Automation Test Engineer

Phase-End Project Problem Statement

Get Certified. Get Ahead.

Phase-End Project 3

Non-Functional Testing Using Postman, REST Assured, and JMeter

Project Agenda: To automate functionalities using https://petstore.swagger.io/ REST API services

Scenario:
You are working as a Test Engineer in XYZ Corp. Your company has decided to automate a few functionalities for one of the **Pet Store** companies.

You have been asked to design an end-to-end functionality to automate three REST API services using Postman and Rest Assured.

Tools Required:

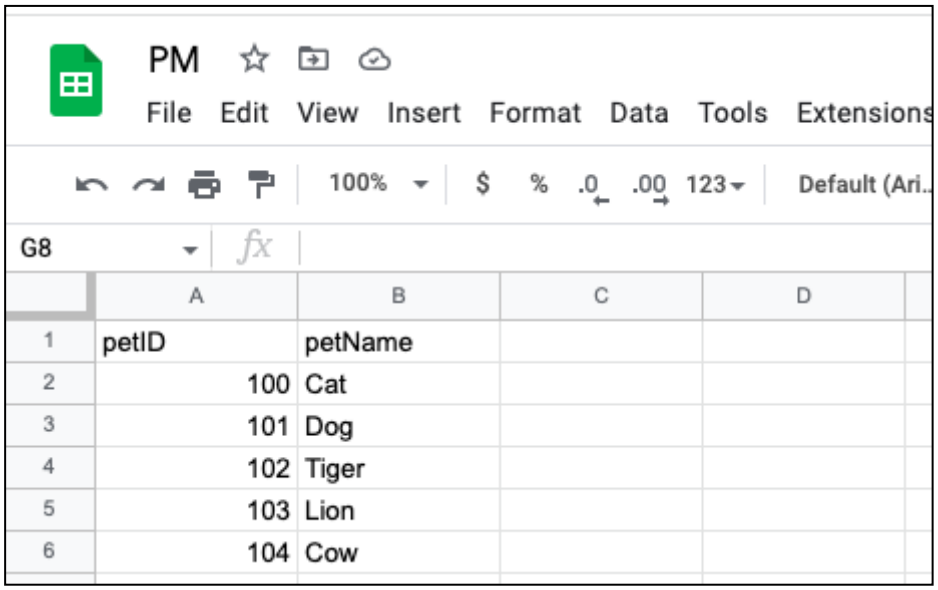
- Postman
- Java 1.8+
- Maven
- Rest Assured Maven dependency version 4.5.1
- TestNG Maven dependency version 7.1.0
- Hamcrest Maven dependency
- Newman for Postman
- JMeter

Expected Deliverables:

Postman Assignment_001:

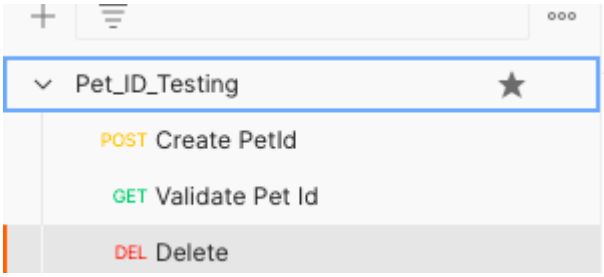
Create petID and PetName:

- Create a CSV file with two columns, petID and petName, having 20 rows' details as below example:



	A	B	C	D
1	petID	petName		
2	100	Cat		
3	101	Dog		
4	102	Tiger		
5	103	Lion		
6	104	Cow		

- Open Postman
- Create a new collection called **Pet_ID_Testing** and arrange three end-to-end services as shown below:



- In the collection, add a new POST service request with following details:
  - URL: https://petstore.swagger.io/v2/pet
  - Service Type: POST
  - JSON Body:

```
{
"id": 344,
"category": {
  "id": 0,
  "name": "string"
},
"name": "Doggie",
"photoUrls": [
  "string"
],
"tags": [
  {
    "id": 0,
    "name": "string"
  }
],
"status": "available"
}
```

- In the JSON body, **id,** and **name**, which are highlighted in YELLOW color, should be parameterized in Postman and runtime data should be driven through the CSV file.
- In the Postman Tests section, validate for this POST call. Response status code should be 200.
- In the Postman Tests section validate for this POST call. Response body should contain text as **'available'**.

Validate petID:

- In the collection, add a new GET service request with the following details:
  - URL: https://petstore.swagger.io/v2/pet/2003
- In the service URL pet **id**, which is highlighted in YELLOW color, should be parameterized in Postman and runtime data should be driven through the CSV file.
- In the Postman Tests section validate for this GET call. Response status code should be 200.

Delete petID:

- In the collection, add a new DELETE service request with the following details:
  - URL: https://petstore.swagger.io/v2/pet/2003
- In the service URL pet **id**, which is highlighted in YELLOW color should be parameterized in Postman and runtime data should be driven through the CSV file.
- In the Postman Tests section validate for this GET call. Response status code should be 200.

Steps to Run:

1. Run collection
2. Select the CSV file to run these End-to-End scenarios 20 times
3. Check the checkbox for Save Response
4. Open Postman Console and Run the collection
5. Validate that all parameterized data [PetID/PetName] should be filled with CSV runtime data
6. All status codes should pass
7. Export the collection as a JSON file
8. Run this JSON collection using the Postman Newman command from cmd/terminal.

Postman Assignment_002:

- Hit a **PUT** call having service URL = https://petstore.swagger.io/v2/pet
- Create a global variable of this URL where **testURL** is the variable name and its value is https://petstore.swagger.io/v2/pet. Use this variable while hitting the URL in Postman.
- PUT Call request JSON body:


```
{
"id": 9223372016900013000, "category": {
"id": 20021,
"name": "string" },
"name": "doggie", "photoUrls": [
"string"
], "tags": [
{
"id": 0,
"name": "string"
}
],
"status": "available_QA"
}
```


- Create 3 test Environments as DEV, QA, PROD. The PUT call JSON body **job** field should be parameterized and its value should change as per environment:
  - When Environment is DEV then "status": "available_DEV"
  - When Environment is QA then "status": "available_QA"
  - When Environment is PROD then "status": "available_PROD"
- Validate id = 20021 in response
- Validate response = 200
- Validate **status** value, if it is changing as per environment in JSON response

Postman Assignment_003:

- Hit a **GET** call with URL: https://petstore.swagger.io/v2/user/Uname001
- Use Uname001 as global parameter
- Validate response as 200 in Postman
- Validate username = Uname001 in JSON response
- Validate email = Positive@Attitude.com in JSON response
- Validate userStatus = 1 in JSON response

Postman Assignment_004:

- Hit a **GET** call with URL: https://petstore.swagger.io/v2/pet/findByStatus
- Use Postman params as **status**
- When **status = available** and if after hitting the URL, response status = 200, then validate for all pet details that their response status = **available**
- When **status = pending** and if after hitting the URL, response status = 200, then validate for all pet details that their response status = **pending**
- When **status = sold** and if after hitting the URL, response status = 200, then validate for all pet details that their response status = **sold**

Postman Assignment_005:

- Hit a **GET** call with URL: https://petstore.swagger.io/v2/user/logout
- Validate response as 200 in Postman
- Validate code = 200 in response
- Validate message = OK in response

REST Assured Assignment:

- Open Eclipse/IntelliJ
- Create a Maven project
- Add Maven dependency for TestNG and REST Assured
- Create a TestNG test with below REST API execution details -
- **POST CALL**
  - URL: https://petstore.swagger.io/v2/pet
  - Service Type: POST
  - JSON Body:

```
{
 "id": 344,
 "category": {
  "id": 0,
  "name": "string"
 },
 "name": "Doggie",
 "photoUrls": [
  "string"
 ],
 "tags": [
  {
   "id": 0,
   "name": "string"
  }
 ],
 "status": "available"
}
```

  - PetID is parameterized
  - Once POST call is successful - validate response code
  - Validate that PetID from response code should be same as request PetID

- **GET CALL**
  - URL: https://petstore.swagger.io/v2/pet/2003
  - Service Type: GET
  - Service URL pet **id**, which is highlighted in YELLOW color should be parameterized and be the same as POST call PetID
  - Validate response code as 200
  - Validate that response JSON body contains keys **status** and **id**
  - Validate **status** value is **available**

- **DELETE CALL**
  - URL: https://petstore.swagger.io/v2/pet/2003
  - Service Type: DELETE
  - Service URL pet **id**, which is highlighted in YELLOW color, should be parameterized and the same as POST call PetID
  - Validate response code is 200
  - Validate that response JSON body contains keys **code** and **message**
  - Validate **message** value is **PetID**
- POST→GET→DELETE services are interrelated. If the POST call fails, then the GET and DELETE call will also fail.
- Note: This TestNG test should run from testNG.xml file
- You can use either TestNG assertion or Hamcrest assertion

Rest Assured Assignment_002:

- Hit a PUT call having service URL = https://petstore.swagger.io/v2/pet
- PUT Call request JSON body:


```
{
 "id": 9223372016900013000, "category": {

                    "id": 20021,
"name": "string"},
"name": "doggie", "photoUrls": [
"string"
], "tags": [
{
 "id": 0,
 "name": "string"
}],
"status": "available_QA"

}
```

- Create three test environments as DEV, QA, PROD, where PUT call JSON body status field should be parameterized and its value should change as per environment:
    - When Environment is DEV then "status": "available_DEV"
    - When Environment is QA then "status": "available_QA"
    - When Environment is PROD then "status": "available_PROD"
- All assertion/validation should be done using Hamcrest.
- Validate response as 200
- Validate id = 20021 in response
- Validate **status** value is changing as per environment in JSON response

Steps to Run this:

1. Create a TestNG test with parameters as environment. Example: putCallTesting(String Env)
2. As per environment name, the value should be read from HasMap and the same value should populate in JSON response.
3. As an example**, putCallTesting("Dev")** method should pick its corresponding value from Hashmap where **Dev** is the **KEY** and its value should be used in the JSON request body **status** field.

Rest Assured Assignment_003:

- Create a TestNG test and implement the scenario
- Hit a GET call with URL: https://petstore.swagger.io/v2/user/Uname001
- All assertion/validation should be done using Hamcrest
- Validate response as 200
- Validate username = Uname001 in JSON response
- Validate email = Positive@Attitude.com in JSON response
- Validate userStatus = 1 in JSON response

Rest Assured Assignment_004:

- Create a TestNG test and implement the below scenario
- Hit a **GET** call with URL: https://petstore.swagger.io/v2/user/login
- Use REST Assured basic authentication with **username** = Uname001 and **password** = @tt!tude
- All assertion/validation should be done using Hamcrest
- Validate code = 200 in response
- Validate response message contains text 'logged in user session'

Rest Assured Assignment_005:

- Create a TestNG test and implement the below scenario
- Hit a **GET** call with URL: https://petstore.swagger.io/v2/pet/findByStatus
- Use REST Assured params as **status**
- All assertion/validation should be done using Hamcrest
- When **status = available** and if after hitting the URL, response status = 200, then validate for all pet details that their response status = **available**
- When **status = pending** and if after hitting the URL, response status = 200, then validate for all pet details that their response status = **pending**
- When **status = sold** and if after hitting the URL, response status = 200, then validate for all pet details that their response status = **sold**

Rest Assured Assignment_006:

- Create a TestNG test and implement the below scenario
- Hit a **GET** call with URL: https://petstore.swagger.io/v2/user/logout
- All assertion/validation should be done using Hamcrest
- Validate response as 200
- Validate code = 200 in response
- Validate message = OK in response

Jmeter Assignment:

- Open JMeter
- Validate https://httpbin.org/basic-auth/user/passwd link using HTTP Authentication manager
- Default username is **user**, and the default password is **passwd**
- Validate response is a JSON file using JSON assertion
- Using HTTP sampler, hit https://www.simplilearn.com and validate xpath as //img[@title='Simplilearn - Online Certification Training Course Provider']
- Increase the thread count and do lead testing