on the left on the right hand side panel you will see the test suite. You can right click on the Smoke Test suite or any test and choose the Run focused item option.



# Task 2:

*API test (REST):*

*The application under test is exposing 3 API endpoints for data manipulation:*

*CREATE* [*http://ec2-34-250-139-60.eu-west-1.compute.amazonaws.com/peps*](http://ec2-34-250-139-60.eu-west-1.compute.amazonaws.com/peps)

*- allows adding politicians data, the response contains the generated id for the entity and other fields*

*READ* [*http://ec2-34-250-139-60.eu-west-1.compute.amazonaws.com/peps*](http://ec2-34-250-139-60.eu-west-1.compute.amazonaws.com/peps)

*- returns the list of the latest 5 politicians, order by date of creation*

*READ* [*http://ec2-34-250-139-60.eu-west-1.compute.amazonaws.com/peps/{id}*](http://ec2-34-250-139-60.eu-west-1.compute.amazonaws.com/peps/%7bid%7d)

*- returns a politician information, by its id*

*The following fields are mandatory: name, country (country code or full name), yob (year of birth), position (string), risk (int number from 1 to 5). Write most relevant test scenarios with the automation tools of your choice, please also describe the scenarios as simple as possible in human-readable format (a good title is enough), report any bugs you encounter. Extra points are added if the automated tests are reporting bugs (in console, html format, text file, test runner, etc.) as test failures. Please explain how the test suite can be run.*

*NOTE: data added from the web form is not accessible through API.*

In order to accomplish this task I made a bit of research and selected the Soap UI Pro open source <https://www.soapui.org/downloads/soapui.html>

This tool is also developed by SmartBear but I found it very easy to use and even though it was the first time using it after looking over a bit of documentation I was able to use it successfully and do the task.

The documentation used was mainly from these locations:

<https://www.youtube.com/playlist?list=PLrA5ciulugn_w0pW1_SSyfucD-k5Fasj_>

<https://www.linkedin.com/learning/api-test-automation-with-soapui/learning-soapui-for-api-testing?u=2287137> but I found plenty of more documentation and tutorials on the web.

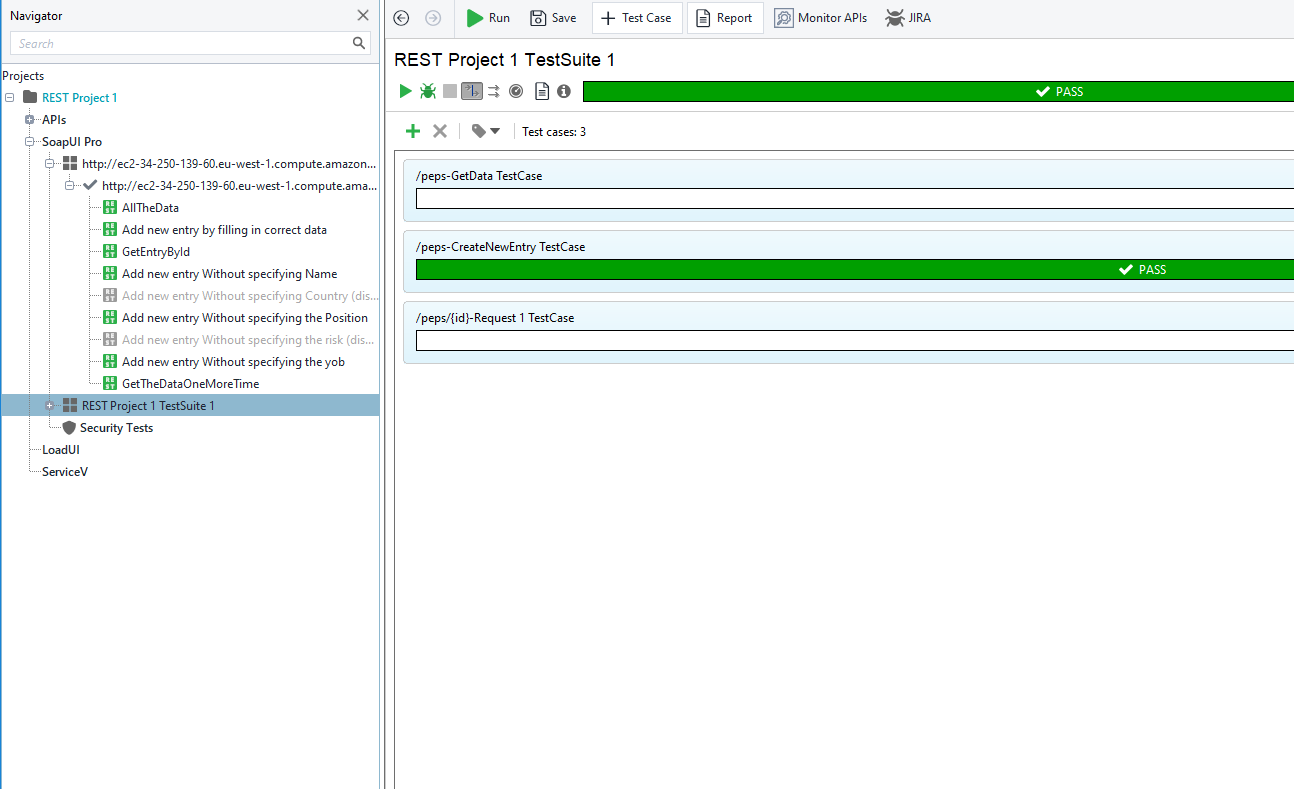
I used Ready API (again a trial version) that integrates beside Soap UI also Load UI and Service: <https://smartbear.com/product/ready-api/free-trial/>

I started by creating the project and then the object that represent the service that I tried to test.

I provided the base URL and then played with the requests by thinking over a way to automate and create test scenarios with these requests and made several assertions.

I used POST Requests to send data to the server and GET Request to get the data from the server.

I grouped the test scenarios inside the:



The project can be found at this location:

<https://github.com/nickoa2001/HaveFunTesting/REST-Project-1-readyapi-project.xml>

You will need to load the project (xml file) inside the Ready API and run the tests. I spotted that there are two bugs:

1. If you don’t specify the country you are allowed to push your data and a new entry is being created **(Bug 1)**. The test is failing but I disabled it in order to have a successful run. In order to check the assertion and run the test you will need to re-enable the test. The data used for testing is the following:

{

"name" : "Nicolae Oprean",

"position" : "QA Engineer",

"risk" : 7,

"yob" : 1982

}

With this input data we receive from the server the message: ***Entity created successfully!*** When it should in fact reply ***Invalid request - missing parameters*** hence the bug.

2. If you don’t specify the risk again you are allowed to push your data and a new entry will be created:

{

"country" : "RO",

"name" : "Nicolae Oprean",

"position" : "QA Engineer",

"yob" : 1982

}

With this input data we receive from the server the message: ***Entity created successfully!*** When it should in fact reply ***Invalid request - missing parameters*** hence the bug.

Test ***Add new entry Without specifying the risk*** is failing.

The test results can be found here and is illustrating the two scenarios that are failing:

<https://github.com/nickoa2001/HaveFunTesting/blob/master/ReadyApiTestReport.pdf>