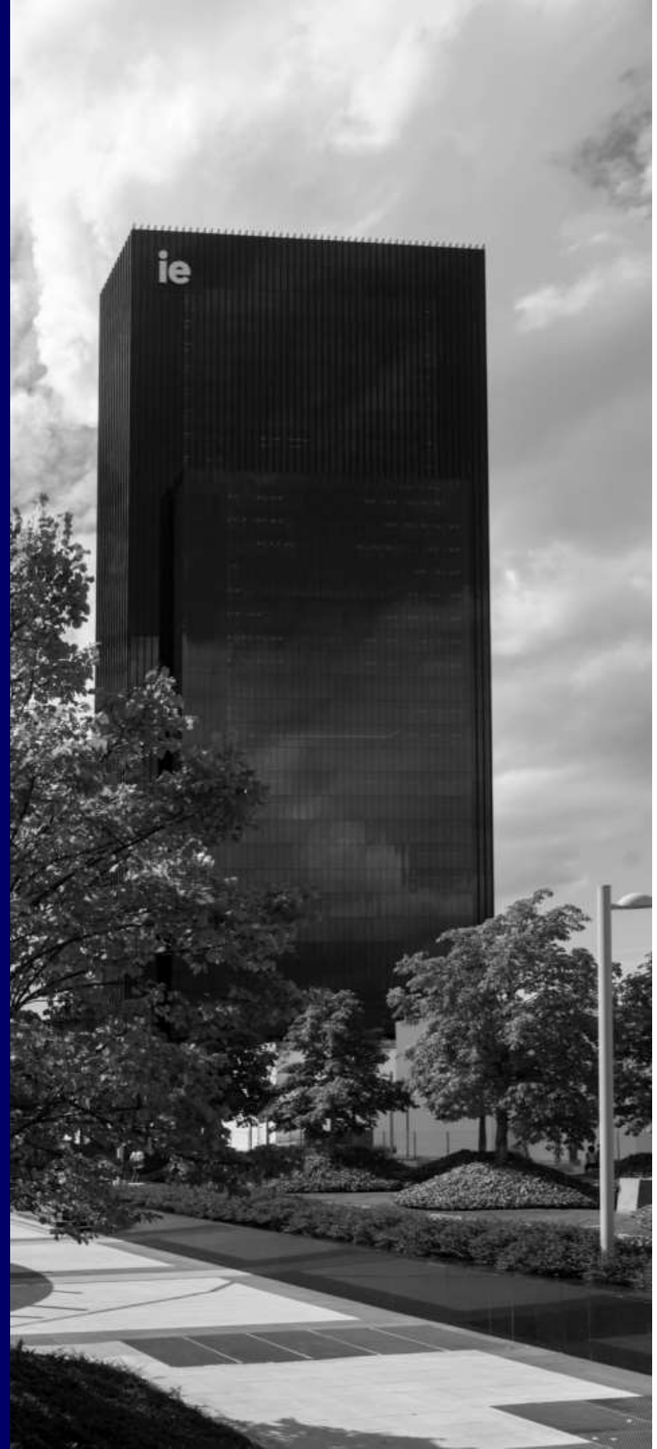

ie UNIVERSITY
SCHOOL OF SCIENCE & TECHNOLOGY

SOFTWARE DEVELOPMENT AND DEVOPS

Testing

Testing

- **Software quality** means how well software meets user needs and conforms to functional and non-functional requirements.
- Key quality dimensions include **correctness, reliability, maintainability,** and **performance.**
- **Testing** is a core part of software quality assurance: it provides evidence that **code behaves as intended**.
- Effective testing improves quality by **detecting defects early**, **preventing regressions**, and increasing **confidence in changes**.

Testing

Type	Purpose	Example	Python Tools
Unit Testing	Test individual functions/classes in isolation	Testing a calculate_total() function	pytest, unittest
Integration Testing	Test how components interact	Function calling database or API	pytest, requests-mock
Regression Testing	Ensure new code doesn't break existing features	Re-run previous tests after new commits	pytest --lf or CI/CD pipelines
Performance Testing	Check speed and scalability	Measure execution time under load	pytest-benchmark
Acceptance Testing	Validate against business requirements	End-to-end test via user workflows	behave, pytest-bdd

Fixtures

Provide reusable setup data or objects for tests

```
import pytest


@pytest.fixture
def api_url():
    """Fixture providing the base API URL"""
    return "https://api.example.com"
```

Parametrized tests

Multiple input/output pairs for a single test

```
import pytest
from app import add


@pytest.mark.parametrize("a,b,expected", [
    (1, 1, 2),
    (2, 3, 5),
    (-1, 1, 0),
])
def test_add_param(a, b, expected):
    assert add(a, b) == expected
```

Patching / Monkey Patching

Overriding calls to dependencies to test individual components

Modifying state at execution time

```
def test_greet(monkeypatch):
    monkeypatch.setenv("USER", "Alice")
    from app.greetings import greet
    assert greet() == "Hello, Alice!"
```

Mocking

Simulating an object's behavior using a mock object that records calls and can return fake data.

```
import pytest


@pytest.fixture
def mock_api_response(mocker):
    """Fixture providing a mock response object"""
    response = mocker.Mock()
    response.status_code = 200
    return response
```

THANK YOU!

ie UNIVERSITY
SCHOOL OF SCIENCE & TECHNOLOGY