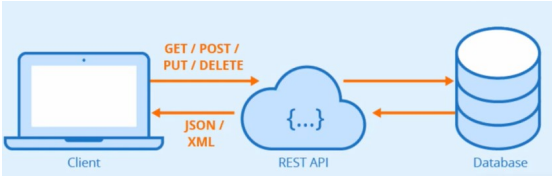
Test Automation using:

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

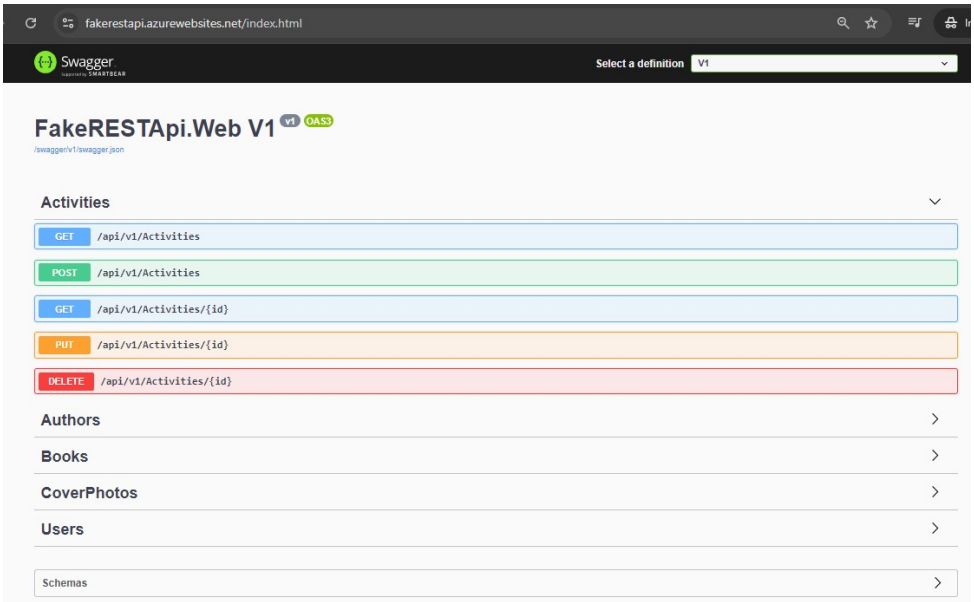
Document highlight:

- 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

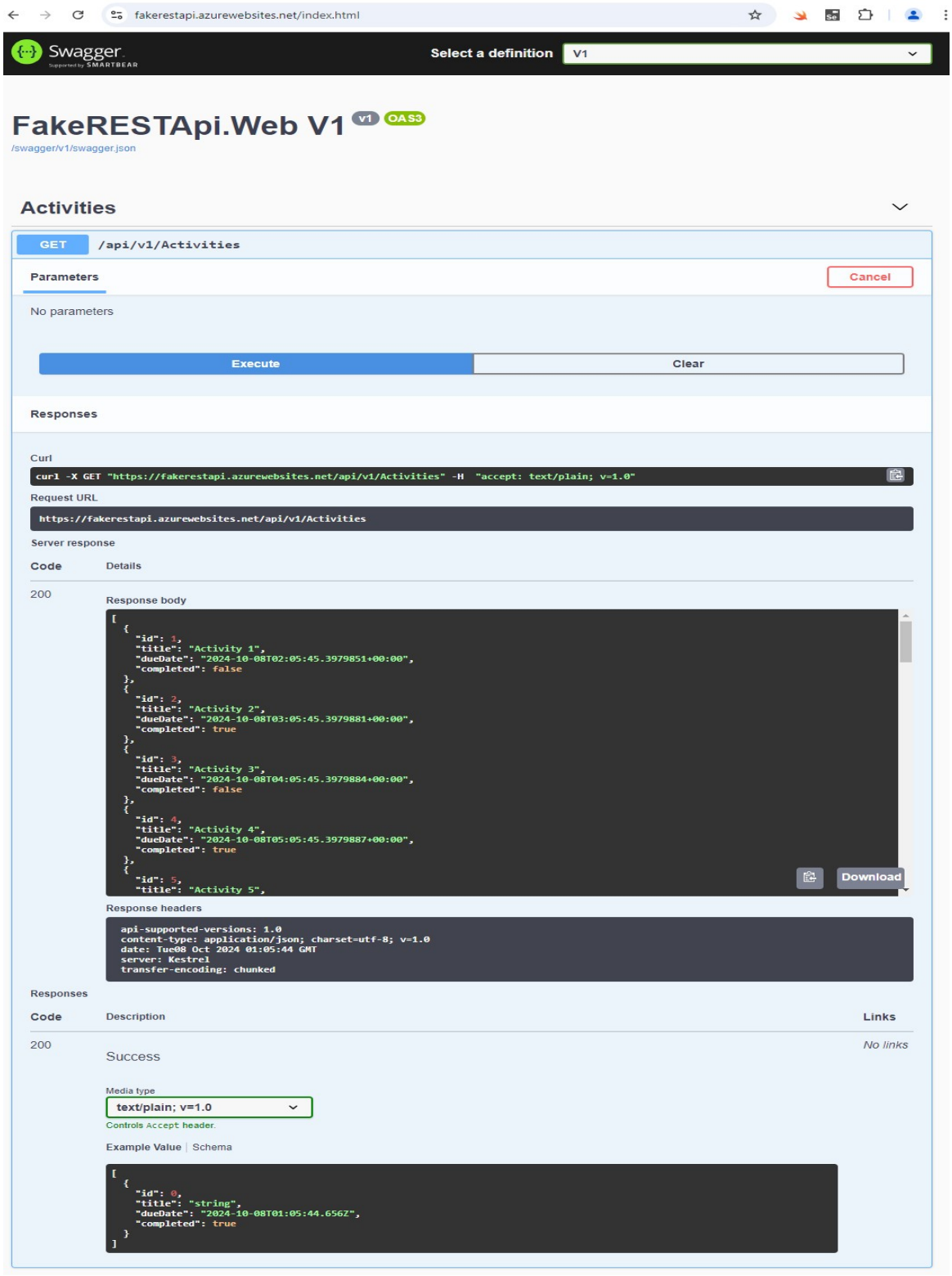
1. API Concept (interface between client and server (database), request and response)



2. Sampling APIs (using Swagger)



3. Executing a GET method

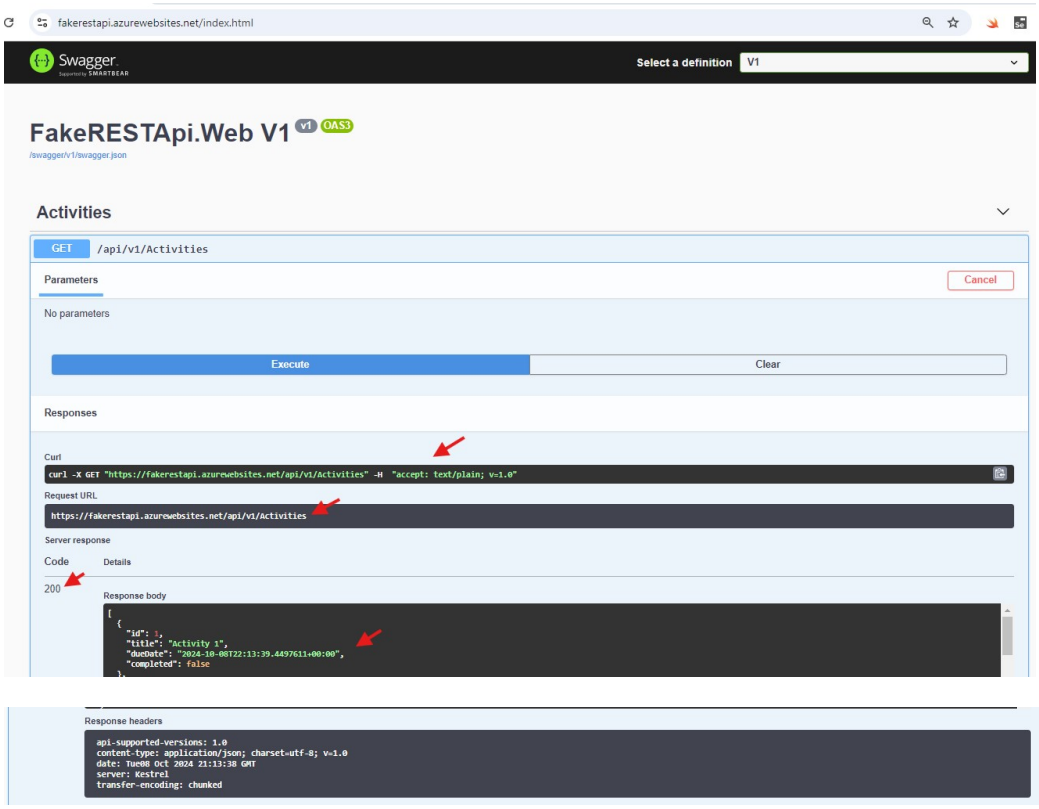


4. Sample GET without parameter (copy the sampling url from Swagger and into Postman)

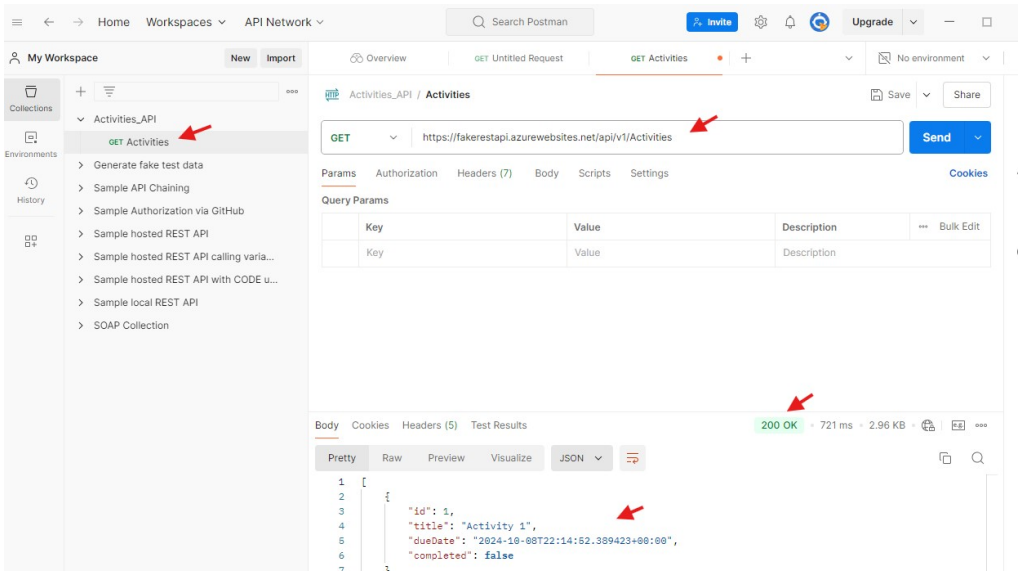
Validate and compare the following response results:

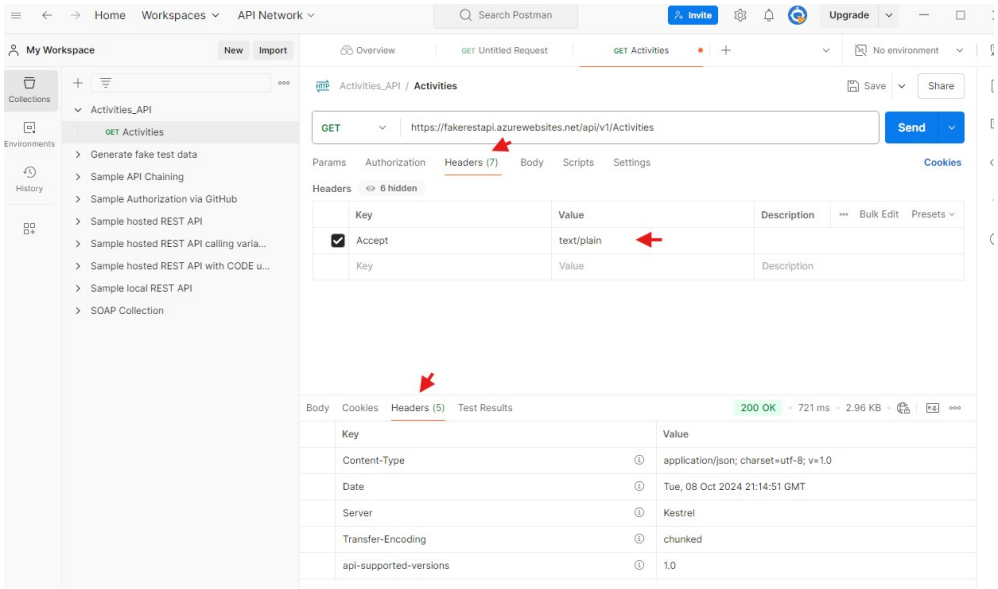
- **Status**
- **Body**
- **Header**
- **Time**

Swagger



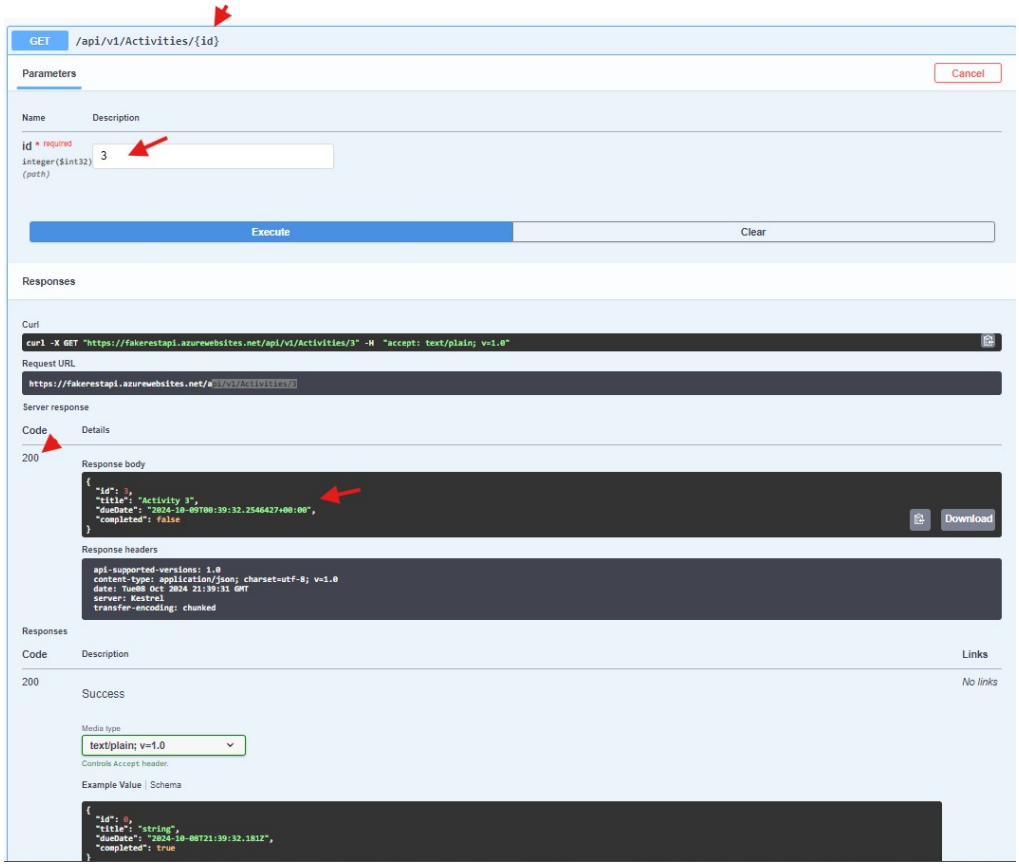
Postman


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
GET Untitled Request
GET Activities
No environment
Collections
Environments
History
Activities_API
GET Activities
Generate fake test data
Sample API Chaining
Sample Authorization via GitHub
Sample hosted REST API
Sample hosted REST API calling varia...
Sample hosted REST API with CODE u...
Sample local REST API
SOAP Collection
Activities_API / Activities
Save
Share
GET
https://fakerestapi.azurewebsites.net/api/v1/Activities
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (5)
Test Results
200 OK
721 ms
2.96 KB
Pretty
Raw
Preview
Visualize
JSON
1 [
2 {
3 "id": 1,
4 "title": "Activity 1",
5 "dueDate": "2024-10-08T22:14:52.389423+00:00",
6 "completed": false

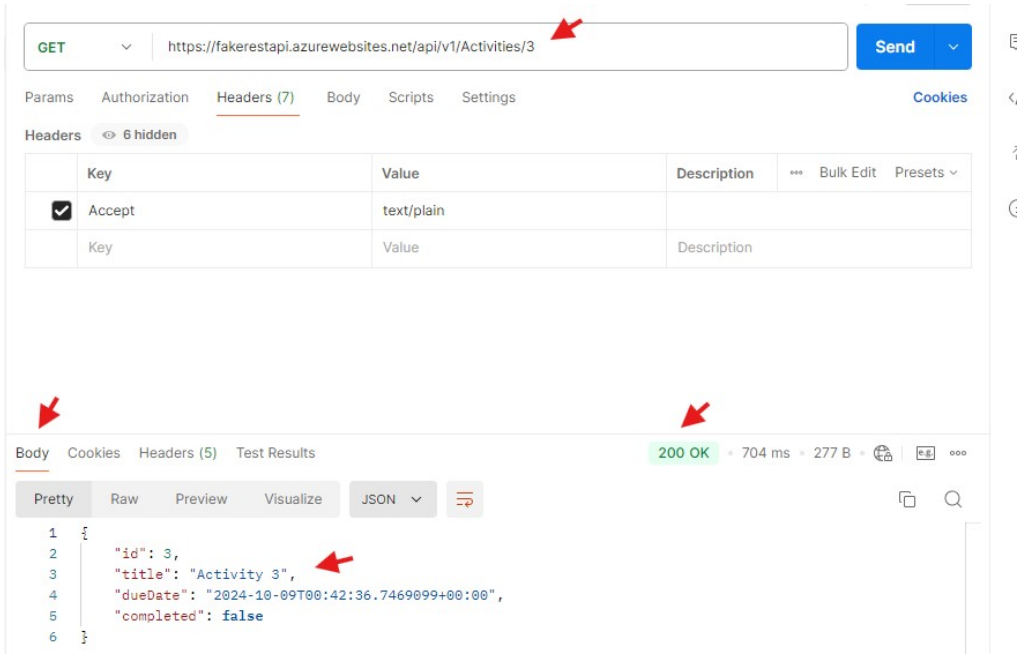

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
GET Untitled Request
GET Activities
No environment
Collections
Environments
History
Activities_API
GET Activities
Generate fake test data
Sample API Chaining
Sample Authorization via GitHub
Sample hosted REST API
Sample hosted REST API calling varia...
Sample hosted REST API with CODE u...
Sample local REST API
SOAP Collection
Activities_API / Activities
Save
Share
GET
https://fakerestapi.azurewebsites.net/api/v1/Activities
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Headers
6 hidden
Key
Value
Description
Bulk Edit
Presets
Accept
text/plain
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
200 OK
721 ms
2.96 KB
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

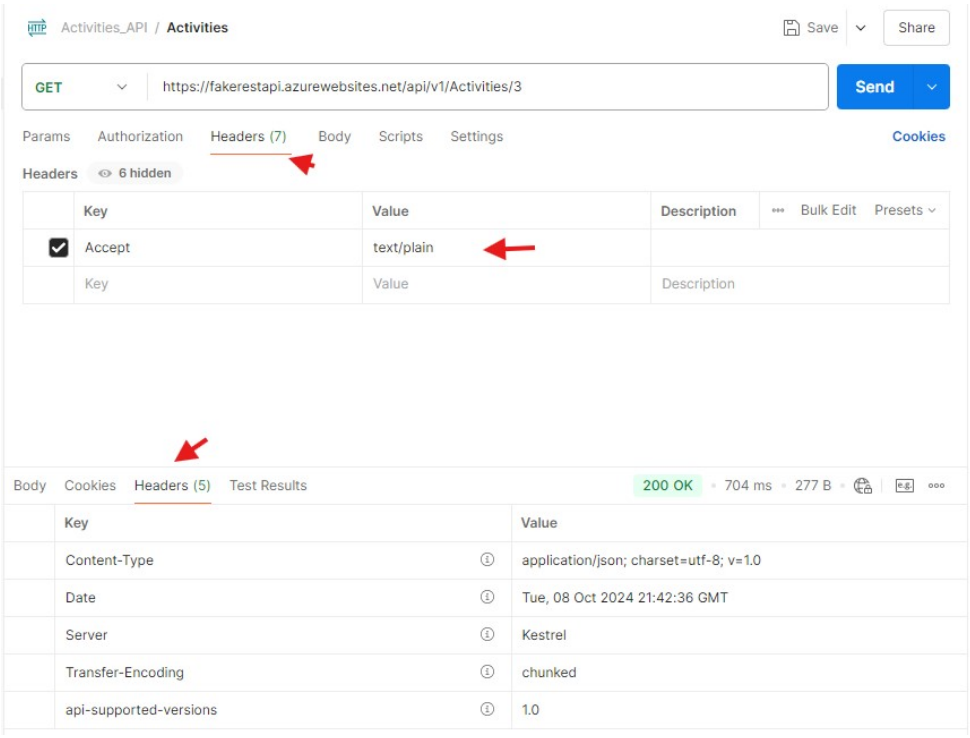
5. Sample GET with parameter (copy the sampling url from Swagger and into Postman)

Swagger



Postman

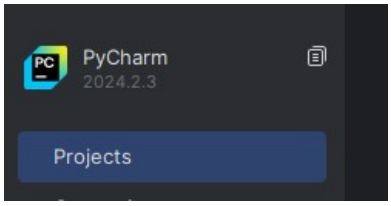




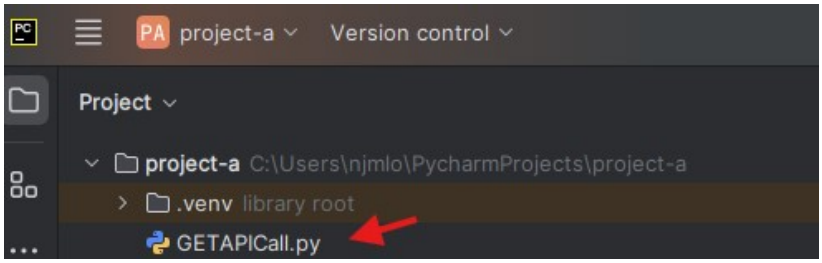
6. Determine Python version, install "requests" library

```
(.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
```

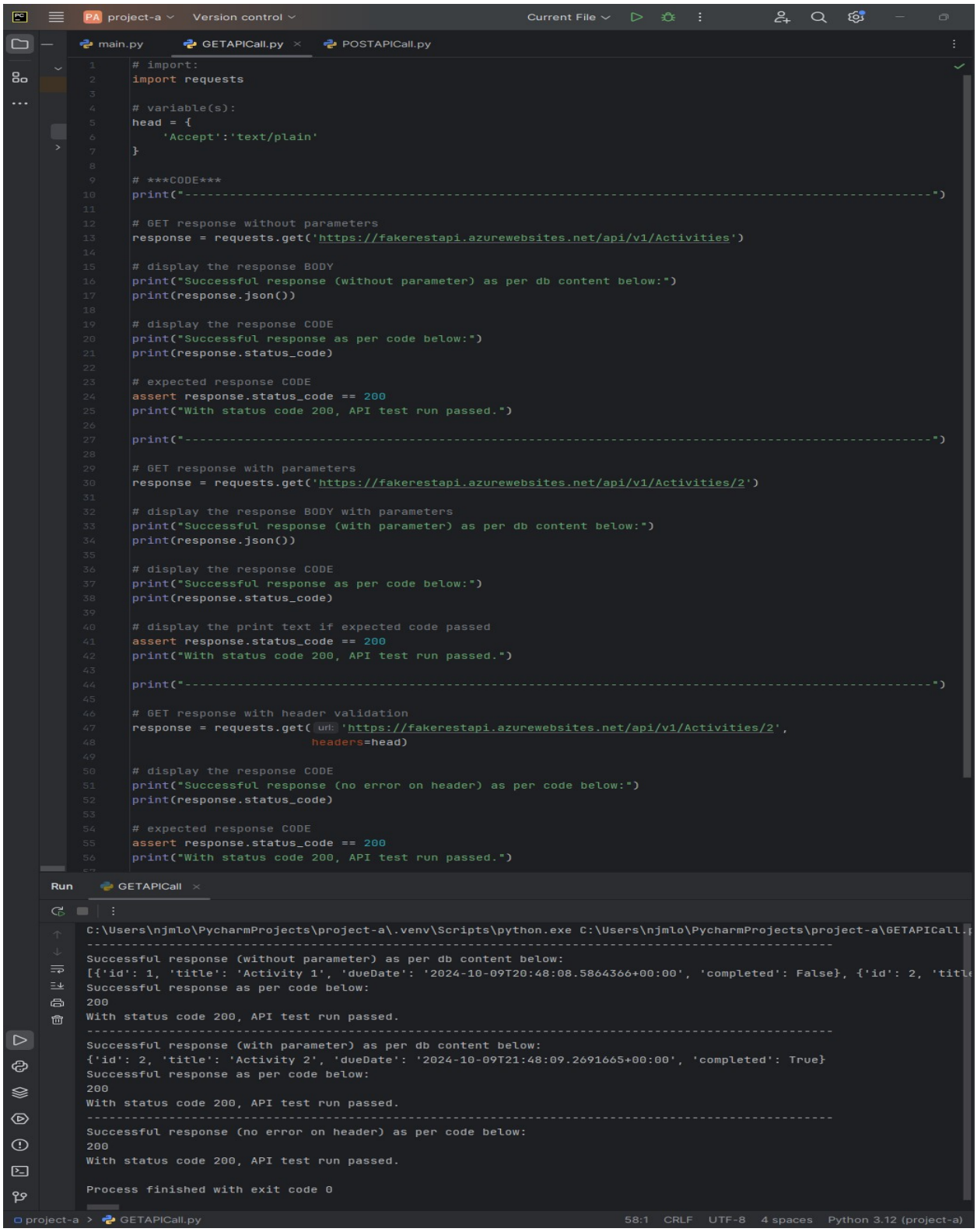
7. PyCharm Community Edition (IDE version at the time of documentation)



8. Create a python file in PyCharm editor

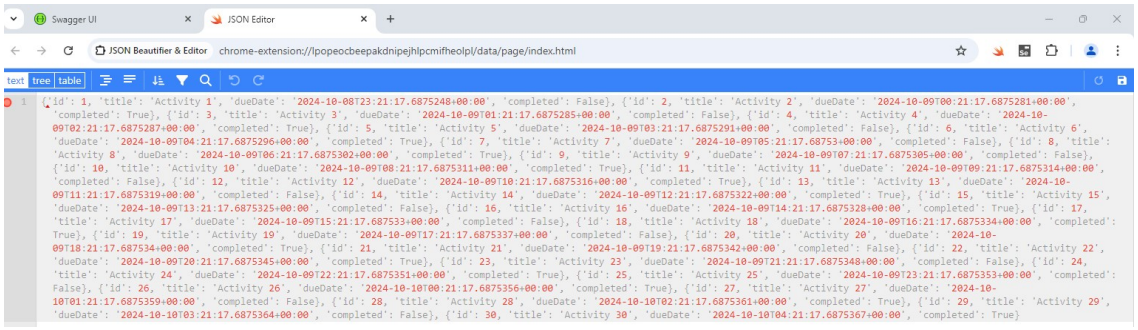


9. Sample GET (in PyCharm using Python code, a sampling of with and without parameters, and header validation)



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

Text view



Tree view

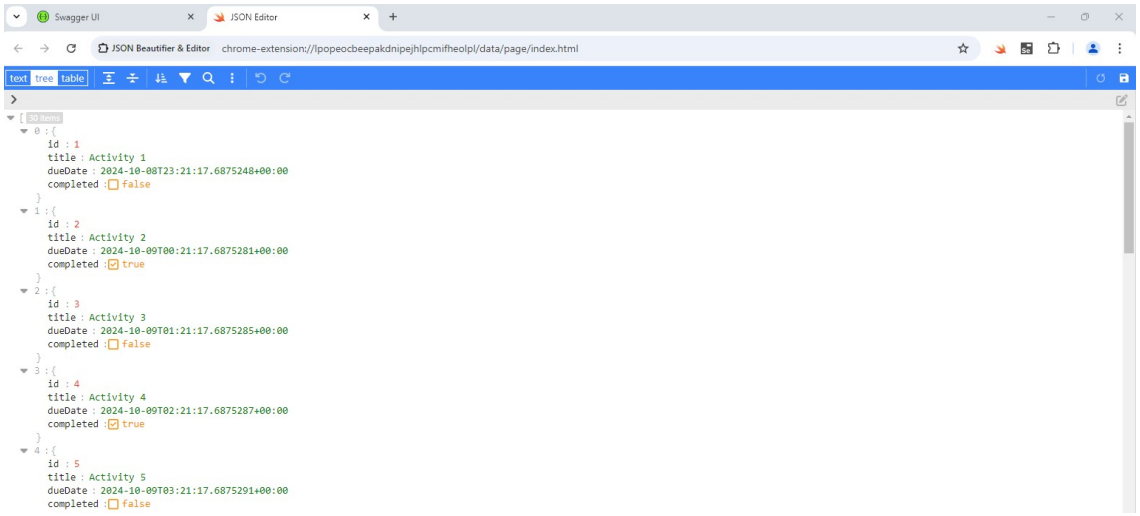
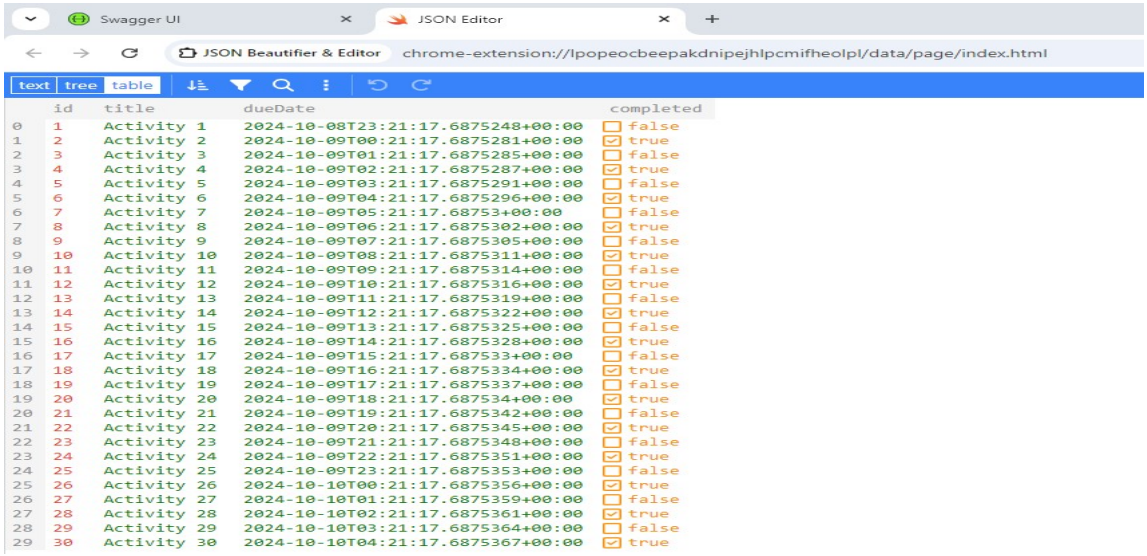


Table view



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

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

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

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

```

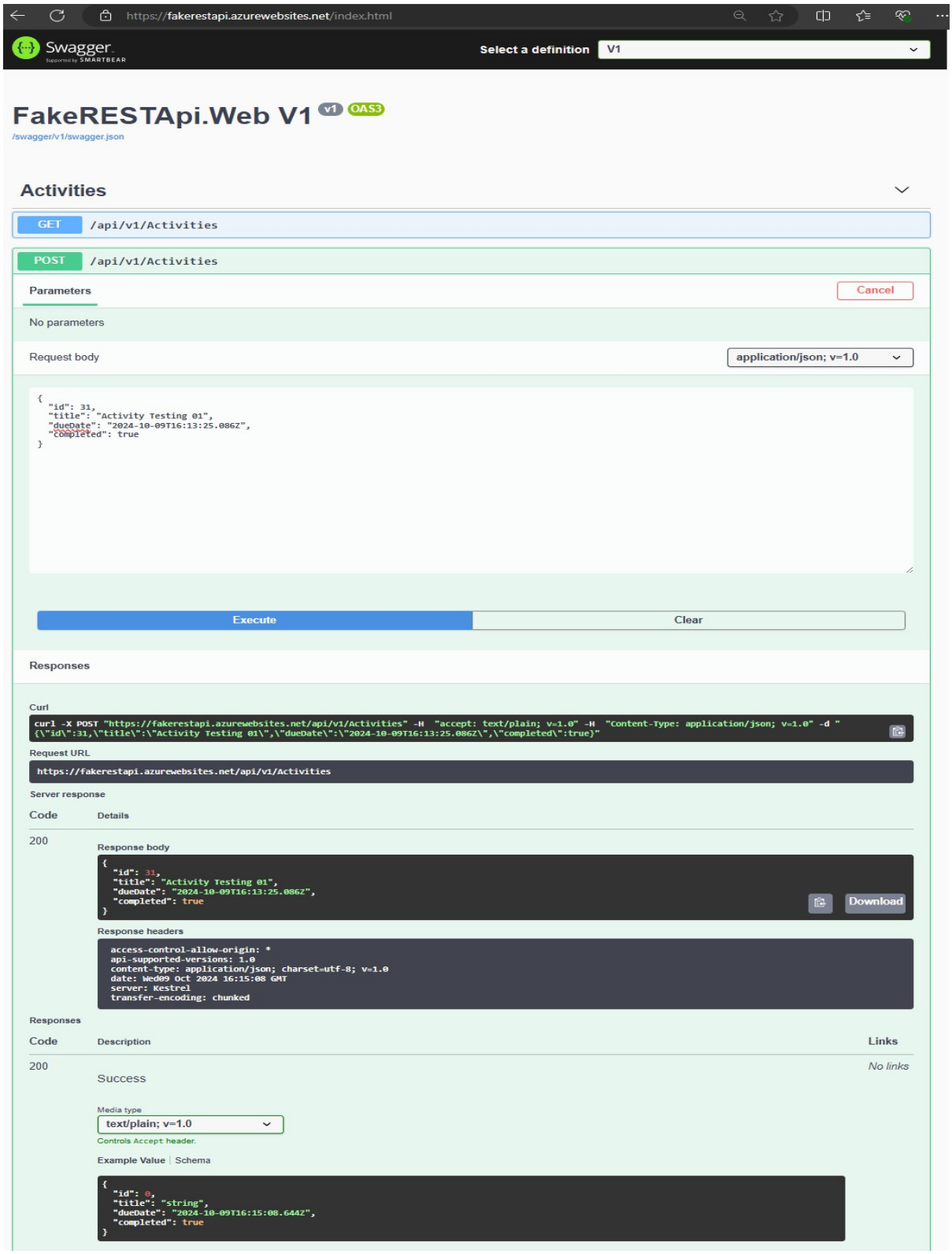
```
-----------------------------------------------------------------------------------------------
Traceback (most recent call last):  Explain with AI
  File "C:\Users\njmlo\PycharmProjects\project-a\GETAPICall.py", line 60, in <module>
    assert response.status_code == 201
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^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



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

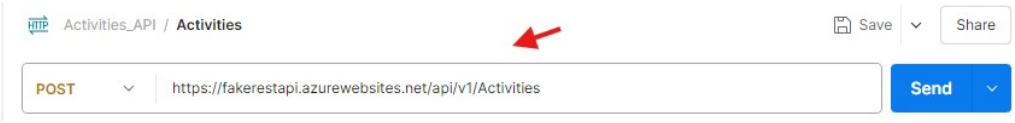
Validate and compare the following response results:

- **Status**
- **Body**
- **Header**
- **Time**

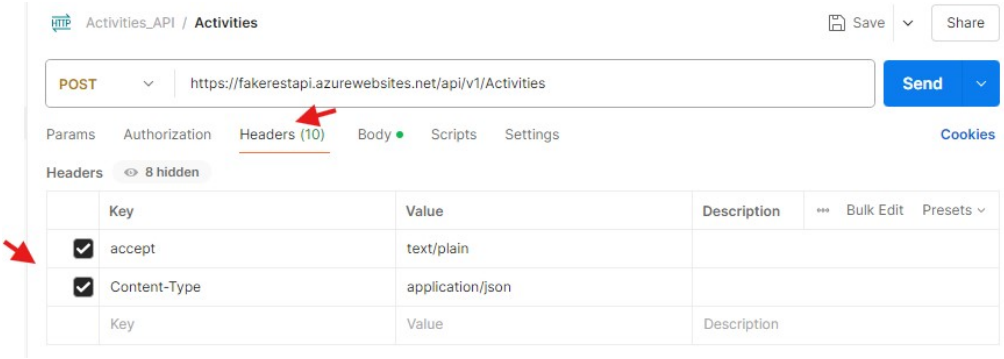
Swagger

Postman

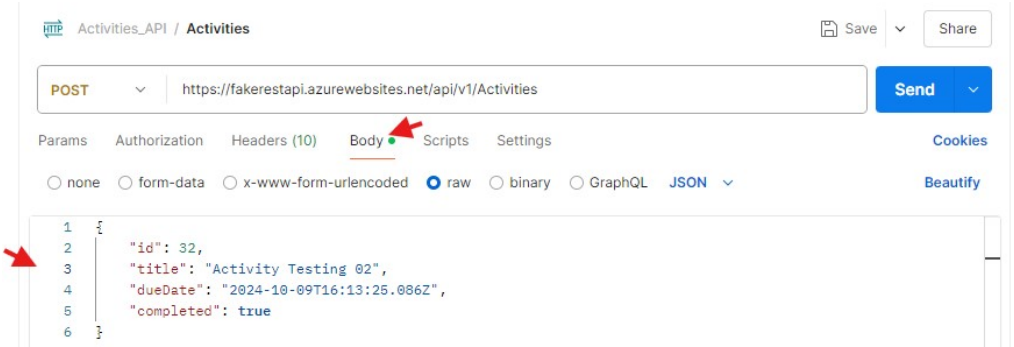
URL



Header

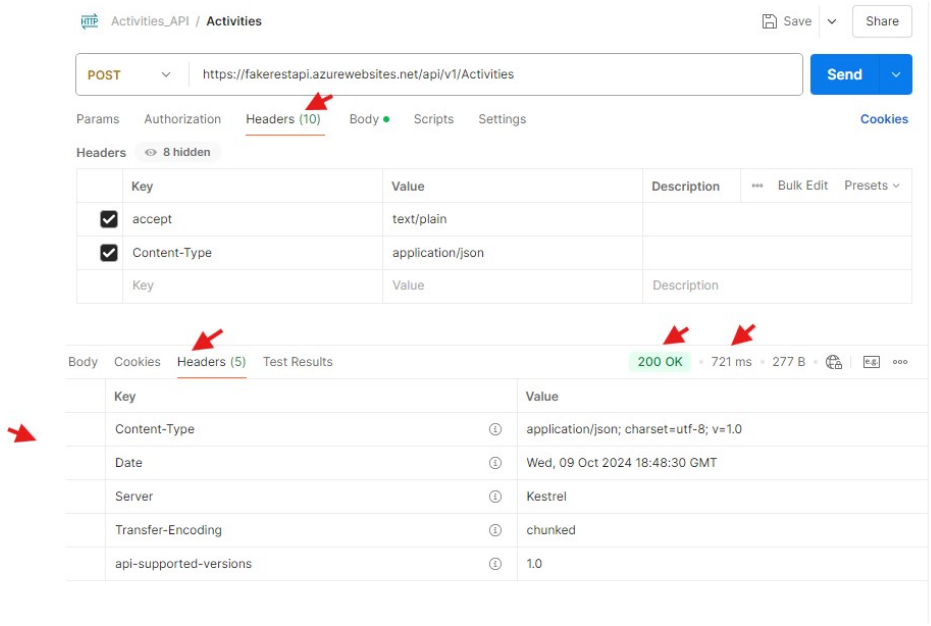


Body

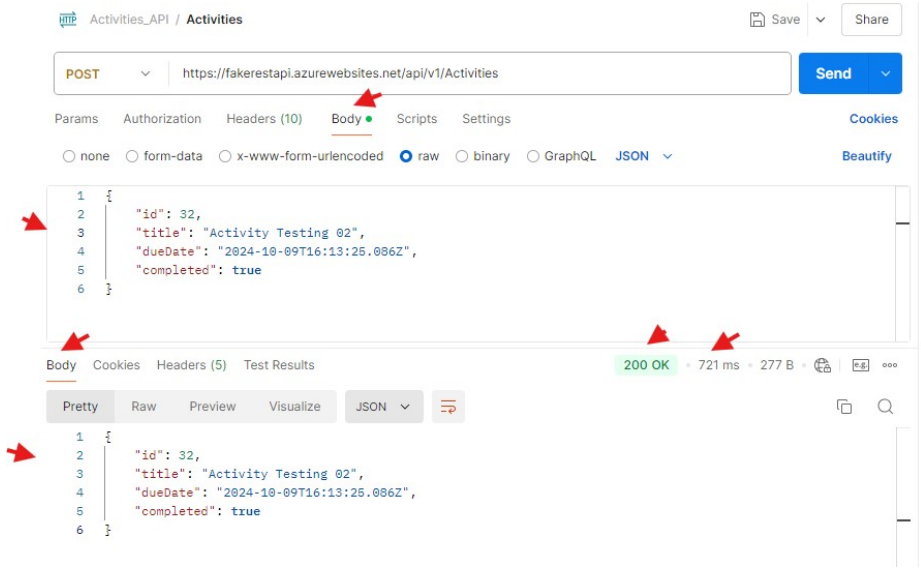


After Send…

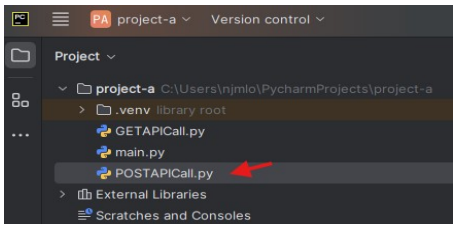
Header



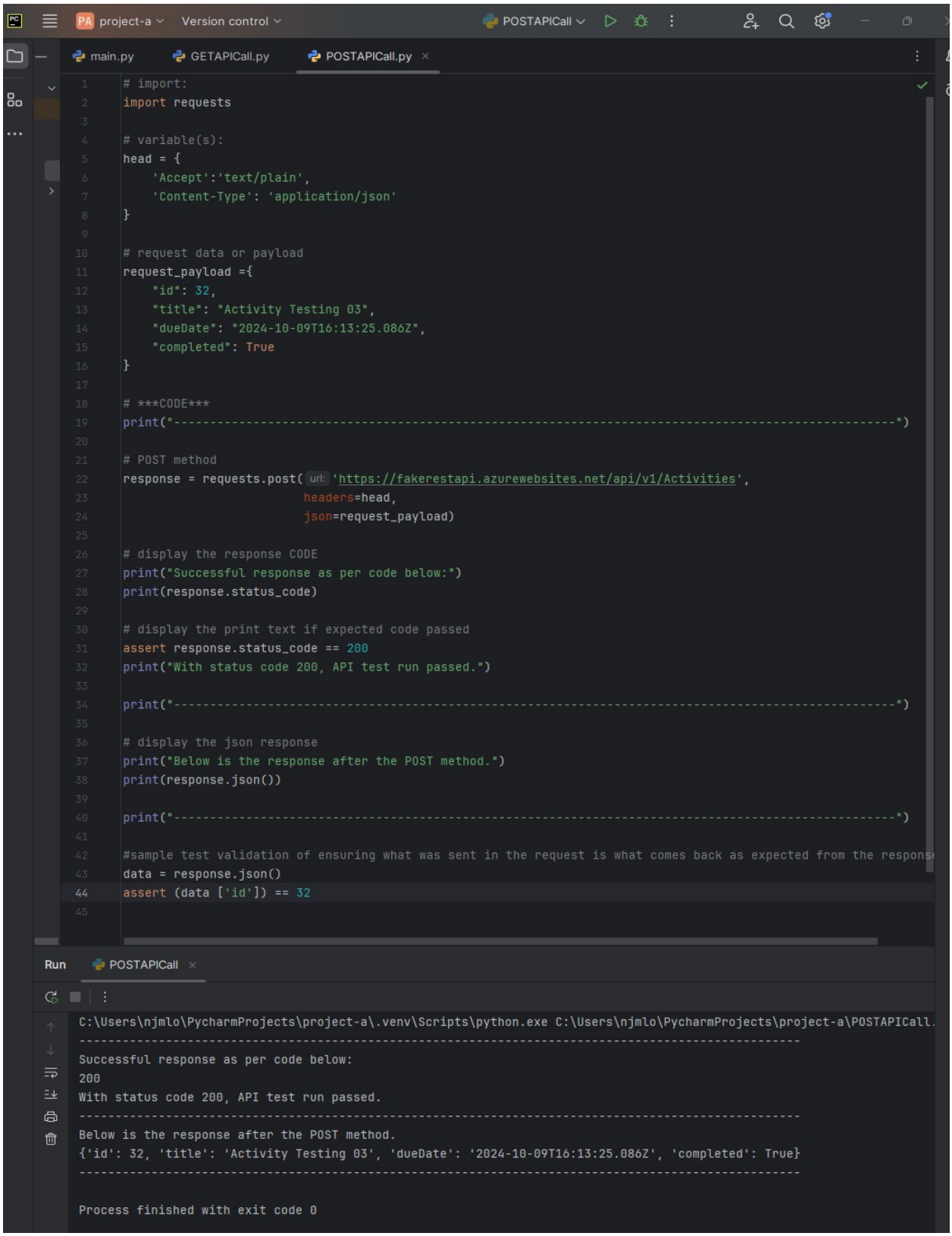
Body



15. Create a python file in PyCharm editor



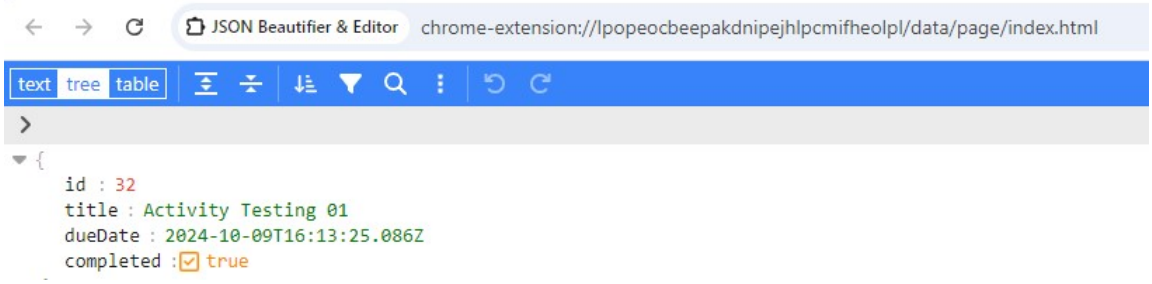
16. Sample POST (in PyCharm using Python code)



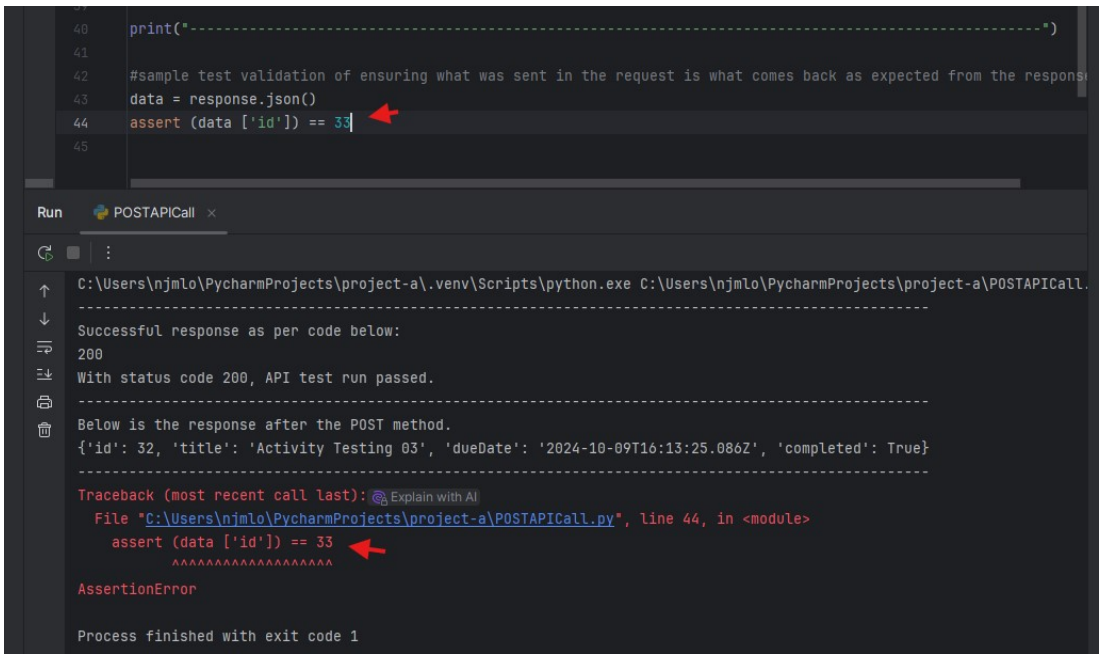
17. Using Json Beautifier to display response result nicely (copy and paste the result)

Text view

Tree view



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

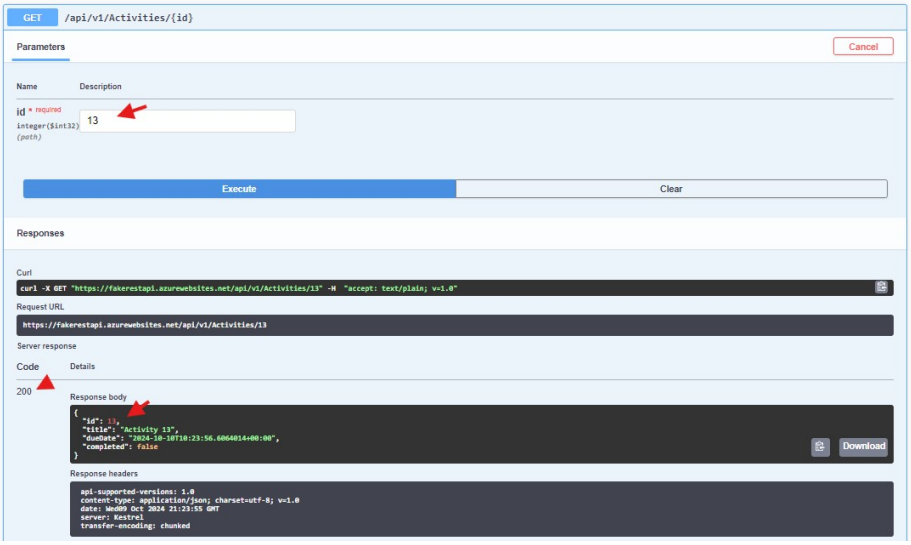


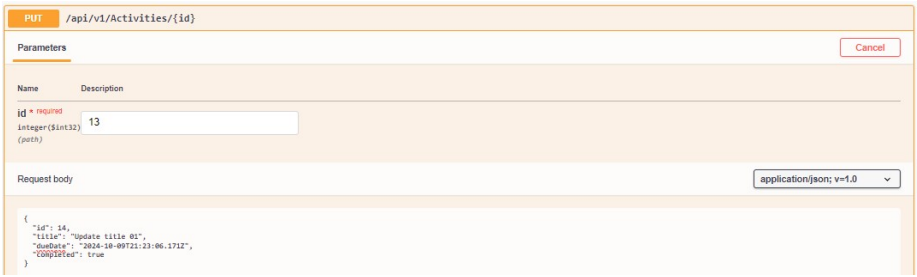
19. Here is the code in text

```
# 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
```

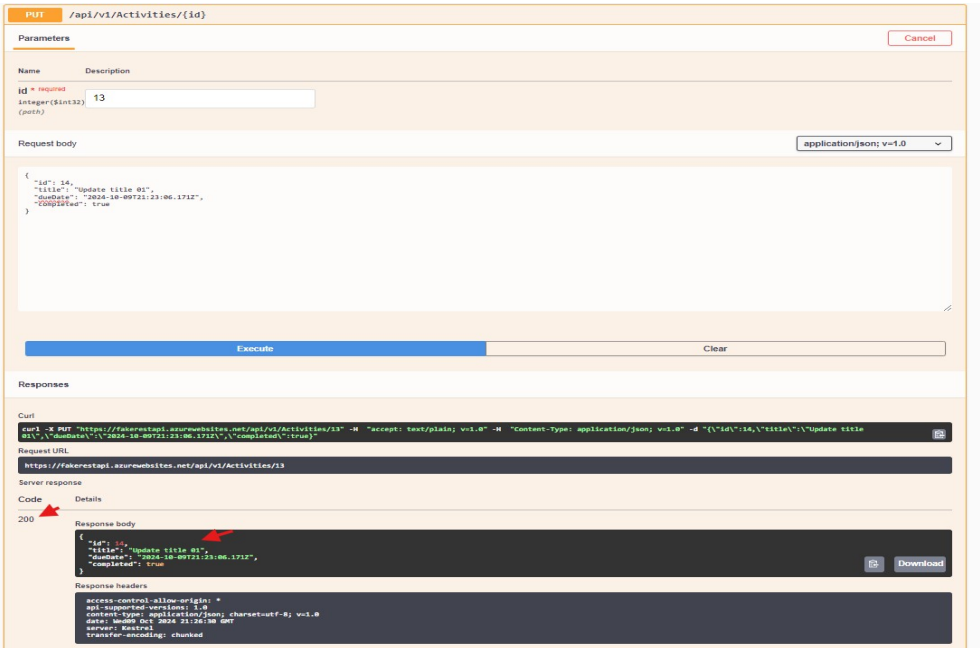
20. Executing a PUT method

In SwaggerUI, before executing the method





In SwaggerUI, after executing the method

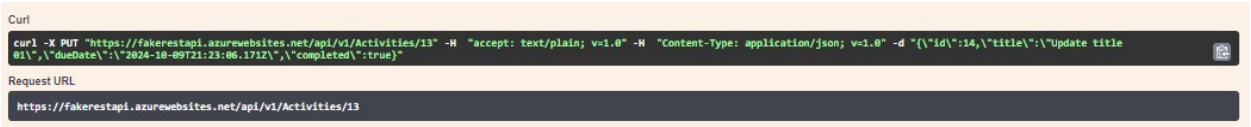


21. Sample PUT (copy the sampling url from Swagger and into Postman)

Validate and compare the following response results:

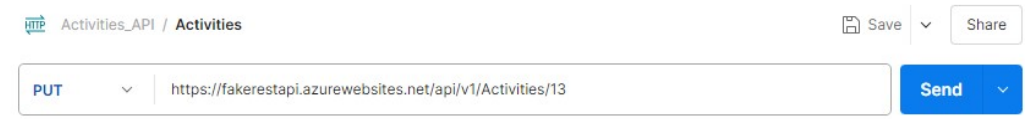
- **Status**
- **Body**
- **Header**
- **Time**

Swagger

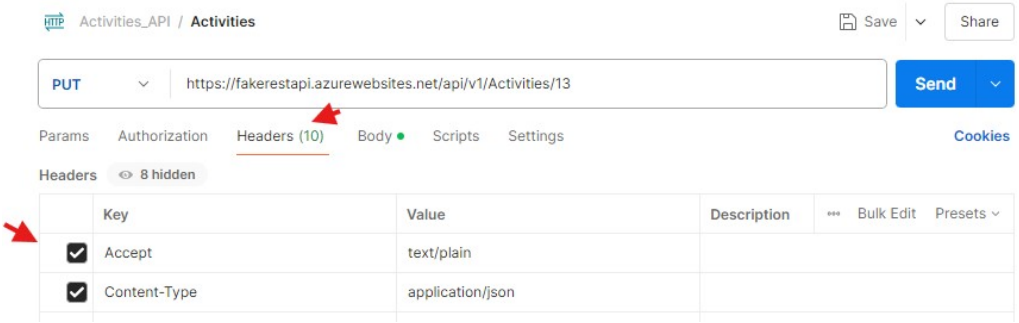


Postman

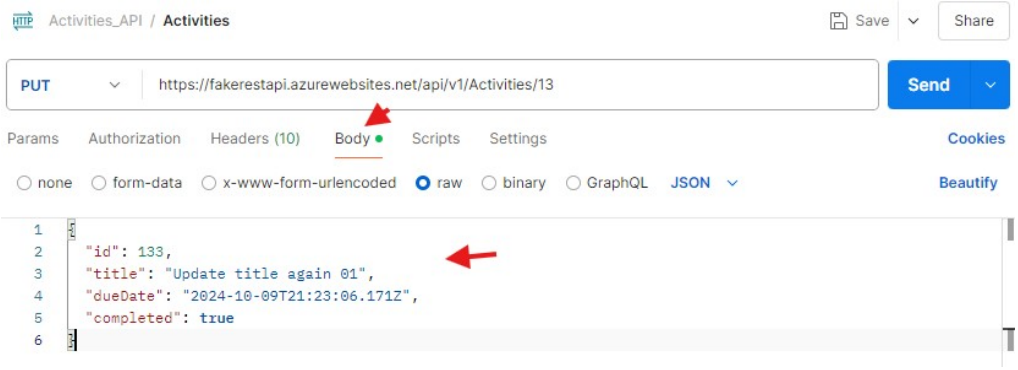
URL



Header

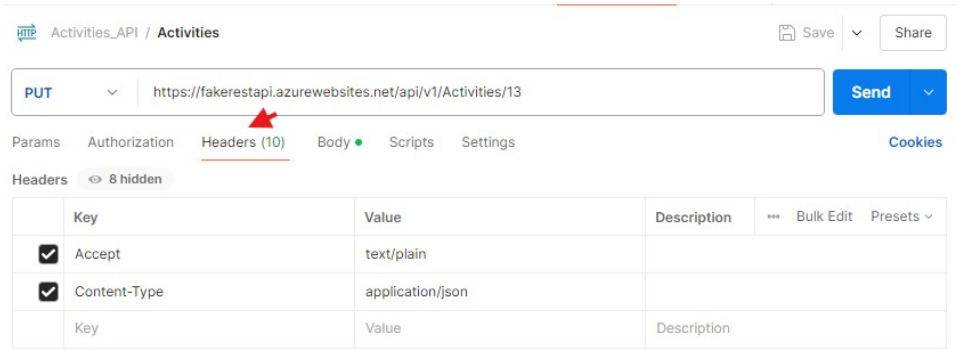


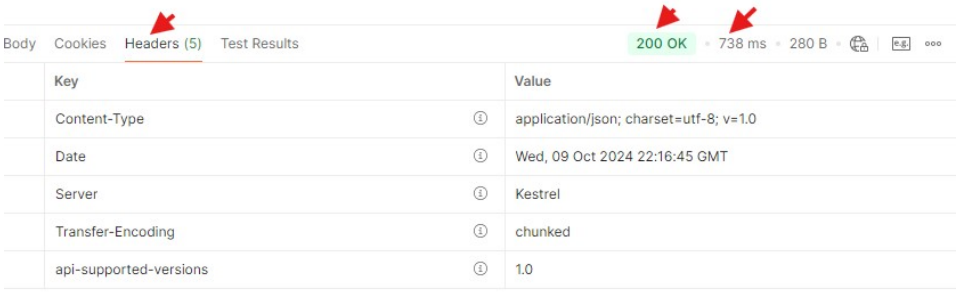
Body



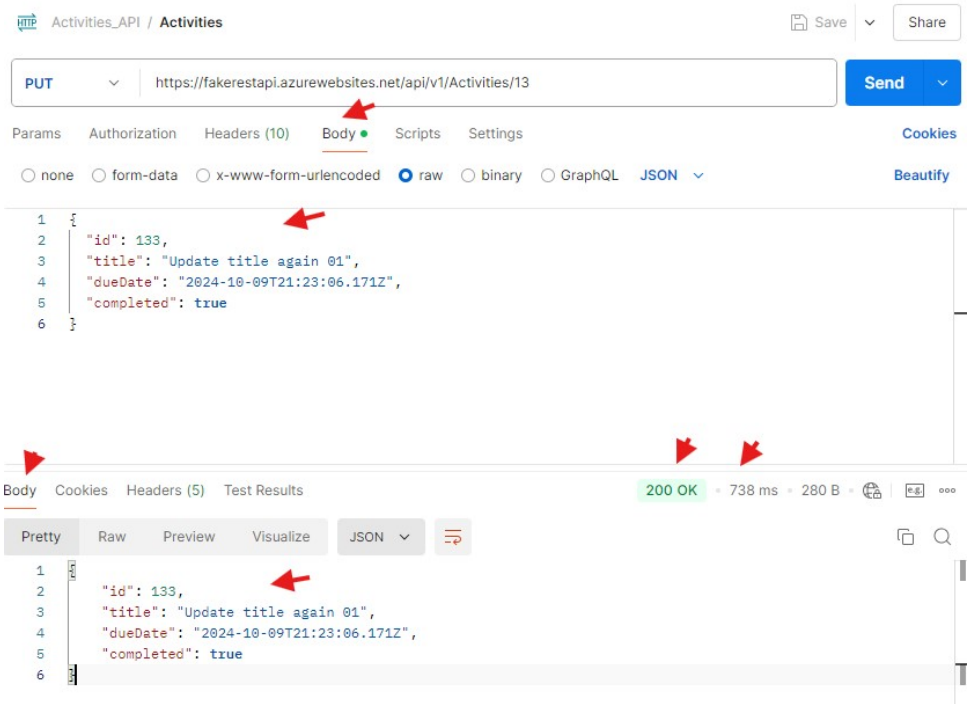
After Send…

Header

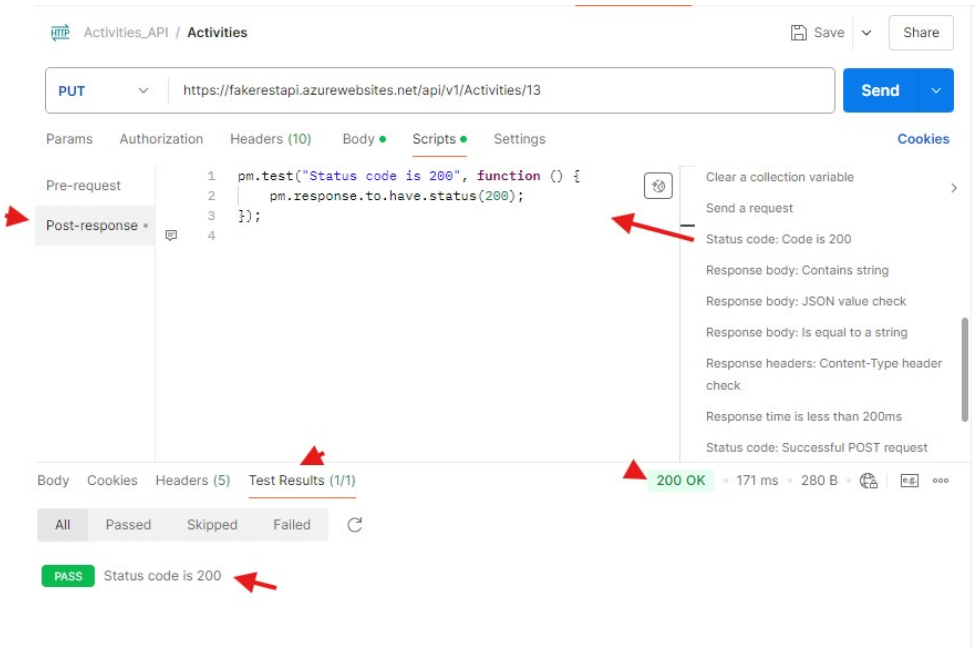




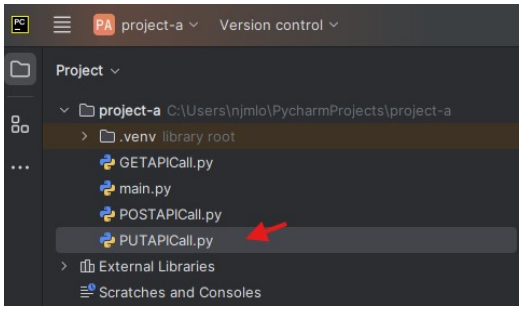
Body



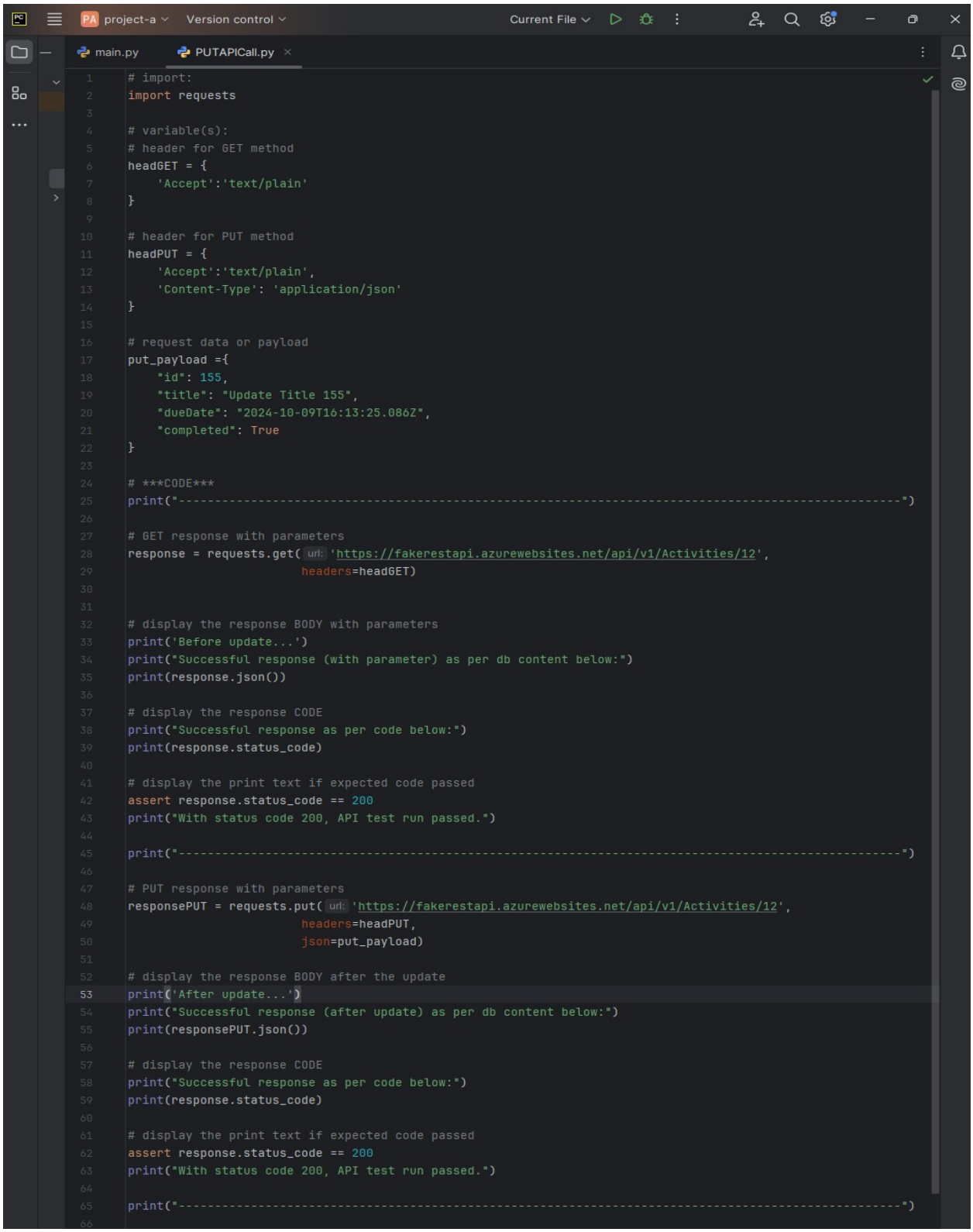
22. Sample Test code (using Snippets in Postman) and validating its Test Results (Post-response)



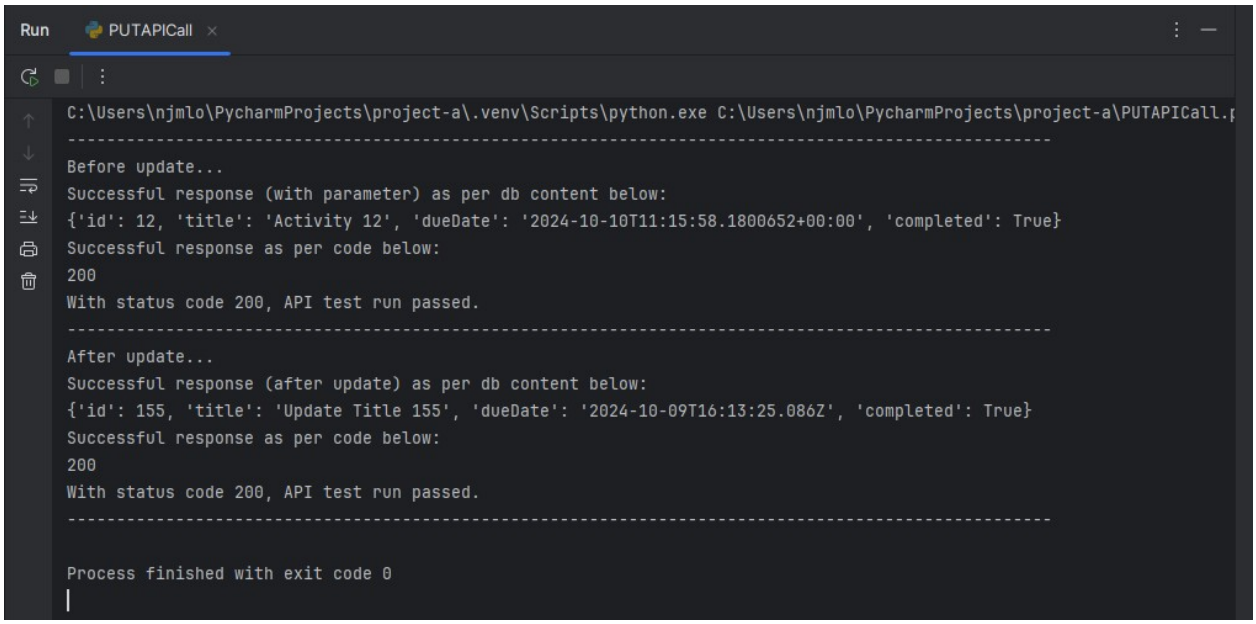
23. Create a python file in PyCharm editor



24. Sample PUT (in PyCharm using Python code)




```python
# 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("-----------------------------------------------------------------------------------------------------")

# GET response with parameters
response = requests.get( url: '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( url: '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("-----------------------------------------------------------------------------------------------------")
```



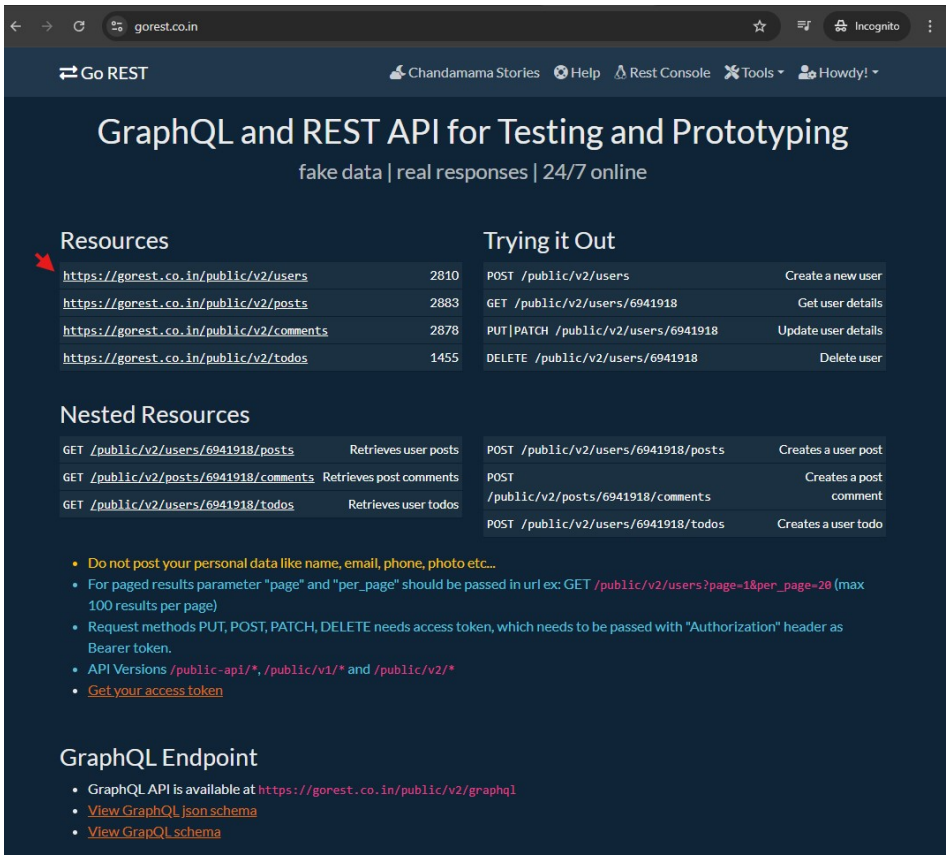
25. Here is the code in text

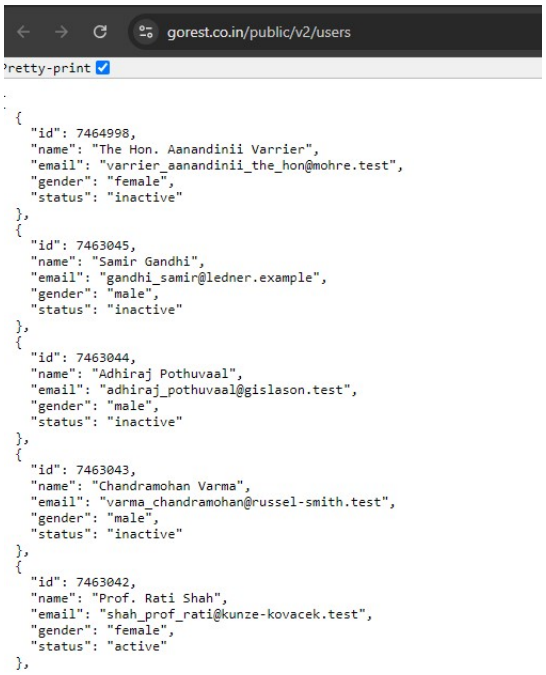
```
# import:
import requests

# 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("--------------------------------------------------------------------------
-----------------------------")

# 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



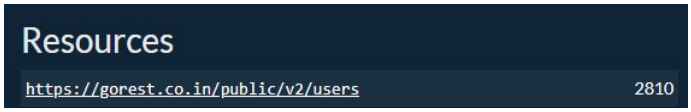


27. Sample POST with OAuth2 bearer key (copy the sampling url from Go REST and into Postman)

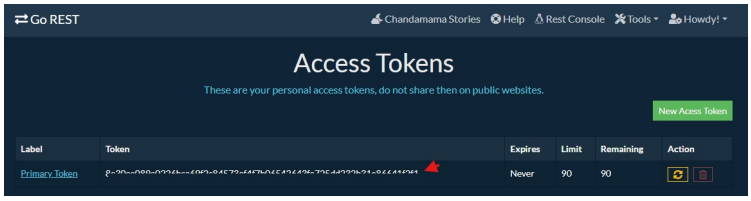
Validate and compare the following response results:

- **Status**
- **Body**
- **Header**
- **Time**

Go REST



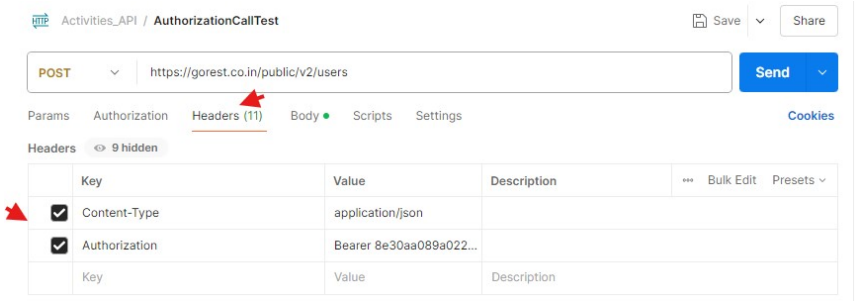
Sample Github token



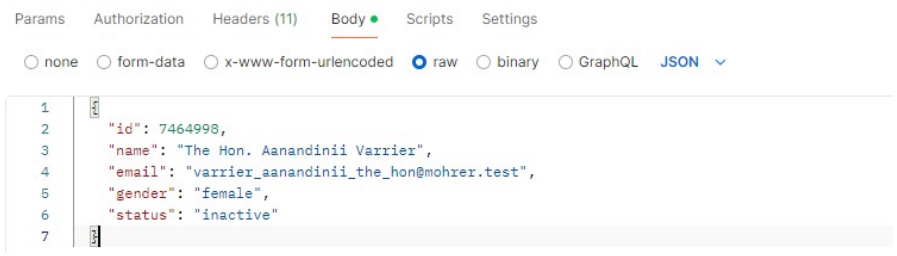
Postman

URL

Header

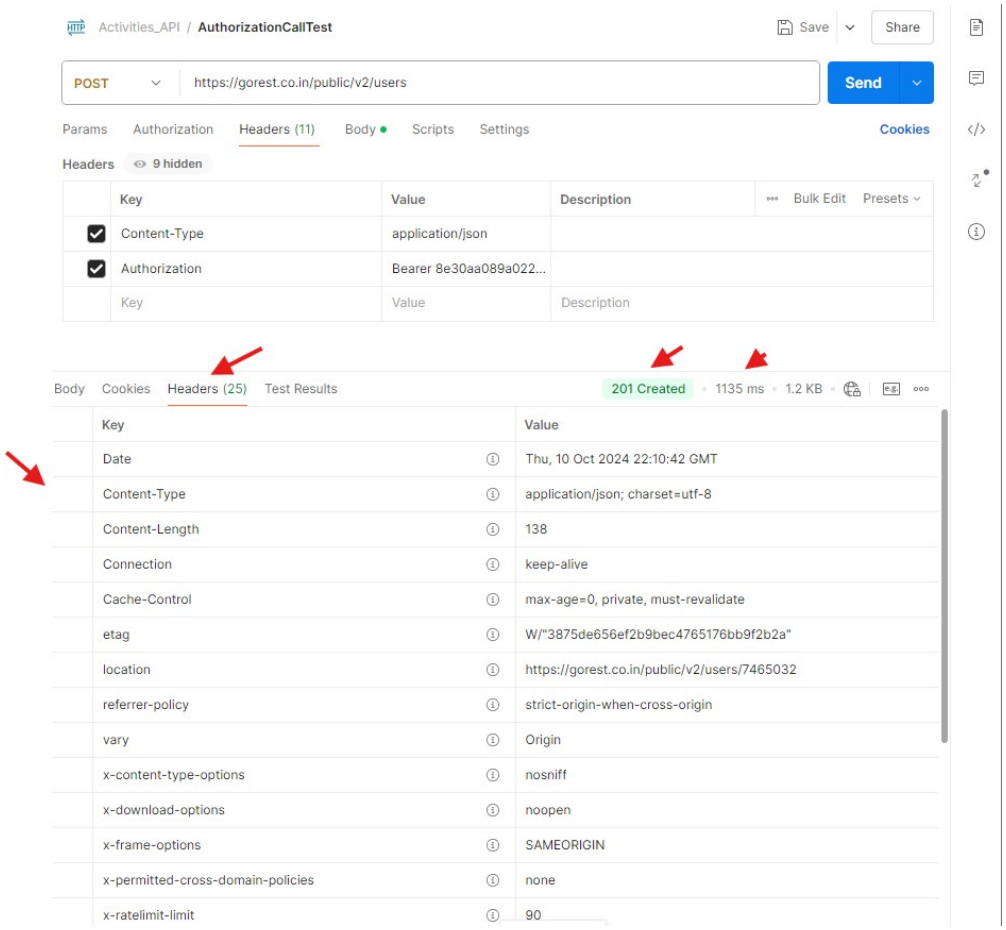


Body

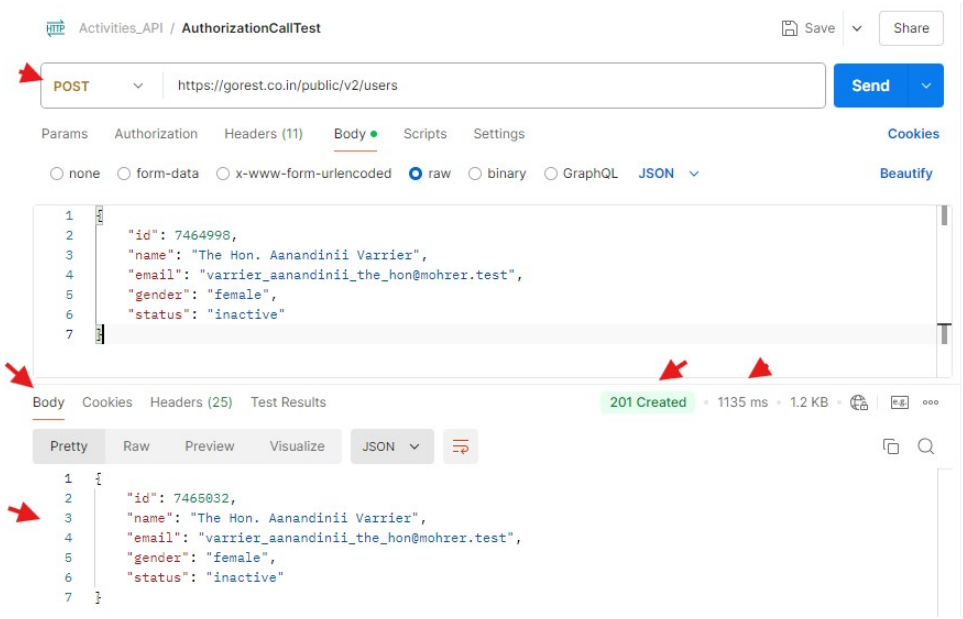


After Send...

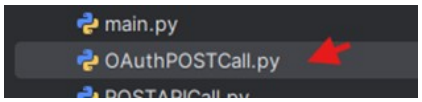
Header



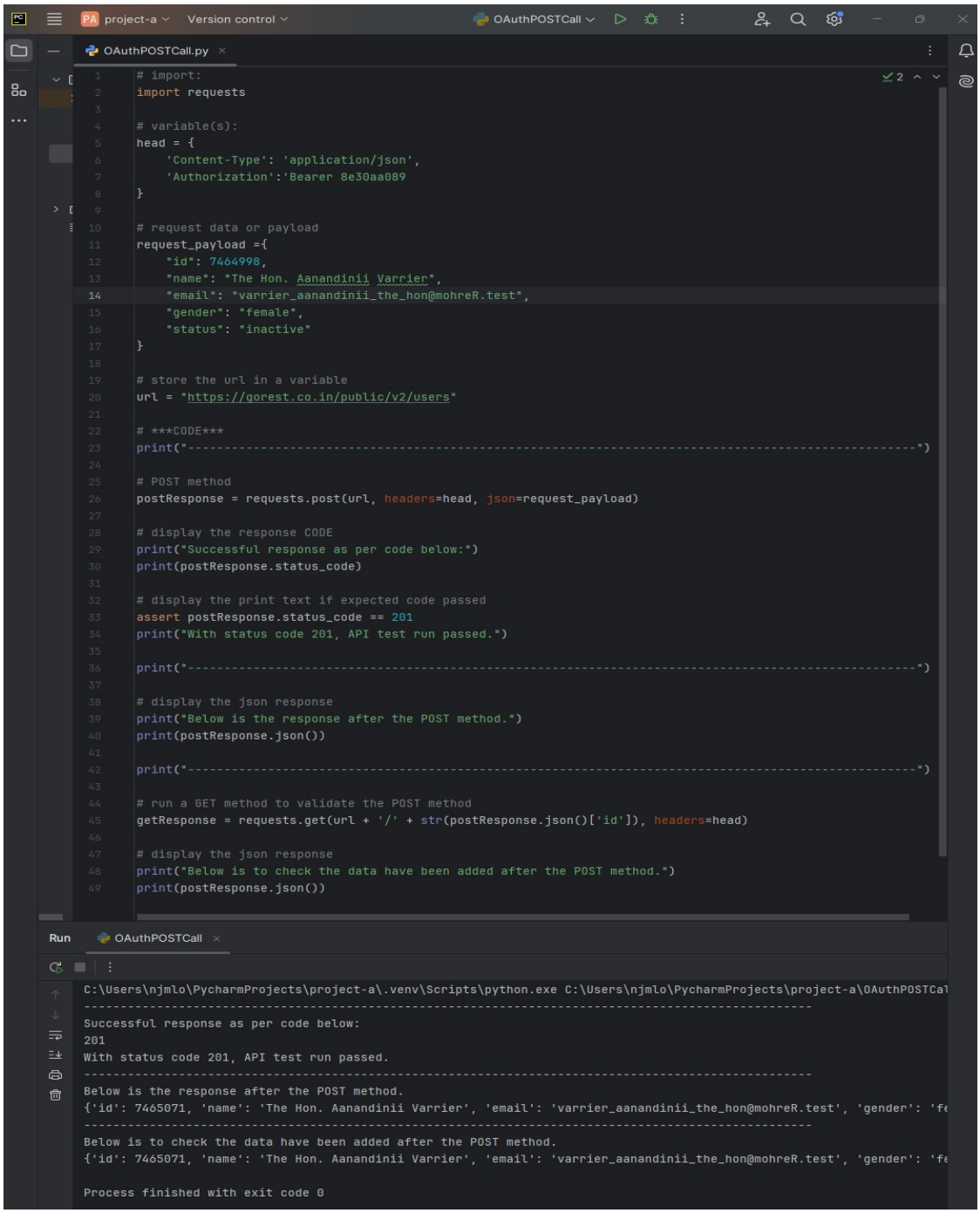
Body



28. Create a python file in PyCharm editor



29. Sample POST (in PyCharm using Python code)

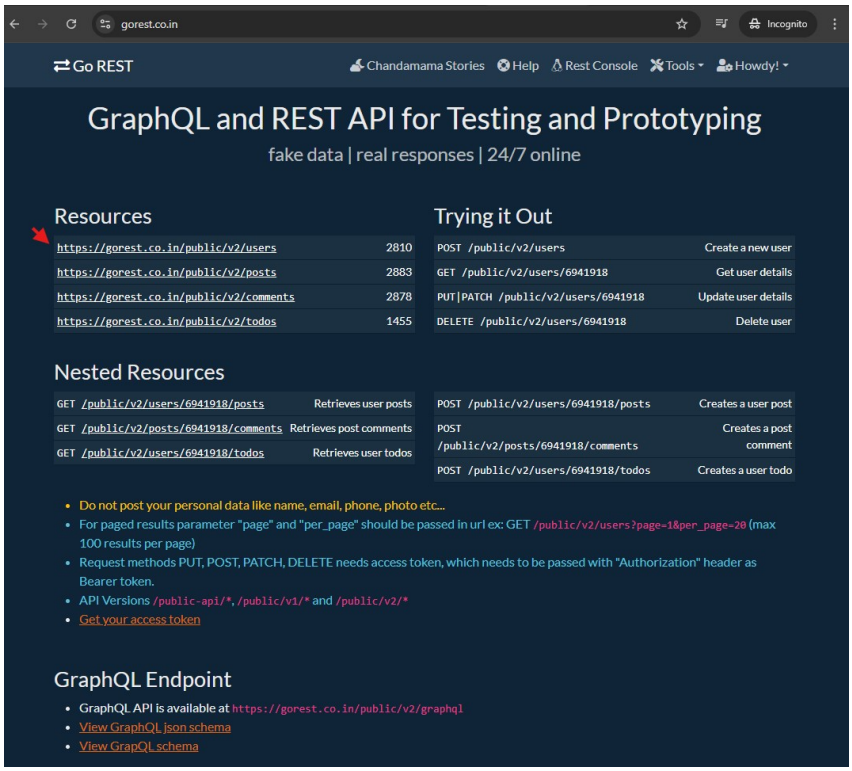


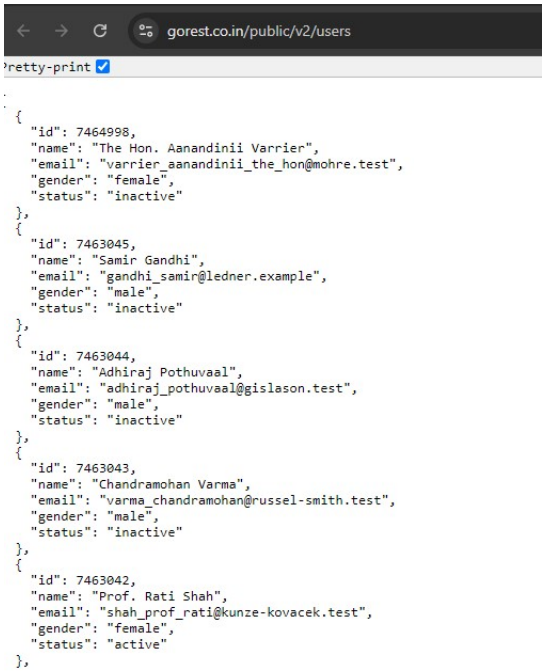
30. Here is the code in text

```
# 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("------------------------------------------------------------------------------
----------------------")

# 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())
```

31. Passing Parameters in URL

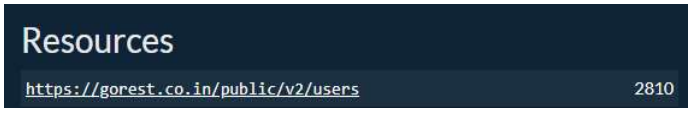
Go REST sampling APIs



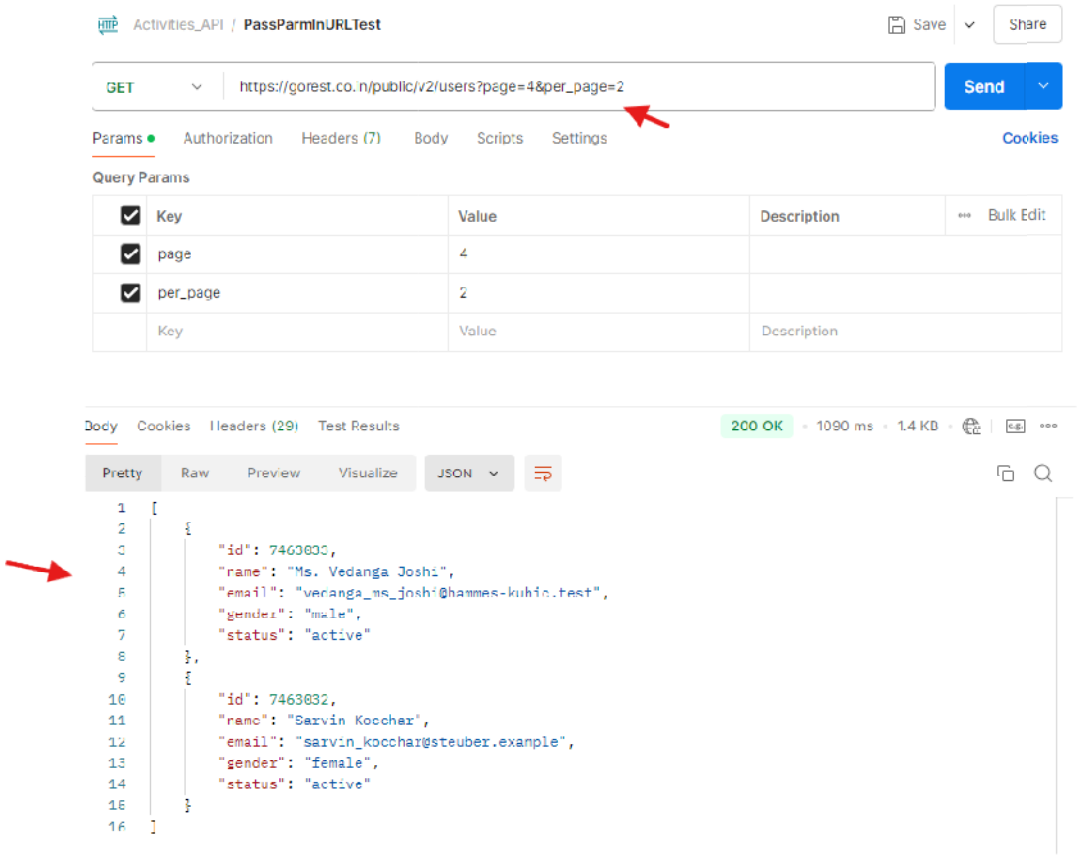


32. Sample GET (copy the sampling url from Go REST and into Postman (add parameters)

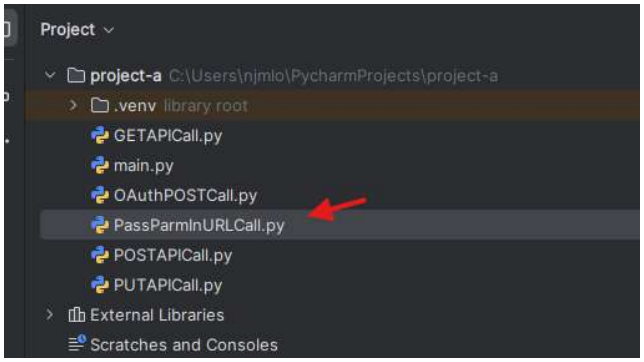
Go REST



Postman



33. Create a python file in PyCharm editor



34. Sample GET (in PyCharm using Python code)

PassParmInURLCall.py

```python
# 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())
```

```
Run  PassParmInURLCall
C:\Users\njmlo\PycharmProjects\project-a\.venv\Scripts\python.exe C:\Users\njmlo\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': 7463041, 'name': 'Amb. Anjushree Desai', 'email': 'anjushree_desai_amb@mosciski.test', 'gender': 'male', 'status': 'inactive'}, {'id': 7463040, 'name': 'Bhisham Pilla', 'email': 'pilla_bhisham@walsh.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)

JSON Beautifier & Editor chrome-extension://lpopeocbeepakdnipejhlpcmifheolpl/data/page/index.html

text tree table

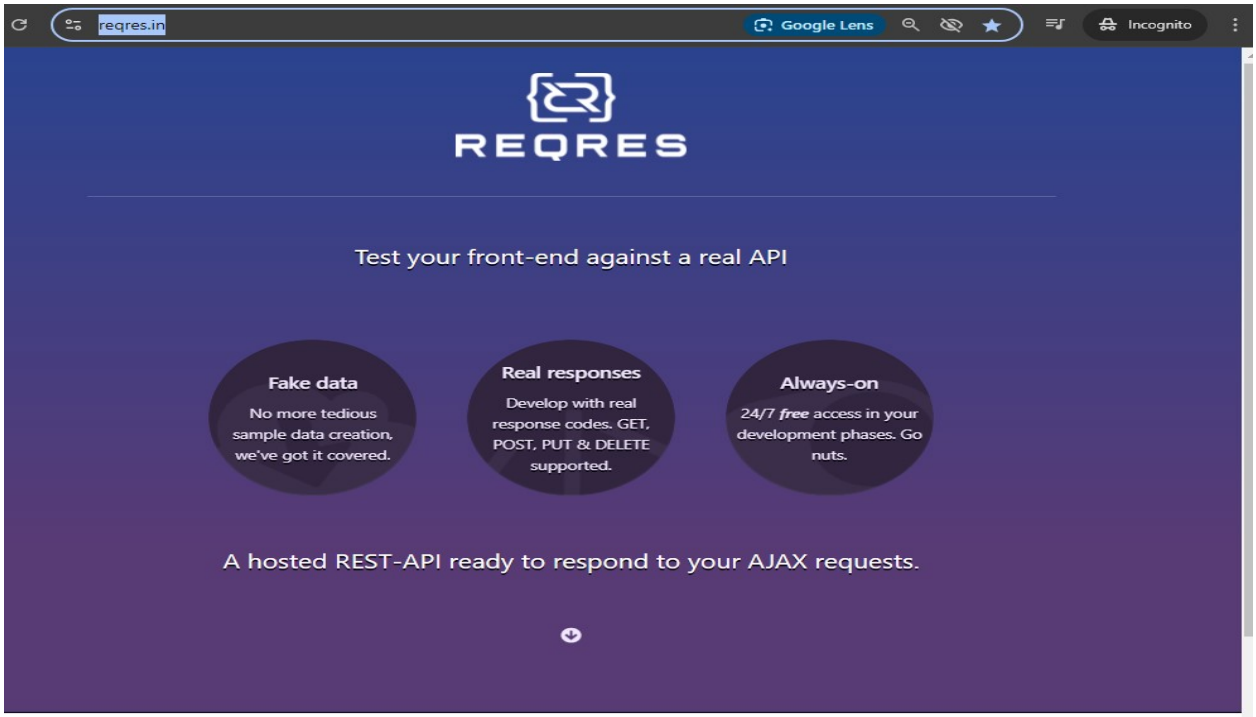
```
[ 2 items
  0 : {
      id : 7463041
      name : Amb. Anjushree Desai
      email : anjushree_desai_amb@mosciski.test
      gender : male
      status : inactive
    }
  1 : {
      id : 7463040
      name : Bhisham Pilla
      email : pilla_bhisham@walsh.example
      gender : female
      status : inactive
    }
]
```

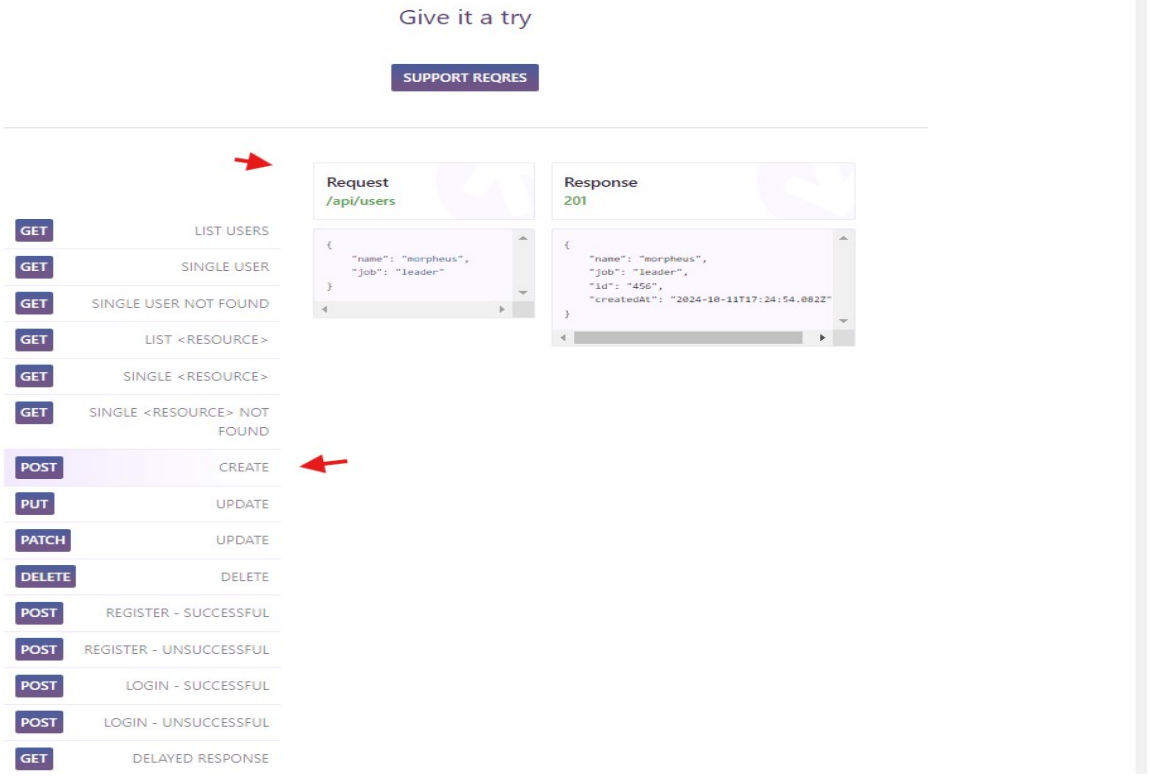
36. Here is the code in text

```
# 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

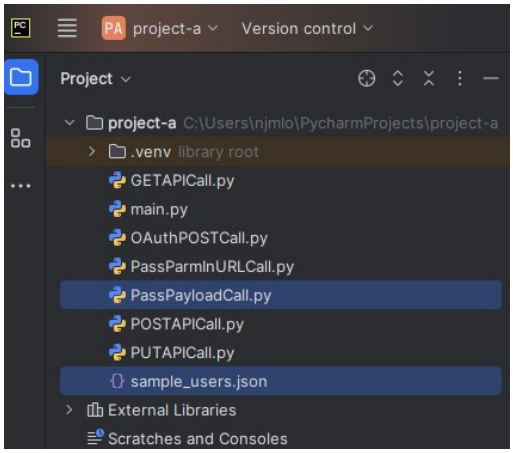
REQRES sampling APIs





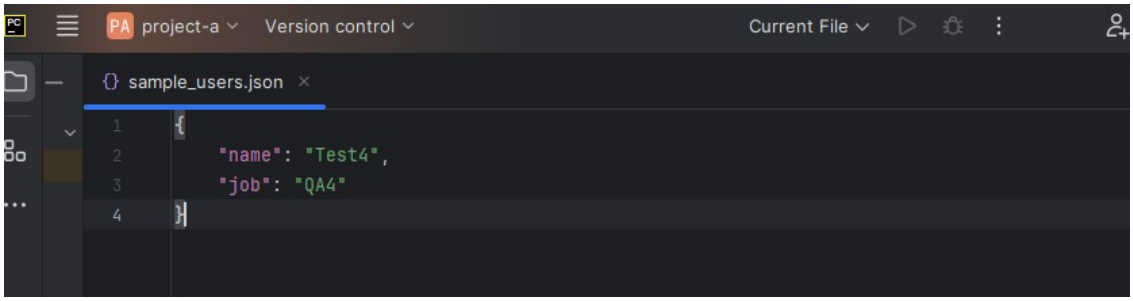
38. Create file(s) in PyCharm editor

Python and JSON file

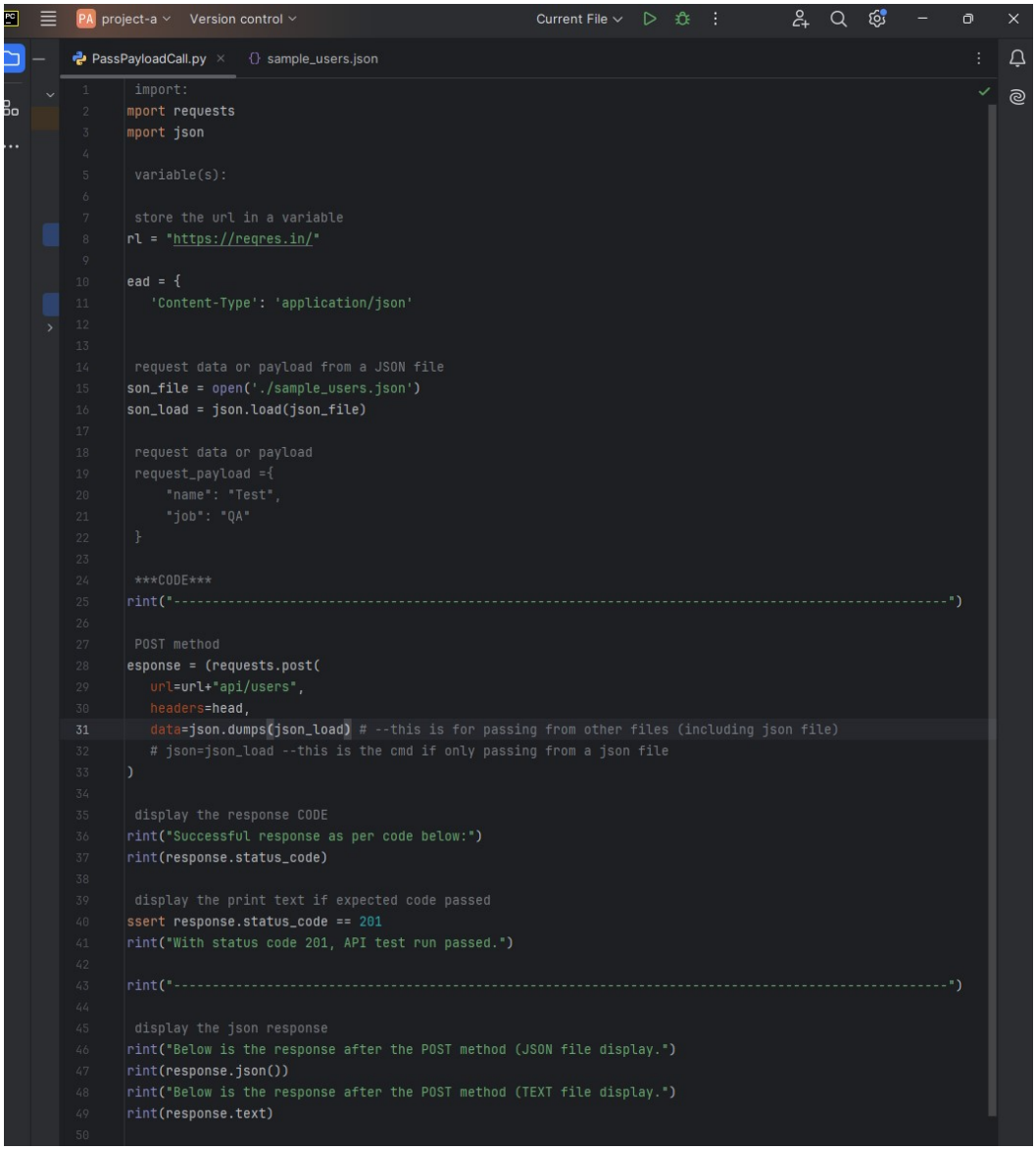


39. Sample POST (in PyCharm using Python code)

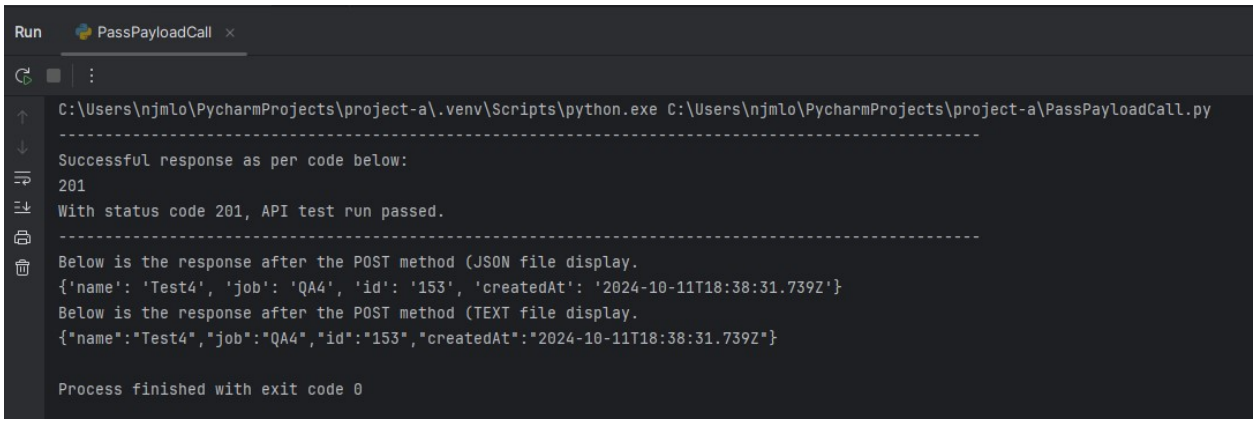
JSON file



Python file



```
import:
mport requests
mport json

variable(s):

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

ead = {
   'Content-Type': 'application/json'


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

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

***CODE***
rint("-----------------------------------------------------------------------------------------------")

POST method
esponse = (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
rint("Successful response as per code below:")
rint(response.status_code)

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

rint("-----------------------------------------------------------------------------------------------")

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



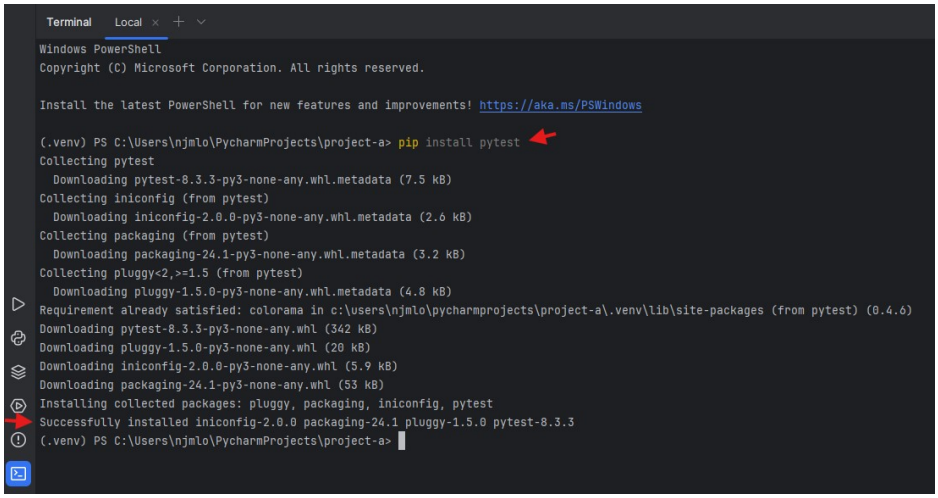
```
C:\Users\njmlo\PycharmProjects\project-a\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\project-a\PassPayloadCall.py
-----------------------------------------------------------------------------------------------
Successful response as per code below:
201
With status code 201, API test run 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

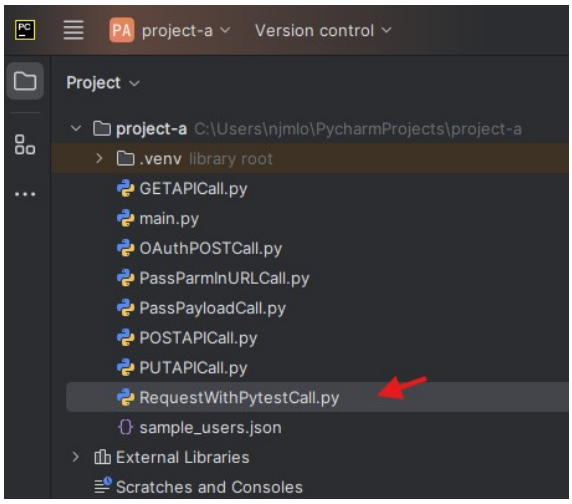
```
# import:
import requests
import json

# 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)



42. Create a python file in PyCharm editor



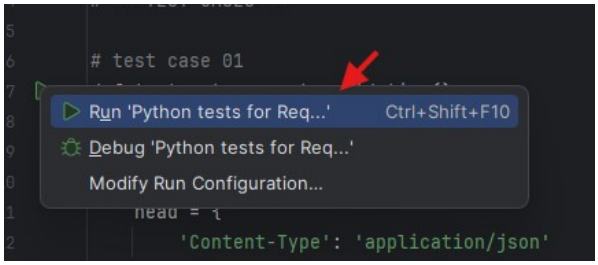
43. Sample GET (in PyCharm using Python code and PyTest to run the “test”)

Python file

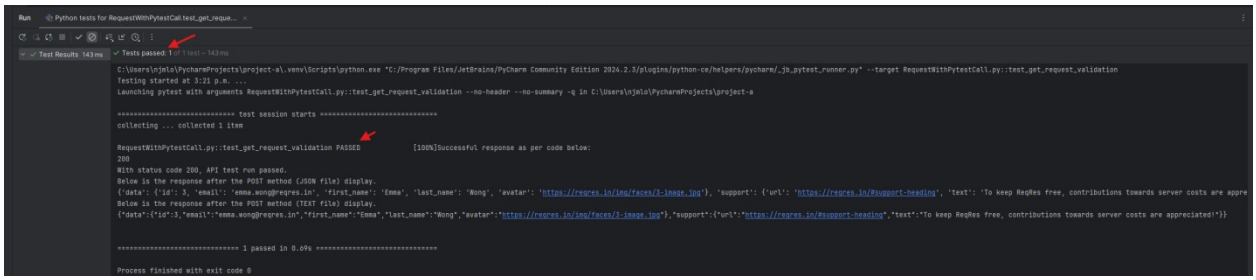
RequestWithPytestCall.py

```python
# 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=str(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

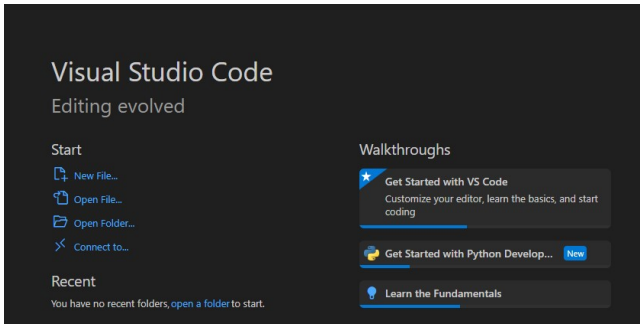


44. Here is the code in text

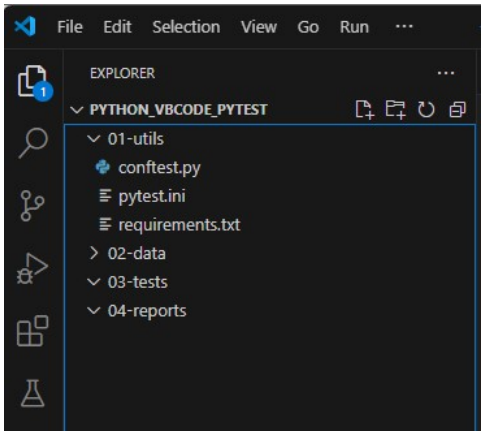
```
# import:
import requests

# ***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=str(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)

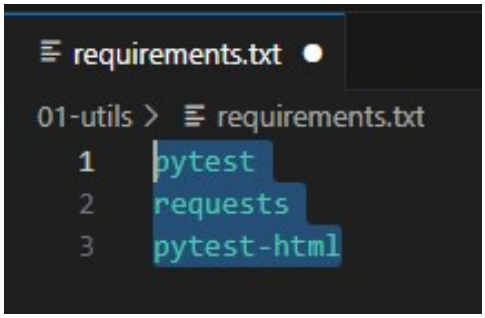
Ensure that the Visual Studio Code is setup



With the following files

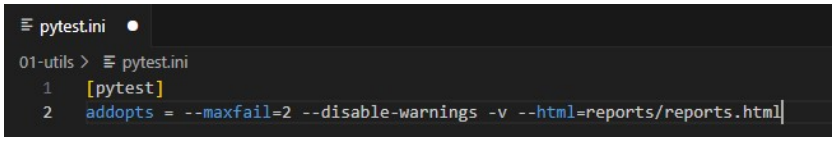


And part of the setup, the following libraries are installed (requirements.txt)



Further, the following are outlined in the following files:

pytest.ini



conftest.py

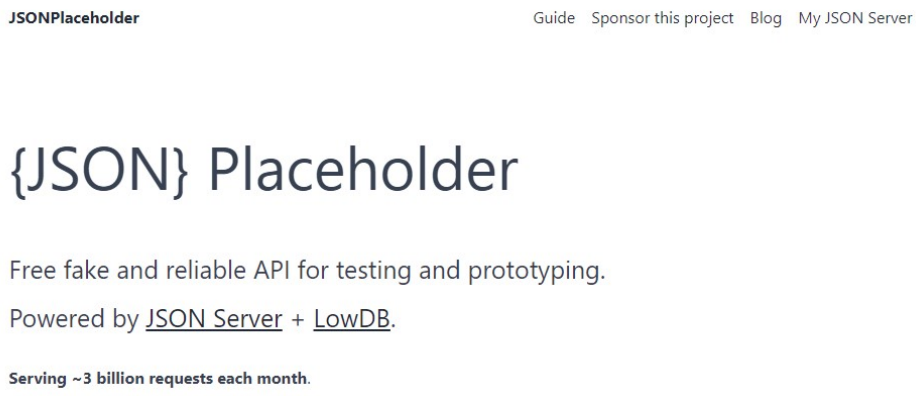
```python
import pytest
from datetime import datetime

@pytest.hookimpl(tryfirst=True)
def pytest_configure(config):
    report_dir="reports"
    now = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    config.option.htmlpath = f"{report_dir}/reports_{now}.html"

@pytest.fixture(scope='session', autouse=True)
def setup_teardown():
    print('Start')
    yield
    print('End')
```

apis.py

```python
import requests

class APIs:
    BASE_URL= "https://jsonplaceholder.typicode.com"

    def __init__(self):
        self.header = {
            "Content-Type":"application/json"
        }

    def get(self,endpoint):
        url=f'{self.BASE_URL}/{endpoint}'
        getResponse = requests.get(url, headers=self.header)
        return getResponse
```

46. Sampling source of get users

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

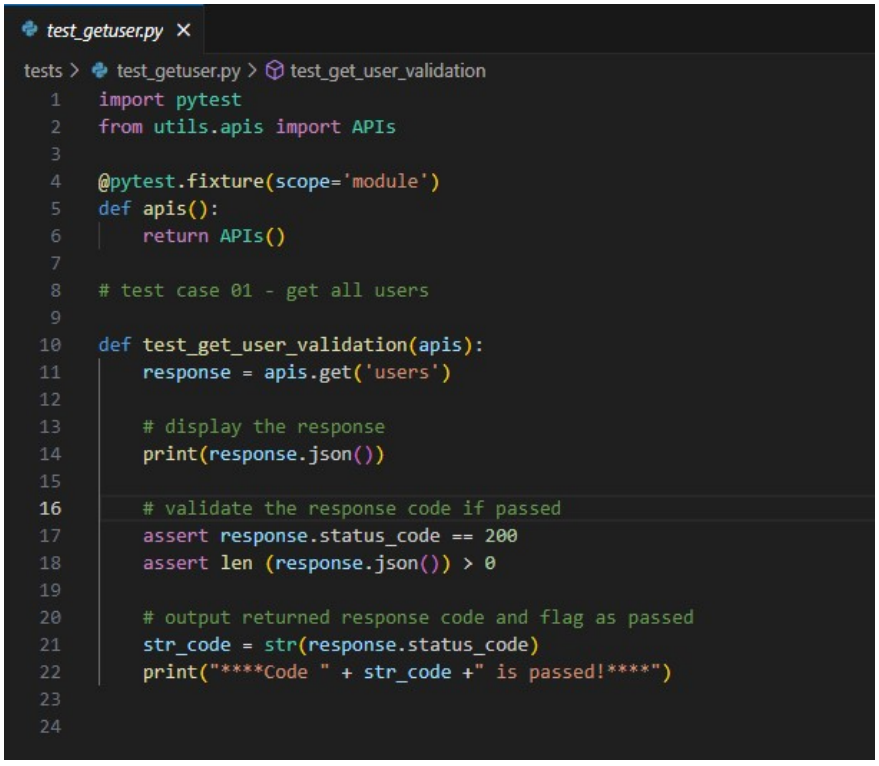


Resources

JSONPlaceholder comes with a set of 6 common resources:

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

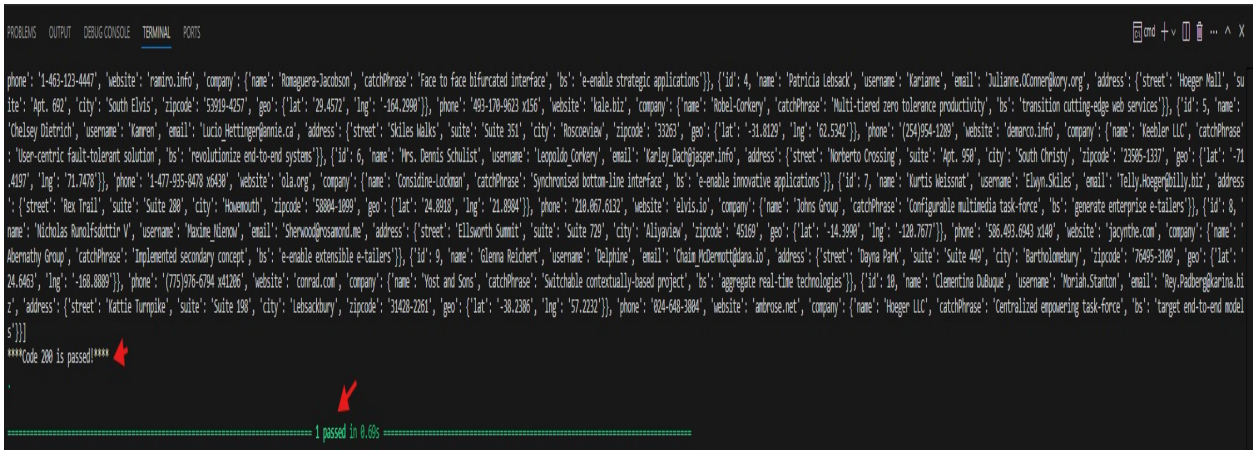
47. Run a test case in Visual Studio Code (get the same set of data (users))



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

Terminal:

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



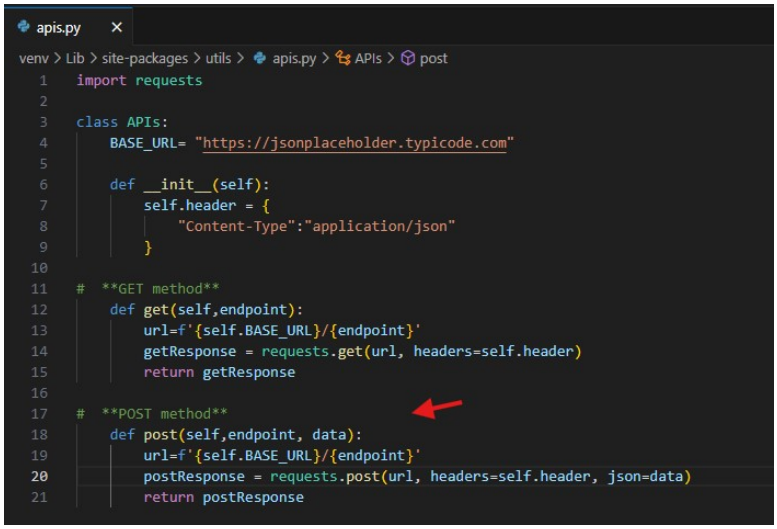
49. Here is the code in text

```python
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:

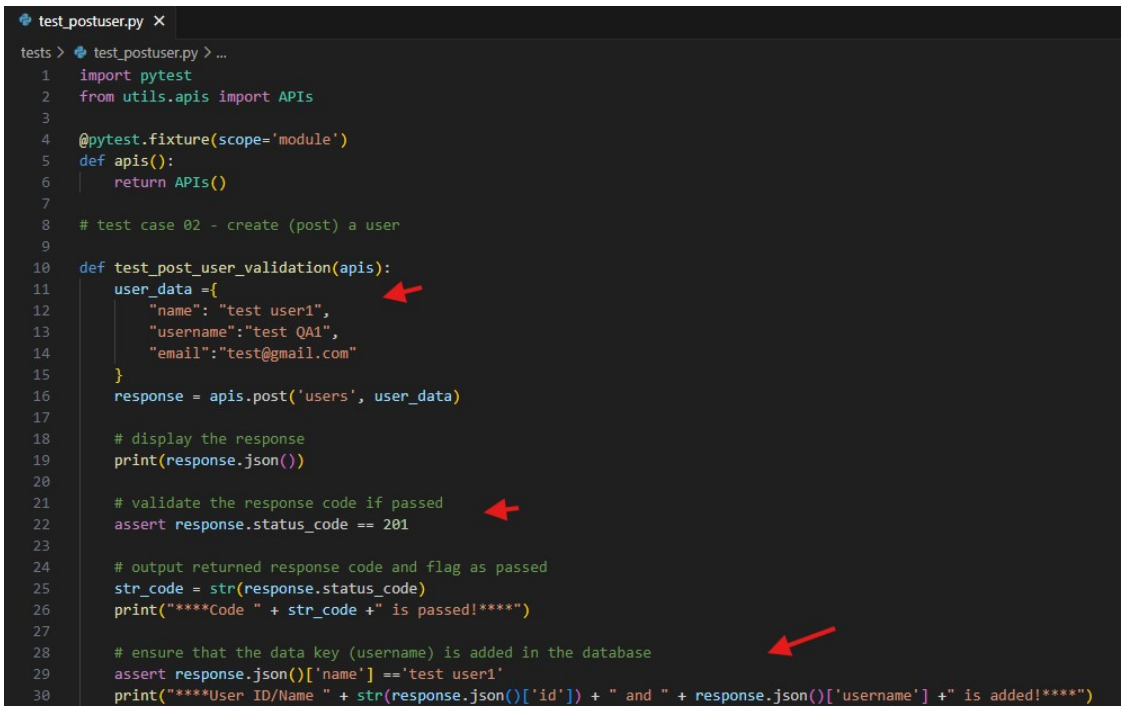
apis.py



```python
import requests

class APIs:
    BASE_URL= "https://jsonplaceholder.typicode.com"

    def __init__(self):
        self.header = {
            "Content-Type":"application/json"
        }

#  **GET method**
    def get(self,endpoint):
        url=f'{self.BASE_URL}/{endpoint}'
        getResponse = requests.get(url, headers=self.header)
        return getResponse

#  **POST method**
    def post(self,endpoint, data):
        url=f'{self.BASE_URL}/{endpoint}'
        postResponse = requests.post(url, headers=self.header, json=data)
        return postResponse
```

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

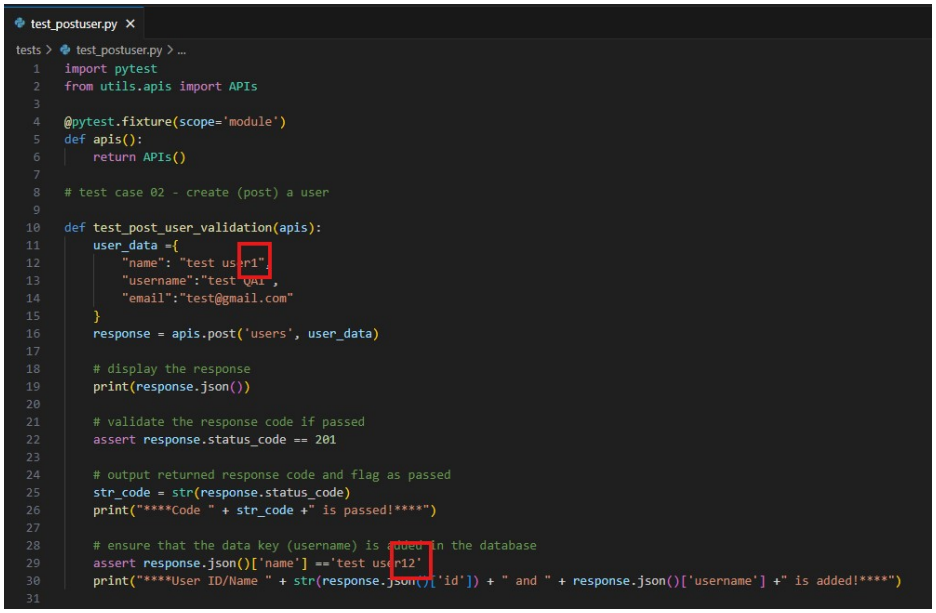
Sampling of no error

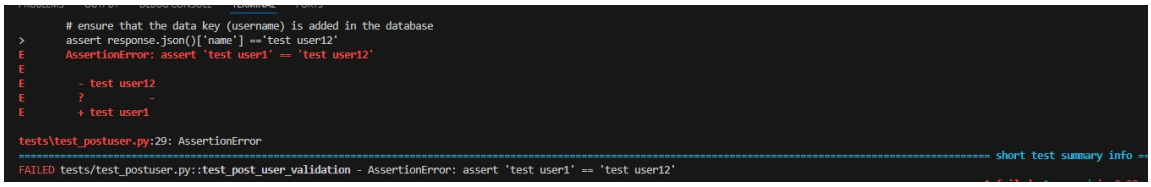


```python
import pytest
from utils.apis import APIs

@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 user1'
    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





52. Here is the code in text

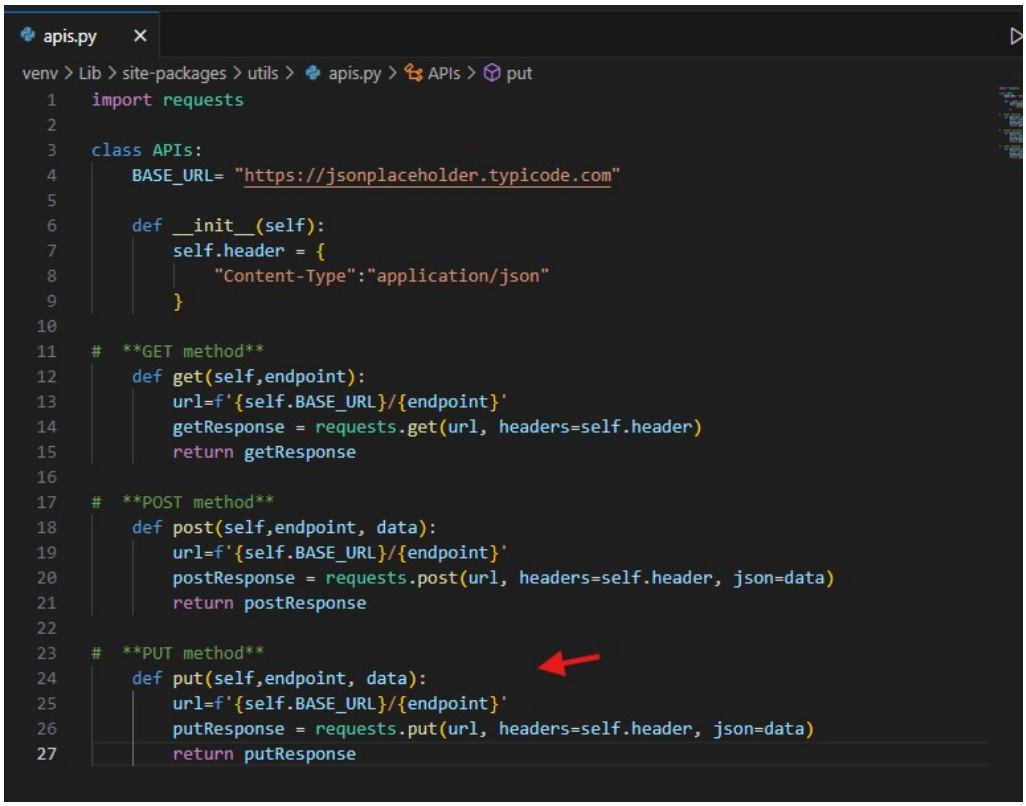
```python
import pytest
from utils.apis import APIs

@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



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

Sampling of no error

test_putuser.py

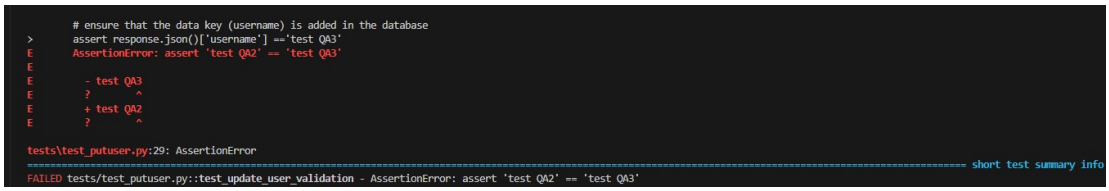
```python
import pytest
from utils.apis import APIs

@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!****")
```

```
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

test_putuser.py

```python
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 QA3'
    print("****User ID/Name " + str(response.json()['id']) + " and " + response.json()['username'] +" is updated!****")
```



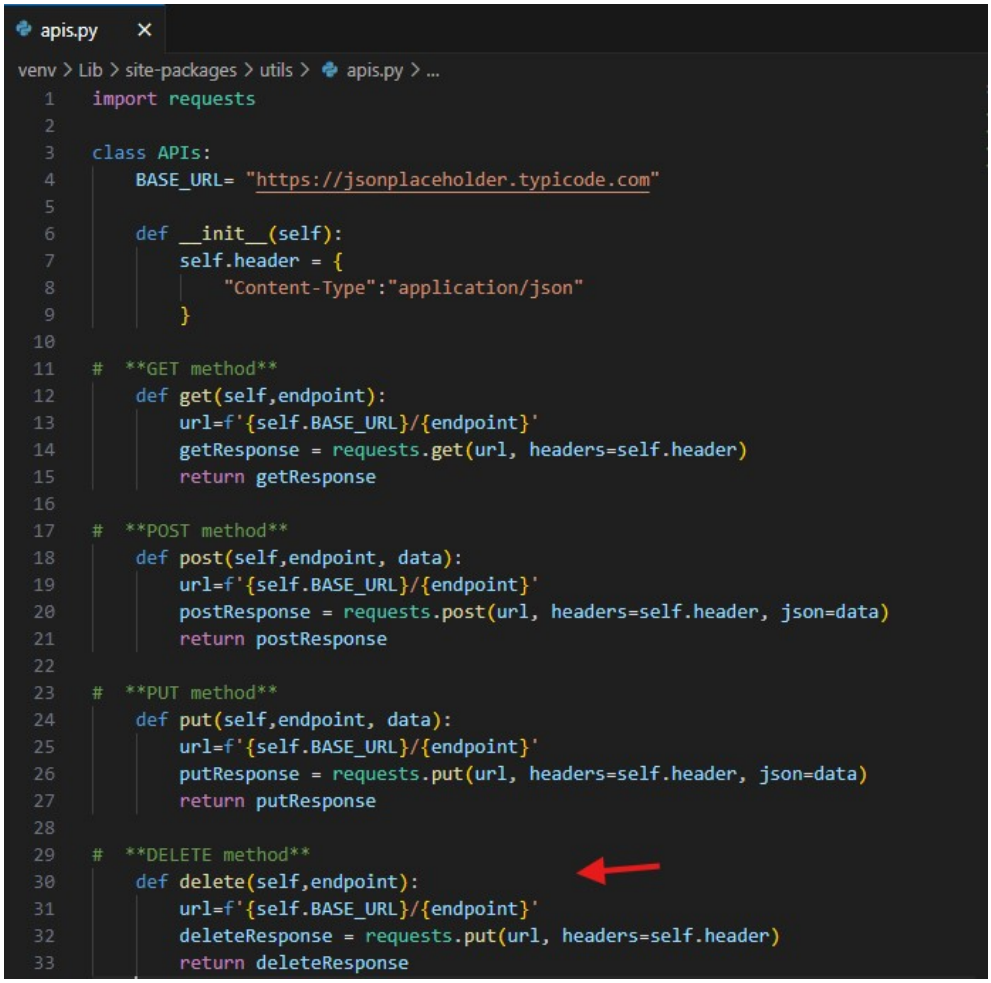
55. Here is the code in text

```
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



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

Sampling of no error

test_deleteuser.py

```
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\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}
****Code 200 is passed (data deleted!****
.
```

Sampling with error

test_deleteuser.py

```
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
```

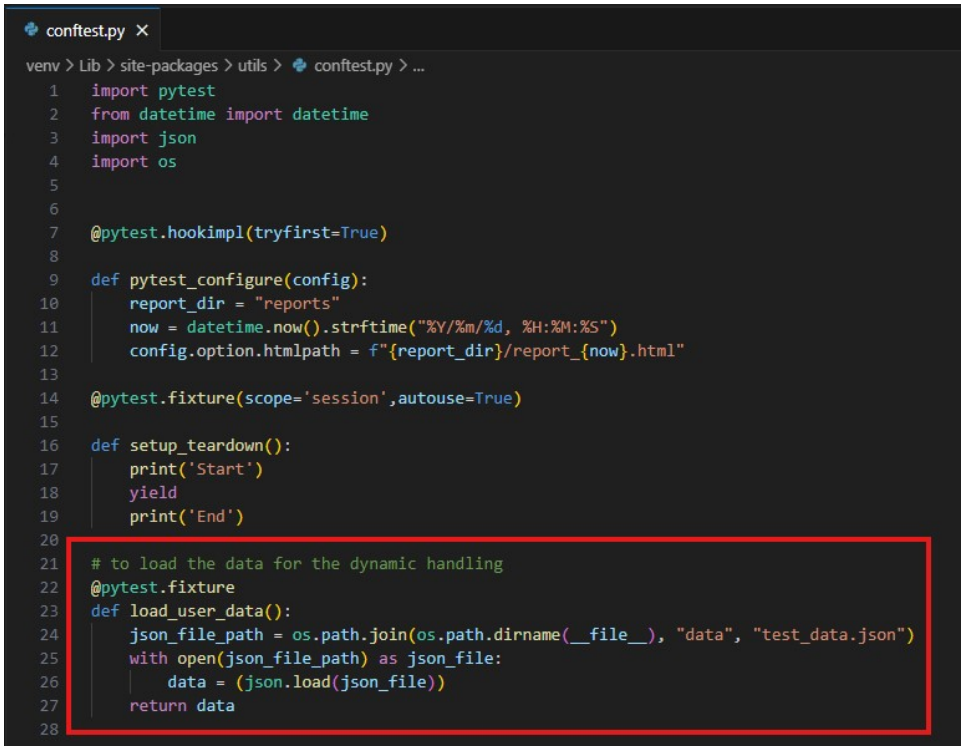
58. Here is the code in text

```python
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)

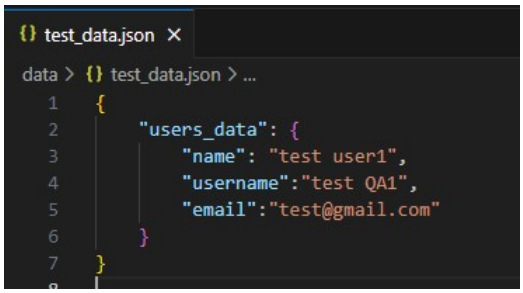
As a continuation of the prior method framework, reuse and update some of the existing files:

conftest.py



Add a new file:

test_data.json



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

Sampling of no error

test_postuser_dyna_handl.py

tests > test_postuser_dyna_handl.py > test_post_user_validation

```python
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!****")

```

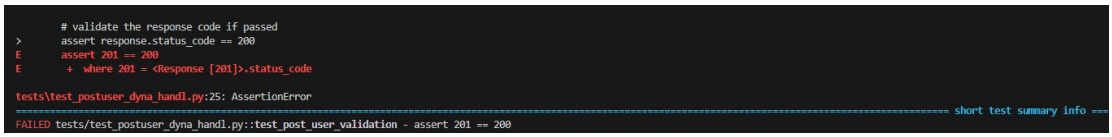
```
tests\test_postuser_dyna_handl.py This is now the new email: ffd1232f@yahoo.com
{'name': 'test user1', 'username': 'test QA1', 'email': 'ffd1232f@yahoo.com', 'id': 11}
****Code 201 is passed!****
****New Email ffd1232f@yahoo.com for ID 11 is added!****
```

Sampling with error

test_postuser_dyna_handl.py

tests > test_postuser_dyna_handl.py > ...

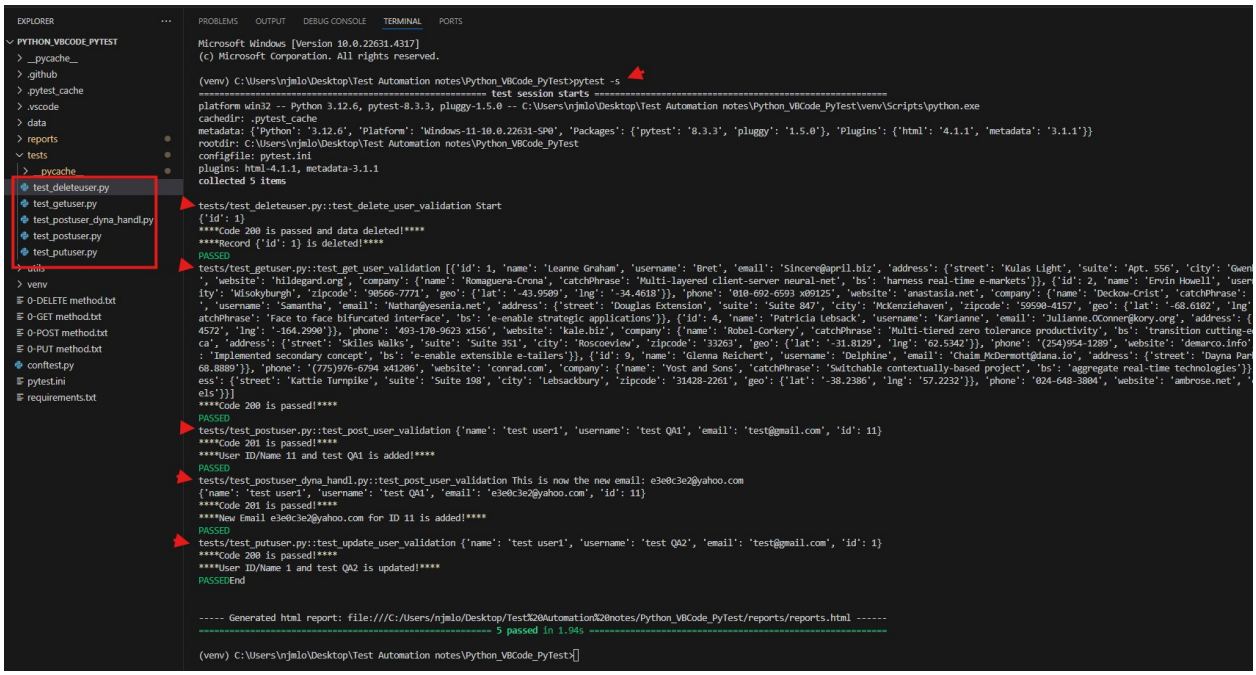
```python
import pytest
from utils.apis import APIs
import uuid

@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 == 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 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!****")
```



61. Here is the code in text

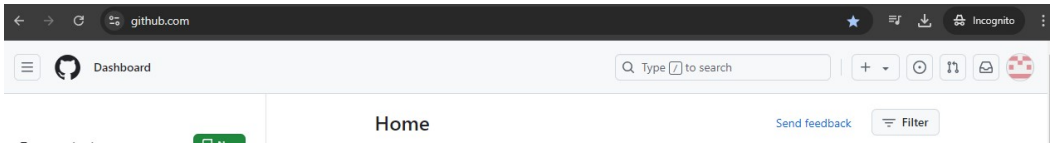
```
import pytest
from utils.apis import APIs
import uuid

@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

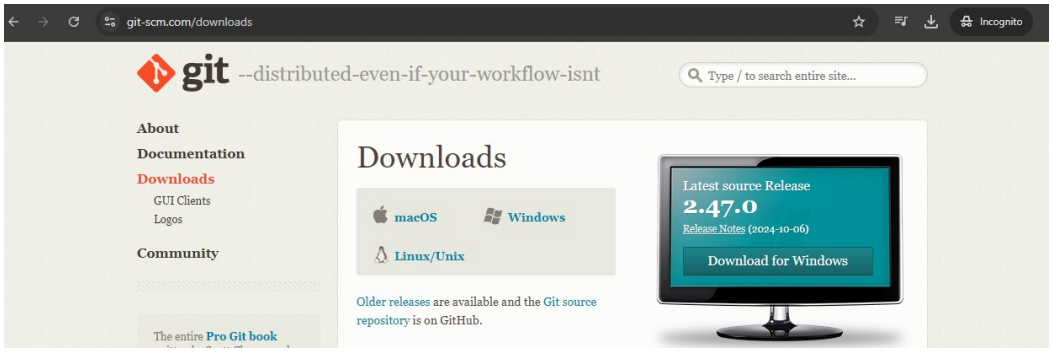


63. GitHub setup (Create, Stage, Commit, and Publish with VS Code)

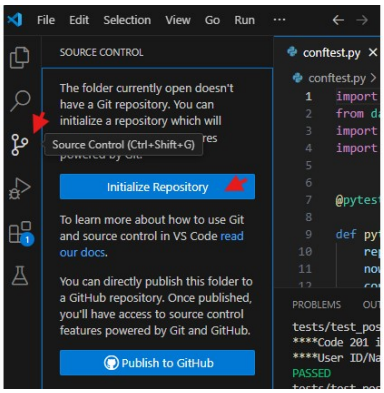
Have a GitHub account

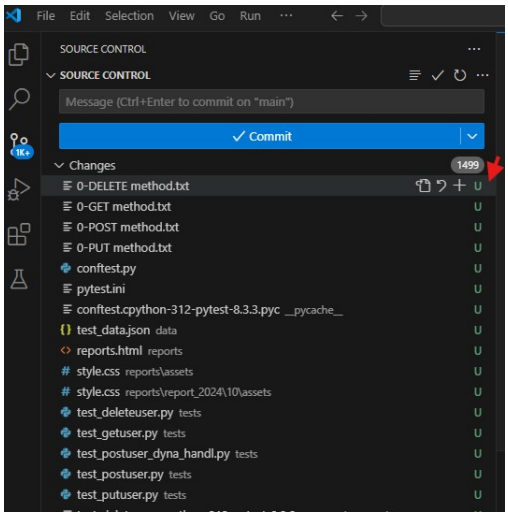


Download and install Git app



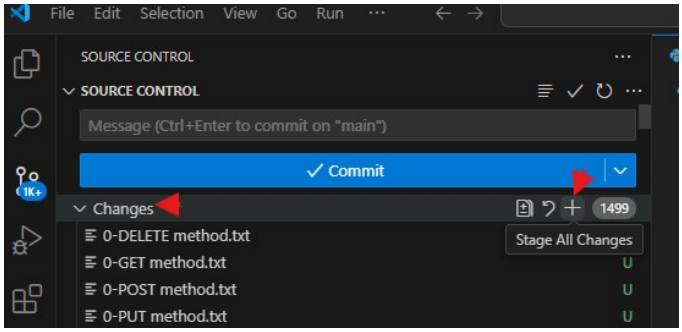
Initialize Repository



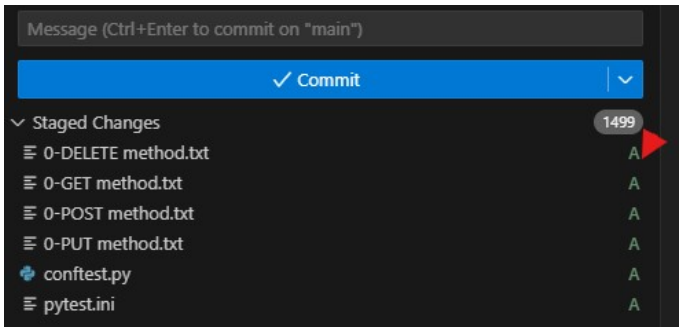


Stage all files

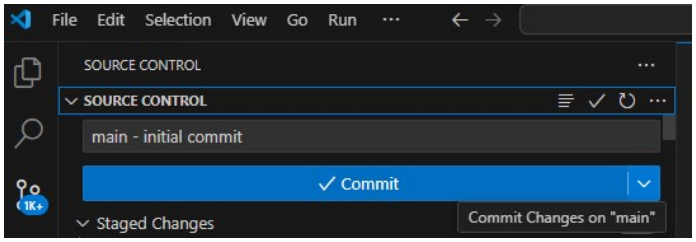
Before

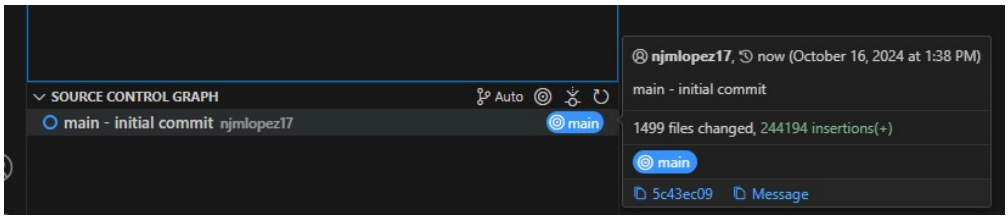


After

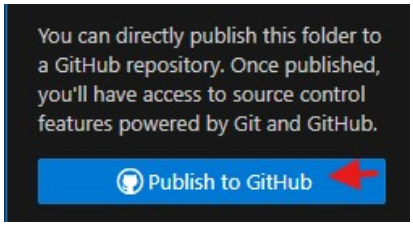


Commit all files

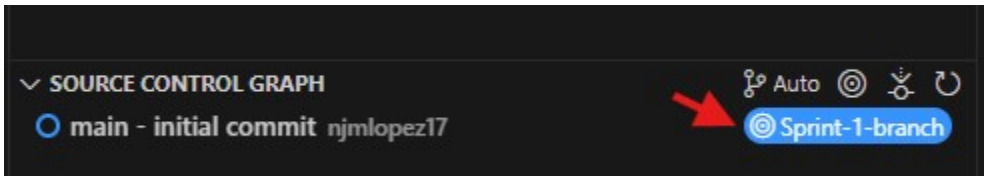




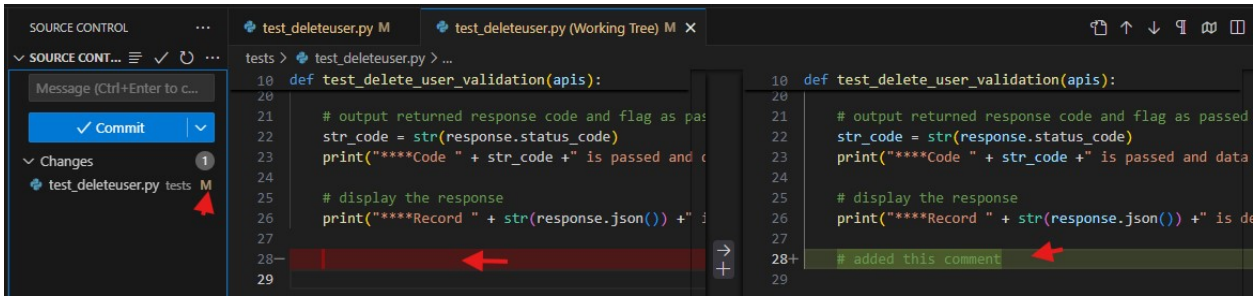
Publish GitHub



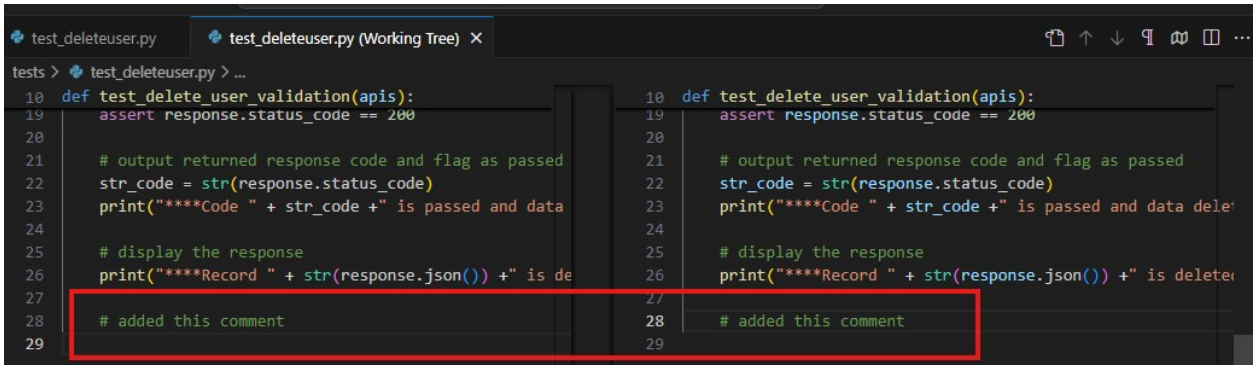
Other than the main branch, create a branch



Sampling code change that is in the branch



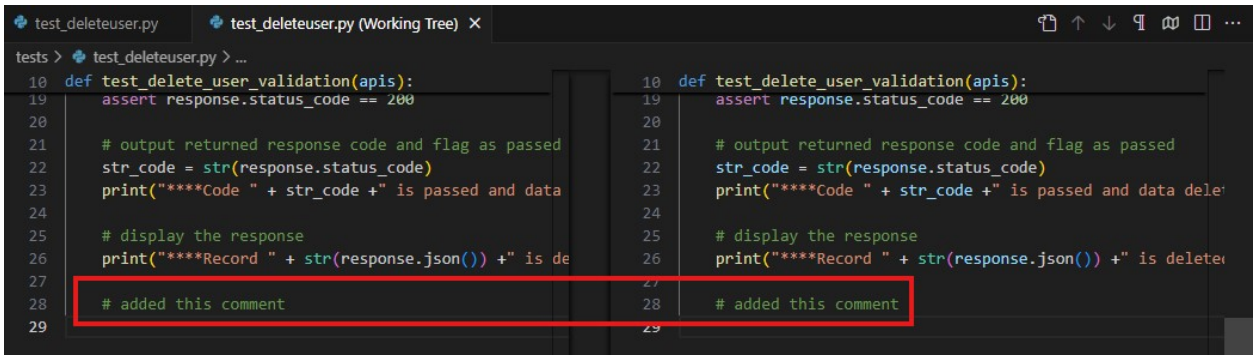
Stage and commit the change



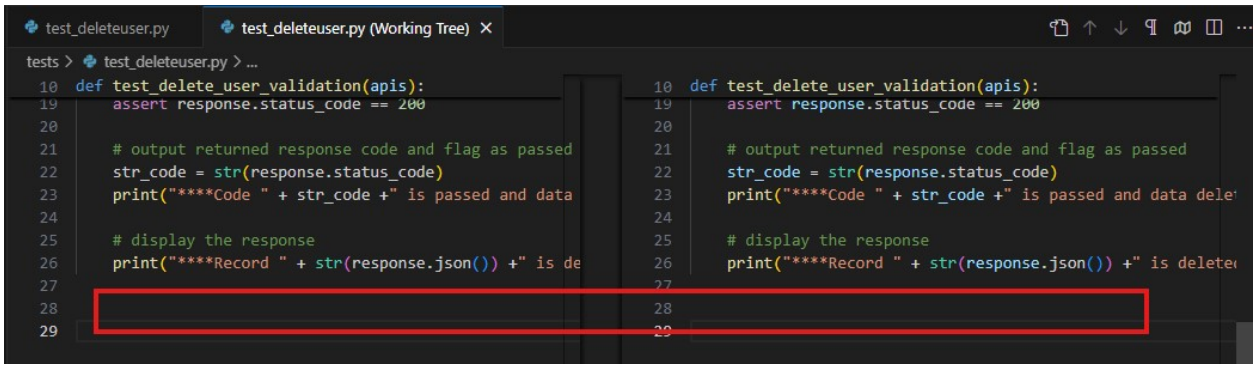
Merge the spring branch into that of the main branch

Before merging:

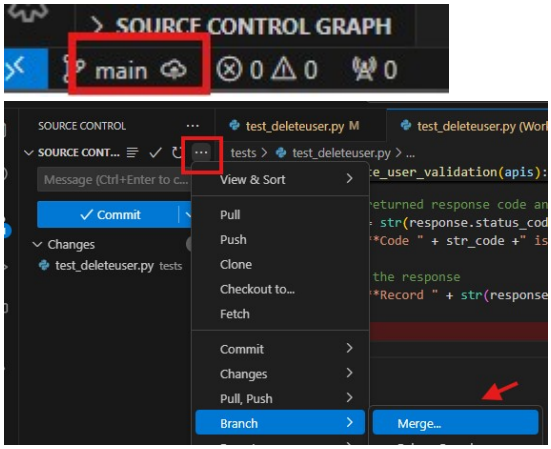
Sprint branch



Main branch

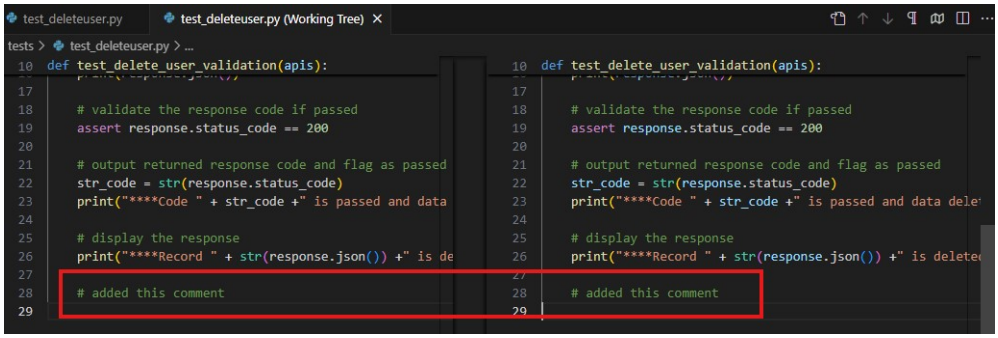


Merge the branch changes

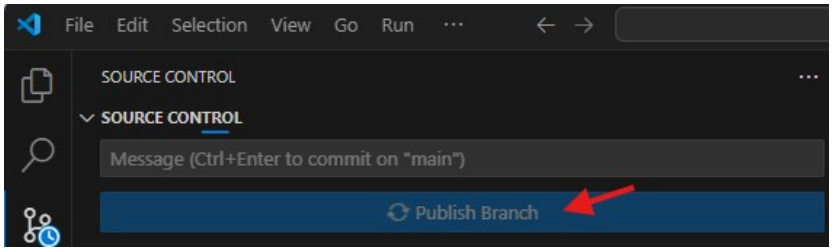


After merging:

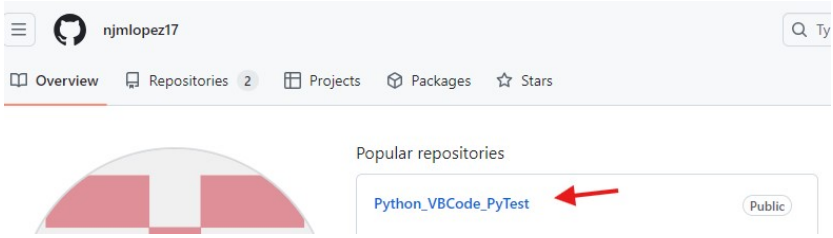
Main branch



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

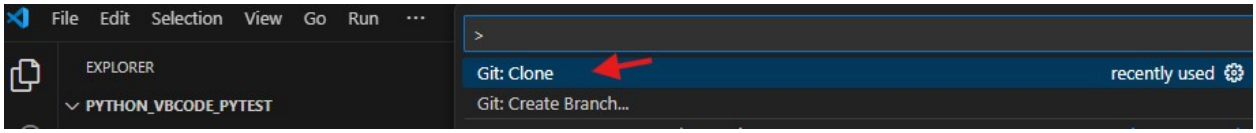


Added in GitHub



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

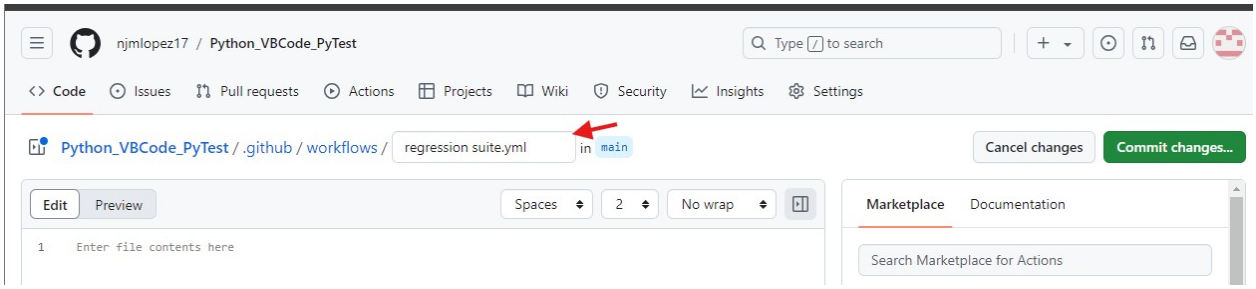
Note: to run this whole sampling of codes, simply clone in Visual Studio Code editor (using the URL above)



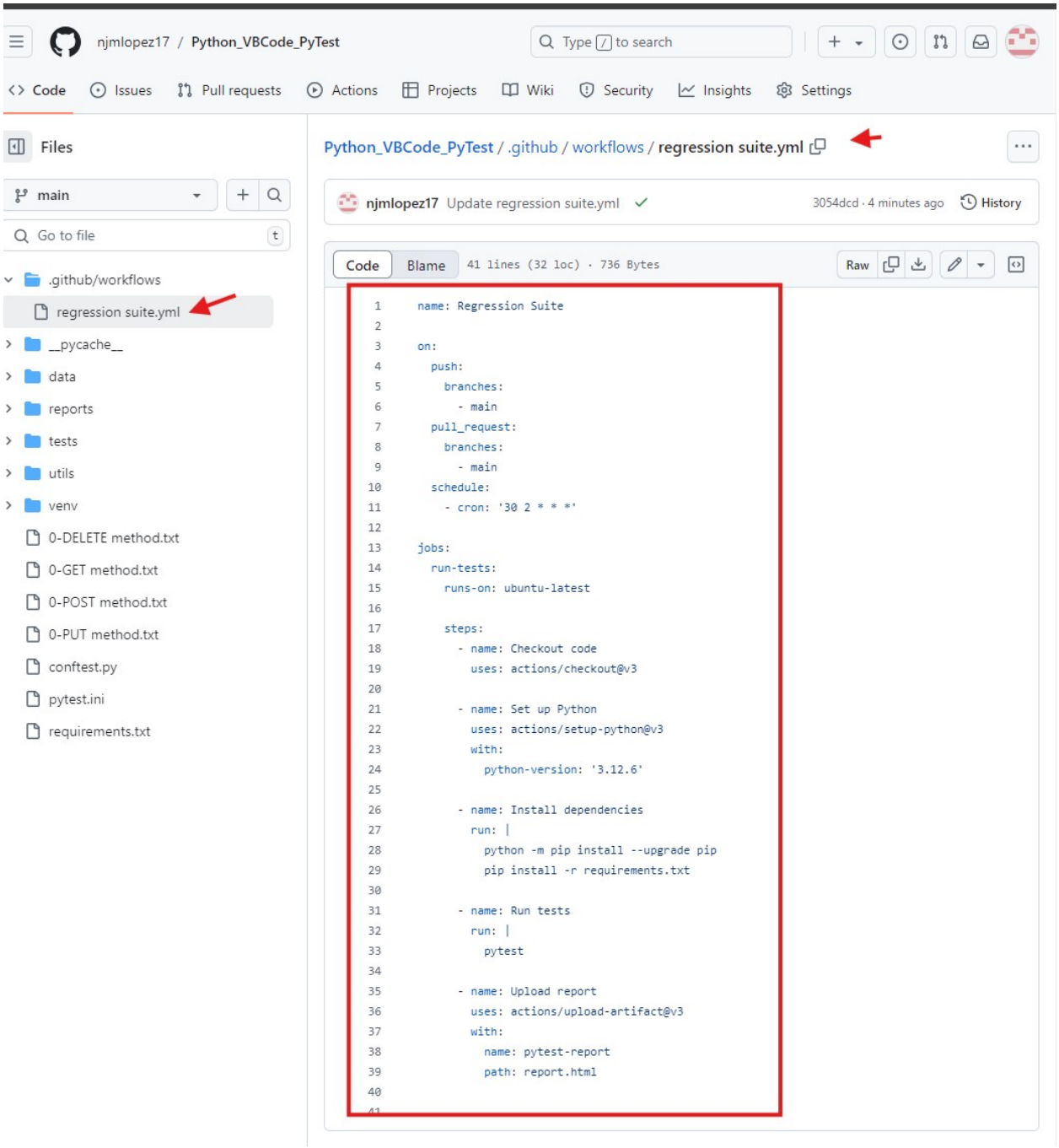
64. CI using GitHub Actions

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

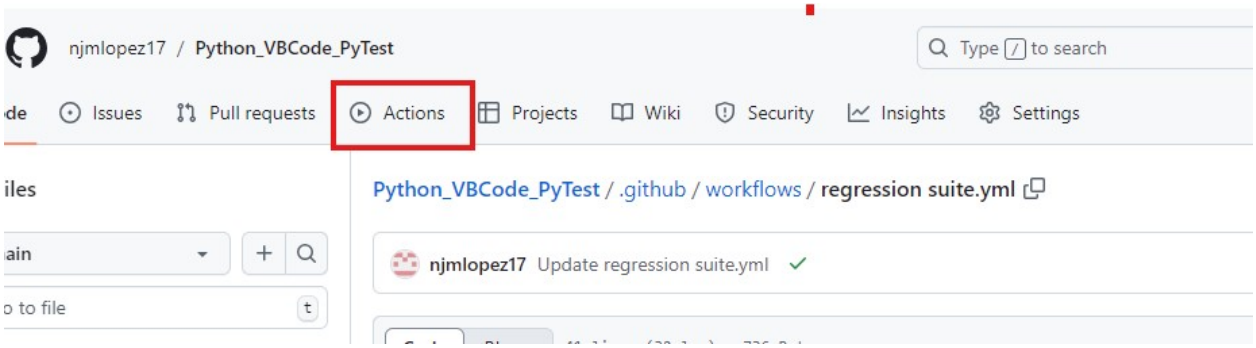
Create a new file in GitHub (and commit changes)

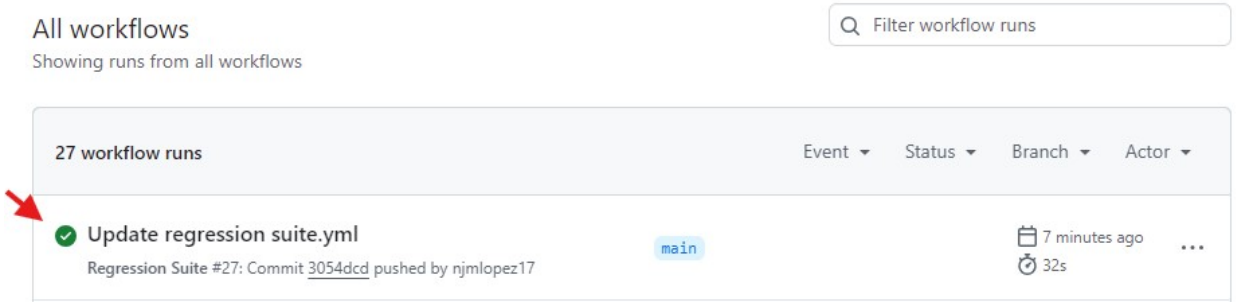


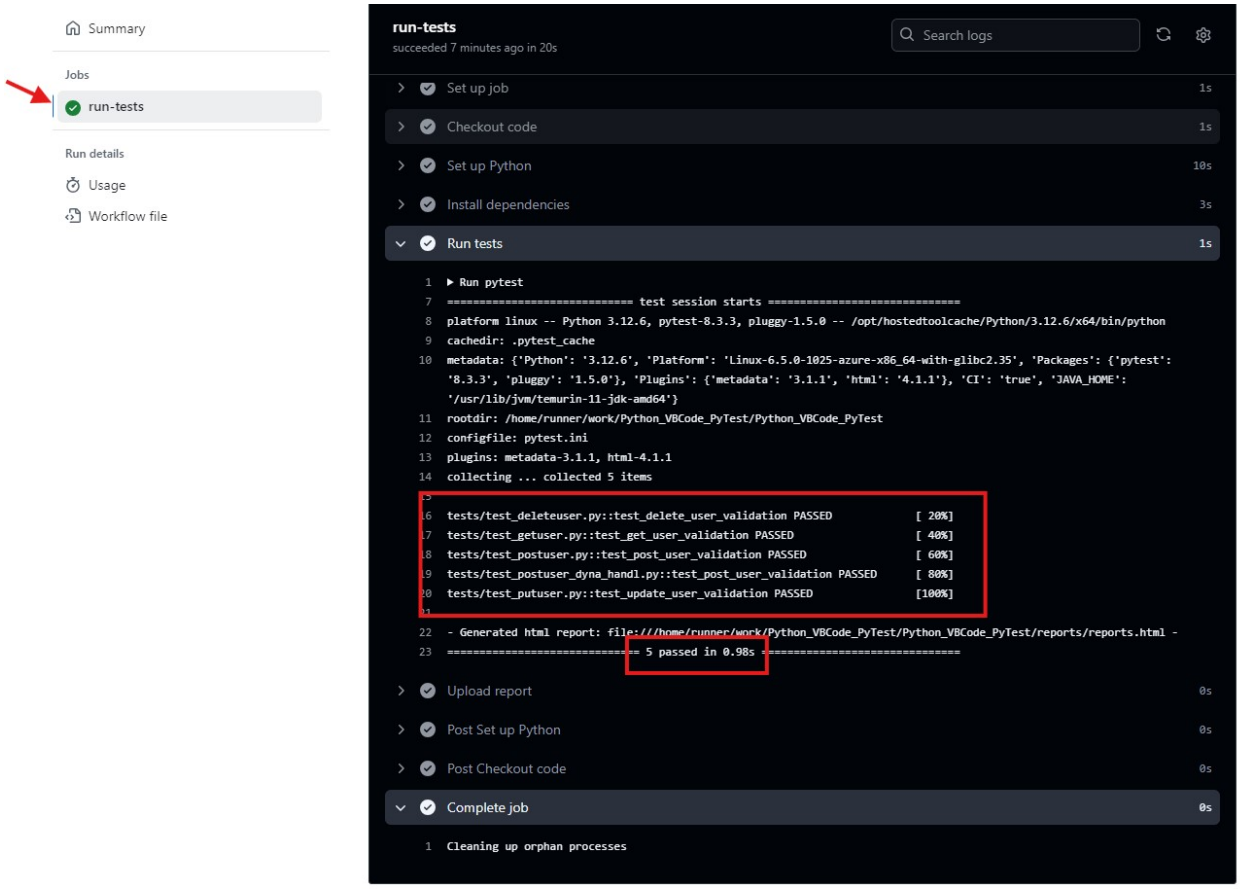
Code context of the file



After a successful run (via Actions button)







**********END**********