

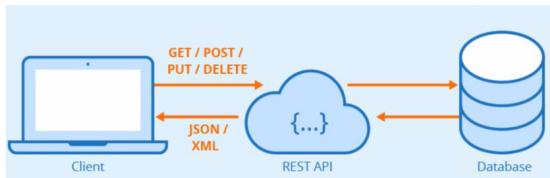
Test Automation using:

- Python
- Swagger
- Go REST
- Postman
- PyCharm
- JSON Beautifier and Editor
- PyTest
- Visual Studio Code
- Libraries (ie., requests, json)
- GitHub repository

Document highlight:

- GET method
- POST method
- PUT method
- OAuth2 Bearer Token request
- Passing Parameters in URL
- Passing payload from a JSON file
- Python request API automation with PyTest
- API Automation framework (GET method) using PyTest, Python, and Requests libraries (in Visual Studio Code)
- API Automation framework (POST method) using PyTest, Python, and Requests libraries (in Visual Studio Code)
- API Automation framework (PUT, DELETE method) plus Dynamic Data handling in POST method using PyTest, Python, and Requests libraries (in Visual Studio Code)
- GitHub setup (Create, Stage, Commit, and Publish with VS Code)
- CI with GitHub Actions

1. API Concept (interface between client and server (database), request and response)



2. Sampling APIs (using Swagger)

3. Executing a GET method

The screenshot shows the Swagger UI for the **FakeRESTApi.Web V1** API. The URL is fakerestapi.azurewebsites.net/index.html. The top navigation bar has a dropdown for "Select a definition" set to "V1".

Activities

GET /api/v1/Activities

Parameters

No parameters

Responses

Curl

```
curl -X GET "https://fakerestapi.azurewebsites.net/api/v1/Activities" -H "accept: text/plain; v=1.0"
```

Request URL

```
https://fakerestapi.azurewebsites.net/api/v1/Activities
```

Server response

Code **Details**

200 Response body

```
[{"id": 1, "title": "Activity 1", "dueDate": "2024-10-08T02:05:45.3979851+00:00", "completed": false}, {"id": 2, "title": "Activity 2", "dueDate": "2024-10-08T03:05:45.3979881+00:00", "completed": true}, {"id": 3, "title": "Activity 3", "dueDate": "2024-10-08T04:05:45.3979884+00:00", "completed": false}, {"id": 4, "title": "Activity 4", "dueDate": "2024-10-08T05:05:45.3979887+00:00", "completed": true}, {"id": 5, "title": "Activity 5", "dueDate": null, "completed": false}]
```

Response headers

```
api-supported-versions: 1.0
content-type: application/json; charset=utf-8; v=1.0
date: Tue, 08 Oct 2024 01:05:44 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	SUCCESS	No links

Media type

text/plain; v=1.0

Controls Accept header.

[Example Value](#) | [Schema](#)

```
[{"id": 0, "title": "string", "dueDate": "2024-10-08T01:05:44.656Z", "completed": true}]
```

4. Sample GET without parameter (copy the sampling url from Swagger and into Postman)

Validate and compare the following response results:

- Status
- Body
- Header
- Time

Swagger

The screenshot shows the Swagger UI for the **FakeRESTApi.Web V1**. The endpoint **/api/v1/Activities** is selected. In the **Responses** section, the **Curl** command and the **Request URL** are displayed. The **Server response** section shows a status code of **200** and a JSON response body containing a single activity object. The **Response headers** section lists standard HTTP headers.

```
curl -X GET "https://fakerestapi.azurewebsites.net/api/v1/Activities" -H "accept: text/plain; v=1.0"
Request URL
https://fakerestapi.azurewebsites.net/api/v1/Activities
```

Server response

Code	Details
200	Response body

```
[{"id": 1, "title": "Activity A", "dueDate": "2024-10-06T22:13:39.4497611+00:00", "completed": false}]
```

Response headers

```
api-supported-version: 1.0
Content-type: application/json; charset=utf-8; v=1.0
date: Tue, 06 Oct 2024 21:13:38 GMT
server: Kestrel
transfer-encoding: chunked
```

Postman

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History.
- Request URL:** GET https://fakerestapi.azurewebsites.net/api/v1/Activities
- Headers:** None
- Body:** JSON response (Pretty) showing a single activity object:

```
1 [  
2 {  
3   "id": 1,  
4   "title": "Activity 1",  
5   "dueDate": "2024-10-08T22:14:52.389423+00:00",  
6   "completed": false  
7 ]
```
- Status:** 200 OK, 721 ms, 2.96 KB

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History.
- Request URL:** GET https://fakerestapi.azurewebsites.net/api/v1/Activities
- Headers:** Accept: text/plain
- Body:** Headers (5) showing the following:

Key	Value
Content-Type	application/json; charset=utf-8; v=1.0
Date	Tue, 08 Oct 2024 21:14:51 GMT
Server	Kestrel
Transfer-Encoding	chunked
api-supported-versions	1.0
- Status:** 200 OK, 721 ms, 2.96 KB

5. Sample GET with parameter (copy the sampling url from Swagger and into Postman)

Swagger

GET /api/v1/Activities/{id}

Parameters

Name	Description
id * required	integer(\$int32) (path)

Responses

Curl
curl -X GET "https://fakerestapi.azurewebsites.net/api/v1/Activities/3" -H "accept: text/plain; v=1.0"

Request URL
https://fakerestapi.azurewebsites.net/api/v1/Activities/3

Server response

Code Details

200 Response body

```
{ "id": 3, "title": "Activity 3", "dueDate": "2024-10-09T00:39:32.2546427+00:00", "completed": false }
```

Response headers

```
api-support-versions: 1.0  
Content-Type: application/json; charset=utf-8; v=1.0  
date: Tue, 09 Oct 2024 21:39:31 GMT  
server: Kestrel  
transfer-encoding: chunked
```

Responses

Code Description

200 Success

Media type

text/plain; v=1.0

Example Value | Schema

```
{ "id": 3, "title": "Activity 3", "dueDate": "2024-10-09T00:39:32.181Z", "completed": true }
```

Postman

GET https://fakerestapi.azurewebsites.net/api/v1/Activities/3

Params Authorization Headers (7) Body Scripts Settings Cookies

Headers

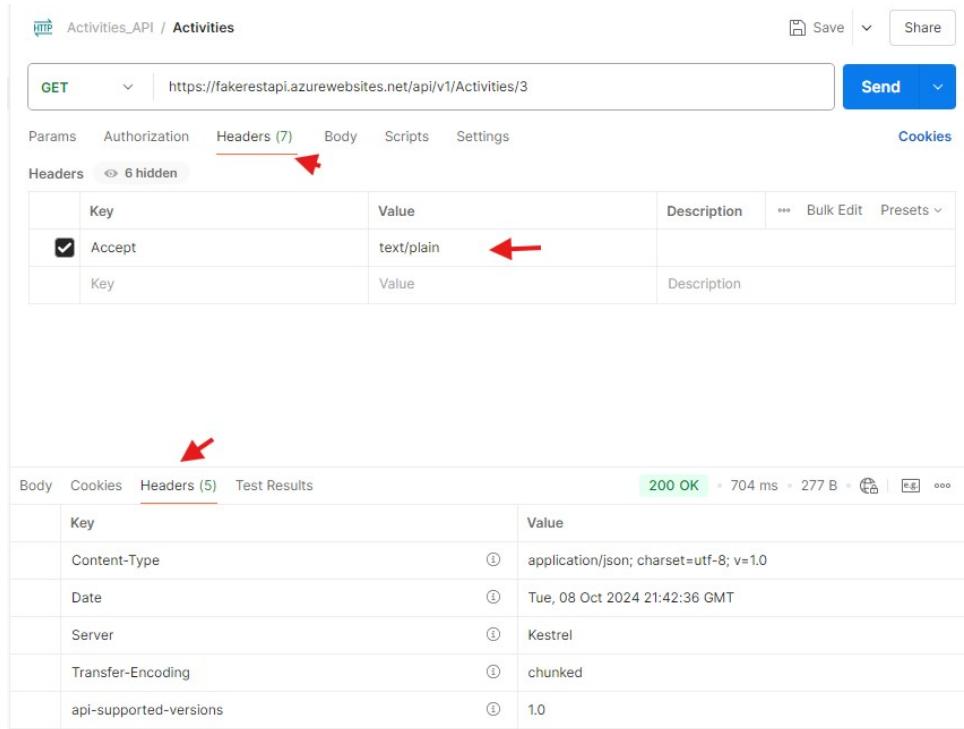
Key	Value	Description	Bulk Edit	Presets
Accept	text/plain			
Key	Value	Description		

Body Cookies Headers (5) Test Results

200 OK 704 ms 277 B

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": 3,  
3   "title": "Activity 3",  
4   "dueDate": "2024-10-09T00:42:36.7469099+00:00",  
5   "completed": false  
6 }
```



Activities_API / Activities

GET https://fakerestapi.azurewebsites.net/api/v1/Activities/3

Params Authorization Headers (7) Body Scripts Settings Cookies

Headers (6 hidden)

Key	Value	Description
Accept	text/plain	
Key	Value	Description

Body Cookies Headers (5) Test Results

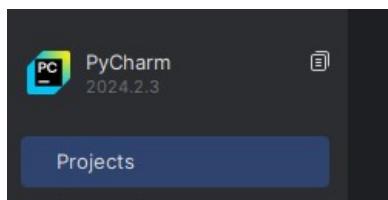
200 OK 704 ms 277 B

Key	Value
Content-Type	application/json; charset=utf-8; v=1.0
Date	Tue, 08 Oct 2024 21:42:36 GMT
Server	Kestrel
Transfer-Encoding	chunked
api-supported-versions	1.0

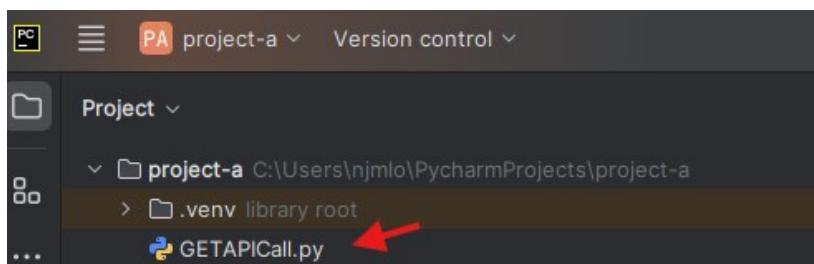
6. Determine Python version, install “requests” library

```
(.venv) PS C:\Users\njmlo\PycharmProjects\project-a> python -V
Python 3.12.6
(.venv) PS C:\Users\njmlo\PycharmProjects\project-a> pip install requests
```

7. PyCharm Community Edition (IDE version at the time of documentation)



8. Create a python file in PyCharm editor



9. Sample GET (in PyCharm using Python code, a sampling of with and without parameters, and header validation)

The screenshot shows the PyCharm IDE interface with the following details:

- Project Structure:** The project is named "project-a". The main file "main.py" is open, along with "GETAPICall.py" and "POSTAPICall.py".
- Code Editor:** The code in "main.py" demonstrates three types of GET requests:
 - Without parameters: `response = requests.get('https://fakerestapi.azurewebsites.net/api/v1/Activitities')`
 - With parameters: `response = requests.get('https://fakerestapi.azurewebsites.net/api/v1/Activitities/2')`
 - With header validation: `response = requests.get(url='https://fakerestapi.azurewebsites.net/api/v1/Activitities/2', headers=head)`
- Terminal Output:** The terminal window shows the results of the API calls.
 - For the first request (without parameters):

```
Successful response (without parameter) as per db content below:  
[{'id': 1, 'title': 'Activity 1', 'dueDate': '2024-10-09T20:48:08.5864366+00:00', 'completed': False}, {'id': 2, 'title': 'Activity 2', 'dueDate': '2024-10-09T21:48:09.2691665+00:00', 'completed': True}]
```
 - For the second request (with parameter):

```
Successful response (with parameter) as per db content below:  
[{'id': 2, 'title': 'Activity 2', 'dueDate': '2024-10-09T21:48:09.2691665+00:00', 'completed': True}]
```
 - For the third request (with header validation):

```
Successful response (no error on header) as per code below:  
200  
With status code 200, API test run passed.
```
- Status Bar:** The bottom right corner shows the status bar with "58:1 CRLF UTF-8 4 spaces Python 3.12 (project-a)".

10. Using Json Beautifier to display response result nicely (copy and paste the result)

Text view

Tree view



The screenshot shows a browser window with two tabs: "Swagger UI" and "JSON Editor". The "JSON Editor" tab is active, displaying a hierarchical tree view of a JSON dataset representing five activities. Each activity object contains fields: id, title, dueDate, and completed. The completed field is represented as a checkbox input.

```
[{"id": 1, "title": "Activity 1", "dueDate": "2024-10-08T23:21:17.6875248+00:00", "completed": false}, {"id": 2, "title": "Activity 2", "dueDate": "2024-10-09T00:21:17.6875281+00:00", "completed": true}, {"id": 3, "title": "Activity 3", "dueDate": "2024-10-09T01:21:17.6875285+00:00", "completed": false}, {"id": 4, "title": "Activity 4", "dueDate": "2024-10-09T02:21:17.6875287+00:00", "completed": true}, {"id": 5, "title": "Activity 5", "dueDate": "2024-10-09T03:21:17.6875291+00:00", "completed": false}]
```

Table view

text	tree	table	⬇️	⬆️	🔍	⋮	↶	↷
0	1	Activity 1	2024-10-08T23:21:17.6875248+00:00	<input type="checkbox"/>	false			
1	2	Activity 2	2024-10-09T00:21:17.6875281+00:00	<input checked="" type="checkbox"/>	true			
2	3	Activity 3	2024-10-09T01:21:17.6875285+00:00	<input type="checkbox"/>	false			
3	4	Activity 4	2024-10-09T02:21:17.6875287+00:00	<input checked="" type="checkbox"/>	true			
4	5	Activity 5	2024-10-09T03:21:17.6875291+00:00	<input type="checkbox"/>	false			
5	6	Activity 6	2024-10-09T04:21:17.6875296+00:00	<input checked="" type="checkbox"/>	true			
6	7	Activity 7	2024-10-09T05:21:17.68753+00:00	<input type="checkbox"/>	false			
7	8	Activity 8	2024-10-09T06:21:17.6875302+00:00	<input checked="" type="checkbox"/>	true			
8	9	Activity 9	2024-10-09T07:21:17.6875305+00:00	<input type="checkbox"/>	false			
9	10	Activity 10	2024-10-09T08:21:17.6875311+00:00	<input checked="" type="checkbox"/>	true			
10	11	Activity 11	2024-10-09T09:21:17.6875314+00:00	<input type="checkbox"/>	false			
11	12	Activity 12	2024-10-09T10:21:17.6875316+00:00	<input checked="" type="checkbox"/>	true			
12	13	Activity 13	2024-10-09T11:21:17.6875319+00:00	<input type="checkbox"/>	false			
13	14	Activity 14	2024-10-09T12:21:17.6875322+00:00	<input checked="" type="checkbox"/>	true			
14	15	Activity 15	2024-10-09T13:21:17.6875325+00:00	<input type="checkbox"/>	false			
15	16	Activity 16	2024-10-09T14:21:17.6875328+00:00	<input checked="" type="checkbox"/>	true			
16	17	Activity 17	2024-10-09T15:21:17.687533+00:00	<input type="checkbox"/>	false			
17	18	Activity 18	2024-10-09T16:21:17.6875334+00:00	<input checked="" type="checkbox"/>	true			
18	19	Activity 19	2024-10-09T17:21:17.6875337+00:00	<input type="checkbox"/>	false			
19	20	Activity 20	2024-10-09T18:21:17.687534+00:00	<input checked="" type="checkbox"/>	true			
20	21	Activity 21	2024-10-09T19:21:17.6875342+00:00	<input type="checkbox"/>	false			
21	22	Activity 22	2024-10-09T20:21:17.6875345+00:00	<input checked="" type="checkbox"/>	true			
22	23	Activity 23	2024-10-09T21:21:17.6875348+00:00	<input type="checkbox"/>	false			
23	24	Activity 24	2024-10-09T22:21:17.6875351+00:00	<input checked="" type="checkbox"/>	true			
24	25	Activity 25	2024-10-09T23:21:17.6875353+00:00	<input type="checkbox"/>	false			
25	26	Activity 26	2024-10-10T00:21:17.6875356+00:00	<input checked="" type="checkbox"/>	true			
26	27	Activity 27	2024-10-10T01:21:17.6875359+00:00	<input type="checkbox"/>	false			
27	28	Activity 28	2024-10-10T02:21:17.6875361+00:00	<input checked="" type="checkbox"/>	true			
28	29	Activity 29	2024-10-10T03:21:17.6875364+00:00	<input type="checkbox"/>	false			
29	30	Activity 30	2024-10-10T04:21:17.6875367+00:00	<input checked="" type="checkbox"/>	true			

11. Sample GET (Python in PyCharm, a sampling of with status error)

```
57     print("-----")
58
59     # expected response CODE is 200 but with error
60     assert response.status_code == 201
61
```

```
-----
Traceback (most recent call last): Explain with AI
  File "C:\Users\njml\PycharmProjects\project-a\GETAPICall.py", line 60, in <module>
    assert response.status_code == 201
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError
```

12. Here is the code in text

```
# import:
import requests

# variable(s):
head = {
    'Accept':'text/plain'
}

# ***CODE***
print("-----")

# GET response without parameters
response =
requests.get('https://fakerestapi.azurewebsites.net/api/v1/Activities')

# display the response BODY
print("Successful response (without parameter) as per db content below:")
print(response.json())

# display the response CODE
print("Successful response as per code below:")
print(response.status_code)

# expected response CODE
assert response.status_code == 200
print("With status code 200, API test run passed.")

print("-----")

# GET response with parameters
response =
```

```
requests.get('https://fakerestapi.azurewebsites.net/api/v1/Activities/2')

# display the response BODY with parameters
print("Successful response (with parameter) as per db content below:")
print(response.json())

# display the response CODE
print("Successful response as per code below:")
print(response.status_code)

# display the print text if expected code passed
assert response.status_code == 200
print("With status code 200, API test run passed.")

print("-----")
-----")

# GET response with header validation
response =
requests.get('https://fakerestapi.azurewebsites.net/api/v1/Activities/2',
               headers=head)

# display the response CODE
print("Successful response (no error on header) as per code below:")
print(response.status_code)

# expected response CODE
assert response.status_code == 200
print("With status code 200, API test run passed.")

print("-----")
-----")

# expected response CODE is 200 but with error
assert response.status_code == 201
```

13. Executing a POST method

The screenshot shows the Swagger UI interface for the **FakeRESTApi.Web V1** API. The URL is <https://fakerestapi.azurewebsites.net/index.html>. The top navigation bar includes a back button, forward button, search icon, star icon, refresh icon, and a dropdown menu labeled "Select a definition" with "V1" selected.

The main title is **FakeRESTApi.Web V1** with a **v1 OAS3** badge. Below it is the path </swagger/v1/swagger.json>.

The **Activities** section contains two buttons: **GET /api/v1/Activities** (blue) and **POST /api/v1/Activities** (green). The **POST** button is currently active.

Parameters: No parameters.

Request body: application/json; v=1.0

```
{ "id": 31, "title": "Activity Testing 01", "dueDate": "2024-10-09T16:13:25.086Z", "completed": true }
```

Buttons: Execute (blue), Clear (white).

Responses:

Curl: curl -X POST "https://fakerestapi.azurewebsites.net/api/v1/Activities" -H "accept: text/plain; v=1.0" -H "Content-type: application/json; v=1.0" -d "{\"id\":31,\"title\":\"Activity Testing 01\",\"dueDate\":\"2024-10-09T16:13:25.086Z\",\"completed\":true}"

Request URL: <https://fakerestapi.azurewebsites.net/api/v1/Activities>

Server response:

Code	Details
200	Response body: <pre>{ "id": 31, "title": "Activity Testing 01", "dueDate": "2024-10-09T16:13:25.086Z", "completed": true }</pre> Download
	Response headers: <pre>access-control-allow-origin: * api-supported-versions: 1.0 content-type: application/json; charset=utf-8; v=1.0 date: Wednesday, 10-Oct-2024 16:15:08 GMT server: Kestrel transfer-encoding: chunked</pre>

Responses:

Code	Description	Links
200	Success	No links

Media type: text/plain; v=1.0

Controls Accept header.

[Example Value](#) | [Schema](#)

```
{ "id": 0, "title": "string", "dueDate": "2024-10-09T16:15:08.644Z", "completed": true }
```

14. Sample POST (copy the sampling url from Swagger and into Postman)

Validate and compare the following response results:

- Status
- Body
- Header
- Time

Swagger

```
Curl
curl -X POST "https://fakerestapi.azurewebsites.net/api/v1/Activities" -H "Accept: text/plain; v=1.0" -H "Content-Type: application/json; v=1.0" -d "{\"id\":33,\"title\":\"Activity Testing 01\",\"dueDate\":\"2024-10-09T16:13:25.086Z\",\"completed\":true}"
Request URL
https://fakerestapi.azurewebsites.net/api/v1/Activities
```

Postman

URL

The screenshot shows the Postman interface. In the top left, it says "HTTP Activities_API / Activities". Below that is a dropdown menu set to "POST" and a URL input field containing "https://fakerestapi.azurewebsites.net/api/v1/Activities". To the right of the URL is a blue "Send" button. A red arrow points to the "Send" button.

Header

The screenshot shows the "Headers" tab in Postman. It lists two checked headers: "accept" with value "text/plain" and "Content-Type" with value "application/json". There is also an uncheckable header row with "Key" and "Value" columns. A red arrow points to the "Headers" tab.

Body

The screenshot shows the "Body" tab in Postman. It has a "raw" JSON input area containing the following code, with a red arrow pointing to the input area:

```
1 {
2   "id": 32,
3   "title": "Activity Testing 02",
4   "dueDate": "2024-10-09T16:13:25.086Z",
5   "completed": true
6 }
```

Below the input area are radio buttons for "none", "form-data", "x-www-form-urlencoded", "raw", "binary", "GraphQL", and "JSON". The "raw" option is selected. To the right of the input area is a "Beautify" button.

After Send...

Header

The screenshot shows the Postman interface after sending a POST request to `https://fakerestapi.azurewebsites.net/api/v1/Activities`. The 'Headers' tab is selected, displaying the following configuration:

Key	Value	Description
accept	text/plain	
Content-Type	application/json	

Below the table, the 'Test Results' section shows the response status as `200 OK` with a response time of `721 ms` and a size of `277 B`.

Key	Value
Content-Type	application/json; charset=utf-8; v=1.0
Date	Wed, 09 Oct 2024 18:48:30 GMT
Server	Kestrel
Transfer-Encoding	chunked
api-supported-versions	1.0

Body

The screenshot shows the Postman interface after sending a POST request to `https://fakerestapi.azurewebsites.net/api/v1/Activities`. The 'Body' tab is selected, showing the raw JSON input:

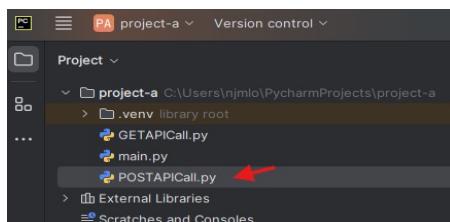
```
1 {
2     "id": 32,
3     "title": "Activity Testing 02",
4     "dueDate": "2024-10-09T16:13:25.086Z",
5     "completed": true
6 }
```

Below the input, the 'Test Results' section shows the response status as `200 OK` with a response time of `721 ms` and a size of `277 B`.

The response body is identical to the input:

```
1 {
2     "id": 32,
3     "title": "Activity Testing 02",
4     "dueDate": "2024-10-09T16:13:25.086Z",
5     "completed": true
6 }
```

15. Create a python file in PyCharm editor



16. Sample POST (in PyCharm using Python code)

The screenshot shows the PyCharm IDE interface with the following details:

- Project Structure:** The project is named "project-a". Inside, there are three files: "main.py", "GETAPICall.py", and "POSTAPICall.py".
- Code Editor:** The "POSTAPICall.py" file is open and contains the following Python code:

```
# import:
import requests

# variable(s):
head = {
    'Accept': 'text/plain',
    'Content-Type': 'application/json'
}

# request data or payload
request_payload ={
    "id": 32,
    "title": "Activity Testing 03",
    "dueDate": "2024-10-09T16:13:25.086Z",
    "completed": True
}

# ***CODE***
print("-----")

# POST method
response = requests.post(url: 'https://fakerestapi.azurewebsites.net/api/v1/Activities',
                         headers=head,
                         json=request_payload)

# display the response CODE
print("Successful response as per code below:")
print(response.status_code)

# display the print text if expected code passed
assert response.status_code == 200
print("With status code 200, API test run passed.")

print("-----")

# display the json response
print("Below is the response after the POST method.")
print(response.json())

print("-----")

#sample test validation of ensuring what was sent in the request is what comes back as expected from the response
data = response.json()
assert (data ['id']) == 32
```

- Run Tab:** The "Run" tab is active, showing the output of the script execution.
- Output:** The terminal window displays the following output:

```
C:\Users\njmlo\PycharmProjects\project-a\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\project-a\POSTAPICall.py
-----
Successful response as per code below:
200
With status code 200, API test run passed.

-----
```

The output shows the successful execution of the POST request, including the status code 200 and a confirmation message.

```
Below is the response after the POST method.
{'id': 32, 'title': 'Activity Testing 03', 'dueDate': '2024-10-09T16:13:25.086Z', 'completed': True}
```

The output concludes with "Process finished with exit code 0".

17. Using Json Beautifier to display response result nicely (copy and paste the result)

Text view

A screenshot of the JSON Beautifier & Editor extension for Google Chrome. The title bar says "JSON Editor". The address bar shows the extension's URL. The toolbar includes tabs for "text", "tree", and "table", along with various icons. The main content area displays a JSON object:

```
{'id': 32, 'title': 'Activity Testing 01', 'dueDate': '2024-10-09T16:13:25.086Z', 'completed': True}
```

Tree view

A screenshot of the JSON Beautifier & Editor extension for Google Chrome, showing the same JSON object but in tree view. The "tree" tab is selected in the toolbar. The JSON structure is shown as a tree with expandable nodes:

```
>
{
  id : 32
  title : Activity Testing 01
  dueDate : 2024-10-09T16:13:25.086Z
  completed : true
```

18. Sample POST (Python in PyCharm, a sampling of with validation error)

A screenshot of PyCharm showing a Python script named "POSTAPICall.py". The code contains an assertion statement:

```
40     print("-----")
41
42     #sample test validation of ensuring what was sent in the request is what comes back as expected from the response
43     data = response.json()
44     assert (data ['id']) == 33
```

The line "assert (data ['id']) == 33" has a red arrow pointing to the value "33" and another red arrow pointing to the line number "44".

The "Run" tool window shows the output of the script:

```
Run POSTAPICall x
C:\Users\njmlo\PycharmProjects\project-a\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\project-a\POSTAPICall.py
-----
Successful response as per code below:
200
With status code 200, API test run passed.
Below is the response after the POST method.
{'id': 32, 'title': 'Activity Testing 03', 'dueDate': '2024-10-09T16:13:25.086Z', 'completed': True}

Traceback (most recent call last):
  File "C:\Users\njmlo\PycharmProjects\project-a\POSTAPICall.py", line 44, in <module>
    assert (data ['id']) == 33
          ^^^^^^^^^^^^^^^^^^
AssertionError

Process finished with exit code 1
```

19. Here is the code in text

```
# import:
import requests

# variable(s):
head = {
    'Accept':'text/plain',
    'Content-Type': 'application/json'
}

# request data or payload
request_payload ={
    "id": 32,
    "title": "Activity Testing 03",
    "dueDate": "2024-10-09T16:13:25.086Z",
    "completed": True
}

# ***CODE***
print("-----")
-----)

# POST method
response =
requests.post('https://fakerestapi.azurewebsites.net/api/v1/Activities',
                headers=head,
                json=request_payload)

# display the response CODE
print("Successful response as per code below:")
print(response.status_code)

# display the print text if expected code passed
assert response.status_code == 200
print("With status code 200, API test run passed.")

print("-----")
-----)

# display the json response
print("Below is the response after the POST method.")
print(response.json())

print("-----")
-----)

#sample test validation of ensuring what was sent in the request is what
comes back as expected from the response
data = response.json()
assert (data ['id']) == 32
```

20. Executing a PUT method

In SwaggerUI, before executing the method

GET /api/v1/Activities/{id}

Parameters

Name	Description
id * required	integer(\$int32) (path)

Responses

Curl
curl -X GET "https://fakertestapi.azurewebsites.net/api/v1/Activities/13" -H "accept: text/plain; v=1.0"

Request URL
https://fakertestapi.azurewebsites.net/api/v1/Activities/13

Server response

Code Details

200 Response body

```
{ "id": 13, "title": "Activity 13", "dueDate": "2024-10-18T10:23:56.6064814+00:00", "completed": false }
```

Response headers

```
api-supported-versions: 1.0
content-type: application/json; charset=utf-8; v=1.0
date: Mon Oct 29 2024 21:23:55 GMT
server: Kestrel
transfer-encoding: chunked
```

PUT /api/v1/Activities/{id}

Parameters

Name	Description
id * required	integer(\$int32) (path)

Request body

```
{ "id": 14, "title": "Update title 01", "dueDate": "2024-10-09T21:23:06.171Z", "Completed": true }
```

application/json; v=1.0

In SwaggerUI, after executing the method

PUT /api/v1/Activities/{id}

Parameters

Name	Description
id * required	integer(\$int32) (path)

Request body

```
{ "id": 14, "title": "Update title 01", "dueDate": "2024-10-09T21:23:06.171Z", "Completed": true }
```

application/json; v=1.0

Responses

Curl
curl -X PUT "https://fakertestapi.azurewebsites.net/api/v1/Activities/13" -H "accept: text/plain; v=1.0" -H "Content-Type: application/json; v=1.0" -d "{\"id\":14,\"title\":\"Update title 01\"}"

Request URL
https://fakertestapi.azurewebsites.net/api/v1/Activities/13

Server response

Code Details

200 Response body

```
{ "id": 14, "title": "Update title 01", "dueDate": "2024-10-09T21:23:06.171Z", "Completed": true }
```

Response headers

```
access-control-allow-origin: *
api-supported-versions: 1.0
content-type: application/json; charset=utf-8; v=1.0
date: Mon Oct 29 2024 21:26:50 GMT
server: Kestrel
transfer-encoding: chunked
```

21. Sample PUT (copy the sampling url from Swagger and into Postman)

Validate and compare the following response results:

- Status
- Body
- Header
- Time

Swagger

```
Curl
curl -X PUT "https://fakerestapi.azurewebsites.net/api/v1/Activities/13" -H "accept: text/plain; v=1.0" -H "Content-Type: application/json; v=1.0" -d "{\"id\":14,\"title\":\"Update title again 01\",\"dueDate\":\"2024-10-09T21:23:06.171Z\",\"completed\":true}"

Request URL
https://fakerestapi.azurewebsites.net/api/v1/Activities/13
```

Postman

URL

HTTP Activities_API / Activities

PUT https://fakerestapi.azurewebsites.net/api/v1/Activities/13 Send

Header

HTTP Activities_API / Activities

PUT https://fakerestapi.azurewebsites.net/api/v1/Activities/13 Send

Headers (10) Params Authorization Body Scripts Settings Cookies

Headers 8 hidden

Key	Value	Description	Bulk Edit	Presets
Accept	text/plain			
Content-Type	application/json			

Body

HTTP Activities_API / Activities

PUT https://fakerestapi.azurewebsites.net/api/v1/Activities/13 Send

Params Authorization Headers (10) Body Scripts Settings Cookies

Body raw binary GraphQL JSON Beautify

```
1
2   "id": 133,
3   "title": "Update title again 01",
4   "dueDate": "2024-10-09T21:23:06.171Z",
5   "completed": true
```

After Send...

Header

The screenshot shows the 'Headers' tab in Postman. There are two checked headers: 'Accept' (text/plain) and 'Content-Type' (application/json). Other headers listed are 'Key' and 'Value'.

Key	Value	Description	Bulk Edit	Presets
Accept	text/plain			
Content-Type	application/json			
Key	Value	Description		

The screenshot shows the 'Headers (5)' section in the 'Test Results' tab. It lists several response headers with their values and descriptions.

Key	Value
Content-Type	application/json; charset=utf-8; v=1.0
Date	Wed, 09 Oct 2024 22:16:45 GMT
Server	Kestrel
Transfer-Encoding	chunked
api-supported-versions	1.0

Body

The screenshot shows the 'Body' tab in Postman. A JSON payload is being sent to the API endpoint. The payload contains fields: id, title, dueDate, and completed.

```
1 {  
2   "id": 133,  
3   "title": "Update title again 01",  
4   "dueDate": "2024-10-09T21:23:06.171Z",  
5   "completed": true  
6 }
```

The screenshot shows the 'Body' section in the 'Test Results' tab. The response is displayed in 'Pretty' format, showing the updated title 'Update title again 01'.

```
1 {  
2   "id": 133,  
3   "title": "Update title again 01",  
4   "dueDate": "2024-10-09T21:23:06.171Z",  
5   "completed": true  
6 }
```

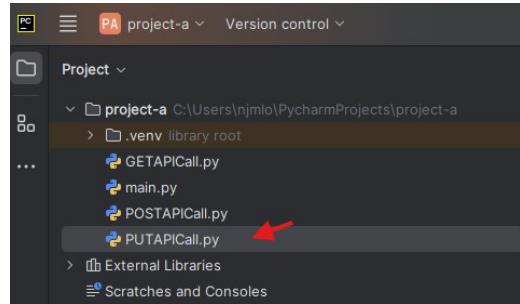
22. Sample Test code (using Snippets in Postman) and validating its Test Results (Post-response)

The screenshot shows the Postman interface for a PUT request to `https://fakerestapi.azurewebsites.net/api/v1/Activities/13`. The 'Scripts' tab is selected, containing the following JavaScript code:

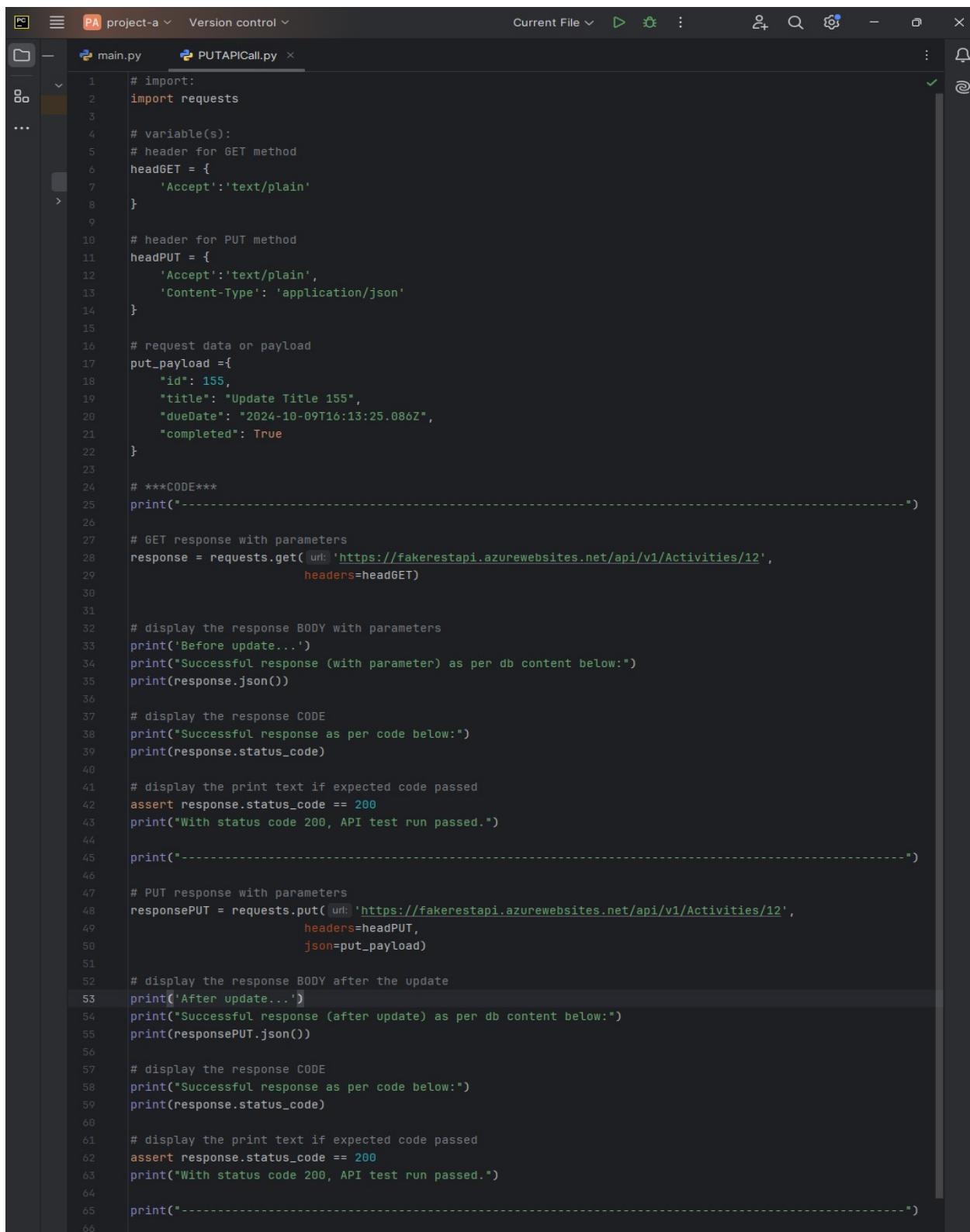
```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

A red arrow points to the 'Test Results' tab at the bottom, which displays a single green 'PASS' status with the message 'Status code is 200'. Another red arrow points to the right-click context menu, which includes options like 'Clear a collection variable', 'Send a request', and 'Status code: Code is 200'.

23. Create a python file in PyCharm editor

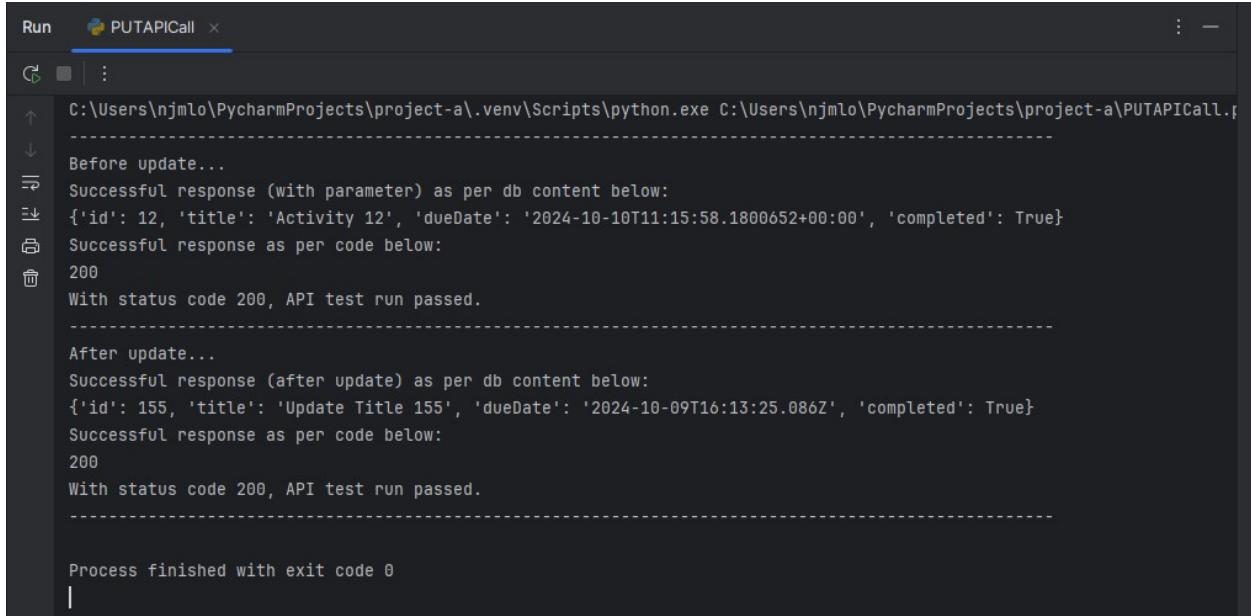


24. Sample PUT (in PyCharm using Python code)



The screenshot shows the PyCharm IDE interface with the following details:

- Project Structure:** The left sidebar shows a project named "project-a" with files "main.py" and "PUTAPICall.py".
- Current File:** The main editor window displays "main.py".
- Code Content:** The Python script performs the following steps:
 - Imports the requests library.
 - Defines variables for headers and payload.
 - Sets up headers for GET and PUT methods.
 - Creates a payload dictionary with fields: id, title, dueDate, and completed.
 - Prints a separator line.
 - Issues a GET request to the specified URL with the defined headers.
 - Prints the response body.
 - Prints the status code.
 - Asserts the status code is 200.
 - Prints a separator line.
 - Issues a PUT request to the specified URL with the defined headers and payload.
 - Prints the response body.
 - Prints the status code.
 - Asserts the status code is 200.
 - Prints a final separator line.



The screenshot shows the PyCharm Run window titled "PUTAPICall". The output pane displays the results of a PUT API call. It starts with the command: "C:\Users\njmlo\PycharmProjects\project-a\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\project-a\PUTAPICall.py". The log then shows two sections: "Before update..." and "After update...". Under "Before update...", it says "Successful response (with parameter) as per db content below:" followed by a JSON object: {"id": 12, "title": "Activity 12", "dueDate": "2024-10-10T11:15:58.1800652+00:00", "completed": True}. It also shows "Successful response as per code below:" with status code 200 and a message: "With status code 200, API test run passed.". Under "After update...", it says "Successful response (after update) as per db content below:" followed by a JSON object: {"id": 155, "title": "Update Title 155", "dueDate": "2024-10-09T16:13:25.086Z", "completed": True}. It also shows "Successful response as per code below:" with status code 200 and a message: "With status code 200, API test run passed.". The log concludes with "Process finished with exit code 0".

25. Here is the code in text

```
# import:
import requests

# variable(s):
# header for GET method
headGET = {
    'Accept':'text/plain'
}

# header for PUT method
headPUT = {
    'Accept':'text/plain',
    'Content-Type': 'application/json'
}

# request data or payload
put_payload ={
    "id": 155,
    "title": "Update Title 155",
    "dueDate": "2024-10-09T16:13:25.086Z",
    "completed": True
}

# ***CODE***
print("-----")
print("-----")

# GET response with parameters
response =
requests.get('https://fakerestapi.azurewebsites.net/api/v1/Activities/12',
```

```
                headers=headGET)

# display the response BODY with parameters
print('Before update... ')
print("Successful response (with parameter) as per db content below:")
print(response.json())

# display the response CODE
print("Successful response as per code below:")
print(response.status_code)

# display the print text if expected code passed
assert response.status_code == 200
print("With status code 200, API test run passed.")

print("-----")
-----")

# PUT response with parameters
responsePUT =
requests.put('https://fakerestapi.azurewebsites.net/api/v1/Activities/12',
              headers=headPUT,
              json=put_payload)

# display the response BODY after the update
print('After update... ')
print("Successful response (after update) as per db content below:")
print(responsePUT.json())

# display the response CODE
print("Successful response as per code below:")
print(response.status_code)

# display the print text if expected code passed
assert response.status_code == 200
print("With status code 200, API test run passed.")

print("-----")
-----")
```

26. Setting up and executing an OAuth2 Bearer Token request

Go REST sampling APIs

The screenshot shows the Go REST interface. On the left, under 'Resources', there is a list of endpoints:

Endpoint	Count
https://gorest.co.in/public/v2/users	2810
https://gorest.co.in/public/v2/posts	2883
https://gorest.co.in/public/v2/comments	2878
https://gorest.co.in/public/v2/todos	1455

On the right, under 'Trying it Out', there is a table of operations:

Method	Endpoint	Description
POST	/public/v2/users	Create a new user
GET	/public/v2/users/6941918	Get user details
PUT PATCH	/public/v2/users/6941918	Update user details
DELETE	/public/v2/users/6941918	Delete user

Below these sections, there is a 'Nested Resources' table:

Method	Endpoint	Description
GET	/public/v2/users/6941918/posts	Retrieves user posts
GET	/public/v2/posts/6941918/comments	Retrieves post comments
GET	/public/v2/users/6941918/todos	Retrieves user todos
POST	/public/v2/users/6941918/posts	Creates a user post
POST	/public/v2/posts/6941918/comments	Creates a post comment
POST	/public/v2/users/6941918/todos	Creates a user todo

At the bottom, there is a note with bullet points:

- Do not post your personal data like name, email, phone, photo etc...
- For paged results parameter "page" and "per_page" should be passed in url ex: GET /public/v2/users?page=1&per_page=20 (max 100 results per page)
- Request methods PUT, POST, PATCH, DELETE needs access token, which needs to be passed with "Authorization" header as Bearer token.
- API Versions /public-api/* and /public/v1/* and /public/v2/*
- [Get your access token](#)

Finally, there is a 'GraphQL Endpoint' section with three items:

- GraphQL API is available at <https://gorest.co.in/public/v2/graphql>
- [View GraphQL json schema](#)
- [View GraphQL schema](#)

The screenshot shows a JSON response from the /public/v2/users endpoint. The response is a list of user objects. The 'pretty-print' checkbox is checked, so the JSON is formatted with indentation and line breaks. The first few users in the list are:

```
{  
  "id": 7464998,  
  "name": "The Hon. Aanandinii Varrier",  
  "email": "varrier_aanandinii_the_hon@mohre.test",  
  "gender": "female",  
  "status": "inactive"  
},  
{  
  "id": 7463045,  
  "name": "Samir Gandhi",  
  "email": "gandhi_samir@ledner.example",  

```

27. Sample POST with OAuth2 bearer key (copy the sampling url from Go REST and into Postman)

Validate and compare the following response results:

- Status
- Body
- Header
- Time

Go REST

The screenshot shows the Go REST interface with a dark header bar. Below it, the word "Resources" is displayed in white. Underneath, there is a URL field containing "https://gorest.co.in/public/v2/users" and a port number "2810".

Sample Github token

The screenshot shows the Go REST interface with a dark header bar. Below it, the word "Access Tokens" is displayed in white. A message below says "These are your personal access tokens, do not share them on public websites." There is a green button labeled "New Access Token". Below is a table with columns: Label, Token, Expires, Limit, Remaining, and Action. A red arrow points to the "Token" column of the first row, which contains a long string of characters.

Postman

URL

The screenshot shows the Postman interface with a dark header bar. Below it, the word "Activities_API / AuthorizationCallTest" is displayed in white. Underneath, there is a "POST" method dropdown and a URL input field containing "https://gorest.co.in/public/v2/users". To the right of the URL is a "Send" button.

Header

The screenshot shows the Postman interface with a dark header bar. Below it, the word "Activities_API / AuthorizationCallTest" is displayed in white. Underneath, there is a "POST" method dropdown and a URL input field containing "https://gorest.co.in/public/v2/users". To the right of the URL is a "Send" button. Below the URL, there is a "Headers (11)" tab. A red arrow points to this tab. The "Headers" section shows two entries: "Content-Type" with value "application/json" and "Authorization" with value "Bearer 8e30aa089a022...".

Body

The screenshot shows the Postman interface with a dark header bar. Below it, the word "Activities_API / AuthorizationCallTest" is displayed in white. Underneath, there is a "POST" method dropdown and a URL input field containing "https://gorest.co.in/public/v2/users". To the right of the URL is a "Send" button. Below the URL, there is a "Body" tab. A red arrow points to this tab. The "Body" section has a "raw" radio button selected. In the text area, there is a JSON object with numbered lines:

```
1  {
2   "id": 7464998,
3   "name": "The Hon. Aanandinii Varrier",
4   "email": "varrier_aanandinii_the_hon@mohrex.test",
5   "gender": "female",
6   "status": "inactive"
7 }
```

After Send...

Header

The screenshot shows the Postman interface after sending a POST request to <https://gorest.co.in/public/v2/users>. The 'Headers' tab is selected, displaying the following response headers:

Key	Value	Description
Date	Thu, 10 Oct 2024 22:10:42 GMT	
Content-Type	application/json; charset=utf-8	
Content-Length	138	
Connection	keep-alive	
Cache-Control	max-age=0, private, must-revalidate	
etag	W/"3875de656ef2b9bec4765176bb9f2b2a"	
location	https://gorest.co.in/public/v2/users/7465032	
referrer-policy	strict-origin-when-cross-origin	
vary	Origin	
x-content-type-options	nosniff	
x-download-options	noopen	
x-frame-options	SAMEORIGIN	
x-permitted-cross-domain-policies	none	
x-ratelimit-limit	90	

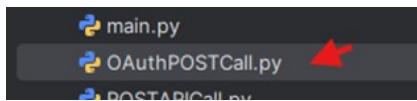
Body

The screenshot shows the Postman interface after sending a POST request to <https://gorest.co.in/public/v2/users>. The 'Body' tab is selected, showing the raw JSON response:

```
1 {  
2   "id": 7464998,  
3   "name": "The Hon. Aanandinii Varrier",  
4   "email": "varrier_aanandinii_the_hon@mohrer.test",  
5   "gender": "female",  
6   "status": "inactive"  
7 }
```

The response status is 201 Created, with a total time of 1135 ms and a size of 1.2 KB.

28. Create a python file in PyCharm editor



29. Sample POST (in PyCharm using Python code)

```
# import:
import requests

# variable(s):
head = {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer 8e30aa089'
}

# request data or payload
request_payload = {
    "id": 7464998,
    "name": "The Hon. Aanandinii Varrier",
    "email": "varrier_aanandinii_the_hon@mohreR.test",
    "gender": "female",
    "status": "inactive"
}

# store the url in a variable
url = "https://gorest.co.in/public/v2/users"

# ***CODE***
print("-----")

# POST method
postResponse = requests.post(url, headers=head, json=request_payload)

# display the response CODE
print("Successful response as per code below:")
print(postResponse.status_code)

# display the print text if expected code passed
assert postResponse.status_code == 201
print("With status code 201, API test run passed.")

print("-----")

# display the json response
print("Below is the response after the POST method.")
print(postResponse.json())

print("-----")

# run a GET method to validate the POST method
getResponse = requests.get(url + '/' + str(postResponse.json()['id']), headers=head)

# display the json response
print("Below is to check the data have been added after the POST method.")
print(postResponse.json())

Process finished with exit code 0
```

30. Here is the code in text

```
# import:
import requests

# variable(s):
head = {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer <place your own token here>' # add your token here
}

# request data or payload
request_payload ={
    "id": 7464998,
    "name": "The Hon. Aanandinii Varrier",
    "email": "varrier_aanandinii_the_hon@mohreR.test",
    "gender": "female",
    "status": "inactive"
}

# store the url in a variable
url = "https://gorest.co.in/public/v2/users"

# ***CODE***
print("-----")
print("-----")

# POST method
postResponse = requests.post(url, headers=head, json=request_payload)

# display the response CODE
print("Successful response as per code below:")
print(postResponse.status_code)

# display the print text if expected code passed
assert postResponse.status_code == 201
print("With status code 201, API test run passed.")

print("-----")
print("-----")

# display the json response
print("Below is the response after the POST method.")
print(postResponse.json())

print("-----")
print("-----")

# run a GET method to validate the POST method
getResponse = requests.get(url + '/' + str(postResponse.json()['id']), headers=head)

# display the json response
print("Below is to check the data have been added after the POST method.")
print(postResponse.json())
```

31. Passing Parameters in URL

Go REST sampling APIs

GraphQL and REST API for Testing and Prototyping
fake data | real responses | 24/7 online

Resources

URL	Status
https://gorest.co.in/public/v2/users	2810
https://gorest.co.in/public/v2/posts	2883
https://gorest.co.in/public/v2/comments	2878
https://gorest.co.in/public/v2/todos	1455

Nested Resources

Resource	Description
GET /public/v2/users/6941918/posts	Retrieves user posts
GET /public/v2/posts/6941918/comments	Retrieves post comments
GET /public/v2/users/6941918/todos	Retrieves user todos

Trying it Out

Method	URL	Description
POST	/public/v2/users	Create a new user
GET	/public/v2/users/6941918	Get user details
PUT PATCH	/public/v2/users/6941918	Update user details
DELETE	/public/v2/users/6941918	Delete user

• Do not post your personal data like name, email, phone, photo etc..
• For pagged results parameter "page" and "per_page" should be passed in url ex: GET [/public/v2/users?page=1&per_page=20](#) (max 100 results per page)
• Request methods PUT, POST, PATCH, DELETE needs access token, which needs to be passed with "Authorization" header as Bearer token.
• API Versions [/public-api/*](#), [/public/v1/*](#) and [/public/v2/*](#)
• [Get your access token](#)

GraphQL Endpoint

- GraphQL API is available at <https://gorest.co.in/public/v2/graphQL>
- [View GraphQL json schema](#)
- [View GraphQL schema](#)

```
{  
  "id": 7464998,  
  "name": "The Hon. Aanandinii Varrier",  
  "email": "varrier_aanandinii_the_hon@mohre.test",  
  "gender": "female",  
  "status": "inactive"  
},  
{  
  "id": 7463045,  
  "name": "Samir Gandhi",  
  "email": "gandhi.samir@ledner.example",  

```

32. Sample GET (copy the sampling url from Go REST and into Postman (add parameters)

Go REST

Resources

<https://gorest.co.in/public/v2/users>

2810

Postman

HTTP Activities_API / PassParmlnURLTest

Save Share

GET https://gorest.co.in/public/v2/users?page=4&per_page=2

Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit:
page	4		
per_page	2		

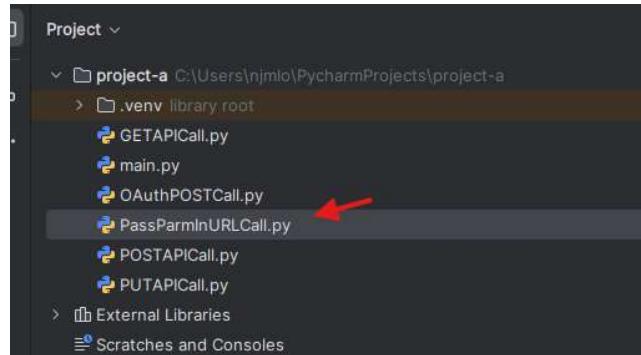
Body Cookies Headers (29) Test Results

200 OK 1090 ms 1.4 KB

Pretty Raw Preview Visualize JSON

```
1 [  
2   {  
3     "id": 7463003,  
4     "name": "Ms. Vedanga Joshi",  
5     "email": "veranga_ms_joshi@hammels-kuhin.test",  
6     "gender": "male",  
7     "status": "active"  
8   },  
9   {  
10    "id": 7463032,  
11    "name": "Sarvin Kocchar",  
12    "email": "sarvin_kocchar@steuber.example",  
13    "gender": "female",  
14    "status": "active"  
15  }  
16 ]
```

33. Create a python file in PyCharm editor



34. Sample GET (in PyCharm using Python code)

```
PassParamInURLCall.py x

1 # import:
2 import requests
3
4 # variable(s):
5
6 # store the url in a variable
7 url = "https://gorest.co.in/public/v2/users"
8
9 # parameters to be passed and part of the request
10 parm = {
11     'page': 1,
12     'per_page': 2
13 }
14
15 # ***CODE***
16 print("-----")
17
18 # GET method
19 getResponse = requests.get(url, params=parm)
20
21 # display the response CODE
22 print("Successful response as per code below:")
23 print(getResponse.status_code)
24
25 # display the print text if expected code passed
26 assert getResponse.status_code == 200
27 print("With status code 200, API test run passed.")
28
29 print("-----")
30
31 # display the json response
32 print("Below is the response after the POST method.")
33 print(getResponse.json())
```

```
Run PassParMinURLCall x
C:\Users\njm1o\PycharmProjects\project-a\.venv\Scripts\python.exe C:\Users\njm1o\PycharmProjects\project-a\PassParMinURLCall.py
=====
Successful response as per code below:
200
With status code 200, API test run passed.
-----
Below is the response after the POST method.
[{"id": 7643041, "name": "Amb. Anjushree Desai", "email": "anjushree_desai@mosciski.test", "gender": "male", "status": "inactive"}, {"id": 7643040, "name": "Bhisham Pilla", "email": "pilla.bhisham@malsh.example", "gender": "female", "status": "inactive"}]
Process finished with exit code 0
```

35. Using Json Beautifier to display response result nicely (copy and paste the result)

36. Here is the code in text

```
# import:
import requests

# variable(s):

# store the url in a variable
url = "https://gorest.co.in/public/v2/users"

# parameters to be passed and part of the request
parm = {
    'page': 1,
    'per_page': 2
}

# ***CODE***
print("-----")
-----)

# GET method
getResponse = requests.get(url, params=parm)

# display the response CODE
print("Successful response as per code below:")
print(getResponse.status_code)

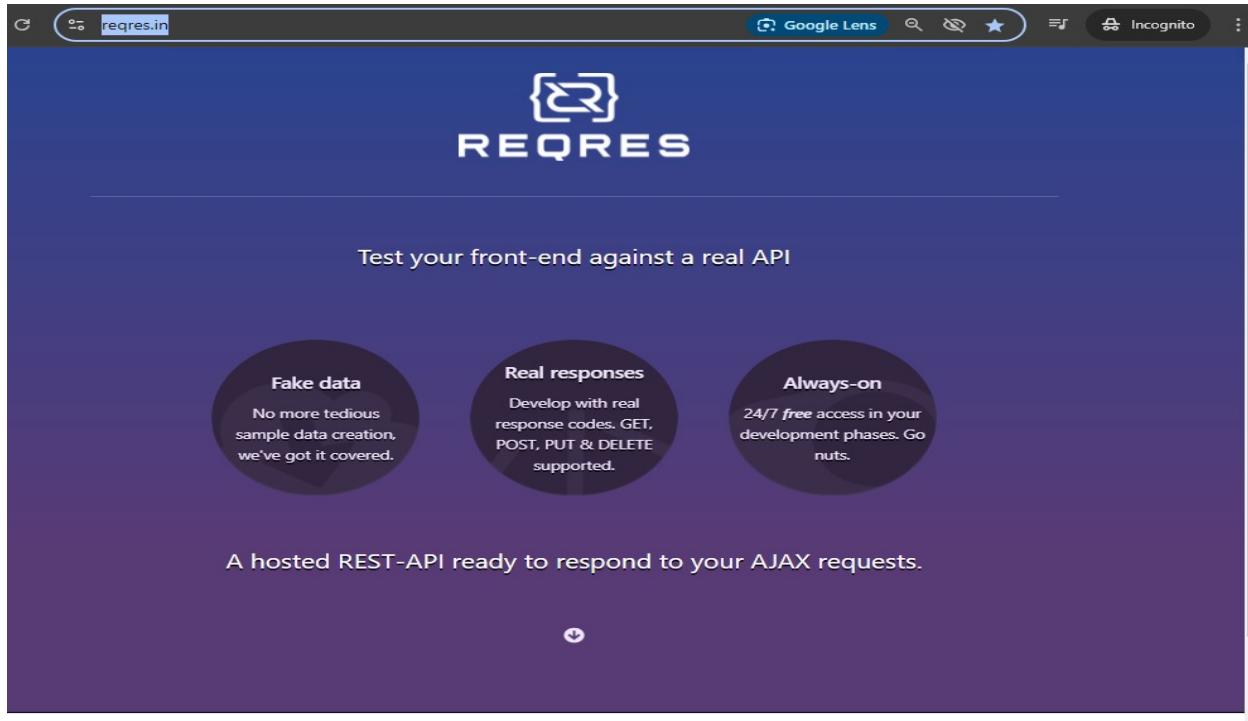
# display the print text if expected code passed
assert getResponse.status_code == 200
print("With status code 200, API test run passed.")

print("-----")
-----)

# display the json response
print("Below is the response after the POST method.")
print(getResponse.json())
```

37. Passing payload from JSON file

REQRES sampling APIs



Give it a try

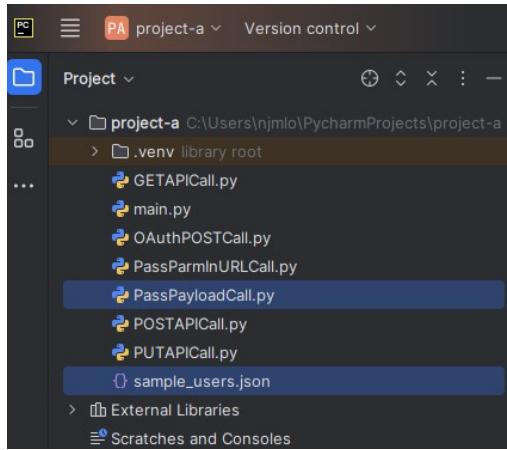
SUPPORT REQRES

The screenshot shows the REQRES API testing interface. On the left, a sidebar lists various API endpoints with their methods and descriptions. Red arrows point to the 'LIST USERS' (GET) and 'CREATE' (POST) endpoints. The main area has two panes: 'Request' and 'Response'. The 'Request' pane shows a JSON payload for creating a user: { "name": "morpheus", "job": "leader" }. The 'Response' pane shows a successful response: 201 { "name": "morpheus", "job": "leader", "id": "456", "createdAt": "2024-10-11T17:24:54.082Z" }.

Method	Endpoint	Description
GET	LIST USERS	
GET	SINGLE USER	
GET	SINGLE USER NOT FOUND	
GET	LIST <RESOURCE>	
GET	SINGLE <RESOURCE>	
GET	SINGLE <RESOURCE> NOT FOUND	
POST	CREATE	
PUT	UPDATE	
PATCH	UPDATE	
DELETE	DELETE	
POST	REGISTER - SUCCESSFUL	
POST	REGISTER - UNSUCCESSFUL	
POST	LOGIN - SUCCESSFUL	
POST	LOGIN - UNSUCCESSFUL	
GET	DELAYED RESPONSE	

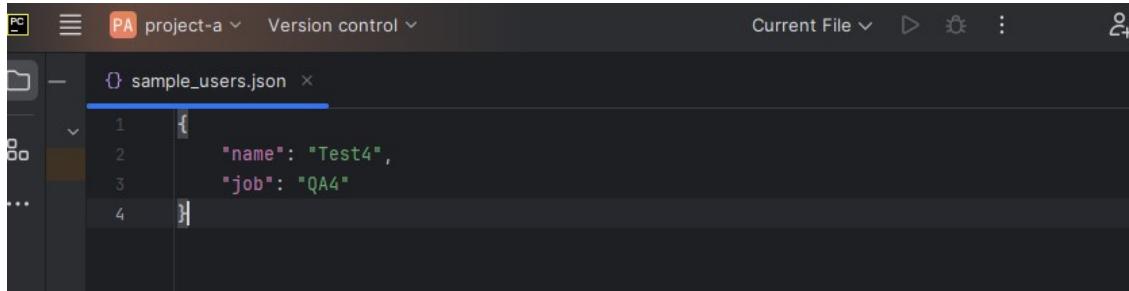
38. Create file(s) in PyCharm editor

Python and JSON file



39. Sample POST (in PyCharm using Python code)

JSON file



Python file

```
Run  PassPayloadCall x

C:\Users\njmlo\PycharmProjects\project-a\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\project-a\PassPayloadCall.py
-----
Successful response as per code below:
201
With status code 201, API test run passed.
-----
Below is the response after the POST method (JSON file display).
{'name': 'Test4', 'job': 'QA4', 'id': '153', 'createdAt': '2024-10-11T18:38:31.739Z'}
Below is the response after the POST method (TEXT file display).
{"name":"Test4","job":"QA4","id":"153","createdAt":"2024-10-11T18:38:31.739Z"}

Process finished with exit code 0
```

40. Here is the code in text

```
# import:
import requests
import json

# variable(s):

# store the url in a variable
url = "https://reqres.in/"

head = {
    'Content-Type': 'application/json'
}

# request data or payload from a JSON file
json_file = open('./sample_users.json')
json_load = json.load(json_file)

# request data or payload
# request_payload ={
#     "name": "Test",
#     "job": "QA"
# }

# ***CODE***
print("-----")
-----)

# POST method
response = (requests.post(
    url=url+"api/users",
    headers=head,
    data=json.dumps(json_load) # --this is for passing from other files
(including json file)
    # json=json_load --this is the cmd if only passing from a json file
))

# display the response CODE
print("Successful response as per code below:")
print(response.status_code)

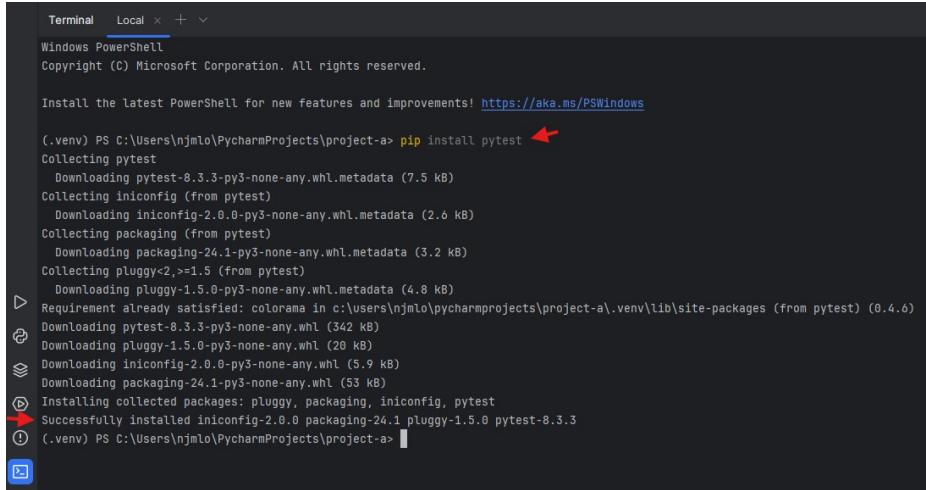
# display the print text if expected code passed
assert response.status_code == 201
print("With status code 201, API test run passed.")

print("-----")
-----)

# display the json response
print("Below is the response after the POST method (JSON file display.)")
print(response.json())
print("Below is the response after the POST method (TEXT file display.)")
print(response.text)
```

41. Python request API automation with PyTest

PyTest installation (in PyCharm Terminal)

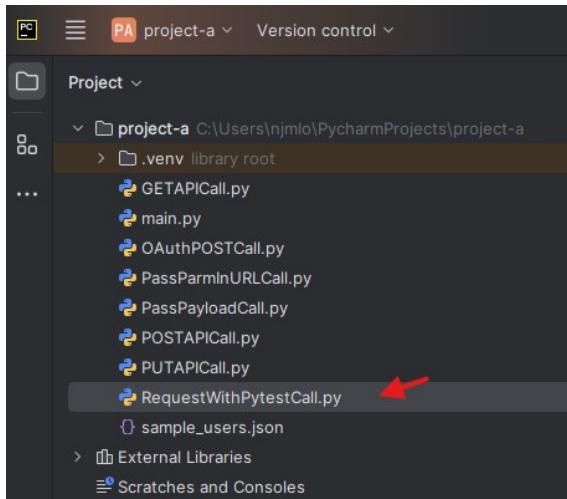


```
Terminal Local x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

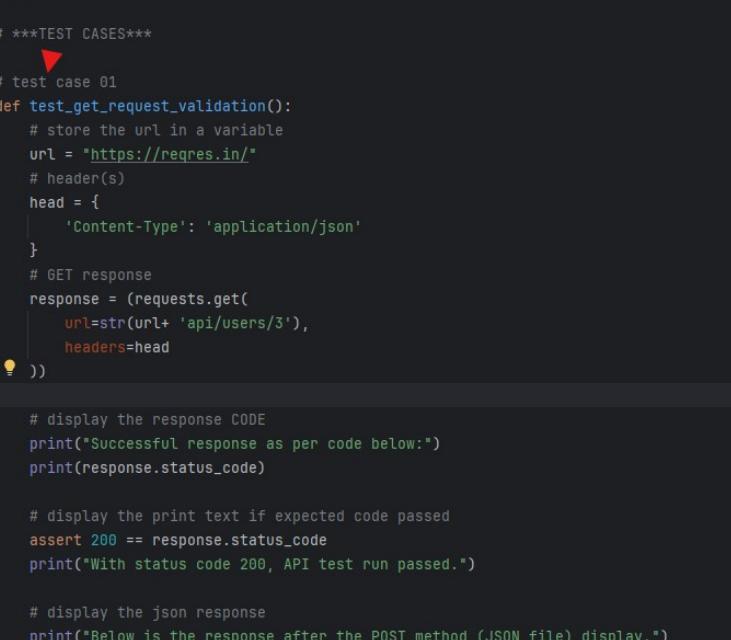
(.venv) PS C:\Users\njmlo\PycharmProjects\project-a> pip install pytest ↵
Collecting pytest
  Downloading pytest-8.3.3-py3-none-any.whl.metadata (7.5 kB)
Collecting iniconfig (from pytest)
  Downloading iniconfig-2.0.0-py3-none-any.whl.metadata (2.6 kB)
Collecting packaging (from pytest)
  Downloading packaging-24.1-py3-none-any.whl.metadata (3.2 kB)
Collecting pluggy<2,>=1.5 (from pytest)
  Downloading pluggy-1.5.0-py3-none-any.whl.metadata (4.8 kB)
D Requirement already satisfied: colorama in c:\users\njmlo\pycharmprojects\project-a\.venv\lib\site-packages (from pytest) (0.4.6)
D Downloading pytest-8.3.3-py3-none-any.whl (342 kB)
D Downloading pluggy-1.5.0-py3-none-any.whl (20 kB)
D Downloading iniconfig-2.0.0-py3-none-any.whl (5.9 kB)
D Downloading packaging-24.1-py3-none-any.whl (53 kB)
B Installing collected packages: pluggy, packaging, iniconfig, pytest
B Successfully installed iniconfig-2.0.0 packaging-24.1 pluggy-1.5.0 pytest-8.3.3
I (.venv) PS C:\Users\njmlo\PycharmProjects\project-a>
```

42. Create a python file in PyCharm editor



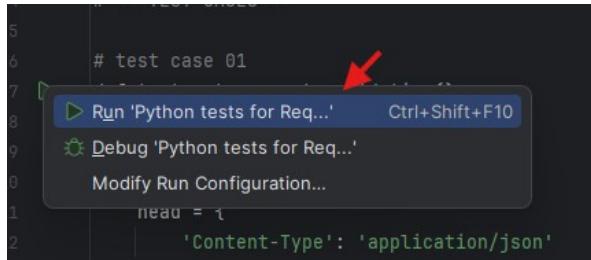
43. Sample GET (in PyCharm using Python code and PyTest to run the “test”)

Python file



```
# import:  
import requests  
  
# ***TEST CASES***  
  
# test case 01  
def test_get_request_validation():  
    # store the url in a variable  
    url = "https://reqres.in/"  
    # header(s)  
    head = {  
        'Content-Type': 'application/json'  
    }  
    # GET response  
    response = (requests.get(  
        url=url+ 'api/users/3'),  
        headers=head  
    )  
  
    # display the response CODE  
    print("Successful response as per code below:")  
    print(response.status_code)  
  
    # display the print text if expected code passed  
    assert 200 == response.status_code  
    print("With status code 200, API test run passed.")  
  
    # display the json response  
    print("Below is the response after the POST method (JSON file) display.")  
    print(response.json())  
    print("Below is the response after the POST method (TEXT file) display.")  
    print(response.text)
```

PyTest test run



Response

```
Run Python tests for RequestWithPytestCall.py::test_get_request_validation ... 
  ✓ Test Results 143ms Tests passed: 1 / 1 Total: 143ms
  C:\Users\user\PycharmProjects\project-a\venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2024.2.3/plugins/python-ce/helpers/pycharm/_jb_pytest_runner.py" --target RequestWithPytestCall.py::test_get_request_validation
  Testing started at 3:51 p.m. ...
  Launching pytest with arguments RequestWithPytestCall.py::test_get_request_validation --no-header --no-summary -q in C:\Users\user\PycharmProjects\project-a
  ===== test session starts =====
  collecting ... collected 1 item

  RequestWithPytestCall.py::test_get_request_validation PASSED [100m]Successful response as per code below:
  200
  WebKit testing code 200, API test run passed.
  Below is the response after the POST method (JSON file display).
  {"data": {"id": 3, "email": "emma.wong@reqres.in", "first_name": "Emma", "last_name": "Wong", "avatar": "https://reqres.in/img/faces/3-image.jpg"}, "support": {"url": "https://reqres.in/#support-heading", "text": "To keep ReqRes free, contributions towards server costs are appreciated."}}
  Below is the response after the POST method (Text file) display.
  {"data": {"id": 3, "email": "emma.wong@reqres.in", "first_name": "Emma", "last_name": "Wong", "avatar": "https://reqres.in/img/faces/3-image.jpg"}, "support": {"url": "https://reqres.in/#support-heading", "text": "To keep ReqRes free, contributions towards server costs are appreciated."}}
  ===== 1 passed in 0.96s =====
  Process finished with exit code 0
```

44. Here is the code in text

```
# import:
import requests

# ****TEST CASES****

# test case 01
def test_get_request_validation():
    # store the url in a variable
    url = "https://reqres.in/"
    # header(s)
    head = {
        'Content-Type': 'application/json'
    }
    # GET response
    response = (requests.get(
        url=url+ 'api/users/3',
        headers=head
    ))

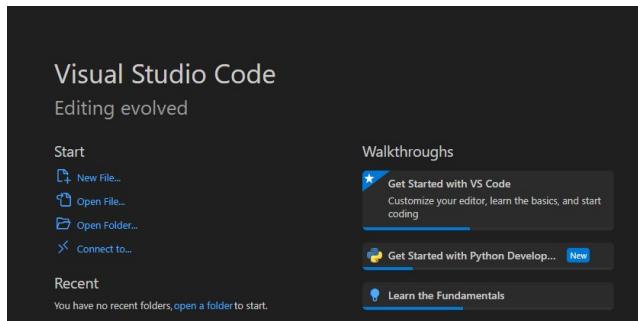
    # display the response CODE
    print("Successful response as per code below:")
    print(response.status_code)

    # display the print text if expected code passed
    assert 200 == response.status_code
    print("With status code 200, API test run passed.")

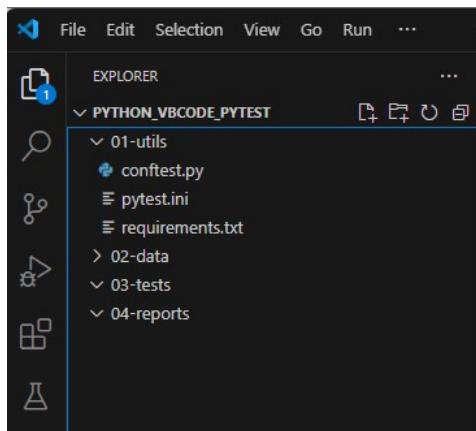
    # display the json response
    print("Below is the response after the POST method (JSON file) display.")
    print(response.json())
    print("Below is the response after the POST method (TEXT file) display.")
    print(response.text)
```

45. API Automation framework (GET method) using PyTest request libraries (in Visual Studio Code)

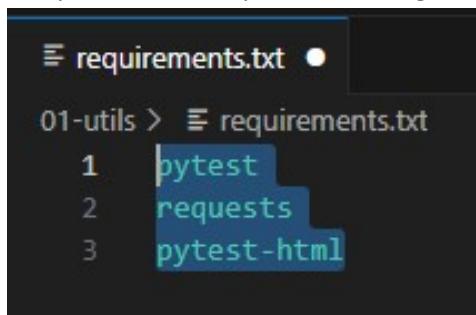
Ensure that the Visual Studio Code is setup



With the following files

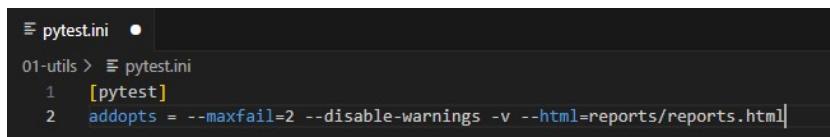


And part of the setup, the following libraries are installed (requirements.txt)

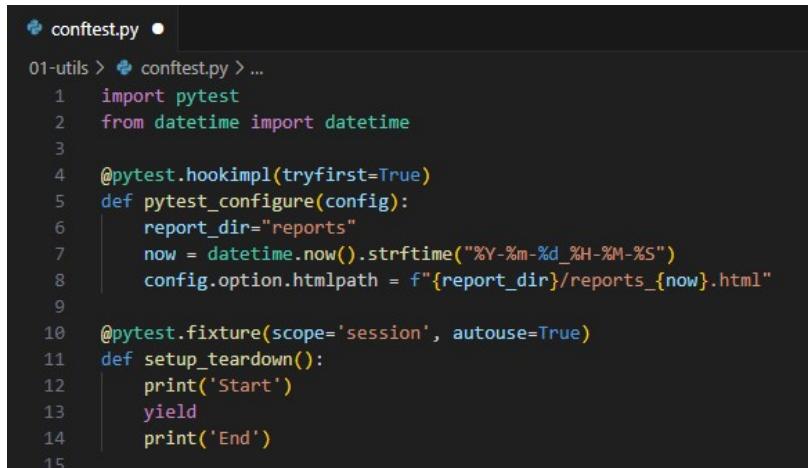


Further, the following are outlined in the following files:

pytest.ini

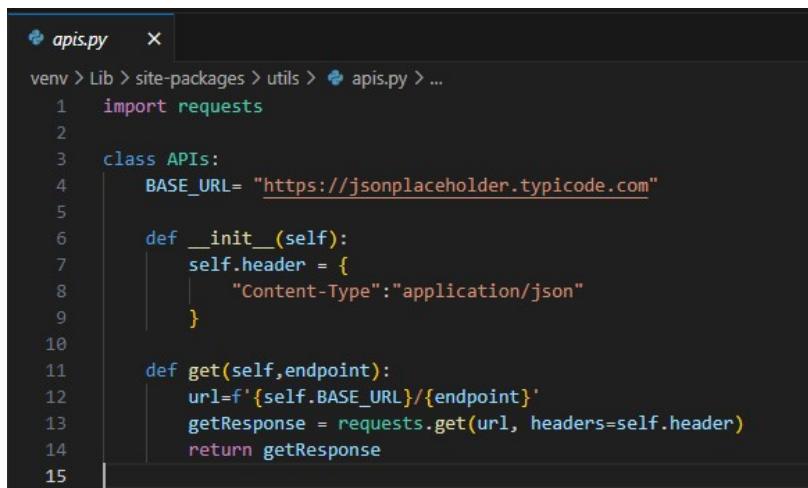


conftest.py



```
conftest.py
01-utils > conftest.py > ...
1 import pytest
2 from datetime import datetime
3
4 @pytest.hookimpl(tryfirst=True)
5 def pytest_configure(config):
6     report_dir="reports"
7     now = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
8     config.option.htmlpath = f"{report_dir}/reports_{now}.html"
9
10 @pytest.fixture(scope='session', autouse=True)
11 def setup_teardown():
12     print('Start')
13     yield
14     print('End')
15
```

apis.py



```
apis.py
venv > Lib > site-packages > utils > apis.py > ...
1 import requests
2
3 class APIs:
4     BASE_URL= "https://jsonplaceholder.typicode.com"
5
6     def __init__(self):
7         self.header = {
8             "Content-Type": "application/json"
9         }
10
11     def get(self,endpoint):
12         url=f'{self.BASE_URL}/{endpoint}'
13         getResponse = requests.get(url, headers=self.header)
14         return getResponse
15
```

46. Sampling source of get users

url: <https://jsonplaceholder.typicode.com/>

JSONPlaceholder

Guide Sponsor this project Blog My JSON Server

{JSON} Placeholder

Free fake and reliable API for testing and prototyping.

Powered by [JSON Server](#) + [LowDB](#).

Serving ~3 billion requests each month.

Resources

JSONPlaceholder comes with a set of 6 common resources:

<u>/posts</u>	100 posts
<u>/comments</u>	500 comments
<u>/albums</u>	100 albums
<u>/photos</u>	5000 photos
<u>/todos</u>	200 todos
<u>/users</u>	10 users 



47. Run a test case in Visual Studio Code (get the same set of data (users))

```
test_getuser.py x
tests > test_getuser.py > test_get_user_validation
1 import pytest
2 from utils.apis import APIs
3
4 @pytest.fixture(scope='module')
5 def apis():
6     return APIs()
7
8 # test case 01 - get all users
9
10 def test_get_user_validation(apis):
11     response = apis.get('users')
12
13     # display the response
14     print(response.json())
15
16     # validate the response code if passed
17     assert response.status_code == 200
18     assert len(response.json()) > 0
19
20     # output returned response code and flag as passed
21     str_code = str(response.status_code)
22     print("****Code " + str_code + " is passed!****")
```

48. Run the PyTest command to get expected response results and response code

Terminal:

(venv) C:\Users\njml0\Desktop\Test Automation notes\Python_VBCode_PyTest>pytest -s

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ⌘ and + × ⌂ ⌂ ... X

phone: '1-463-123-4447', 'website': 'hamro.info', 'company': {'name': 'Romaguera-Jacobson', 'catchPhrase': 'Face to face bifurcated interface', 'bs': 'e-enable strategic applications'}, ('id': 4, 'name': 'Patricia Lebsack', 'username': 'Karianne', 'email': 'Julianne.O'Connor@kory.org', 'address': {'street': 'Hoeger Wall', 'suite': 'Suite 400', 'city': 'South Elvis', 'zipCode': '53910-4257', 'geo': {'lat': '29.4572', 'lng': '-104.2990'}}, 'phone': '483-178-9623 x156', 'website': 'kale.biz', 'company': {'name': 'Rachael-Corkery', 'catchPhrase': 'Multi-tiered zero tolerance productivity', 'bs': 'transition cutting-edge web services'}, ('id': 5, 'name': 'Chelsey Dietrich', 'username': 'Karren', 'email': 'Lucio.Lettingjamie.ca', 'address': {'street': 'Skiles Walks', 'suite': 'Suite 351', 'city': 'Roscoeview', 'zipCode': '33263', 'geo': {'lat': '-31.829', 'lng': '62.5342'}}, 'phone': '(254)954-1289', 'website': 'deanaro.info', 'company': {'name': 'Keebler LLC', 'catchPhrase': 'User-centric fault-tolerant solution', 'bs': 'revolutionize end-to-end systems'}, ('id': 6, 'name': 'Mrs. Dennis Schulist', 'username': 'Leopoldo.Corkery', 'email': 'Karley.Dietrichjasper.info', 'address': {'street': 'Norberto Crossing', 'suite': 'Apt. 990', 'city': 'South Christy', 'zipCode': '23895-1337', 'geo': {'lat': '-71.4197', 'lng': '71.7478'}}, 'phone': '1-477-935-8478 x1438', 'website': 'ola.org', 'company': {'name': 'Considine-Lockman', 'catchPhrase': 'Synchronised bottom-line interface', 'bs': 'e-enable innovative applications'}, ('id': 7, 'name': 'Kurtis Weissnat', 'username': 'Elwyn.Skiles', 'email': 'Telly.Hoeger@billy.biz', 'address': {'street': 'Rex Trail', 'suite': 'Suite 288', 'city': 'Howemouth', 'zipCode': '58804-1889', 'geo': {'lat': '24.2818', 'lng': '21.8984'}}, 'phone': '210.867.6332', 'website': 'elvis.io', 'company': {'name': 'Johns Group', 'catchPhrase': 'Configurable multimedia task-force', 'bs': 'generate enterprise e-tailers'}, ('id': 8, 'name': 'Nicholas Runa.FordhamV', 'username': 'Xavine_Kueen', 'email': 'Shewww@yrsandme.com', 'address': {'street': 'Ellsworth Summit', 'suite': 'Suite 729', 'city': 'Alivipar', 'zipCode': '45169', 'geo': {'lat': '14.3999', 'lng': '-120.7657'}}, 'phone': '586.493.6493 x148', 'website': 'jayzthe.com', 'company': {'name': 'Abenathy Group', 'catchPhrase': 'Implemented secondary concept', 'bs': 'e-enable extensible e-tailers'}, ('id': 9, 'name': 'Glenra Reichert', 'username': 'Delphine', 'email': 'Chain.McDermottJalana.io', 'address': {'street': 'Dayna Park', 'suite': 'Suite 449', 'city': 'Bartholemey', 'zipCode': '76495-3108', 'geo': {'lat': '24.6451', 'lng': '168.8898'}}, 'phone': '(775)976-6794 x41268', 'website': 'conrad.com', 'company': {'name': 'Yost and Sons', 'catchPhrase': 'Switchable contextually-based project', 'bs': 'aggregate real-time technologies'}, ('id': 10, 'name': 'Clementine DuBuque', 'username': 'Moriah.Stanton', 'email': 'Rey.Padberg@karina.biz', 'address': {'street': 'Kattie Turnpike', 'suite': 'Suite 190', 'city': 'Lebsackbury', 'zipCode': '31428-2861', 'geo': {'lat': '-38.2386', 'lng': '57.2229'}}}, 'phone': '924.648-3894', 'website': 'ambrose.net', 'company': {'name': 'Hoeger LLC', 'catchPhrase': 'Centralized empowering task-force', 'bs': 'target end-to-end model $'})}
***'Code 200 is passed'****
```

49. Here is the code in text

```
import pytest
from utils.apis import APIs

@pytest.fixture(scope='module')
def apis():
    return APIs()

# test case 01 - get all users

def test_get_user_validation(apis):
    response = apis.get('users')

    # display the response
    print(response.json())

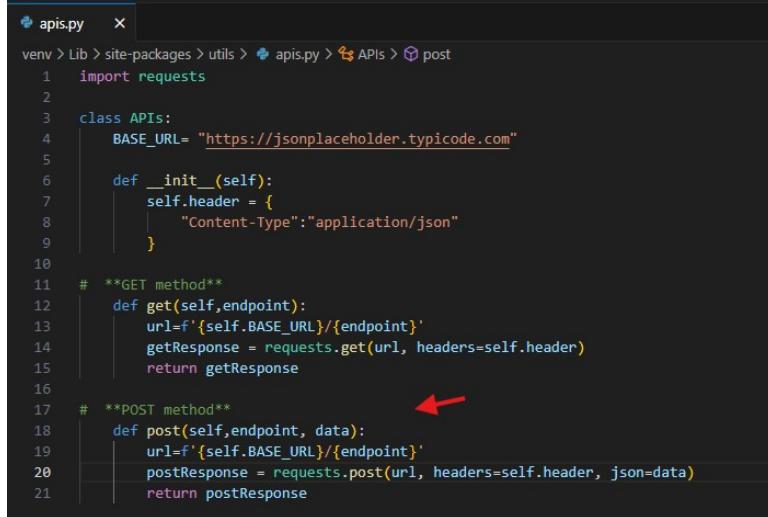
    # validate the response code if passed
    assert response.status_code == 200
    assert len(response.json()) > 0

    # output returned response code and flag as passed
    str_code = str(response.status_code)
    print("****Code " + str_code + " is passed!****")
```

50. API Automation framework (POST method) using PyTest request libraries (in Visual Studio Code)

As a continuation of the prior method framework, reuse and update some of the existing files:

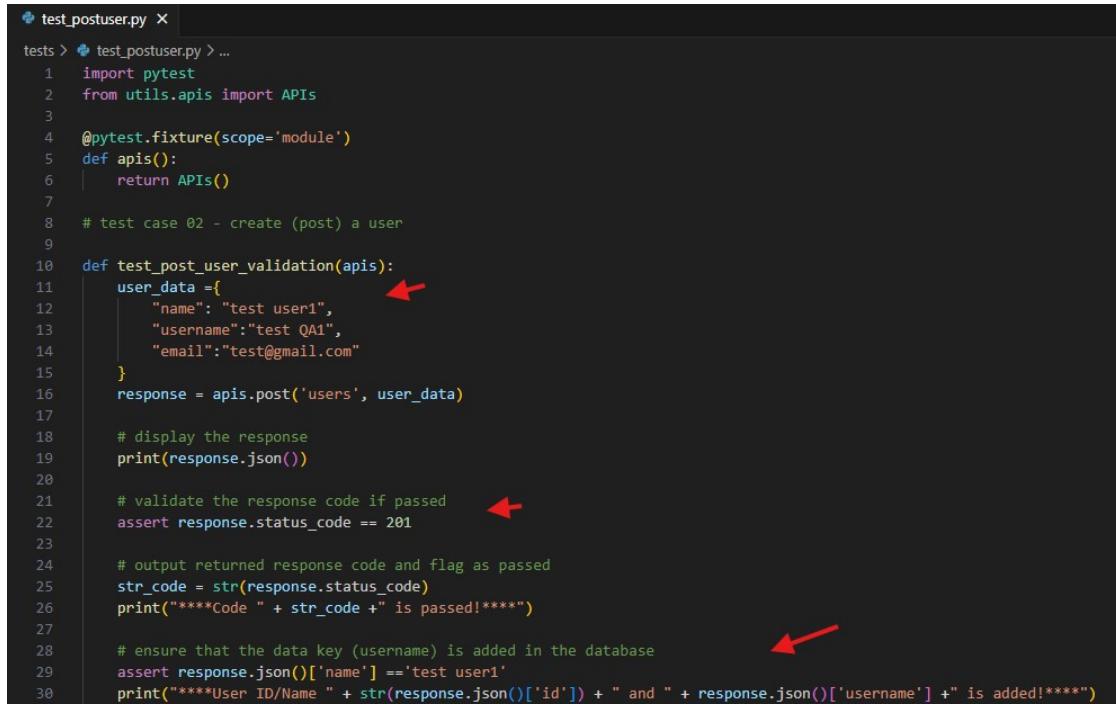
apis.py



```
venv > Lib > site-packages > utils > apis.py > APIs > post
1 import requests
2
3 class APIs:
4     BASE_URL= "https://jsonplaceholder.typicode.com"
5
6     def __init__(self):
7         self.header = {
8             "Content-Type": "application/json"
9         }
10
11     # **GET method**
12     def get(self,endpoint):
13         url=f'{self.BASE_URL}/{endpoint}'
14         getResponse = requests.get(url, headers=self.header)
15         return getResponse
16
17     # **POST method** ←
18     def post(self,endpoint, data):
19         url=f'{self.BASE_URL}/{endpoint}'
20         postResponse = requests.post(url, headers=self.header, json=data)
21         return postResponse
```

51. Run the test case in Visual Studio Code (add a user)

Sampling of no error



```
tests > test_postuser.py > ...
1 import pytest
2 from utils.apis import APIs
3
4 @pytest.fixture(scope='module')
5 def apis():
6     return APIs()
7
8 # test case 02 - create (post) a user
9
10 def test_post_user_validation(apis): ←
11     user_data ={←
12         "name": "test user1",
13         "username": "test QAI",
14         "email": "test@gmail.com"
15     }
16     response = apis.post('users', user_data)
17
18     # display the response
19     print(response.json())
20
21     # validate the response code if passed ←
22     assert response.status_code == 201
23
24     # output returned response code and flag as passed
25     str_code = str(response.status_code)
26     print("****Code " + str_code +" is passed!****")
27
28     # ensure that the data key (username) is added in the database ←
29     assert response.json()['name'] =='test user1'
30     print("****User ID/Name " + str(response.json()['id']) + " and " + response.json()['username'] + " is added!****")
```

```
tests\test_postuser.py {'name': 'test user1', 'username': 'test QA1', 'email': 'test@gmail.com', 'id': 11}
****Code 201 is passed!****
****User ID/Name 11 and test QA1 is added!****
```

Sampling with error

```
* test_postuser.py x
tests > ⚡ test_postuser.py > ...
1  import pytest
2  from utils.apis import APIs
3
4  @pytest.fixture(scope='module')
5  def apis():
6      return APIs()
7
8  # test case 02 - create (post) a user
9
10 def test_post_user_validation(apis):
11     user_data ={
12         "name": "test user1"
13         "username": "test QA1",
14         "email": "test@gmail.com"
15     }
16     response = apis.post('users', user_data)
17
18     # display the response
19     print(response.json())
20
21     # validate the response code if passed
22     assert response.status_code == 201
23
24     # output returned response code and flag as passed
25     str_code = str(response.status_code)
26     print("****Code " + str_code + " is passed!****")
27
28     # ensure that the data key (username) is added in the database
29     assert response.json()['name'] == 'test user1'
30     print("****User ID/Name " + str(response.json()['id']) + " and " + response.json()['username'] + " is added!****")
31
```

```
tests\test_postuser.py:29: AssertionError
# ensure that the data key (username) is added in the database
assert response.json()['name'] == 'test user1'
AssertionError: assert 'test user1' == 'test user2'

E   - test user2
E   ?
E   + test user1

tests\test_postuser.py:29: AssertionError
=====
short test summary info --
```

52. Here is the code in text

```
import pytest
from utils.apis import APIs

@pytest.fixture(scope='module')
def apis():
    return APIs()

# test case 02 - create (post) a user

def test_post_user_validation(apis):
    user_data ={
        "name": "test user1",
        "username": "test QA1",
        "email": "test@gmail.com"
    }
    response = apis.post('users', user_data)

    # display the response
```

```

print(response.json())

# validate the response code if passed
assert response.status_code == 201

# output returned response code and flag as passed
str_code = str(response.status_code)
print("****Code " + str_code +" is passed!****")

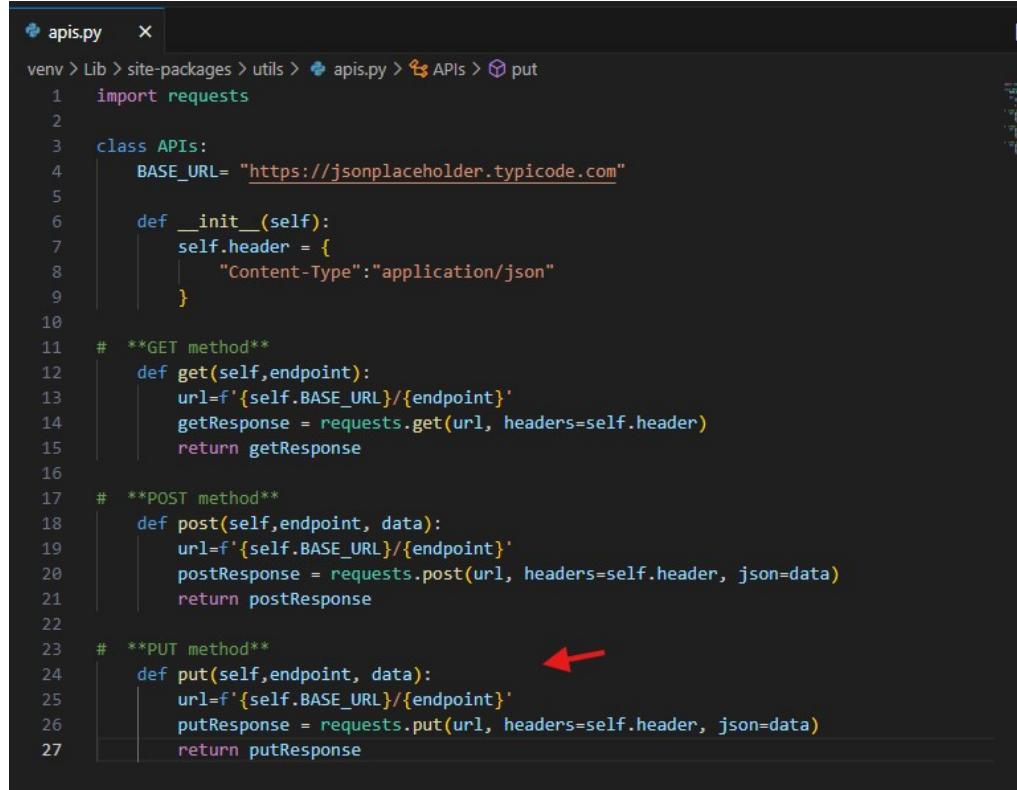
# ensure that the data key (username) is added in the database
assert response.json()['name'] =='test user12'
print("****User ID/Name " + str(response.json()['id']) + " and " +
response.json()['username'] +" is added!****")

```

53. API Automation framework (PUT method) using PyTest, Python, and Requests libraries (in Visual Studio Code)

As a continuation of the prior method framework, reuse and update some of the existing files:

apis.py



```

apis.py  X
venv > Lib > site-packages > utils > apis.py > APIs > put
1 import requests
2
3 class APIs:
4     BASE_URL= "https://jsonplaceholder.typicode.com"
5
6     def __init__(self):
7         self.header = {
8             "Content-Type": "application/json"
9         }
10
11    # **GET method**
12    def get(self,endpoint):
13        url=f'{self.BASE_URL}/{endpoint}'
14        getResponse = requests.get(url, headers=self.header)
15        return getResponse
16
17    # **POST method**
18    def post(self,endpoint, data):
19        url=f'{self.BASE_URL}/{endpoint}'
20        postResponse = requests.post(url, headers=self.header, json=data)
21        return postResponse
22
23    # **PUT method**
24    def put(self,endpoint, data):
25        url=f'{self.BASE_URL}/{endpoint}'
26        putResponse = requests.put(url, headers=self.header, json=data)
27        return putResponse

```

54. Run the test case in Visual Studio Code (update a user)

Sampling of no error

```
tests/test_putuser.py x
tests > tests/test_putuser.py > ...
1 import pytest
2 from utils.apis import APIs
3
4 @pytest.fixture(scope='module')
5 def apis():
6     return APIs()
7
8 # test case 03 - update (put) a user
9
10 def test_update_user_validation(apis):
11     user_data ={
12         "name": "test user1",
13         "username": "test QA2",
14         "email": "test@gmail.com"
15     }
16     response = apis.put('users/1', user_data)
17
18     # display the response
19     print(response.json())
20
21     # validate the response code if passed
22     assert response.status_code == 200
23
24     # output returned response code and flag as passed
25     str_code = str(response.status_code)
26     print("****Code " + str_code + " is passed!****")
27
28     # ensure that the data key (username) is updated in the database
29     assert response.json()['username'] == 'test QA2'
30     print("****User ID/Name 1 and test QA2 is updated!****")
31
```

```
tests\test_putuser.py {'name': 'test user1', 'username': 'test QA2', 'email': 'test@gmail.com', 'id': 1}
****Code 200 is passed!
****User ID/Name 1 and test QA2 is updated!
```

Sampling with error

```
tests/test_putuser.py x
tests > tests/test_putuser.py > test_update_user_validation
1 import pytest
2 from utils.apis import APIs
3
4 @pytest.fixture(scope='module')
5 def apis():
6     return APIs()
7
8 # test case 03 - update (put) a user
9
10 def test_update_user_validation(apis):
11     user_data ={
12         "name": "test user1",
13         "username": "test QA2",
14         "email": "test@gmail.com"
15     }
16     response = apis.put('users/1', user_data)
17
18     # display the response
19     print(response.json())
20
21     # validate the response code if passed
22     assert response.status_code == 200
23
24     # output returned response code and flag as passed
25     str_code = str(response.status_code)
26     print("****Code " + str_code + " is passed!****")
27
28     # ensure that the data key (username) is updated in the database
29     assert response.json()['username'] == 'test QA3'
30     print("****User ID/Name 1 and test QA3 is updated!****")
31
```

```

>     # ensure that the data key (username) is added in the database
>     assert response.json()['username'] == 'test QA3'
E   AssertionError: assert 'test QA2' == 'test QA3'
E
E     - test QA3
E     ?
E     + test QA2
E     ?

tests\test_putuser.py:29: AssertionError
=====
short test summary info
FAILED tests\test_putuser.py::test_update_user_validation - AssertionError: assert 'test QA2' == 'test QA3'

```

55. Here is the code in text

```

import pytest
from utils.apis import APIs

@pytest.fixture(scope='module')
def apis():
    return APIs()

# test case 03 - update (put) a user

def test_update_user_validation(apis):
    user_data ={
        "name": "test user1",
        "username": "test QA2",
        "email": "test@gmail.com"
    }
    response = apis.put('users/1', user_data)

    # display the response
    print(response.json())

    # validate the response code if passed
    assert response.status_code == 200

    # output returned response code and flag as passed
    str_code = str(response.status_code)
    print("****Code " + str_code + " is passed!****")

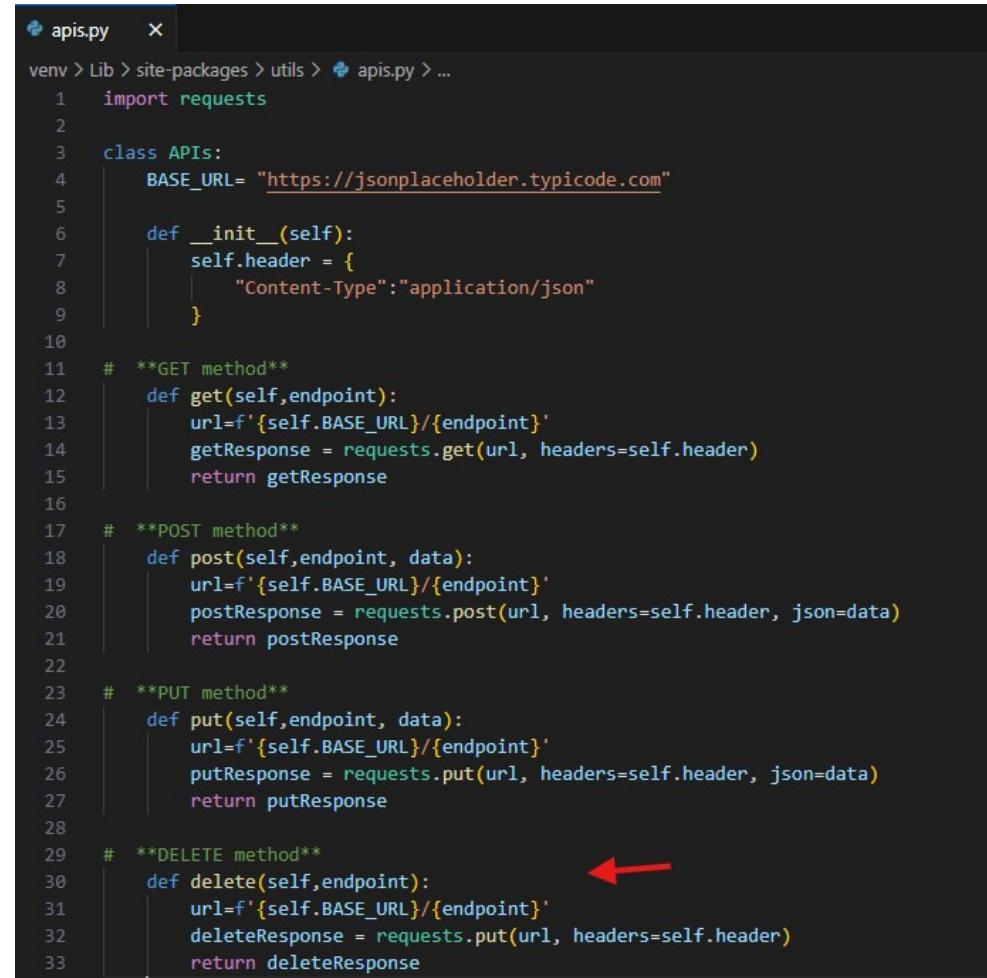
    # ensure that the data key (username) is updated in the database
    assert response.json()['username'] == 'test QA2'
    print("****User ID/Name " + str(response.json()['id']) + " and " +
response.json()['username'] + " is updated!****")

```

56. API Automation framework (DELETE method) using PyTest, Python, and Requests libraries (in Visual Studio Code)

As a continuation of the prior method framework, reuse and update some of the existing files:

apis.py



```
apis.py  X

venv > Lib > site-packages > utils > apis.py > ...

1  import requests
2
3  class APIs:
4      BASE_URL= "https://jsonplaceholder.typicode.com"
5
6      def __init__(self):
7          self.header = {
8              "Content-Type": "application/json"
9          }
10
11     # **GET method**
12     def get(self,endpoint):
13         url=f'{self.BASE_URL}/{endpoint}'
14         getResponse = requests.get(url, headers=self.header)
15         return getResponse
16
17     # **POST method**
18     def post(self,endpoint, data):
19         url=f'{self.BASE_URL}/{endpoint}'
20         postResponse = requests.post(url, headers=self.header, json=data)
21         return postResponse
22
23     # **PUT method**
24     def put(self,endpoint, data):
25         url=f'{self.BASE_URL}/{endpoint}'
26         putResponse = requests.put(url, headers=self.header, json=data)
27         return putResponse
28
29     # **DELETE method**
30     def delete(self,endpoint): ←
31         url=f'{self.BASE_URL}/{endpoint}'
32         deleteResponse = requests.delete(url, headers=self.header)
33         return deleteResponse
```

57. Run the test case in Visual Studio Code (delete a user)

Sampling of no error

```
test_deleteuser.py X
tests > test_deleteuser.py > ...
1 import pytest
2 from utils.apis import APIs
3
4 @pytest.fixture(scope='module')
5 def apis():
6     return APIs()
7
8 # test case 03 - delete a user
9
10 def test_delete_user_validation(apis):
11
12     # select and delete the record/data
13     response = apis.delete('users/1')
14
15     # display the response
16     print(response.json())
17
18     # validate the response code if passed
19     assert response.status_code == 200
20
21     # output returned response code and flag as passed
22     str_code = str(response.status_code)
23     print("****Code " + str_code + " is passed (data deleted!)****")
24
```

```
(venv) C:\Users\njml0\Desktop\Test Automation notes\Python_VBCode_PyTest>pytest -s
=====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\njml0\Desktop\Test Automation notes\Python_VBCode_PyTest
plugins: html-4.1.1, metadata-3.1.1
collected 4 items

tests\test_deleteuser.py {'id': 1}
****Code 200 is passed (data deleted!)****
```

Sampling with error

```
test_deleteuser.py X
tests > test_deleteuser.py > test_delete_user_validation
1 import pytest
2 from utils.apis import APIs
3
4 @pytest.fixture(scope='module')
5 def apis():
6     return APIs()
7
8 # test case 03 - delete a user
9
10 def test_delete_user_validation(apis):
11
12     # select and delete the record/data
13     response = apis.delete('users/1')
14
15     # display the response
16     print(response.json())
17
18     # validate the response code if passed
19     assert response.status_code == 201 ←
20
21     # output returned response code and flag as passed
22     str_code = str(response.status_code)
23     print("****Code " + str_code + " is passed (data deleted!)****")
24
```

```
(venv) C:\Users\njmlo\Desktop\Test Automation notes\Python_VBCode_PyTest>pytest -s
=====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\njmlo\Desktop\Test Automation notes\Python_VBCode_PyTest
plugins: html-4.1.1, metadata-3.1.1
collected 4 items

tests\test_deleteuser.py {'id': 1}
F
```

```
# validate the response code if passed
>     assert response.status_code == 201
E     assert 200 == 201
E       +  where 200 = <Response [200]>.status_code

tests\test_deleteuser.py:19: AssertionError
=====
FAILED tests/test_deleteuser.py::test_delete_user_validation - assert 200 == 201
```

58. Here is the code in text

```
import pytest
from utils.apis import APIs

@pytest.fixture(scope='module')
def apis():
    return APIs()

# test case 03 - delete a user

def test_delete_user_validation(apis):

    # select and delete the record/data
    response = apis.delete('users/1')

    # display the response
    print(response.json())

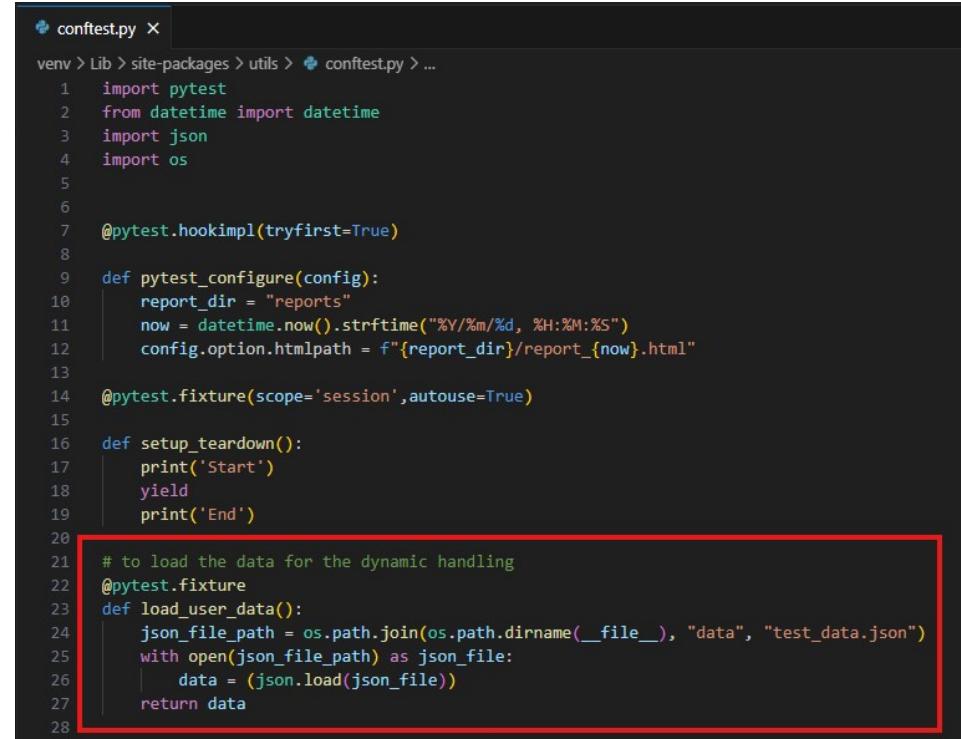
    # validate the response code if passed
    assert response.status_code == 200

    # output returned response code and flag as passed
    str_code = str(response.status_code)
    print("****Code " + str_code +" is passed (data deleted!)****")
```

59. API Automation framework using the dynamic data handling in POST method (in Visual Studio Code)

As a continuation of the prior method framework, reuse and update some of the existing files:

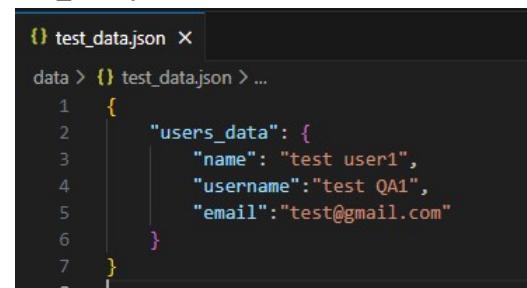
conftest.py



```
conftest.py X
venv > Lib > site-packages > utils > conftest.py > ...
1 import pytest
2 from datetime import datetime
3 import json
4 import os
5
6
7 @pytest.hookimpl(tryfirst=True)
8
9 def pytest_configure(config):
10     report_dir = "reports"
11     now = datetime.now().strftime("%Y/%m/%d, %H:%M:%S")
12     config.option.htmlpath = f"{report_dir}/report_{now}.html"
13
14 @pytest.fixture(scope='session', autouse=True)
15
16 def setup_teardown():
17     print('Start')
18     yield
19     print('End')
20
21 # to load the data for the dynamic handling
22 @pytest.fixture
23 def load_user_data():
24     json_file_path = os.path.join(os.path.dirname(__file__), "data", "test_data.json")
25     with open(json_file_path) as json_file:
26         data = json.load(json_file)
27     return data
28
```

Add a new file:

test_data.json



```
{} test_data.json X
data > {} test_data.json > ...
1 {
2     "users_data": {
3         "name": "test user1",
4         "username": "test QA1",
5         "email": "test@gmail.com"
6     }
7 }
```

60. Run the test case in Visual Studio Code (create a user)

Sampling of no error

```
test_postuser_dyna_handl.py X
tests > test_postuser_dyna_handl.py > test_post_user_validation
1 import pytest
2 from utils.apis import APIs
3 import uuid
4
5 @pytest.fixture(scope='module')
6 def apis():
7     return APIs()
8
9 # test case 02 - create (post) a user
10
11 def test_post_user_validation(apis, load_user_data):
12
13 # import test data from a file instead of getting locally from within this file
14 user_data = load_user_data["users_data"]
15 unique_email = f"{uuid.uuid4().hex[:8]}@yahoo.com"
16 user_data["email"] = unique_email
17 print("This is now the new email: " + unique_email)
18
19 response = apis.post('users', user_data)
20
21 # display the response
22 print(response.json())
23
24 # validate the response code if passed
25 assert response.status_code == 201
26
27 # output returned response code and flag as passed
28 str_code = str(response.status_code)
29 print("****Code " + str_code + " is passed!****")
30
31 # ensure that the data key (username) is added in the database
32 assert response.json()['name'] == 'test user1'
33 print("****New Email " + str(response.json()['email']) + " for ID " + str(response.json()['id']) + " is added!****")
34
```

```
tests\test_postuser_dyna_handl.py This is now the new email: ffd1232f@yahoo.com
{'name': 'test user1', 'username': 'test QA1', 'email': 'ffd1232f@yahoo.com', 'id': 11}
****Code 201 is passed!
****New Email ffd1232f@yahoo.com for ID 11 is added!
```

Sampling with error

```
test_postuser_dyna_handl.py X
tests > test_postuser_dyna_handl.py > ...
1 import pytest
2 from utils.apis import APIs
3 import uuid
4
5 @pytest.fixture(scope='module')
6 def apis():
7     return APIs()
8
9 # test case 02 - create (post) a user
10
11 def test_post_user_validation(apis, load_user_data):
12
13 # import test data from a file instead of getting locally from within this file
14 user_data = load_user_data["users_data"]
15 unique_email = f"{uuid.uuid4().hex[:8]}@yahoo.com"
16 user_data["email"] = unique_email
17 print("This is now the new email: " + unique_email)
18
19 response = apis.post('users', user_data)
20
21 # display the response
22 print(response.json())
23
24 # validate the response code if passed
25 assert response.status_code == 200 ←
26
27 # output returned response code and flag as passed
28 str_code = str(response.status_code)
29 print("****Code " + str_code + " is passed!****")
30
31 # ensure that the data key (username) is added in the database
32 assert response.json()['name'] == 'test user1'
33 print("****New Email " + str(response.json()['email']) + " for ID " + str(response.json()['id']) + " is added!****")
```

```

        # validate the response code if passed
>     assert response.status_code == 200
E     +   where 201 = <Response [201]>.status_code
tests\test_postuser_dyna_handl.py:25: AssertionError
=====
FAILED tests\test_postuser_dyna_handl.py::test_post_user_validation - assert 201 == 200
=====
short test summary info -->

```

61. Here is the code in text

```

import pytest
from utils.apis import APIs
import uuid

@pytest.fixture(scope='module')
def apis():
    return APIs()

# test case 02 - create (post) a user

def test_post_user_validation(apis, load_user_data):

    # import test data from a file instead of getting locally from within this file
    user_data = load_user_data["users_data"]
    unique_email = f"{uuid.uuid4().hex[:8]}@yahoo.com"
    user_data["email"] = unique_email
    print("This is now the new email: " + unique_email)

    response = apis.post('users', user_data)

    # display the response
    print(response.json())

    # validate the response code if passed
    assert response.status_code == 201

    # output returned response code and flag as passed
    str_code = str(response.status_code)
    print("****Code " + str_code +" is passed!****")

    # ensure that the data key (username) is added in the database
    assert response.json()['name'] =='test user1'
    print("****New Email " + str(response.json()['email']) + " for ID " +
str(response.json()['id']) + " is added!****")

```

62. Clean (end to end) 5 test cases run

63. GitHub setup (Create, Stage, Commit, and Publish with VS Code)

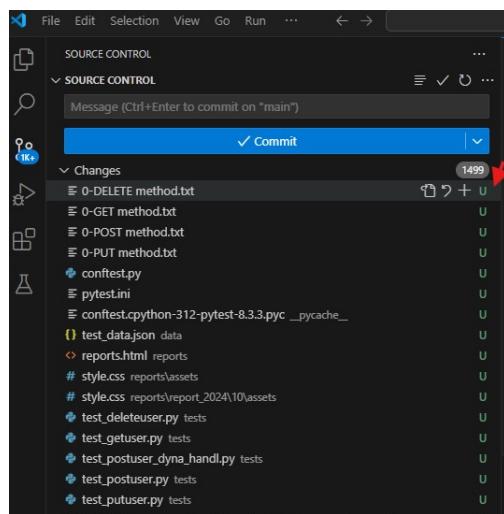
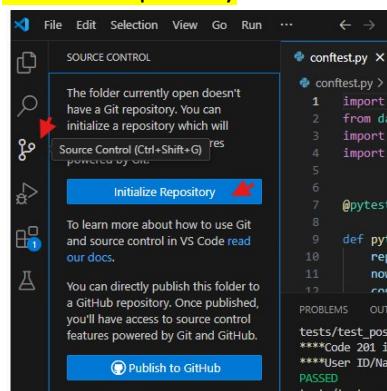
Have a GitHub account



Download and install Git app

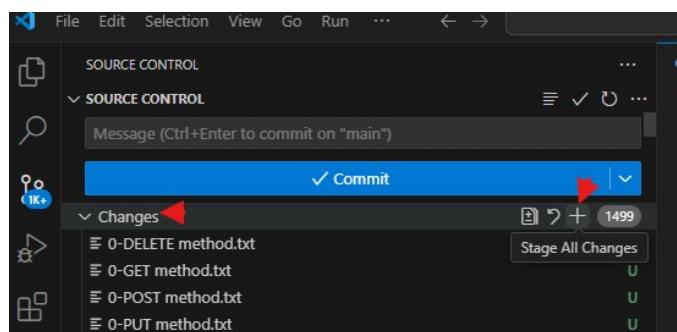


Initialize Repository

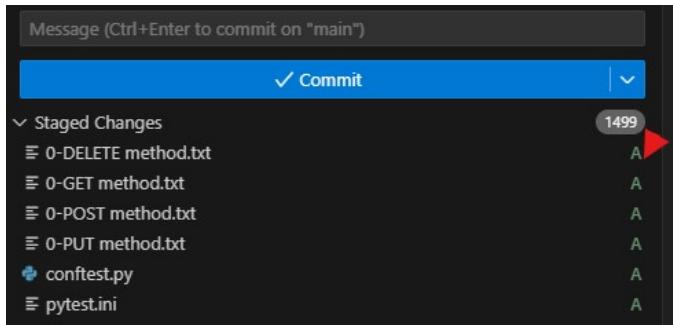


Stage all files

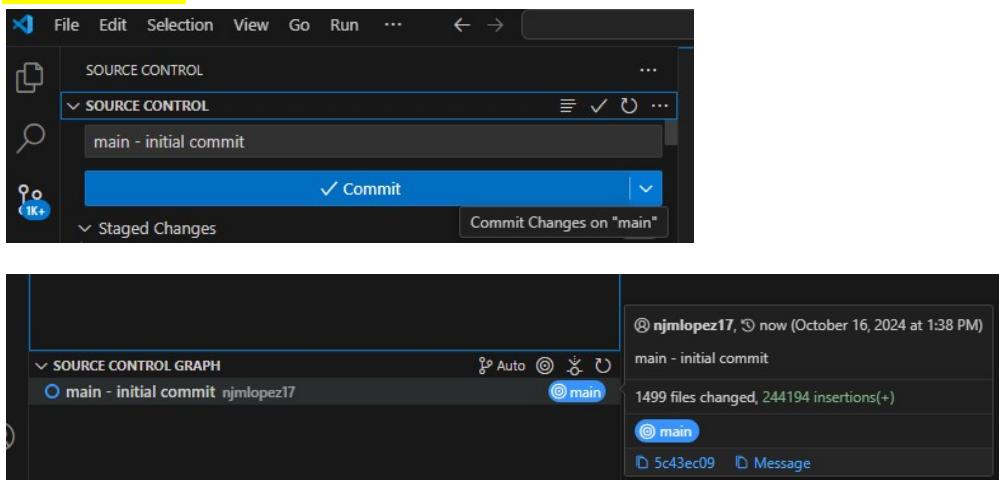
Before



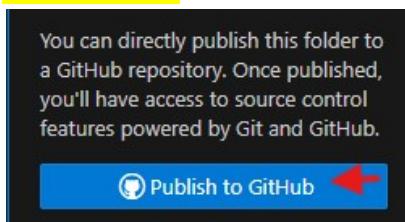
After



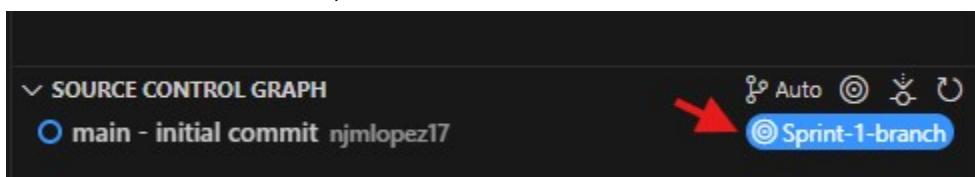
Commit all files



Publish GitHub



Other than the main branch, create a branch



Sampling code change that is in the branch

The screenshot shows a code editor with two tabs: 'test_deleteuser.py M' and 'test_deleteuser.py (Working Tree) M'. The 'test_deleteuser.py (Working Tree)' tab contains Python code for a test function. A red arrow points to the 'Changes' section in the source control panel on the left, which shows a single file named 'test_deleteuser.py' with one change. Another red arrow points to the bottom of the code editor, highlighting the commit message 'Message (Ctrl+Enter to c...)' and the commit button '✓ Commit'.

```
10 def test_delete_user_validation(apis):
11     # output returned response code and flag as passed
12     str_code = str(response.status_code)
13     print("****Code " + str_code +" is passed and data deleted")
14
15     # display the response
16     print("****Record " + str(response.json()) +" is deleted")
17
18- # added this comment
19+
20 def test_delete_user_validation(apis):
21     assert response.status_code == 200
22
23     # output returned response code and flag as passed
24     str_code = str(response.status_code)
25     print("****Code " + str_code +" is passed and data deleted")
26
27     # display the response
28     print("****Record " + str(response.json()) +" is deleted")
29
30 # added this comment
```

Stage and commit the change

The screenshot shows a code editor with two tabs: 'test_deleteuser.py' and 'test_deleteuser.py (Working Tree)'. The 'test_deleteuser.py (Working Tree)' tab displays the same Python code as before. A red box highlights the entire code block, indicating it is staged for commit. The commit message 'Message (Ctrl+Enter to c...)' and the commit button '✓ Commit' are visible in the source control panel on the left.

```
10 def test_delete_user_validation(apis):
11     assert response.status_code == 200
12
13     # output returned response code and flag as passed
14     str_code = str(response.status_code)
15     print("****Code " + str_code +" is passed and data deleted")
16
17     # display the response
18     print("****Record " + str(response.json()) +" is deleted")
19
20 # added this comment
```

Merge the spring branch into that of the main branch

Before merging:

Sprint branch

The screenshot shows a code editor with two tabs: 'test_deleteuser.py' and 'test_deleteuser.py (Working Tree)'. The 'test_deleteuser.py (Working Tree)' tab displays the same Python code as before. A red box highlights the entire code block, indicating it is staged for commit. The commit message 'Message (Ctrl+Enter to c...)' and the commit button '✓ Commit' are visible in the source control panel on the left.

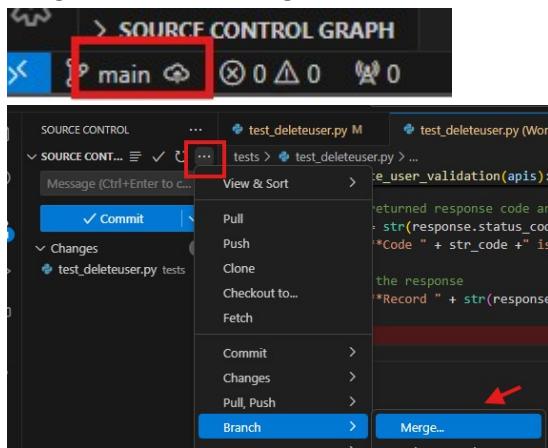
```
10 def test_delete_user_validation(apis):
11     assert response.status_code == 200
12
13     # output returned response code and flag as passed
14     str_code = str(response.status_code)
15     print("****Code " + str_code +" is passed and data deleted")
16
17     # display the response
18     print("****Record " + str(response.json()) +" is deleted")
19
20 # added this comment
```

Main branch

```
tests > test_deleteuser.py > ...
10 def test_delete_user_validation(apis):
11     assert response.status_code == 200
12
13     # output returned response code and flag as passed
14     str_code = str(response.status_code)
15     print("****Code " + str_code +" is passed and data")
16
17     # display the response
18     print("****Record " + str(response.json()) +" is deleted")
19
20
21
22
23
24
25
26
27
28
29
```

```
tests > test_deleteuser.py (Working Tree) > ...
10 def test_delete_user_validation(apis):
11     assert response.status_code == 200
12
13     # output returned response code and flag as passed
14     str_code = str(response.status_code)
15     print("****Code " + str_code +" is passed and data")
16
17     # display the response
18     print("****Record " + str(response.json()) +" is deleted")
19
20
21
22
23
24
25
26
27
28
29
```

Merge the branch changes



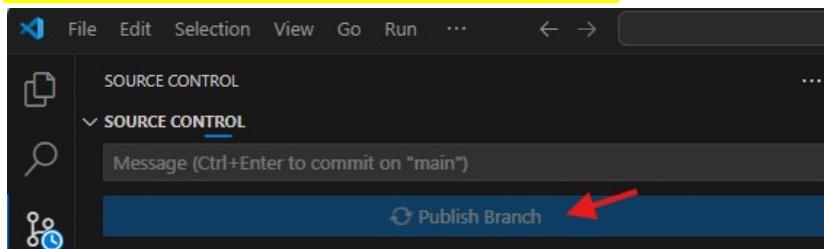
After merging:

Main branch

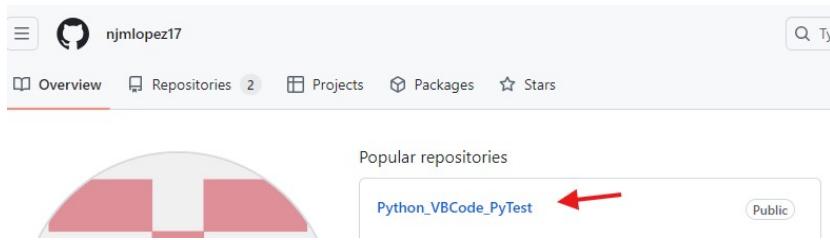
```
tests > test_deleteuser.py > ...
10 def test_delete_user_validation(apis):
11
12     # validate the response code if passed
13     assert response.status_code == 200
14
15     # output returned response code and flag as passed
16     str_code = str(response.status_code)
17     print("****Code " + str_code +" is passed and data")
18
19     # display the response
20     print("****Record " + str(response.json()) +" is deleted")
21
22
23
24
25
26
27     # added this comment
28
29
```

```
tests > test_deleteuser.py (Working Tree) > ...
10 def test_delete_user_validation(apis):
11
12     # validate the response code if passed
13     assert response.status_code == 200
14
15     # output returned response code and flag as passed
16     str_code = str(response.status_code)
17     print("****Code " + str_code +" is passed and data")
18
19     # display the response
20     print("****Record " + str(response.json()) +" is deleted")
21
22
23
24
25
26
27     # added this comment
28
29
```

Publishing the branch (for this sampling, it is public)



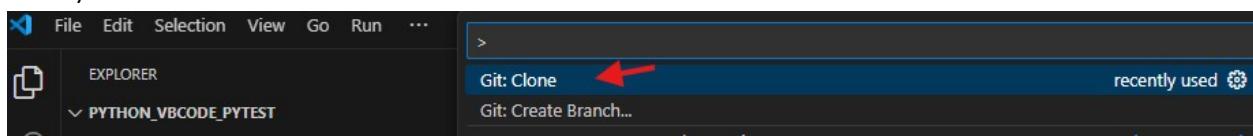
Added in GitHub



Public URL:

https://github.com/njmlopez17/Python_VBCode_PyTest.git

Note: to run this whole sampling of codes, simply clone in Visual Studio Code editor (using the URL above)

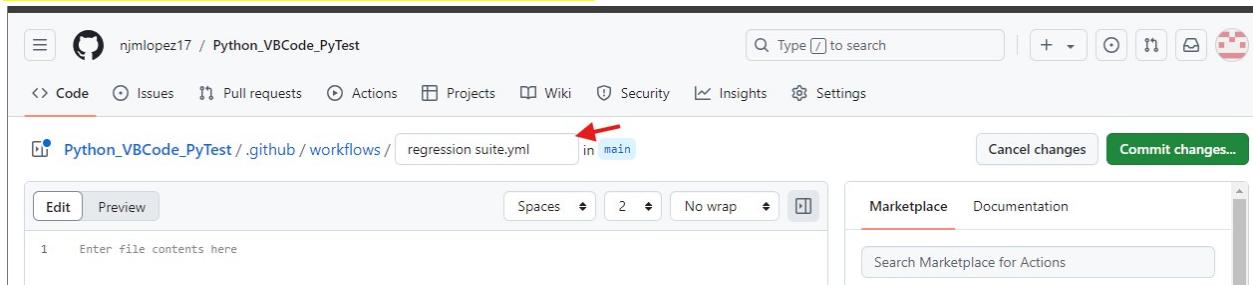


64. CI using GitHub Actions

Public URL:

https://github.com/njmlopez17/Python_VBCode_PyTest.git

Create a new file in GitHub (and commit changes)



Code context of the file

The screenshot shows a GitHub repository interface for the project "Python_VBCode_PyTest". The repository path is "njmlopez17 / Python_VBCode_PyTest". The main navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings.

The left sidebar displays the repository's file structure:

- Files: main (selected), +, Go to file, ...
- .github/workflows: regression suite.yml (highlighted with a red arrow)
- __pycache__
- data
- reports
- tests
- utils
- venv
- 0-DELETE method.txt
- 0-GET method.txt
- 0-POST method.txt
- 0-PUT method.txt
- conftest.py
- pytest.ini
- requirements.txt

The right panel shows the content of the "regression suite.yml" file, which defines a GitHub Action workflow for running tests and generating reports. The code is highlighted with a red box.

```
name: Regression Suite
on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main
schedule:
  - cron: '30 2 * * *'
jobs:
  run-tests:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3
      - name: Set up Python
        uses: actions/setup-python@v3
        with:
          python-version: '3.12.6'
      - name: Install dependencies
        run:
          python -m pip install --upgrade pip
          pip install -r requirements.txt
      - name: Run tests
        run:
          pytest
      - name: Upload report
        uses: actions/upload-artifact@v3
        with:
          name: pytest-report
          path: report.html
```

After a successful run (via Actions button)

The screenshot shows the GitHub Actions interface for the repository "njmlopez17 / Python_VBCode_PyTest". The "Actions" tab is selected, indicated by a red box. A specific workflow run titled "Update regression suite.yml" is highlighted with a green checkmark and a red arrow pointing to it. The run was triggered by a commit pushed by njmlopez17. The workflow details show a series of steps: "run-tests", "Upload report", "Post Set up Python", "Post Checkout code", and "Complete job". The "Run tests" step is expanded, showing the command "pytest" and its output, which includes several test cases passing with 100% coverage. A red box highlights the "5 passed in 0.98s" message at the bottom of the log.

27 workflow runs

Event ▾ Status ▾ Branch ▾ Actor ▾

Update regression suite.yml

Regression Suite #27: Commit [3054dcd](#) pushed by njmlopez17

main

7 minutes ago

32s

Summary

Jobs

run-tests

Run details

Usage

Workflow file

run-tests

succeeded 7 minutes ago in 20s

Search logs

Run tests

1 ► Run pytest

7 ===== test session starts =====

8 platform linux -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0 -- /opt/hostedtoolcache/Python/3.12.6/x64/bin/python

9 cachedir: .pytest_cache

10 metadata: {'Python': '3.12.6', 'Platform': 'Linux-6.5.0-1025-azure-x86_64-with-glibc2.35', 'Packages': {'pytest': '8.3.3', 'pluggy': '1.5.0'}, 'Plugins': {'metadata': '3.1.1', 'html': '4.1.1'}, 'CI': 'true', 'JAVA_HOME': '/usr/lib/jvm/temurin-11-jdk-amd64'}

11 rootdir: /home/runner/work/Python_VBCode_PyTest/Python_VBCode_PyTest

12 configfile: pytest.ini

13 plugins: metadata-3.1.1, html-4.1.1

14 collecting ... collected 5 items

15

16 tests/test_deleteuser.py::test_delete_user_validation PASSED [20%]

17 tests/test_getuser.py::test_get_user_validation PASSED [40%]

18 tests/test_postuser.py::test_post_user_validation PASSED [60%]

19 tests/test_postuser_dynd1.py::test_post_user_validation PASSED [80%]

20 tests/test_putuser.py::test_update_user_validation PASSED [100%]

21

22 - Generated html report: file:///home/runner/work/Python_VBCode_PyTest/Python_VBCode_PyTest/reports/reports.html -

23 ====== 5 passed in 0.98s ======

1 Cleaning up orphan processes

Upload report

Post Set up Python

Post Checkout code

Complete job

Cleaning up orphan processes

*****END*****