

Xiaojun Ruan

CS472

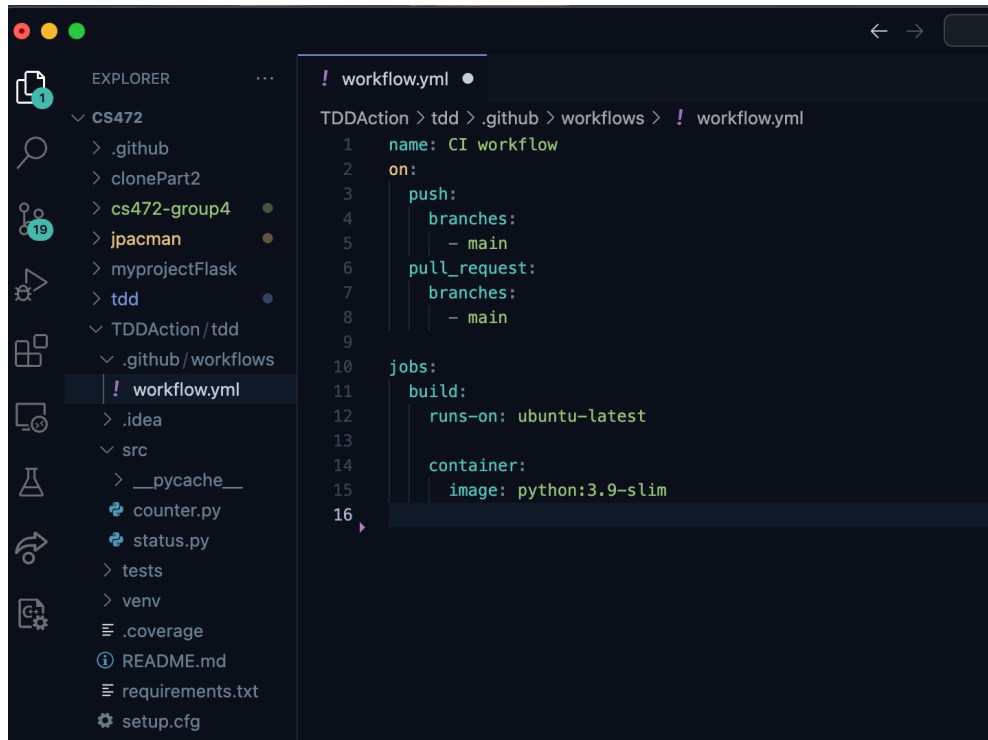
Feb 10, 2024

<https://github.com/junxjr/tdd>

Continuous Integration Report

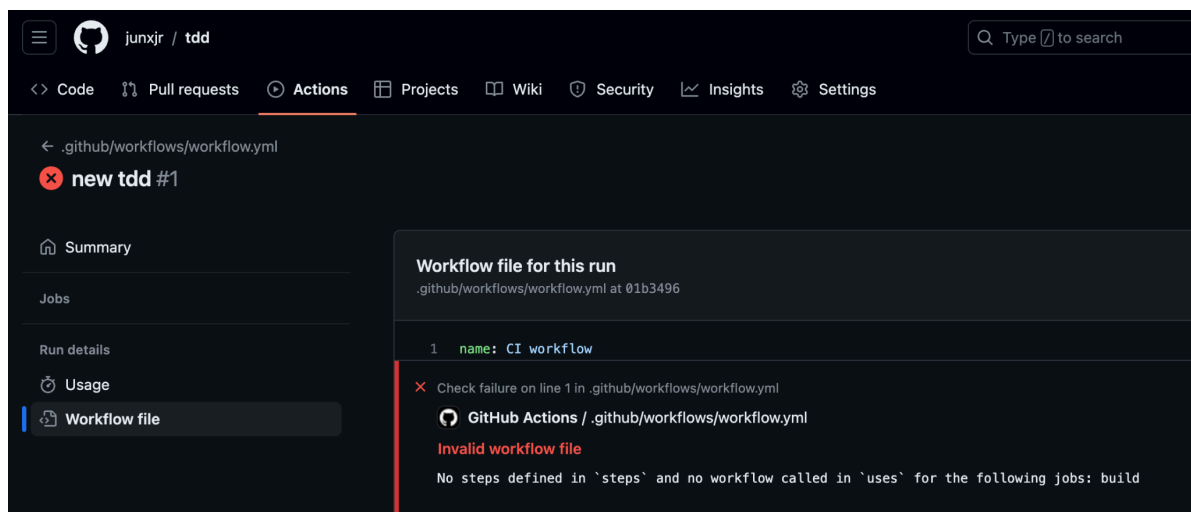
Task 1:

For this task, I followed all the instructions, and created workflow.yml. After I pushed the file to github, the actions tab failed as stated. Here are screenshots of my workflow and action tab.



The screenshot shows the VS Code interface with the Explorer and Source Control views on the left. The Explorer view shows the project structure, including the .github/workflows directory. The Source Control view shows the workflow.yml file. The main editor displays the content of workflow.yml:

```
1 name: CI workflow
2 on:
3   push:
4     branches:
5       - main
6   pull_request:
7     branches:
8       - main
9
10 jobs:
11   build:
12     runs-on: ubuntu-latest
13
14     container:
15       image: python:3.9-slim
16
```



The screenshot shows the GitHub Actions page for the repository junxjr / tdd. The Actions tab is selected, and the workflow file .github/workflows/workflow.yml is shown. The workflow run is labeled "new tdd #1" and is in a failed state, indicated by a red "X" icon. The failure message is:

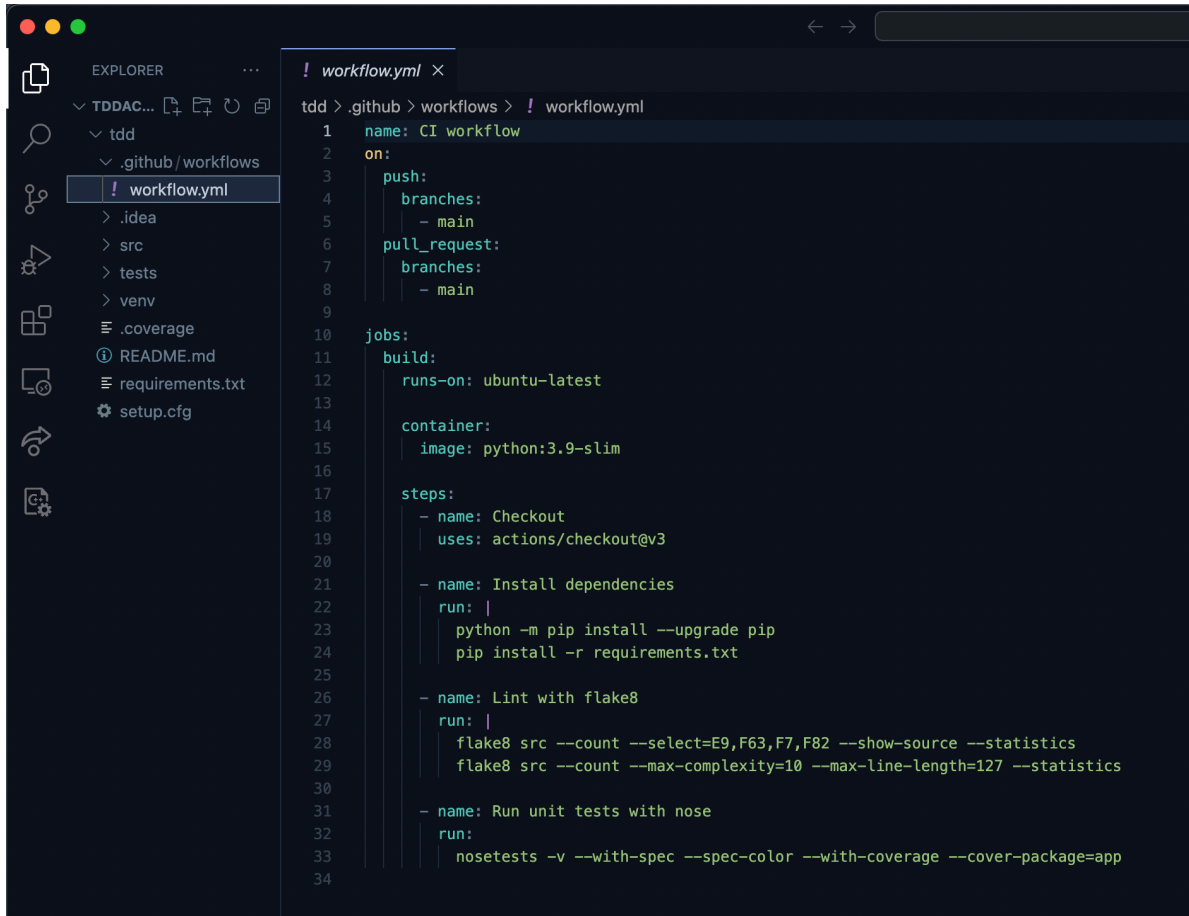
```
Workflow file for this run
.github/workflows/workflow.yml at 01b3496

1 name: CI workflow

X Check failure on line 1 in .github/workflows/workflow.yml
GitHub Actions / .github/workflows/workflow.yml
Invalid workflow file
No steps defined in 'steps' and no workflow called in 'uses' for the following jobs: build
```

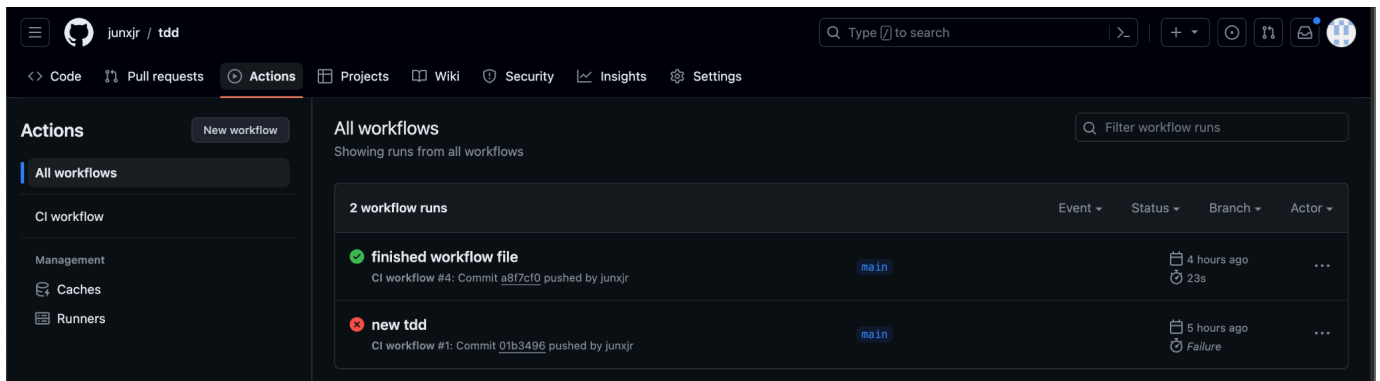
Task 2:

Moving on to task 2, I simply followed the instructions and added the steps. After I pushed my workflow to github, my actions passed successfully.



The screenshot shows a VS Code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a `.github/workflows` directory containing `! workflow.yml`. The code editor displays the content of `workflow.yml`, which is a GitHub Actions workflow for CI. The workflow is named "CI workflow" and is triggered on push to the main branch or pull requests to the main branch. It runs on the `ubuntu-latest` runner, uses a `python:3.9-slim` container, and includes four steps: Checkout, Install dependencies, Lint with flake8, and Run unit tests with nose.

```
1 name: CI workflow
2 on:
3   push:
4     branches:
5       - main
6   pull_request:
7     branches:
8       - main
9
10 jobs:
11   build:
12     runs-on: ubuntu-latest
13
14     container:
15       image: python:3.9-slim
16
17     steps:
18       - name: Checkout
19         uses: actions/checkout@v3
20
21       - name: Install dependencies
22         run: |
23           python -m pip install --upgrade pip
24           pip install -r requirements.txt
25
26       - name: Lint with flake8
27         run: |
28           flake8 src --count --select=E9,F63,F7,F82 --show-source --statistics
29           flake8 src --count --max-complexity=10 --max-line-length=127 --statistics
30
31       - name: Run unit tests with nose
32         run:
33           nosetests -v --with-spec --spec-color --with-coverage --cover-package=app
34
```



The screenshot shows the GitHub Actions interface for the repository `junxjr / tdd`. The "Actions" tab is selected, and the "All workflows" view is shown. The interface displays a table of workflow runs for the "CI workflow".

Event	Status	Branch	Actor
finished workflow file	Success	main	junxjr
new tdd	Failure	main	junxjr

The table shows two workflow runs. The first run, "finished workflow file", was triggered by a push to the main branch and completed successfully. The second run, "new tdd", was triggered by a push to the main branch and failed.

Task 2: Delete a counter

The RED Phase for delete a counter. I added a new test to delete a counter in my `test_counter.py` file. First, I created a counter using POST, then I checked if the return code was successful. After that, I deleted the counter using DELETE, and ensured that the return code is

204_NO_CONTENT. However, since I have not implemented the counter.py yet, it is in the RED phase. Here are the screenshots:

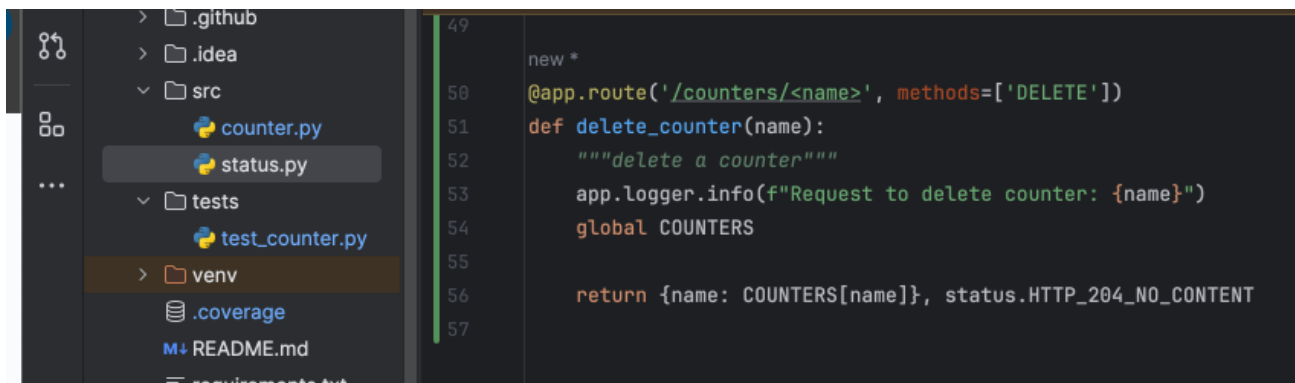
```
new *
def test_delete_a_counter(self):
    """It should delete a counter"""
    create = self.client.post('/counters/delete')
    self.assertEqual(create.status_code, status.HTTP_201_CREATED)

    delete = self.client.delete('/counters/delete')
    self.assertEqual(delete.status_code, status.HTTP_204_NO_CONTENT)
```

```
Counter tests
- It should create a counter
- It should delete a counter (FAILED)
- It should return an error for duplicates
- It should read a counter
- It should update a counter

=====
FAIL: It should delete a counter
-----
Traceback (most recent call last):
  File "/Users/xiaojunrui/Desktop/CS472/TDDAction/tdd/tests/test_counter.py", line 69, in test_delete_a_counter
    self.assertEqual(delete.status_code, status.HTTP_204_NO_CONTENT)
AssertionError: 405 != 204
```

The GREEN Phase for delete a counter. In the counter.py, I first created a route to delete a counter for the method. With the global counter, I returned it with the 204_NO_CONTENT code for delete successfully.



```
new *
49
50 @app.route('/counters/<name>', methods=['DELETE'])
51 def delete_counter(name):
52     """delete a counter"""
53     app.logger.info(f"Request to delete counter: {name}")
54     global COUNTERS
55
56     return {name: COUNTERS[name]}, status.HTTP_204_NO_CONTENT
57
```

Counter tests

- It should create a counter
- It should delete a counter
- It should return an error for duplicates
- It should read a counter
- It should update a counter

Name	Stmts	Miss	Cover	Missing
src/counter.py	24	0	100%	
src/status.py	6	0	100%	
TOTAL	30	0	100%	

Ran 5 tests in 0.297s

OK

The Refactoring Phase for delete a counter was simply checking the code was formatted correctly and ensuring the coverage is 100%. Other than that, I didn't really change the code.

The screenshot shows an IDE with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project structure with files like `counter.py`, `status.py`, `test_counter.py`, `venv`, `coverage`, `README.md`, `requirements.txt`, `setup.cfg`, `External Libraries`, and `Scratches and Consoles`. The code editor shows the implementation of the `delete_counter` function in `src/counter.py` and a test function `test_delete_a_counter` in `test_counter.py`. The terminal window displays the output of running the tests, showing a list of tests, a table of coverage results, and the total number of tests passed.

```
new *
@app.route('/counters/<name>', methods=['DELETE'])
def delete_counter(name):
    """delete a counter"""
    app.logger.info(f"Request to delete counter: {name}")
    global COUNTERS
    return {name: COUNTERS[name]}, status.HTTP_204_NO_CONTENT
S7

new *
def test_delete_a_counter(self):
    """It should delete a counter"""
    create = self.client.post('/counters/delete')
    self.assertEqual(create.status_code, status.HTTP_201_CREATED)

    delete = self.client.delete('/counters/delete')
    self.assertEqual(delete.status_code, status.HTTP_204_NO_CONTENT)
70
```

CounterTest : test_update_a_counter()

Terminal Local x + v

```
tions', group='nose.plugins'): No module named 'pkg_resources'
warn("Unable to load plugin %s: %s" % (ep, e),

Counter tests
- It should create a counter
- It should delete a counter
- It should return an error for duplicates
- It should read a counter
- It should update a counter

Name          Stmts  Miss  Cover   Missing
-----
src/counter.py  24      0   100%
src/status.py   6      0   100%
TOTAL          30      0   100%

Ran 5 tests in 0.297s

OK
```