README

KARATE\_TEST AUTOMATION

The project below was part of a technical Assessment for ABSA

 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

***ABSA Technical Assessment***

1. *Using java karate framework*
2. *Construct/Send the following xml payload to this endpoint*[*https://verifye.co.za/response.php*](https://verifye.co.za/response.php)*using an xml parser/library*
3. *The endpoint will return the message payload you send it, validate the response that is returned using xpath validation for tags/values.*
4. *Display use of parameterization, test data, repeatable and reusable functions.*
5. *Check your framework into****github****and provide me with the repository link*
6. *Ensure there is a read me file with instructions on how to clone, install, execute your project, and view the test results*
7. *Please feel free to include anything over and above what I’ve outlined*

Clone:

Clone the repository with this command:

git clone <https://github.com/Teeran/TeeranAbsaAssignment.git>

**Folder Structure**:

Text

Description automatically generated

**How to Execute**:

1. Feature File: - Go to specific feature and click on the execute button next to the scenario

Note: You can either run at feature level or at Scenario.

Text

Description automatically generated

1. Test Runner: - Edit runner file with correct @tag and execute runner file

Graphical user interface, text

Description automatically generated

1. MAVEN: - mvn clean test -D tags="@RunwithTagCheck"

Graphical user interface, text, application

Description automatically generated

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Parameterisation:**

#(Placeholder) was implemented to parameterise the whole payload which is mapped in an XML file that was created under “payloadValidation” folder. Using the payload provided.

Text

Description automatically generated

For every execution, the data set from the Examples were used to populate the payload which would in turn be used to hit the endpoint. [Feature file]

**Validation**:

Validation was done on all the fields for the response received. All the fields were validated against the data set from the Example [Feature file]

Please note all the Validation steps are separated in a different feature file;

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

The above feature is then called in the main feature file (MyFirstTest.feature) for it to be executed all the way at the end just after the response has been received.

**RE-USE**:

A screenshot of a computer

Description automatically generated

1. Re-use of Method from JAVA class
2. Re-use of functions from one Feature to another.

Made use of random Number (**JAVA Method**) and Current Date (reusable **Feature function**) to concatenate with an already present value for MSGID starting value such as “021/DDINP/Puleng/”

**REPORT**:

After execution, a report is generated to:

*Src/Target/report*

*A picture containing text

Description automatically generated*

And the full path will be printed in the console, paste into browser to view:

*/target/cucumber-html-reports/overview-features.html*

Note: Reuse a snippet of code from internet to convert cucumber json report to a more presentable HTMl report, rather than the normal karate HTML report.

MORE:

@tag, Scenario Outline, Examples were used