Setup the Environment

```
cd playwright-python-automation
sh setup_env.sh
```

Start the mock frontend server

```
cd  playwright-python-automation/mock_frontend
python -m http.server 8000
```

Generate the report and run the tests

```
ENV_NAME=qa .venv/bin/pytest tests/ --html=reports/report.html --self-contained-html
```

1. API Test Suite

CRUD operations testing

   >>> GET and POST endpoints for users and transaction
    Support for mock and execute the real API

Error scenario handling

   >>> Invalid input data (400)
   >>> Unauthorized access (401 / 403)
   >>>Resource not found (404)
   >>>Server errors (500)

Data validation tests

   >>> DataAccessCompare to compares response and validation payload
   >> mappers transaction_config.json, users_config.json - response and validation payload

Authentication/authorization tests

   >> Invalid token , expiry token , cross users access, limited access of the user

2. UI Test Suite (can use mock frontend)

User registration flow

  >> src/page_objects/users_page.py
 >> tests/ui/test_user_API.py
 >> testdata/testdadta_qa.py

Transaction creation flow

 >> src/page_objects/transaction_page.py
 >> tests/ui/test_transaction_API.py
 >> testdata/testdadta_qa.py

Error message validation

 >>> data validation , empty data, invalid data

3. Test Utilities

 Test data factories - user and transaction
 >> test/factories/TransactionFactory.py
 >> test/factories/UserFactory.py

Helper functions

 src/utils

Environment configuration

 .env.qa
 .env.staging

Custom assertions

 src/utils/CustomAssert

4. Reporting

Test results (Bonus -multiple formats)

>>> playwright-python-automation/reports/report.html?sort=result

Screenshots for UI failures

>>> screenshot on failure -path seen in the testcase failure

Example - Path
playwright-python-automation/reports/screenshots/test_user_creation_error[users2]_20251031_051930.png

API response logging

>>> Logger class

Screenshot of the Result