PLAYWRIGHT

API AUTOMATION

Scenario 1: Validate GET API returns the status code 200, and correct JSON data. Also handling the Authentication in order to capture the Access Token which in turn can be used for further requests to get the data info.

Steps:

1. Perform the login
2. Gather the Access Token
3. Use the Access Token as the Authentication to get the user details

Given Data:

Login URL: “https://dummyjson.com/auth/login”

Example credentials:

- username: emilys
- password: emilyspass

GET URL: “https://dummyjson.com/auth/me”

Code:

```python
#Scenario 1: Validate GET API returns the status code 200, and correct JSON data.
from playwright.sync_api import Playwright


def test_TC1(playwright:Playwright):

    API_Session = playwright.request.new_context()

    #Login: [POST]
    Login_API_Response = API_Session.post( url: "https://dummyjson.com/auth/login",
                                data={"username":"emilys", "password":"emilyspass"},
                                headers={"accept":"application/json"})

    print(f"POST Response: {Login_API_Response.json()}")

    Login_JSON = Login_API_Response.json()

    AccessToken = Login_JSON["accessToken"] #Capture the Access Token

    #Using the Access Token to get the data:
    GET_API_Response = API_Session.get( url: "https://dummyjson.com/auth/me",
                                       headers={"Authorization":AccessToken, "accept":"application/json"})

    assert GET_API_Response.status == 200
    print(f"GET Response: {GET_API_Response.json()}")

    GetData_JSON = GET_API_Response.json()

    assert GetData_JSON["email"] == "emily.johnson@x.dummyjson.com"
```

JSON Response in the Output:

```
PS D:\Python\PythonProject6\API_Automate> pytest -v -s test_scenario1.py
============================= test session starts =============================
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\HP\AppData\Local\Programs\Python\Python314\python.exe
cachedir: .pytest_cache
metadata: {'Python': '3.14.0', 'Platform': 'Windows-11-10.0.26100-SP0', 'Packages': {'pytest': '9.0.1', 'pluggy': '1.6.0'}, 'Plugins': {'base-url': '2.1.0', 'bdd': '8.1.0', 'html':
 '4.1.1', 'metadata': '3.1.1', 'playwright': '0.7.2'}, 'JAVA_HOME': 'C:\\Program Files\\Java\\jdk-21', 'Base URL': ''}
rootdir: D:\Python\PythonProject6\API_Automate
plugins: base-url-2.1.0, bdd-8.1.0, html-4.1.1, metadata-3.1.1, playwright-0.7.2
collected 1 item

test_scenario1.py::test_TC1 POST Response: {'accessToken': 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwidXNlcm5hbWUiOiJlbWlseXMiLCJlbWFpbCI6ImVtaWx5LmpvaG5zb25AeC5kdW1teWpzb24
uY29tIiwiZmlyc3ROYW1lIjoiRW1pbHkiLCJsYXN0TmFtZSI6IkpvaG5zb24iLCJnZW5kZXIiOiJmZW1hbGUiLCJpbWFnZSI6Imh0dHBzOi8vZHVtbXlqc29uLmNvbS9pY29uL2VtaWx5cy8xMjgiLCJpYXQiOjE3Njg2NzE1OTAsImV4cCI
6MTc2ODY3NTE5MH0.wbf9sAHNglOt71h8Og6oZZPrTacIBRnSowi9RIIpTQY', 'refreshToken': 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwidXNlcm5hbWUiOiJlbWlseXMiLCJlbWFpbCI6ImVtaWx5LmpvaG5
zb25AeC5kdW1teWpzb24uY29tIiwiZmlyc3ROYW1lIjoiRW1pbHkiLCJsYXN0TmFtZSI6IkpvaG5zb24iLCJnZW5kZXIiOiJmZW1hbGUiLCJpbWFnZSI6Imh0dHBzOi8vZHVtbXlqc29uLmNvbS9pY29uL2VtaWx5cy8xMjgiLCJpYXQiOjE
3Njg2NzE1OTAsImV4cCI6MTc3MTI2MzU5MH0.ECk5UQCZdqO3gU2howZRCH7CXvy5F3ZbP6wjes97hrw', 'id': 1, 'username': 'emilys', 'email': 'emily.johnson@x.dummyjson.com', 'firstName': 'Emily', 'l
astName': 'Johnson', 'gender': 'female', 'image': 'https://dummyjson.com/icon/emilys/128'}
GET Response: {'id': 1, 'firstName': 'Emily', 'lastName': 'Johnson', 'maidenName': 'Smith', 'age': 29, 'gender': 'female', 'email': 'emily.johnson@x.dummyjson.com', 'phone': '+81 9
65-431-3024', 'username': 'emilys', 'password': 'emilyspass', 'birthDate': '1996-5-30', 'image': 'https://dummyjson.com/icon/emilys/128', 'bloodGroup': 'O-', 'height': 193.24, 'wei
ght': 63.16, 'eyeColor': 'Green', 'hair': {'color': 'Brown', 'type': 'Curly'}, 'ip': '42.48.100.32', 'address': {'address': '626 Main Street', 'city': 'Phoenix', 'state': 'Mississi
ppi', 'stateCode': 'MS', 'postalCode': '29112', 'coordinates': {'lat': -77.16213, 'lng': -92.084824}, 'country': 'United States'}, 'macAddress': '47:fa:41:18:ec:eb', 'university':
'University of Wisconsin--Madison', 'bank': {'cardExpire': '05/28', 'cardNumber': '3693233511855044', 'cardType': 'Diners Club International', 'currency': 'GBP', 'iban': 'GB74MH2UZ
LR9TRPHYNU8F8'}, 'company': {'department': 'Engineering', 'name': 'Dooley, Kozey and Cronin', 'title': 'Sales Manager', 'address': {'address': '263 Tenth Street', 'city': 'San Fran
cisco', 'state': 'Wisconsin', 'stateCode': 'WI', 'postalCode': '37657', 'coordinates': {'lat': 71.814525, 'lng': -161.150263}, 'country': 'United States'}}, 'ein': '977-175', 'ssn
': '900-590-289', 'userAgent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36', 'crypto': {'coin': 'Bitcoi
n', 'wallet': '0xb9fc2fe63b2a6c003f1c324c3bfa53259162181a', 'network': 'Ethereum (ERC20)'}, 'role': 'admin'}
PASSED

============================== 1 passed in 1.64s ==============================
```

Scenario 2: Perform the POST request to validate the Response and Response Status Code

Step:

1. Send the Post Request
2. Validate the JSON Response
3. Validate the Status Code

Given Data: For a Single Set of Data

POST: " https://jsonplaceholder.typicode.com/posts"

Payload:

{"title":"QA Automation","body":"Testing POST method","userId":1}

Code:

```
def test_TC2(playwright:Playwright):

    API_Session = playwright.request.new_context()

    Post_response = API_Session.post( url: "https://jsonplaceholder.typicode.com/posts",
                                      headers={"accept":"application/json"},
                                      data= {"title":"QA Automation","body":"Testing POST method","userId":1})


    assert Post_response.status == 201
    print(f"POST Response: {Post_response.json()}")

    Response = Post_response.json()

    assert Response["title"] == "QA Automation"
```

JSON in Output:

```
============================================================================== 1 passed in 1.22s ===============================================================================
PS D:\Python\PythonProject6\API_Automate> pytest -v -s test_scenario1.py::test_TC2
============================================================================= test session starts ==============================================================================
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\HP\AppData\Local\Programs\Python\Python314\python.exe
cachedir: .pytest_cache
metadata: {'Python': '3.14.0', 'Platform': 'Windows-11-10.0.26100-SP0', 'Packages': {'pytest': '9.0.1', 'pluggy': '1.6.0'}, 'Plugins': {'base-url': '2.1.0', 'bdd': '8.1.0', 'html
 '4.1.1', 'metadata': '3.1.1', 'playwright': '0.7.2'}, 'JAVA_HOME': 'C:\\Program Files\\Java\\jdk-21', 'Base URL': ''}
rootdir: D:\Python\PythonProject6\API_Automate
plugins: base-url-2.1.0, bdd-8.1.0, html-4.1.1, metadata-3.1.1, playwright-0.7.2
collected 1 item

test_scenario1.py::test_TC2 POST Response: {'title': 'QA Automation', 'body': 'Testing POST method', 'userId': 1, 'id': 101}
PASSED
```

Given Data: For multi-set payload

POST: "https://dummyjson.com/products/add"

Payload:

```
{
        "products": [
          { "title": "Laptop", "price": 1200 },
          { "title": "Smartphone", "price": 800 },
          { "title": "Headphones", "price": 150 }
        ]
      }
```

Code:

```
def test_TC3(playwright:Playwright):
    TC3_DataSet = {"products": [{ "title": "Laptop", "price": 1200 },{ "title": "Mobile", "price": 600}]}
    API_Session = playwright.request.new_context()
    Post_Response = API_Session.post( url: "https://dummyjson.com/products/add",
                                      data = TC3_DataSet, headers={"accept":"application/json"})

    assert  Post_Response.status == 201
    print(f"POST Response: {Post_Response.json()}")
```

If we had a response as:

response = {"products": [{ "title": "Laptop", "price": 1200 },{ "title": "Mobile", "price": 600}]}

Then, accessing the value "Mobile":

response["products"][1]["title"]

Scenario 3: Asserting the invalid requests

Given Data:

Login URL: “https://dummyjson.com/auth/me”

Example credentials:

- username: TESTuser #invalid user name
- password: emilyspass

Code:

```
def test_TC4(playwright:Playwright):
    API_Session = playwright.request.new_context()
    GET_API_Response = API_Session.get( url: "https://dummyjson.com/auth/me",
                                        headers={"accept": "application/json"})


    assert GET_API_Response.status == 401
    print(f"GET Response: {GET_API_Response.json()}")
```

Output:

```
PS D:\Python\PythonProject6\API_Automate> pytest -v -s test_scenario1.py::test_TC4
============================= test session starts =============================
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\HP\AppData\Local\Programs\Python\Python314\python.exe
cachedir: .pytest_cache
metadata: {'Python': '3.14.0', 'Platform': 'Windows-11-10.0.26100-SP0', 'Packages': {'pytest': '9.0.1', 'pluggy': '1.6.0'}, 'Plugins': {'base-url': '2.1.0', 'bdd': '8.1.0', 'html': '4.1.1', 'metadata': '3.1.1', 'playwright': '0.7.2'}, 'JAVA_HOME': 'C:\\Program Files\\Java\\jdk-21', 'Base URL': ''}
rootdir: D:\Python\PythonProject6\API_Automate
plugins: base-url-2.1.0, bdd-8.1.0, html-4.1.1, metadata-3.1.1, playwright-0.7.2
collected 1 item

test_scenario1.py::test_TC4 GET Response: {'message': 'Access Token is required'}
PASSED

============================== 1 passed in 1.94s ==============================
```

Scenario 4: Automating the Chained API calls

Given Data:

Call List API
GET: https://jsonplaceholder.typicode.com/users
→ Extract the IDs of the users.

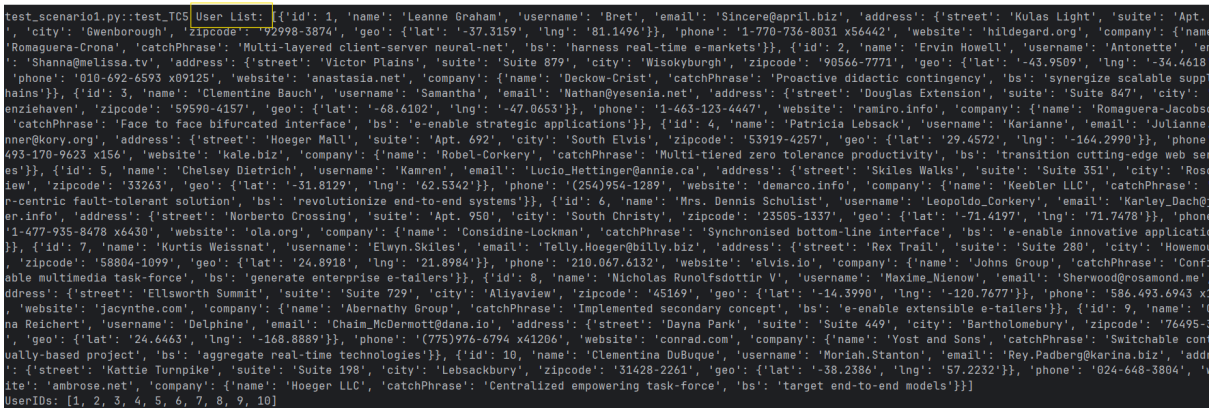
Use ID in Detail API
GET: https://jsonplaceholder.typicode.com/posts?userId={{id}}
→ Fetch all posts by the user with IDs captured.

Code:

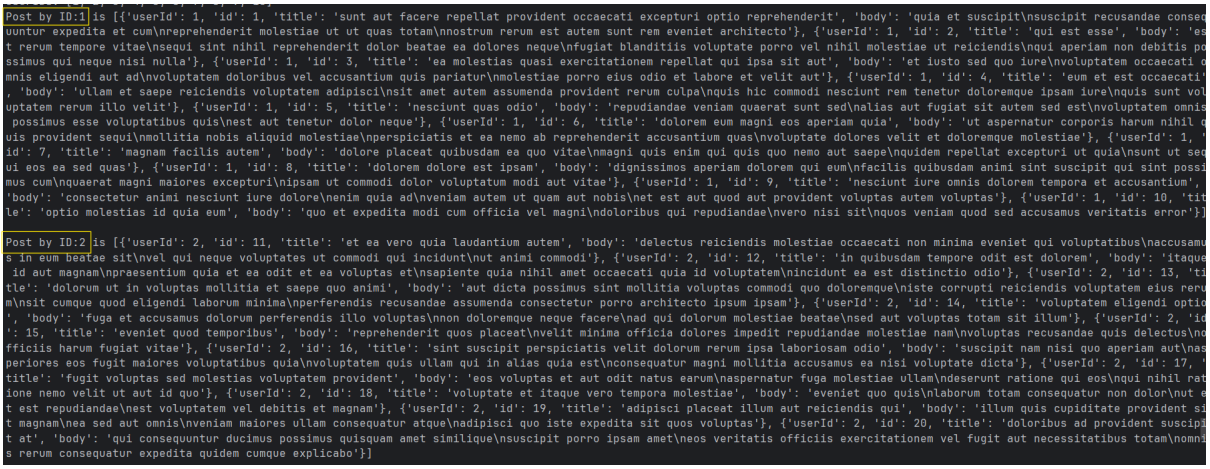
```
def test_TC5(playwright:Playwright):
    API_Session = playwright.request.new_context()

    #Capture the User ID:

    UserList = API_Session.get( url: "https://jsonplaceholder.typicode.com/users",
                               headers={"accept": "application/json"})

    UserList_JSON = UserList.json()
    print(f"User List: {UserList_JSON}")

    UserIDs = []
    for user in UserList_JSON:
        UserIDs.append(user["id"])

    print(f"UserIDs: {UserIDs}")

    # Using the IDs to fetch the Post details for each:

    for ID in UserIDs:

        response = API_Session.get(f"https://jsonplaceholder.typicode.com/posts?userId={ID}")

        print(f"Post by ID:{ID} is {response.json()}\n")
```

Output:

#Capturing the IDs



#Post by IDs



Scenario 5: Checking the Response Time

Code:

```python
def test_TC6(playwright:Playwright):
    API_Session = playwright.request.new_context()

    start_time = time.time() #Records the Start time
    API_Response = API_Session.get( url: "https://jsonplaceholder.typicode.com/users",
                                headers={"accept": "application/json"})

    end_time = time.time() #Records the End time

    Total_time_ms = (end_time - start_time)*1000 #Convert to ms
    Total_time_s = (end_time - start_time)

    print(f"Response_Time in seconds: {Total_time_s}, Response_Time in milli-seconds: {Total_time_ms}")

    #conversion to 2-decimal places:
    print(f"Response_Time: {round(Total_time_ms,2)}")
```

Output:

```
============================================================================== 1 passed in 2.04s ===============================================================================
PS D:\Python\PythonProject6\API_Automate> pytest -v -s test_scenario1.py::test_TC6
============================================================================== test session starts ==============================================================================
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\HP\AppData\Local\Programs\Python\Python314\python.exe
cachedir: .pytest_cache
metadata: {'Python': '3.14.0', 'Platform': 'Windows-11-10.0.26100-SP0', 'Packages': {'pytest': '9.0.1', 'pluggy': '1.6.0'}, 'Plugins': {'base-url': '2.1.0', 'bdd': '8.1.0', 'html':
'4.1.1', 'metadata': '3.1.1', 'playwright': '0.7.2'}, 'JAVA_HOME': 'C:\\Program Files\\Java\\jdk-21', 'Base URL': ''}
rootdir: D:\Python\PythonProject6\API_Automate
plugins: base-url-2.1.0, bdd-8.1.0, html-4.1.1, metadata-3.1.1, playwright-0.7.2
collected 1 item

test_scenario1.py::test_TC6 Response_Time in seconds: 0.9729018211364746, Response_Time in milli-seconds: 972.9018211364746
Response_Time: 972.9
PASSED

============================================================================== 1 passed in 2.19s ===============================================================================
```

Scenario 6: Perform a Basic CRUD Operation

Given Data:

GET: https://jsonplaceholder.typicode.com/posts

- ➔ Returns the List

POST: https://jsonplaceholder.typicode.com/posts

- ➔ Create a new resource
- ➔ Payload: {"title": "foo", "body": "bar", "userId": 1}
- ➔ Returns created post with an ID

PUT: https://jsonplaceholder.typicode.com/posts/1

- ➔ Update the resource for the ID
- ➔ Data: {"id": 1, "title": "updated title", "body": "updated body", "userId": 1}
- ➔ Returns the updated post

Delete: https://jsonplaceholder.typicode.com/posts/1

- ➔ Delete the resource
- ➔ Returns an empty response

Code:

```python
def test_TC7(playwright:Playwright):
    API_Session = playwright.request.new_context()

    #GET Data:
    GET_API_Response = API_Session.get( url: "https://jsonplaceholder.typicode.com/posts",
                            headers={"accept": "application/json"})

    print(f"GET Response: {GET_API_Response.json()}\n")

    #POST Updates:
    POST_DATA = {"title": "foo", "body": "bar", "userId": 1}
    POST_API_Response = API_Session.post( url: "https://jsonplaceholder.typicode.com/posts",
                                        headers={"accept": "application/json"},
                                        data= POST_DATA)

    print(f"POST Response: {POST_API_Response.json()}\n")

    ## Hitting the GET again to check the Updated data is present in the response:
    GET_API_Response_2 = API_Session.get( url: "https://jsonplaceholder.typicode.com/posts",
                                    headers={"accept": "application/json"})

    ID_list = [] #Checking the new ID added

    for ID in GET_API_Response_2.json():
        ID_list.append(ID["id"])

    print(f"ID_list: {ID_list}\n")
```

```python
## Hitting the GET again to check the Updated data is present in the response:
GET_API_Response_2 = API_Session.get( url: "https://jsonplaceholder.typicode.com/posts",
                                    headers={"accept": "application/json"})

ID_list = [] #Checking the new ID added

for ID in GET_API_Response_2.json():
    ID_list.append(ID["id"])

print(f"ID_list: {ID_list}\n")


#Updating the Existing data:
PUT_Data = {"id": 1,  "title": "updated title", "body": "updated body", "userId": 1}
PUT_API_Response = API_Session.put( url: "https://jsonplaceholder.typicode.com/posts/1",
                                    headers={"accept": "application/json"},
                                    data=PUT_Data)

print(f"PUT Response: {PUT_API_Response.json()}\n")

#Delete the Data:
Delete_API_Resonse = API_Session.delete( url: "https://jsonplaceholder.typicode.com/posts/1",
                                        headers={"accept": "application/json"})

print(f"DELETE Response: {Delete_API_Resonse.json()}\n")
```