

INSTRUCTION

Project name: Automation-Test-Project

Author: vdphuoc

Framework: Pytest – Playwright

Programming language: Python

CI/CD tool: Github Actions

Source: <https://github.com/vdphuoc/Automation-Test-Project.git>

Purpose: Demo automation source code on ecommerce website (<https://saucedemo.com>), user can run the test cases or implement more test cases base on project. Also, the project is already setup to trigger pipeline running with Github Actions after any changes.

1. Setup:

- Install python version 3.x (3.8/3.9)
- Install the library necessary (pip, pytest, playwright,..)
Ref: <https://playwright.dev/python/docs/intro>
- Clone source code: git clone <https://github.com/vdphuoc/Automation-Test-Project.git>
- Run command: python run_tests.py

2. How to setup CI/CD pipeline

Reason to select Github Actions: It is integrated into github repository, easy to use. No have much time to setup

Setup:

- Create workflow setup file as below: .github/workflows/python_app.yml

Explain:

```
name: Github Action pipeline  # Name of pipeline
on: [push, pull_request]      # Trigger action to run pipeline
jobs:                          # start workflow
  test:                        # define job name test
    runs-on: ubuntu-latest     # environment to run
    steps:                     # steps to run
      - name: Checkout code     # step name
        uses: actions/checkout@v2 # checkout to code

      - name: Set up Python      # step name
        uses: actions/setup-python@v2 # Setup Python Version, can change to newest version :
                                     setup-python@v5
```

```

with:
  python-version: '3.8'      # Specific version
- name: Install dependencies # step name
  run: |                     # execute command to install dependencies
    python -m pip install --upgrade pip      # install/upgrade the newest pip version
    pip install pytest pytest-html playwright # install pytest - playwright
    playwright install                       # install playwright lib
- name: Run tests and generate reports # step name
  run: python run_tests.py               # execute command to run test cases
- name: Upload Test Report              # step name
  uses: actions/upload-artifact@v2      # upload test result/report
with:
  name: test-report                    # step name
  path: reports                        # location get report file

```

Run the pipeline:

- Push any commits to repository, the pipeline will be triggered automatically

3. Execute test cases on local environment

Run specific test case:

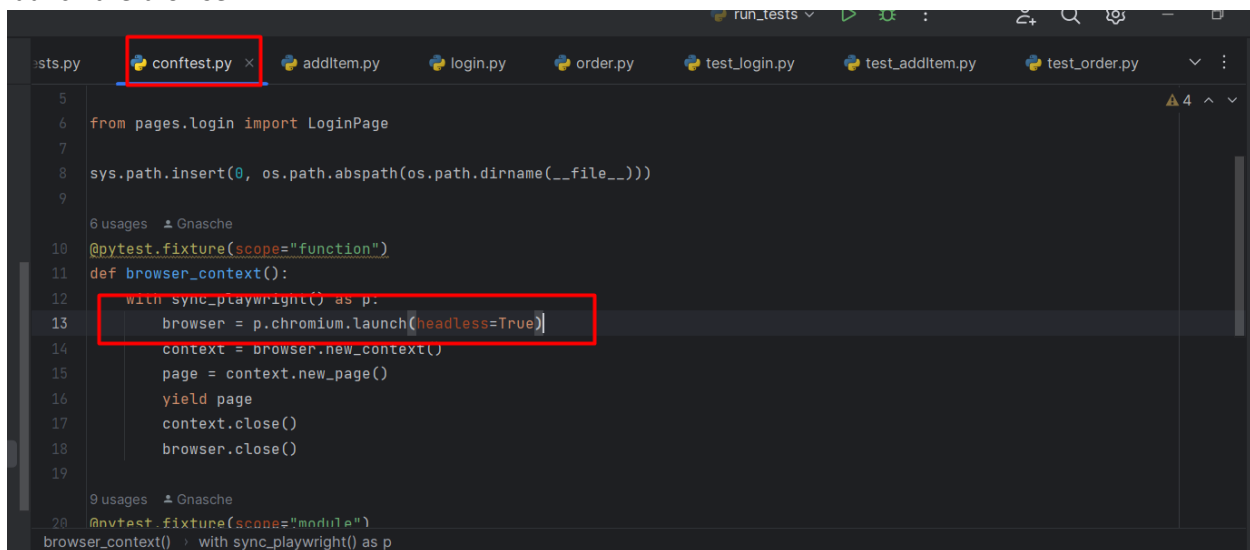
pytest tests/<test cases file>.py --html=reports/<report name>.html

Example: pytest tests/test_addItem.py --html=reports/report.html

Run all test cases:

python run_test.py

****Note:** Current mode is Headless mode, you need to turn off headless mode if you want to launch the browser.



```

5
6 from pages.login import LoginPage
7
8 sys.path.insert(0, os.path.abspath(os.path.dirname(__file__)))
9
10 @pytest.fixture(scope="function")
11 def browser_context():
12     with sync_playwright() as p:
13         browser = p.chromium.launch(headless=True)
14         context = browser.new_context()
15         page = context.new_page()
16         yield page
17         context.close()
18         browser.close()
19
20 @pytest.fixture(scope="module")
21 browser_context()
22 with sync_playwright() as p

```

4. Test cases and reports

- Number of test cases: 6 Test cases

- TC 1: Login successful
- TC 2: Login failed
- TC 3: Sort item by price (low to high)
- TC 4: Add to cart
- TC 5: Order successful (Failed test case – Assert failed)
- TC 6: Order failed (Failed test case – Can not find element, timeout after 10 seconds)

- Report:

- **Type:** HTML
- **Components:**
 - Environment
 - Total test cases executed on suite
 - Status of test case
 - Test case name
 - Time be executed
 - Filter option (Passed, Failed, Skipped, ..)
 - Log captured (after selecting test case)

report_test_addItem.html

Report generated on 16-Jun-2024 at 05:40:10 by [pytest-html](#) v4.1.1

Environment

Python	3.8.18
Platform	Linux-6.5.0-1021-azure-x86_64-with-glibc2.34
Packages	<ul style="list-style-type: none">• pytest: 8.2.2• pluggy: 1.5.0
Plugins	<ul style="list-style-type: none">• html: 4.1.1• metadata: 3.1.1
CI	true
JAVA_HOME	/usr/lib/jvm/temurin-11-jdk-amd64

Summary

1 test took 811 ms.

(Un)check the boxes to filter the results.

☐ 0 Failed, ☒ 1 Passed, ☐ 0 Skipped, ☐ 0 Expected failures, ☐ 0 Unexpected passes, ☐ 0 Errors, ☐ 0 Retuns

[Show all details](#) / [Hide all details](#)

Result	Test	Duration	Links
Passed	tests/test_additem.py::test_add_item_to_cart	811 ms	
No log output captured.			

☒ 2 Failed, ☐ 0 Passed, ☐ 0 Skipped, ☐ 0 Expected failures, ☐ 0 Unexpected passes, ☐ 0 Errors, ☐ 0 Retuns

[Show all details](#) / [Hide all details](#)

Result	Test	Duration	Links
Failed	tests/test_order.py::test_order_success	00:00:01	
<pre>order_page.checkout(undo['valid']['test_name'], undo['valid']['test_name'], undo['valid']['code']) order_page.finish_order(timeout=10000) # Verify order is successful success_message = order_page.get_success_message() > assert success_message == "THANK YOU FOR YOUR ORDER", "Order was not successful." E AssertionError: Order was not successful. E assert 'Thank you for your order!' == 'THANK YOU FOR YOUR ORDER' E E + THANK YOU FOR YOUR ORDER E + Thank you for your order! tests/test_order.py:22: AssertionError</pre>			
Failed	tests/test_order.py::test_order_failure	00:00:11	
<pre>task = asyncio.current_task(self._loop) st: list[inspect.FrameInfo] = getattr(task, "_pw_stack_", inspect.stack()) parsed_st = _extract_stack_trace_information_from_stack(st, is_internal) self._api_name.set(parsed_st) try: return await ob() except Exception as error: raise rewrite_error(error, f"[{parsed_st['apiName']}]: {error}") from None > playwright._impl._errors.TimeoutError: Page.click: Timeout 10000ms exceeded. E Call log: E waiting for locator("#finish") /opt/hostedtoolcache/Python/3.8.18/x64/lib/python3.8/site-packages/playwright/_impl/_connection.py:514: TimeoutError</pre>			