Lab: Testing and Version Control with Git

Context

You and your partner are part of a small development team working on a Python calculator library.
Your goals are to apply version control practices with Git and build confidence in testing (unit and integration) — exactly as in a real collaborative workflow.

Section 1 – Git Setup and Collaboration Basics

1. One student creates a new local Git repository and shares it via GitHub.
2. Both partners clone the project locally and create a virtual environment (`venv`).
3. Create a `.gitignore` file that ignores `venv`, `__pycache__`, and temporary files.
4. In `src/utils.py`, partner A implements add and subtract functions taking "unlimited" number of params.
5. Partner B creates a new branch called `feature/multiply_divide` and adds the multiply and divide functions.
6. Use branches and merges to integrate the code. Simulate a code review before merging into `main`.

Section 2 – Unit Testing & Quality Assurance

1. Install the tools:

```
pip install pytest pytest-cov flake8
```

2. Create the folder `tests/` and a file `tests/test_utils.py`:

3. Each partner writes tests for their own functions.
4. Include normal and edge cases (e.g., division by zero).
5. Run linting with flake8 and tests with pytest.
6. Add a `requirements.txt` file and commit everything.

Section 3 – Integration & Advanced Git Collaboration

Part A

1. Create `src/calculator.py` that:

1. Prints a menu of operations (1–4).
2. Accepts two numbers as input.
3. Calls the correct function from `utils.py`.
4. Handles invalid input (non-numeric, wrong choice, divide by zero).
5. Asks if the user wants another calculation.

Hint: Use `try/except` for robustness.

2. Create tests/test_calculator_integration.py and simulate user interaction.

Part B

1. Both partners create a new branch from main:
    a. Partner A: feature/improve-add
    b. Partner B: feature/improve-subtract
2. Each edits the same line in utils.py (e.g., add print statements).
3. Merge both branches into main to intentionally trigger a conflict.
4. Work together to resolve the conflict correctly.
5. Run all tests again to confirm functionality remains intact.

Deliverables

A document containing the link to the github repository as well as for:

- **Section A –** a clean Git history showing branch merges (e.g., `git log --graph --oneline --all`)
- **Section B –** Screenshot or text output showing all tests passing, coverage ≥ 90%, and no linter errors.
- **Section C –** Passing integration tests proving the calculator runs end-to-end.