**Karate feasibility document**

[REST-Assured](http://rest-assured.io/) is one of the most established API testing tools in the market. It has a powerful Java-based API that leverages [Hamcrest](http://hamcrest.org/) as the matcher tool. It also has a simple set of methods that allows method chaining and Pseudo-Gherkin Syntax.

Karate , on the other hand, is a Gherkin-based API (readable API specifications)testing tool. It’s ideal for quick prototyping as well as simpler testing that can be as in-depth as you want. Karate includes a simple quick start mode, integrated dashboards, and other goodies that are gaining attention from the testing community.

Why we need Karate:

1.Open source

2.Api testing and mocking..

3.Performance testing also

4.Ui automation and reporting

5.Parallel execution

6.BDD Style, supports JSON/XML also

7.Readable syntax

8.Mix API and UI test- automation

9.Dashboard for report

Karate automation test framework:

Karate using visual studio-

How to setup:

1.Go to <https://github.com/karatelabs/karate#quickstart>

Run below command on terminal(make sure java,jdk,mvn(brew install maven…check mvn —version) is installed in your machine).

Go To documents/karate-demo

mvn archetype:generate \

-DarchetypeGroupId=com.intuit.karate \

-DarchetypeArtifactId=karate-archetype \

-DarchetypeVersion=1.3.1 \

-DgroupId=com.mycompany \

-DartifactId=myproject

Above will create project from running karate feature files. You can change myproject to kdemo or some name.

Have visual studio code installed.Install karate runner from extensions.

Open the above generated demo project in visual studio

pom.xml - all plugin dependencies are added, here they are added already

karate-config.js - you can set it for env dev or something else.

Karate works with feature file and runner file.

How to run:

Go to kdemo on terminal

mvn test

Creates report

How to see Report:

Go to targets folder in visual studio

Under it surefire-reports

Karate-summary .html- open in default browser.Install open in default browser from extensions if not there.

Reporting:

Decent dashboard is available for the report.Karate report can be integrated with allure framework. Also cucumber reporting can be integrated in Jenkin .

Karate Report integration with Allure:

Refer below link for the same,

https://www.linkedin.com/pulse/karate-integration-allure-reporting-kushan-shalindra-amarasiri-

Also karate.log

Gives details of all http requests and responses along with headers, params, response time etc…

Response,response status,response headers ,response cookie values can be printed.

Eclipse or IntelliJ IDEA can also be used for developing Karate tests. Both IDEs offer support for JUnit and Cucumber, which Karate can leverage not only for editing but also for running tests.

Karate Pro-30 days trial or paid version for visual studio comes with many benefits(api doc generation so on)..

Reference:

<https://www.youtube.com/watch?v=89WoVbsFyQU>

Comparison of karate vs rest assured:

<https://gorillalogic.com/blog/karate-vs-rest-assured-comparing-powerful-api-testing-frameworks/>

Some useful links referred for the study:

<https://www.softwaretestinghelp.com/api-testing-with-karate-framework/>

<https://github.com/karatelabs/karate/tree/master/karate-demo#example-report>

For response matching use cases,refer below link,

<https://www.baeldung.com/karate-rest-api-testing>

Checks done:

1.Sending multiform data and file and other params- Working

2. Kyc upload and save for at least one category(PAN)- Working

3.Currently running all the feature files by default,check how to run 1 file..(eclipse cucumber integration solves this, check on other editors like visual studio).

Done-This works by having separate runnable test files under each folder.

4.Assertions and error handling part -checked

My PAN upload and save use case related feature file contents:

Feature: sample karate test script for kyc

for help, see: https://github.com/intuit/karate/wiki/IDE-Support

Background:

\* header Accept = 'application/json'

\* header category = 'ID'

\* header Device-Id = 'new-user'

\* header yubi-user-id = 'bcfce898-bd1b-46f1-a088-c5b6764dbd29'

Scenario: upload and save kyc

Given url 'https://retail-qa-api.myyubi.com/kyc/api/v1/flow/kyc/YUBIFIN/documents'

#provide info on location of the file,name of the file,content-type header value

#\*def fileLocation = '../kycPAN/pancard.jpg'

# And multipart file document = { read:'#(fileLocation)', filename:'pancard.jpg', Content-type:'multipart/form-data' }

And multipart file document = { read:'pancard.jpg', filename:'pancard.jpg', Content-type:'multipart/form-data' }

And multipart field category = 'ID'

# And multipart text = { category: 'ID' }

When method post

Then status 200

And print response

\* def documentid = response.documentId

\* print 'document id is: ', documentid

#And match response.message contains 'error'

\* def items =

"""

{

"category": "ID",

"documentType": "PAN",

"flowType": "DOCUMENT",

"documents": [

{

"pageNumber": 0,

"docId": '#(documentid)',

"fileExtension": "jpg"

}

]

}

"""

# items[0].documents.docId = documentid

Given url 'https://retail-qa-api.myyubi.com/kyc/api/v1/flow/kyc/YUBIFIN/items'

And request items

And header yubi-user-id = 'bcfce898-bd1b-46f1-a088-c5b6764dbd29'

And header Device-Id = 'new-user'

When method post

Then status 200

And print response