# **GITHUB ACTIONS – Assignment Documentation**

## **Procedure I Followed**

- 1. I started by creating a new repository on GitHub for my assignment and then cloned it to my local system using Visual Studio Code.
- 2. Inside the repository, I created a small **Python project** that contained two basic functions one for addition and one for subtraction.
- 3. I created a file named math operations.py inside a folder called src and added two functions:

```
o def add(a, b):
    return a + b

o def sub(a, b):
    return a - b
```

4. I then created a separate Python file for **unit testing** using pytest.

This file imported the functions from src and verified their correctness with multiple test cases for both addition and subtraction.

- 5. Next, I created a **requirements.txt** file to list the dependencies used in the project, which included:
  - o pandas
  - o pytest
  - o flake8
- 6. After preparing the code and tests, I moved on to creating the **GitHub Actions workflow** to automate the build and test process.
- 7. Inside the repository, I made a directory structure:

```
.github/workflows/
```

and created a YAML file named main.yml inside it.

- 8. In the main.yml file, I defined a workflow named **Python application** that would:
  - o Trigger automatically on every **push** and **pull request** to the main branch.
  - o Use an **Ubuntu environment** to run the job.
  - o Set up **Python 3.10**.
  - o Install all project dependencies.
  - o Run **flake8** for linting.
  - o Run **pytest** for testing.
- 9. After writing and saving the workflow file, I committed all my changes and pushed them to the main branch on GitHub.

- 10. Once pushed, I opened the **Actions tab** in my GitHub repository to verify that the workflow was triggered automatically.
- 11. The pipeline successfully ran all the steps setting up Python, installing dependencies, linting, and running all tests without any errors.
- 12. Finally, I updated the **README file** of my repository to describe:
  - o The purpose of the project.
  - o The role of GitHub Actions.
  - o The steps included in the workflow (build, lint, and test).
- 13. This completed the setup and verification of my **GitHub Actions CI pipeline**, ensuring that every time code is pushed to the main branch, it automatically checks for syntax errors and verifies functionality through tests.

#### FILES:-

#### **Unittest.yaml**

```
name: Python application
  branches: [ "main" ]
 contents: read
    - uses: actions/checkout@v4
    - name: Set up Python 3.10
     uses: actions/setup-python@v3
     name: Install dependencies
       python -m pip install --upgrade pip
       pip install flake8 pytest
       if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
    - name: Lint with flake8
       # stop the build if there are Python syntax errors or undefined names
        flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics
        # exit-zero treats all errors as warnings. The GitHub editor is 127 chars wide
       flake8 . --count --exit-zero --max-complexity=10 --max-line-length=127 --statistics
     name: Test with pytest
       pytest
```

## Math\_operations.py

```
def add(a,b):
    return a + b

def sub(a,b):
    return a-b
```

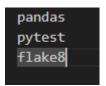
#### Test\_operations.py

```
from src.math_operations import add, sub

def test_add():
    assert add(2,3)==5
    assert add(-1,1)==0

def test_sub():
    assert sub(5,3)==2
    assert sub(2,3)==-1
    assert sub(3,3)==0
    assert sub(-2,-1)==-1
```

#### Requirements.txt



#### **Results:-**

