

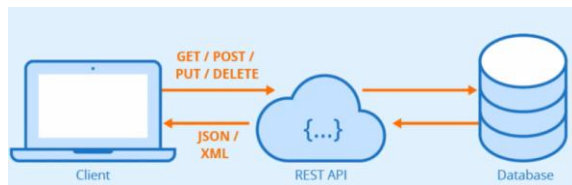
Test Automation using:

- Python
- pytest
- Visual Studio Code
- Libraries (ie., requests, json, uuid, os, datetime)
- Git and GitHub repository

Document highlight:

- 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)

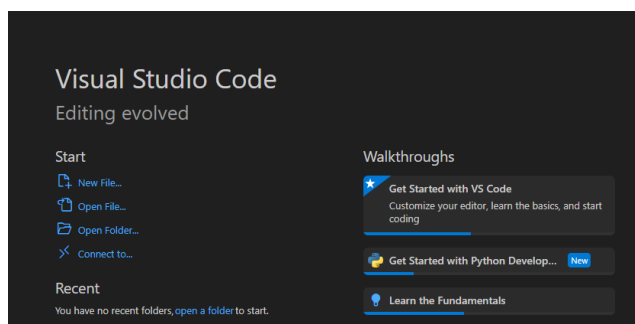


2. API Automation framework (GET method) using pytest request libraries (in Visual Studio Code)

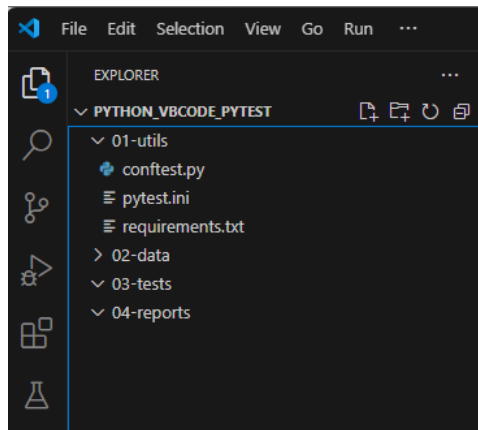
Ensure that the Visual Studio Code is setup

Download:

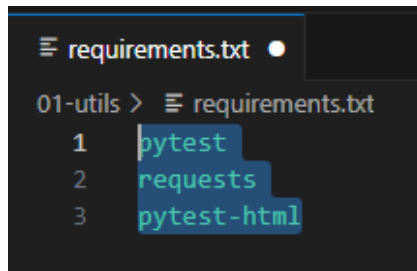
<https://code.visualstudio.com/download>



With the following files

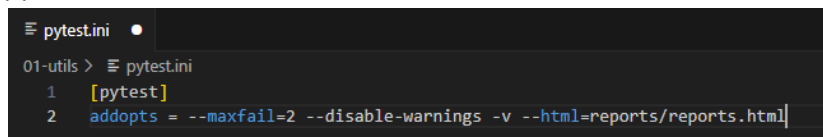


And part of the setup, the following libraries are installed (requirements.txt)

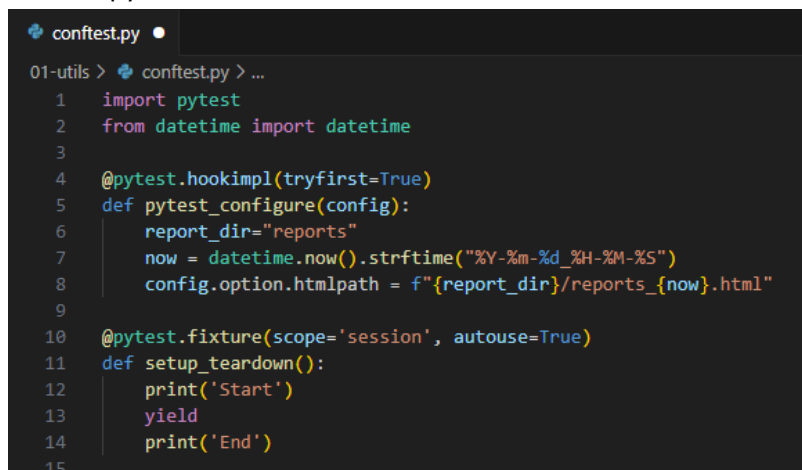


Further, the following are outlined in the following files:

pytest.ini



conftest.py



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     def get(self,endpoint):
12         url=f'{self.BASE_URL}/{endpoint}'
13         getResponse = requests.get(url, headers=self.header)
14         return getResponse
15
```

3. 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:

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



```

+ ↻ jsonplaceholder.typicode.com/users
Pretty print
[{"id":1,"name":"Leanne Graham","username":"Bret","email":"Sincere@april.biz","address":{"street":"Nulas Light","suite":"Apt. 556","city":"Gwenborough","zipcode":"92998-3874","geo":{"lat":"-37.3159","lng":"81.1496"},"phone":"1-770-736-8081 x6442","website":"hildegard.org","company":{"name":"Romaguera-Crona","catchPhrase":"Multi-layered client-server neural-net","bs":"harness real-time e-markets"}}, {"id":2,"name":"Ervin Howell","username":"Antonette","email":"Shanna@ellisla.tv","address":{"street":"Victor Plains","suite":"Suite 879","city":"Wisokyburgh","zipcode":"96566-7771","geo":{"lat":"-43.9989","lng":"-34.4011"},"phone":"010-692-6593 x69125","website":"anastasia.net","company":{"name":"Deckow-Crist","catchPhrase":"Proactive dynamic contingency","bs":"synergize scalable supply-chains"}}, {"id":3,"name":"Clementine Bauch","username":"Samantha","email":"Nathan@yesenia.net","address":{"street":"Douglas Extension","suite":"Suite 847","city":"Kielikiehaven","zipcode":"39599-4157","geo":{"lat":"-48.6102","lng":"-47.0653"},"phone":"1-463-123-4447","website":"ramiro.info","company":{"name":"Romaguera-Jacobson","catchPhrase":"Face to face bifurcated interface","bs":"e-enable strategic applications"}}, {"id":4,"name":"Patricia Lebsack","username":"Vivianne","email":"Julianne.Conner@my.org","address":{"street":"Heger Hall","suite":"Apt. 682","city":"South Elvira","zipcode":"55919-4257","geo":{"lat":"19.4572","lng":"-184.2989"},"phone":"403-470-9632 x156","website":"hale.biz","company":{"name":"Kobel-Corkery","catchPhrase":"Multi-tiered zero tolerance productivity","bs":"transition cutting-edge web services"}}, {"id":5,"name":"Chelsey Dietrich","username":"Kamren","email":"Lucio.Nettinger@qonda.ca","address":{"street":"Skiles Walks","suite":"Suite 551","city":"Kossowville","zipcode":"33383","geo":{"lat":"-31.8129","lng":"81.5341"},"phone":"(254)954-1289","website":"denarco.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 Corney","email":"Marley.Dach@jasper.info","address":{"street":"Norberto Crossing","suite":"Apt. 950","city":"South Christy","zipcode":"23965-1337","geo":{"lat":"-71.4197","lng":"71.7478"},"phone":"1-477-935-8478 x6430","website":"kila.org","company":{"name":"Considine-Lodman","catchPhrase":"Synchronised bottom-line interface","bs":"e-enable innovative applications"}}, {"id":7,"name":"Kurtis Welsch","username":"Elynn Skiles","email":"Telly.Hoeger@illy.biz","address":{"street":"Rex Trail","suite":"Suite 200","city":"Hovenmouth","zipcode":"58094-1099","geo":{"lat":"24.8918","lng":"71.8984"},"phone":"210.867.6332","website":"clivis.io","company":{"name":"Johns Group","catchPhrase":"Configurable multimedia task-force","bs":"generate enterprise e-tailers"}}, {"id":8,"name":"Nicholas Runolfsson Jr","username":"Maxine Wilems","email":"Shervood@osmond.me","address":{"street":"Ellsworth Summit","suite":"Suite 729","city":"Alijonville","zipcode":"45169","geo":{"lat":"-14.3998","lng":"-120.7677"},"phone":"586.493.694 x140","website":"jacythe.com","company":{"name":"Hernathy Group","catchPhrase":"Implemented secondary concept","bs":"e-enable extensible e-tailers"}}, {"id":9,"name":"Glenn Reichert","username":"Delphine","email":"Chaim.McDermott@dana.io","address":{"street":"Dayna Park","suite":"Suite 449","city":"Bartholomew","zipcode":"76495-1108","geo":{"lat":"24.6463","lng":"-168.8889"},"phone":"(775)976-6794 x41106","website":"conrad.com","company":{"name":"Lost and Sons","catchPhrase":"Switchable contextually-based project","bs":"aggregate real-time technologies"}}, {"id":10,"name":"Clementine DuBoise","username":"Vivian Stanton","email":"Key.Padberg@karina.biz","address":{"street":"Kattie Turnip","suite":"Suite 130","city":"Lebsackbury","zipcode":"31410-1261","geo":{"lat":"-38.3386","lng":"51.1332"},"phone":"004-648-3804","website":"ambrose.net","company":{"name":"Hoeger LLC","catchPhrase":"Centralized empowering task-force","bs":"target end-to-end models"}]}]

```

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

```

test_getuser.py ×
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!****")
23
24

```

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

Terminal:

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

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
phone: '1-463-123-4447', 'website': 'ramiro.info', 'company': {'name': 'Romaguera-Jacobson', 'catchPhrase': 'Face to face bifurcated interface', 'bs': 'e-enable strategic applications'}}, {'id': 4, 'name': 'Patricia Lebsack', 'username': 'Vernie', 'email': 'Julianne.Corn@borg.org', 'address': {'street': 'Hoeger Mall', 'suite': 'Apt. 692', 'city': 'South Elvis', 'zipcode': '53919-4257', 'geo': {'lat': '29.4572', 'lng': '-104.2998'}}, 'phone': '493-170-4023 x156', 'website': 'Kale.biz', 'company': {'name': 'Nobel-Corkery', 'catchPhrase': 'Multi-tiered zero tolerance productivity', 'bs': 'transition cutting-edge web services'}}, {'id': 5, 'name': 'Chelsey Dietrich', 'username': 'Kamren', 'email': 'Lucio.Hettinger@amle.ca', 'address': {'street': 'Skiles Walks', 'suite': 'Suite 351', 'city': 'Roscoeview', 'zipcode': '33263', 'geo': {'lat': '31.8129', 'lng': '62.5342'}}, 'phone': '(254)954-1289', 'website': 'banarco.info', 'company': {'name': 'Weebler LLC', 'catchPhrase': 'User-centric fault-tolerant solution', 'bs': 'revolutionize end-to-end systems'}}, {'id': 6, 'name': 'Mrs. Dennis Schulist', 'username': 'Leopoldo Corkery', 'email': 'Marley.Jacobi@jaguar.info', 'address': {'street': 'Norberto Crossing', 'suite': 'Apt. 990', 'city': 'South Christy', 'zipcode': '22995-1337', 'geo': {'lat': '-71.4397', 'lng': '71.7478'}}, 'phone': '1-477-935-8478 x6438', 'website': 'ola.org', 'company': {'name': 'Considine-Lockman', 'catchPhrase': 'Synchronized bottom-line interface', 'bs': 'e-enable innovative applications'}}, {'id': 7, 'name': 'Kurtis Weissnat', 'username': 'Elwyn.Skiles', 'email': 'Telly.Hoeger@billy.biz', 'address': {'street': 'Rex Trill', 'suite': 'Suite 288', 'city': 'Houmoult', 'zipcode': '33884-4939', 'geo': {'lat': '24.8918', 'lng': '21.8984'}}, 'phone': '720.867.6132', 'website': 'elvis.id', 'company': {'name': 'Johns Group', 'catchPhrase': 'Configurable multimedia task-force', 'bs': 'generate enterprise e-tailers'}}, {'id': 8, 'name': 'Nicholas Runolfsson', 'username': 'Maxine Nienow', 'email': 'Sherwood@rosamond.me', 'address': {'street': 'Ellsworth Summit', 'suite': 'Suite 729', 'city': 'Aliyaview', 'zipcode': '45169', 'geo': {'lat': '-14.3989', 'lng': '-120.7677'}}, 'phone': '586.493.6943 x148', 'website': 'jacynte.com', 'company': {'name': 'Alternathy Group', 'catchPhrase': 'Implemented secondary concept', 'bs': 'e-enable extensible e-tailers'}}, {'id': 9, 'name': 'Glenna Reichert', 'username': 'Delphine', 'email': 'Chaim.McDermott@iana.io', 'address': {'street': 'Dayna Park', 'suite': 'Suite 448', 'city': 'Bartholomew', 'zipcode': '76495-3198', 'geo': {'lat': '24.6463', 'lng': '-150.8889'}}, 'phone': '(775)976-6794 x41286', 'website': 'conrad.com', 'company': {'name': 'Vost and Sons', 'catchPhrase': 'Switchable contextually-based project', 'bs': 'aggregate real-time technologies'}}, {'id': 10, 'name': 'Clementine DuBuque', 'username': 'Noriah Stanton', 'email': 'Ney.Padberg@karina.biz', 'address': {'street': 'Mattie Turnpike', 'suite': 'Suite 198', 'city': 'Lebsackbury', 'zipcode': '31420-2261', 'geo': {'lat': '-38.2386', 'lng': '57.2232'}}, 'phone': '824-640-3894', 'website': 'ambrose.net', 'company': {'name': 'Hoeger LLC', 'catchPhrase': 'Centralized empowering task-force', 'bs': 'target end-to-end models'}}
****Code 200 is passed!****
===== 1 passed in 0.69s =====
```

6. 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!****")
```

7. 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

```
apis.py x
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
```

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

Sampling of no 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!****")
```

```
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 user12'
30     print("****User ID/Name " + str(response.json()['id']) + " and " + response.json()['username'] + " is added!****")
31
```

```

> # ensure that the data key (username) is added in the database
> assert response.json()['name'] == 'test user12'
E   AssertionError: assert 'test user1' == 'test user12'
E
E   - test user12
E   ?           -
E   + test user1

tests\test_postuser.py:29: AssertionError
===== short test summary info =====
FAILED tests\test_postuser.py::test_post_user_validation - AssertionError: assert 'test user1' == 'test user12'
```

9. 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!***")

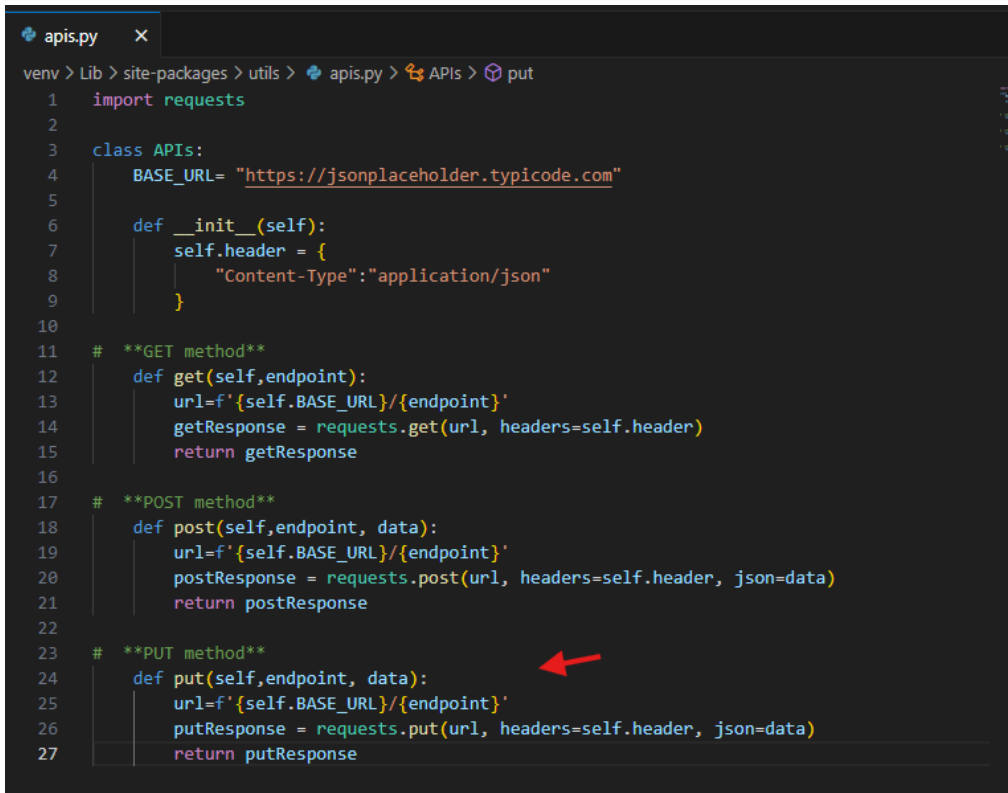
# ensure that the data key (username) is added in the database
assert response.json()['name'] == 'test user12'
print("***User ID/Name " + str(response.json()['id']) + " and " +
response.json()['username'] + " is added!***")

```

10. 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

```


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

Sampling of no error

```
test_putuser.py X
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 " + str(response.json()['id']) + " and " + response.json()['username'] + " 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

```
test_putuser.py X
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 " + str(response.json()['id']) + " and " + response.json()['username'] + " 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 ?      ^
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'

```

12. 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!****")

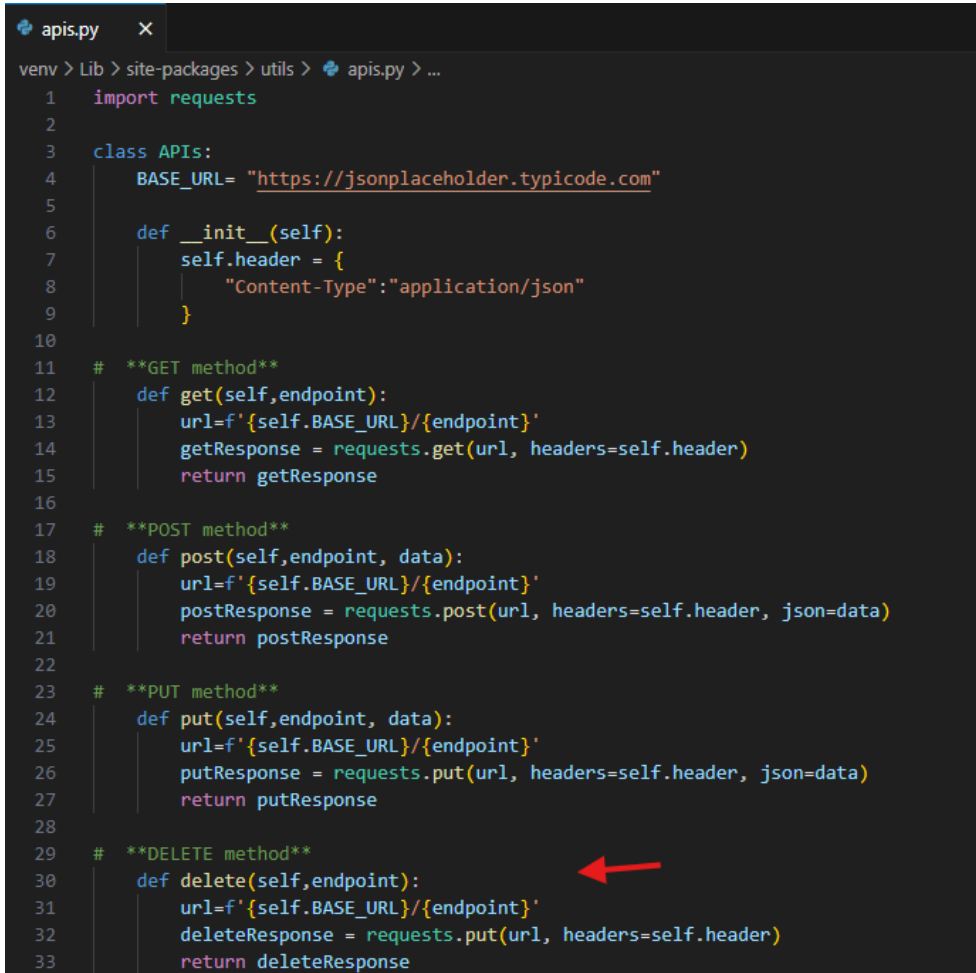
    # ensure that the data key (username) is updated in the database
    assert response.json()['username'] == 'test QA2'
    print("****User ID/Name " + str(response.json()['id']) + " and " +
    response.json()['username'] + " is updated!****")

```

13. 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
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.put(url, headers=self.header)
33         return deleteResponse
```

14. 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\njml\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\njml\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

```

15. 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!****)")
```

16. 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  }
```

17. 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!****")
34
```

```

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

tests/test_postuser_dyna_handl.py:25: AssertionError
===== short test summary info =====
FAILED tests/test_postuser_dyna_handl.py::test_post_user_validation - assert 201 == 200

```

18. 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****")

```


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

```
EXPLORER    PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
PYTHON_VCODE_PYTEST
  .pycache
  .github
  pytest_cache
  .vscode
  data
  cachedir: .pytest_cache
  metadata: {'Python': '3.12.6', 'Platform': 'Windows-11-10.0.22631-SP0', 'Packages': {'pytest': '8.3.3', 'pluggy': '1.5.0'}, 'Plugins': {'html': '4.1.1', 'metadata': '3.1.1'}}
  reports
  pytest
  configfile: pytest.ini
  plugins: html-4.1.1, metadata-3.1.1
  collected 5 items

test_deleteuser.py
test_getuser.py
test_postuser_dyna_handl.py
test_postuser.py
test_putuser.py
venv
  .venv
  .DELETE.method.txt
  .OGET.method.txt
  .OPOST.method.txt
  .OPUT.method.txt
  .OPUT.method.txt
  .confnet.py
  .pytest.ini
  .requirements.txt

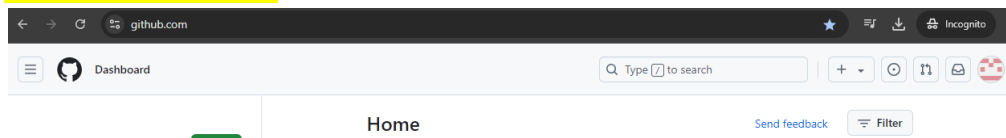
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\njml\Desktop\Test Automation notes\Python_VBCode\PyTest\pytest> -
===== test session starts =====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\njml\Desktop\Test Automation notes\Python_VBCode\PyTest\venv\Scripts\python.exe
cachedir: .pytest_cache
metadata: {'Python': '3.12.6', 'Platform': 'Windows-11-10.0.22631-SP0', 'Packages': {'pytest': '8.3.3', 'pluggy': '1.5.0'}, 'Plugins': {'html': '4.1.1', 'metadata': '3.1.1'}}
rootdir: C:\Users\njml\Desktop\Test Automation notes\Python_VBCode\PyTest
configfile: pytest.ini
plugins: html-4.1.1, metadata-3.1.1
collected 5 items

tests/test_deleteuser.py::test_delete_user_validation Start
{'id': 1}
****Code 200 is passed and data deleted!****
****Record {'id': 1} is deleted!****
PASSED
tests/test_getuser.py::test_get_user_validation [{"id": 1, "name": "Leanne Graham", "username": "Bret", "email": "Sincere@april.biz", "address": {"street": "Kulas Light", "suite": "Apt. 556", "city": "Gwen", "website": "Hildagard.org", "company": {"name": "Romaguera-Crona", "catchPhrase": "Multi-layered client-server neural-net", "bs": "harness real-time e-markets"}}, {"id": 2, "name": "Ervin Howell", "user", "city": "Miskolc", "zipcode": "90566-7771", "geo": {"lat": "-43.9698", "lng": "-34.4618"}, {"phone": "010-692-6993 x09125", "website": "anastasia.net", "company": {"name": "Deckow-Crist", "catchPhrase": "User", "username": "Samantha", "address": {"street": "Nathansyenia.net", "address": {"street": "Douglas Extension", "suite": "Suite 847", "city": "McKenziehaven", "zipcode": "59590-4157", "geo": {"lat": "-68.6192", "lng": "atchewase": "Face to face bifurcated interfaces"}, {"id": 4, "name": "Patricia Lebsack", "email": "Bullane.O'Connell@xoxo.org", "address": {"street": "145", "city": "145", "zipcode": "4572", "phone": "164.2996"}, {"id": 5, "name": "Katie Rompke", "suite": "Suite 198", "city": "Lebsackbury", "zipcode": "31428-2261", "geo": {"lat": "-38.2386", "lng": "57.2232"}, {"phone": "024-648-3804", "website": "ambrose.net", "bs": "transition cutting-edge", "address": {"street": "Skiles Walks", "suite": "Suite 351", "city": "Roscowview", "zipcode": "33263", "geo": {"lat": "-31.8129", "lng": "62.5342"}, {"phone": "(254)954-1289", "website": "demarco.info", "bs": "Implemented secondary concept", "bs": "e-enable extensible e-tailors"}, {"id": 9, "name": "Glenna Reichert", "username": "Delphine", "email": "Chain.MDermott@diana.io", "address": {"street": "Dayna Park", "city": "68.8889"}, {"phone": "(775)978-6794 x12060", "website": "conrad.com", "company": {"name": "Vost and Sons", "catchPhrase": "Switchable contextually-based project", "bs": "aggregate real-time technologies"}}, {"id": 11, "name": "Ervin Howell", "user", "city": "Miskolc", "zipcode": "90566-7771", "geo": {"lat": "-43.9698", "lng": "-34.4618"}, {"phone": "010-692-6993 x09125", "website": "anastasia.net", "company": {"name": "Deckow-Crist", "catchPhrase": "User", "username": "Samantha", "address": {"street": "Nathansyenia.net", "address": {"street": "Douglas Extension", "suite": "Suite 847", "city": "McKenziehaven", "zipcode": "59590-4157", "geo": {"lat": "-68.6192", "lng": "atchewase": "Face to face bifurcated interfaces"}, {"id": 4, "name": "Patricia Lebsack", "email": "Bullane.O'Connell@xoxo.org", "address": {"street": "145", "city": "145", "zipcode": "4572", "phone": "164.2996"}, {"id": 5, "name": "Katie Rompke", "suite": "Suite 198", "city": "Lebsackbury", "zipcode": "31428-2261", "geo": {"lat": "-38.2386", "lng": "57.2232"}, {"phone": "024-648-3804", "website": "ambrose.net", "bs": "transition cutting-edge", "address": {"street": "Skiles Walks", "suite": "Suite 351", "city": "Roscowview", "zipcode": "33263", "geo": {"lat": "-31.8129", "lng": "62.5342"}, {"phone": "(254)954-1289", "website": "demarco.info", "bs": "Implemented secondary concept", "bs": "e-enable extensible e-tailors"}, {"id": 9, "name": "Glenna Reichert", "username": "Delphine", "email": "Chain.MDermott@diana.io", "address": {"street": "Dayna Park", "city": "68.8889"}, {"phone": "(775)978-6794 x12060", "website": "conrad.com", "company": {"name": "Vost and Sons", "catchPhrase": "Switchable contextually-based project", "bs": "aggregate real-time technologies"}}, {"id": 11, "name": "Ervin Howell", "user", "city": "Miskolc", "zipcode": "90566-7771", "geo": {"lat": "-43.9698", "lng": "-34.4618"}, {"phone": "010-692-6993 x09125", "website": "anastasia.net", "company": {"name": "Deckow-Crist", "catchPhrase": "User", "username": "Samantha", "address": {"street": "Nathansyenia.net", "address": {"street": "Douglas Extension", "suite": "Suite 847", "city": "McKenziehaven", "zipcode": "59590-4157", "geo": {"lat": "-68.6192", "lng": "atchewase": "Face to face bifurcated interfaces"}, {"id": 4, "name": "Patricia Lebsack", "email": "Bullane.O'Connell@xoxo.org", "address": {"street": "145", "city": "145", "zipcode": "4572", "phone": "164.2996"}, {"id": 5, "name": "Katie Rompke", "suite": "Suite 198", "city": "Lebsackbury", "zipcode": "31428-2261", "geo": {"lat": "-38.2386", "lng": "57.2232"}, {"phone": "024-648-3804", "website": "ambrose.net", "bs": "transition cutting-edge", "address": {"street": "Skiles Walks", "suite": "Suite 351", "city": "Roscowview", "zipcode": "33263", "geo": {"lat": "-31.8129", "lng": "62.5342"}, {"phone": "(254)954-1289", "website": "demarco.info", "bs": "Implemented secondary concept", "bs": "e-enable extensible e-tailors"}, {"id": 9, "name": "Glenna Reichert", "username": "Delphine", "email": "Chain.MDermott@diana.io", "address": {"street": "Dayna Park", "city": "68.8889"}, {"phone": "(775)978-6794 x12060", "website": "conrad.com", "company": {"name": "Vost and Sons", "catchPhrase": "Switchable contextually-based project", "bs": "aggregate real-time technologies"}}, {"id": 11, "name": "Ervin Howell", "user", "city": "Miskolc", "zipcode": "90566-7771", "geo": {"lat": "-43.9698", "lng": "-34.4618"}, {"phone": "010-692-6993 x09125", "website": "anastasia.net", "company": {"name": "Deckow-Crist", "catchPhrase": "User", "username": "Samantha", "address": {"street": "Nathansyenia.net", "address": {"street": "Douglas Extension", "suite": "Suite 847", "city": "McKenziehaven", "zipcode": "59590-4157", "geo": {"lat": "-68.6192", "lng": "atchewase": "Face to face bifurcated interfaces"}, {"id": 4, "name": "Patricia Lebsack", "email": "Bullane.O'Connell@xoxo.org", "address": {"street": "145", "city": "145", "zipcode": "4572", "phone": "164.2996"}, {"id": 5, "name": "Katie Rompke", "suite": "Suite 198", "city": "Lebsackbury", "zipcode": "31428-2261", "geo": {"lat": "-38.2386", "lng": "57.2232"}, {"phone": "024-648-3804", "website": "ambrose.net", "bs": "transition cutting-edge", "address": {"street": "Skiles Walks", "suite": "Suite 351", "city": "Roscowview", "zipcode": "33263", "geo": {"lat": "-31.8129", "lng": "62.5342"}, {"phone": "(254)954-1289", "website": "demarco.info", "bs": "Implemented secondary concept", "bs": "e-enable extensible e-tailors"}, {"id": 9, "name": "Glenna Reichert", "username": "Delphine", "email": "Chain.MDermott@diana.io", "address": {"street": "Dayna Park", "city": "68.8889"}, {"phone": "(775)978-6794 x12060", "website": "conrad.com", "company": {"name": "Vost and Sons", "catchPhrase": "Switchable contextually-based project", "bs": "aggregate real-time technologies"}}, {"id": 11, "name": "Ervin Howell", "user", "city": "Miskolc", "zipcode": "90566-7771", "geo": {"lat": "-43.9698", "lng": "-34.4618"}, {"phone": "010-692-6993 x09125", "website": "anastasia.net", "company": {"name": "Deckow-Crist", "catchPhrase": "User", "username": "Samantha", "address": {"street": "Nathansyenia.net", "address": {"street": "Douglas Extension", "suite": "Suite 847", "city": "McKenziehaven", "zipcode": "59590-4157", "geo": {"lat": "-68.6192", "lng": "atchewase": "Face to face bifurcated interfaces"}, {"id": 4, "name": "Patricia Lebsack", "email": "Bullane.O'Connell@xoxo.org", "address": {"street": "145", "city": "145", "zipcode": "4572", "phone": "164.2996"}, {"id": 5, "name": "Katie Rompke", "suite": "Suite 198", "city": "Lebsackbury", "zipcode": "31428-2261", "geo": {"lat": "-38.2386", "lng": "57.2232"}, {"phone": "024-648-3804", "website": "ambrose.net", "bs": "transition cutting-edge", "address": {"street": "Skiles Walks", "suite": "Suite 351", "city": "Roscowview", "zipcode": "33263", "geo": {"lat": "-31.8129", "lng": "62.5342"}, {"phone": "(254)954-1289", "website": "demarco.info", "bs": "Implemented secondary concept", "bs": "e-enable extensible e-tailors"}, {"id": 9, "name": "Glenna Reichert", "username": "Delphine", "email": "Chain.MDermott@diana.io", "address": {"street": "Dayna Park", "city": "68.8889"}, {"phone": "(775)978-6794 x12060", "website": "conrad.com", "company": {"name": "Vost and Sons", "catchPhrase": "Switchable contextually-based project", "bs": "aggregate real-time technologies"}}, {"id": 11, "name": "Ervin Howell", "user", "city": "Miskolc", "zipcode": "90566-7771", "geo": {"lat": "-43.9698", "lng": "-34.4618"}, {"phone": "010-692-6993 x09125", "website": "anastasia.net", "company": {"name": "Deckow-Crist", "catchPhrase": "User", "username": "Samantha", "address": {"street": "Nathansyenia.net", "address": {"street": "Douglas Extension", "suite": "Suite 847", "city": "McKenziehaven",
```

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

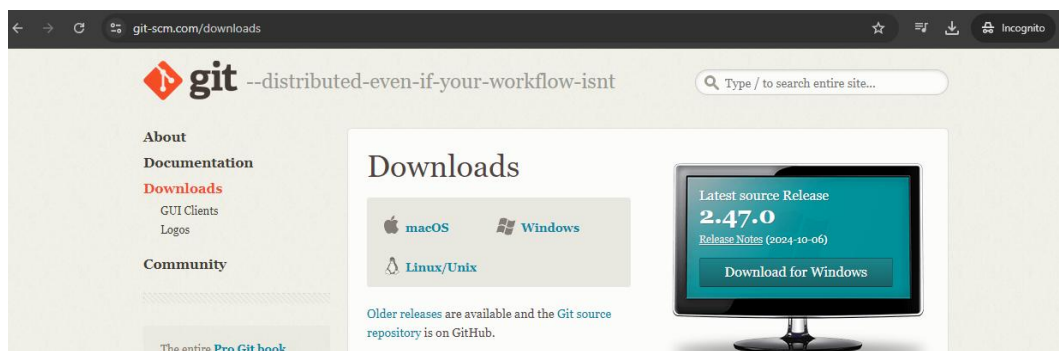
Have a GitHub account



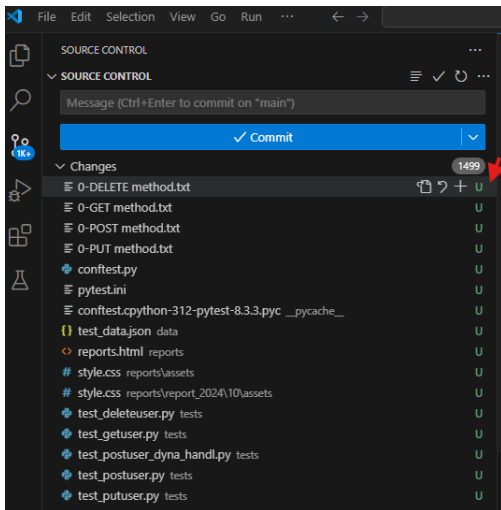
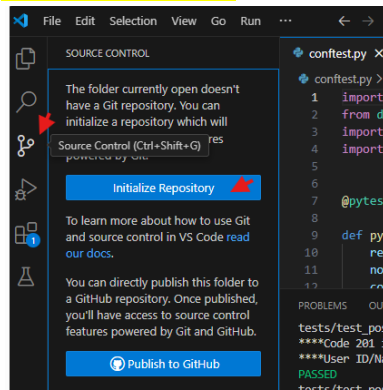
Download and install Git app

Download:

<https://git-scm.com/downloads>

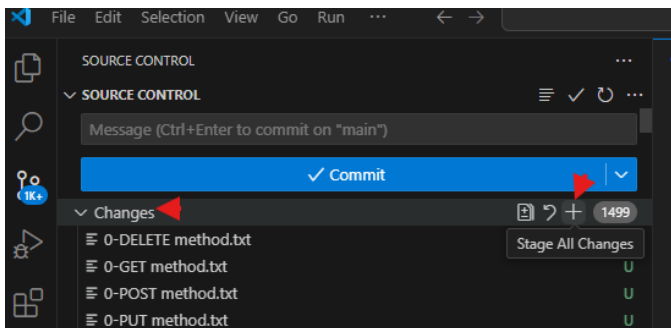


Initialize Repository

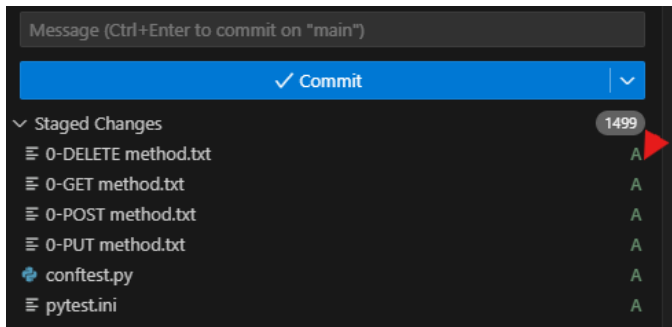


Stage all files

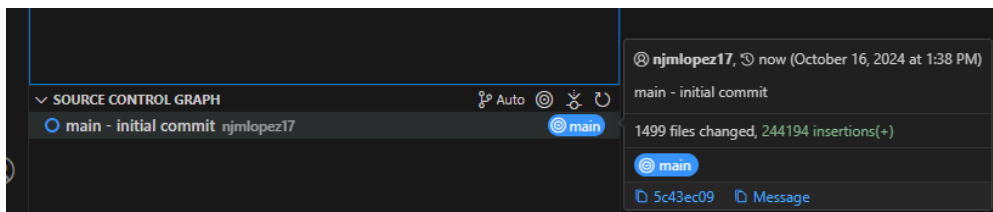
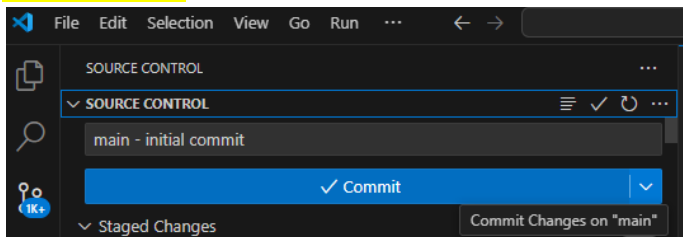
Before



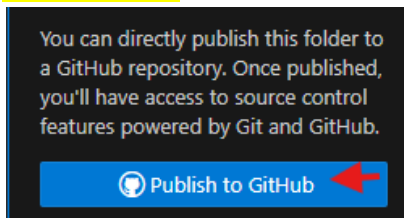
After



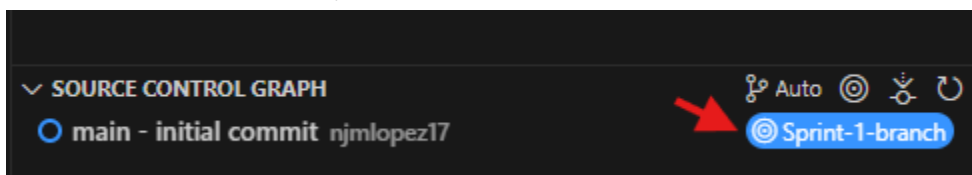
Commit all files



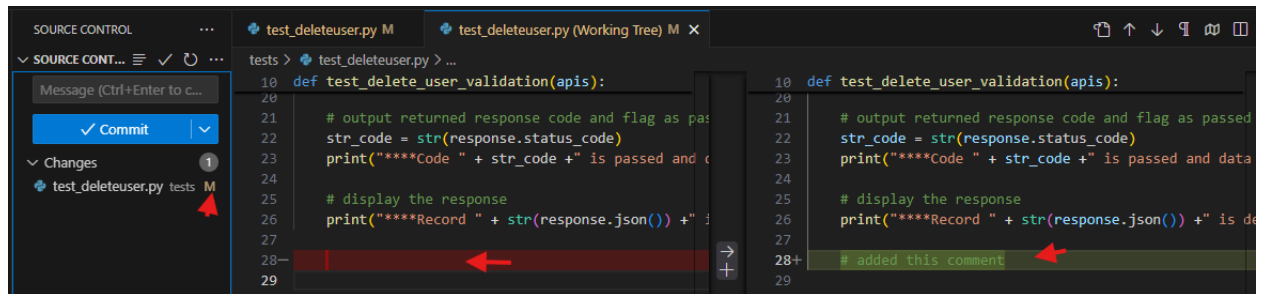
Publish GitHub



Other than the main branch, create a branch

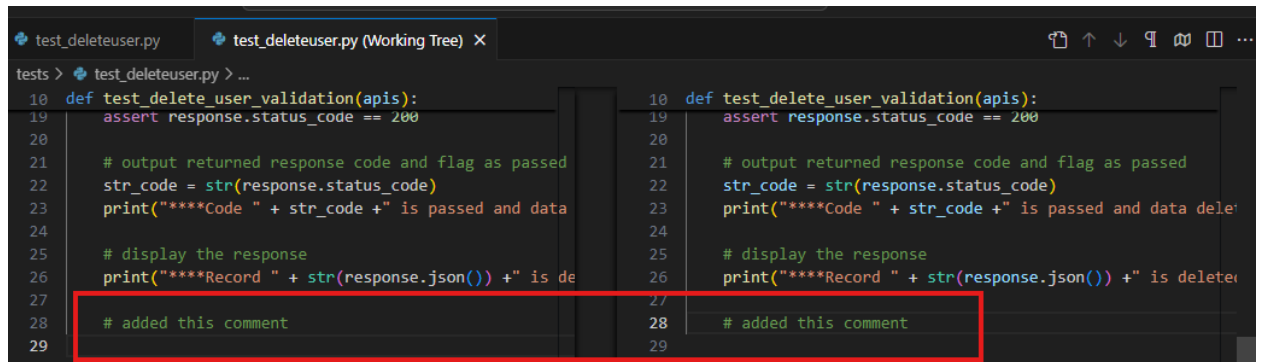


Sampling code change that is in the branch



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

Stage and commit the change

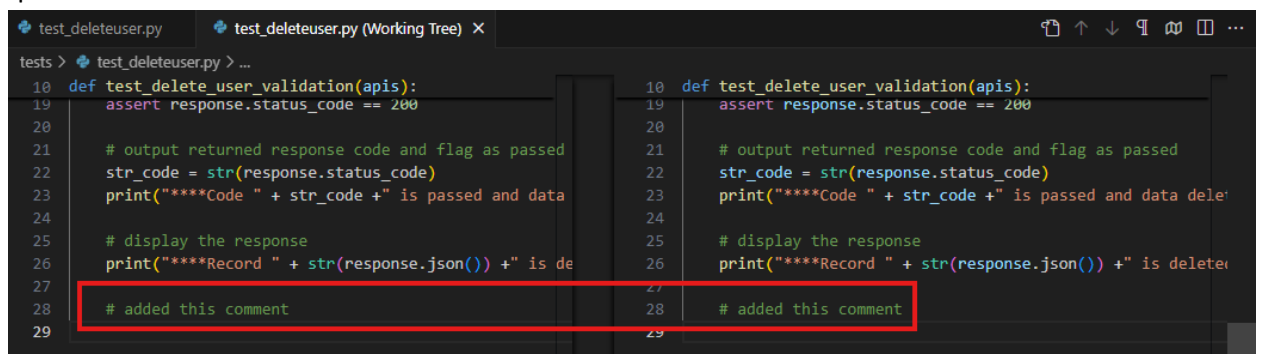


```
10 def test_delete_user_validation(apis):
19     assert response.status_code == 200
20
21     # output returned response code and flag as passed
22     str_code = str(response.status_code)
23     print("****Code " + str_code + " is passed and data deleted")
24
25     # display the response
26   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

Merge the spring branch into that of the main branch

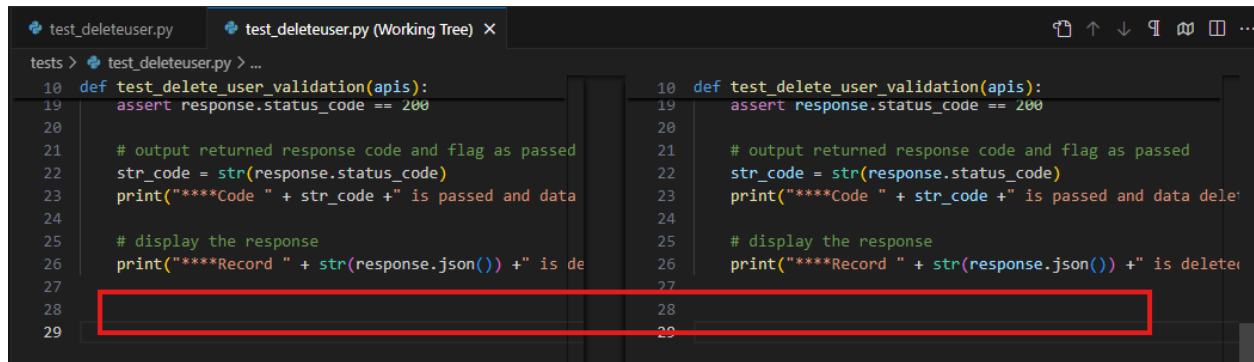
Before merging:

Sprint branch



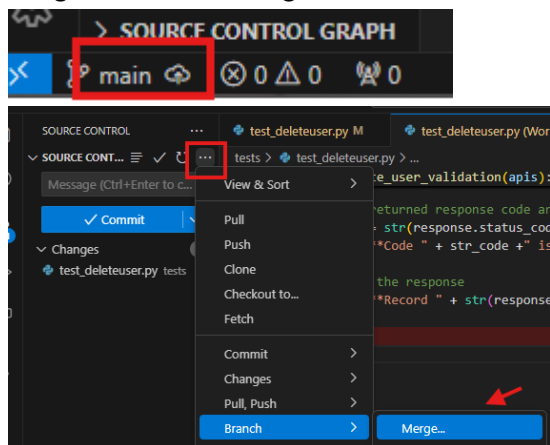
```
10 def test_delete_user_validation(apis):
19     assert response.status_code == 200
20
21     # output returned response code and flag as passed
22     str_code = str(response.status_code)
23     print("****Code " + str_code + " is passed and data deleted")
24
25     # display the response
26   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

Main branch



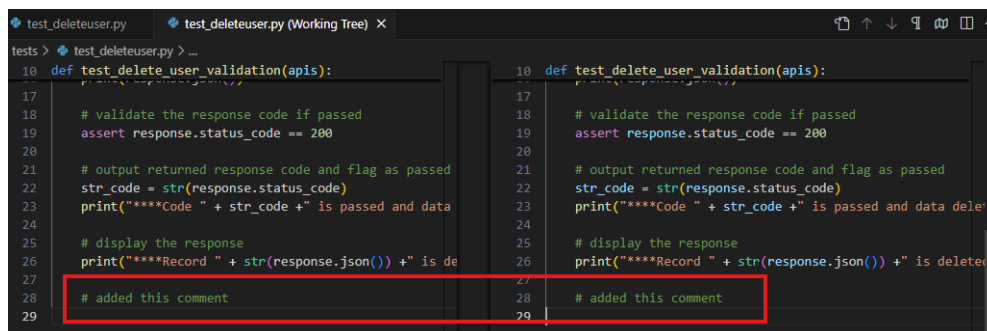
```
10 def test_delete_user_validation(apis):
19     assert response.status_code == 200
20
21     # output returned response code and flag as passed
22     str_code = str(response.status_code)
23     print("****Code " + str_code + " is passed and data deleted")
24
25     # display the response
26     print("****Record " + str(response.json()) + " is deleted")
27
28
29
```

Merge the branch changes



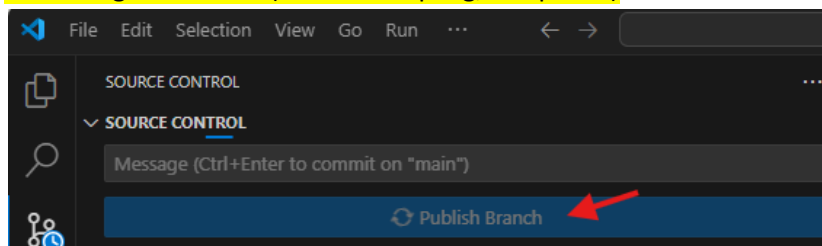
After merging:

Main branch

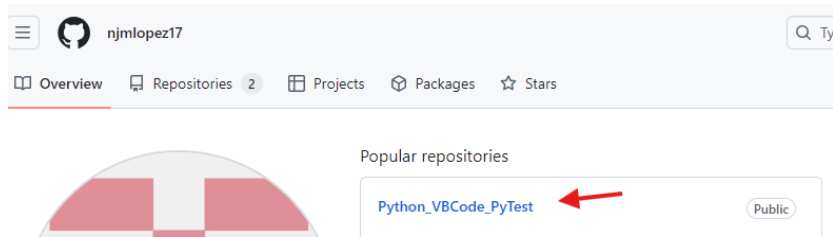


```
10 def test_delete_user_validation(apis):
17     # validate the response code if passed
18     assert response.status_code == 200
19
20     # output returned response code and flag as passed
21     str_code = str(response.status_code)
22     print("****Code " + str_code + " is passed and data deleted")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

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



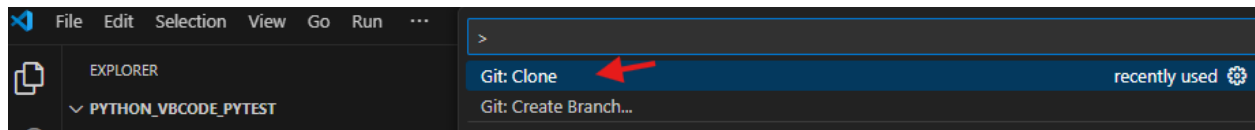
Added in GitHub



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)

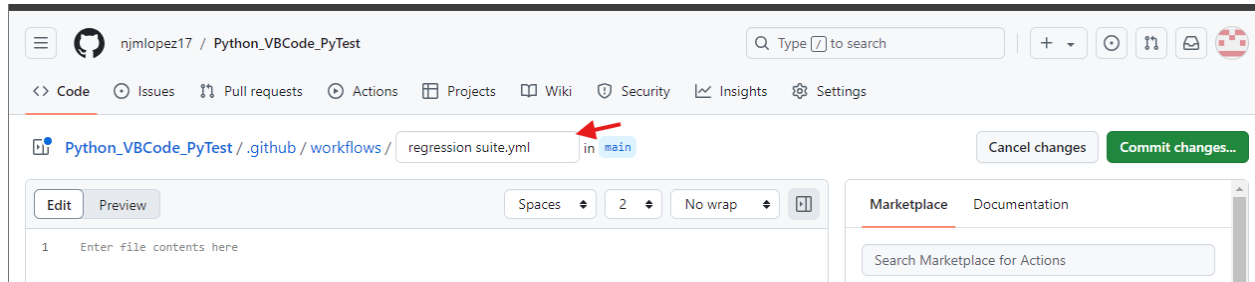


21. CI using GitHub Actions

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 for 'njmlopez17 / Python_VBCode_PyTest'. The file explorer on the left shows the directory structure, with '.github/workflows/regression suite.yml' selected. The main editor area displays the content of this file, which is a GitHub Actions workflow named 'Regression Suite'. The workflow is triggered on push to the main branch or pull requests to the main branch, and also runs on a scheduled cron job. The workflow includes jobs for running tests and uploading reports.

```
1  name: Regression Suite
2
3  on:
4    push:
5      branches:
6        - main
7    pull_request:
8      branches:
9        - main
10   schedule:
11     - cron: '30 2 * * *'
12
13   jobs:
14     run-tests:
15       runs-on: ubuntu-latest
16
17       steps:
18         - name: Checkout code
19           uses: actions/checkout@v3
20
21         - name: Set up Python
22           uses: actions/setup-python@v3
23           with:
24             python-version: '3.12.6'
25
26         - name: Install dependencies
27           run: |
28             python -m pip install --upgrade pip
29             pip install -r requirements.txt
30
31         - name: Run tests
32           run: |
33             pytest
34
35         - name: Upload report
36           uses: actions/upload-artifact@v3
37           with:
38             name: pytest-report
39             path: report.html
40
41
```

After a successful run (via Actions button)

Python_VBCode_PyTest / .github / workflows / regression suite.yml

njmlopez17 Update regression suite.yml ✓

All workflows

Showing runs from all workflows

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

Set up job 1s

Checkout code 1s

Set up Python 10s

Install dependencies 3s

Run tests 1s

```
1 ▶ Run pytest
2 ===== test session starts =====
3 platform linux -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0 -- /opt/hostedtoolcache/Python/3.12.6/x64/bin/python
4 cachedir: .pytest_cache
5 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'}
6 rootdir: /home/runner/work/Python_VBCode_PyTest/Python_VBCode_PyTest
7 configfile: pytest.ini
8 plugins: metadata-3.1.1, html-4.1.1
9 collecting ... collected 5 items
10
11 tests/test_deleteuser.py::test_delete_user_validation PASSED [ 20%]
12 tests/test_getuser.py::test_get_user_validation PASSED [ 40%]
13 tests/test_postuser.py::test_post_user_validation PASSED [ 60%]
14 tests/test_postuser_dyna_handl.py::test_post_user_validation PASSED [ 80%]
15 tests/test_putuser.py::test_update_user_validation PASSED [100%]
16
17 - Generated html report: file:///home/runner/work/Python_VBCode_PyTest/Python_VBCode_PyTest/reports/reports.html -
18 ===== 5 passed in 0.98s =====
```

Upload report 0s

Post Set up Python 0s

Post Checkout code 0s

Complete job 0s

1 Cleaning up orphan processes

*****END*****